D. Konstantas
J.-P. Bourrières
M. Léonard
N. Boudjlida
Editors

Interoperability of Enterprise Software and Applications

WITH CD-ROM

Springer

**Return or exchange
only possible when packaging unopened**

Interoperability of Enterprise Software and Applications

Dimitri Konstantas, Jean-Paul Bourrières,
Michel Léonard and Nacer Boudjlida (Eds.)

Interoperability of Enterprise Software and Applications

With 162 Figures

INTEROP-ESA'05

First International Conference on Interoperability of Enterprise Software and Applications
February 23-25, 2005, Geneva Switzerland

Springer

Dimitri Konstantas, Prof. Dr. Eng.
Centre Universitaire d'Informatique
Department of Information Systems
Université de Genève
24 rue du Général-Dufour
CH-1211 Genève 4
Switzerland

Jean-Paul Bourrières, Prof. Dr.
Université Bordeaux 1
Laboratoire d'Automatique et de Productique
Signal et Images
351, cours de la Libération
33405 Talence cedex
France

Michel Léonard, Prof. Dr.
Centre Universitaire d'Informatique
Department of Information Systems
Université de Genève
24 rue du Général-Dufour
CH-1211 Genève 4
Switzerland

Nacer Boudjlida, Prof. Dr.
Université Henri Poincaré Nancy 1
Bâtiment LORIA
BP 239
54506 Vandœuvre Lès Nancy cedex
France

British Library Cataloguing in Publication Data
INTEROP-ESA 2005 (Conference: 2005)
Interoperability of enterprise software and applications
1. Internetworking (Telecommunications) - Congresses
I. Title II. Konstantas, Dimitri, 1958-
004.6
ISBN-13: 9781846281518
ISBN-10: 1846281512

Library of Congress Control Number: 2005931919

ISBN-10: 1-84628-151-2 e-ISBN 1-84628-152-0 Printed on acid-free paper
ISBN-13: 978-1-84628-151-8

© Springer-Verlag London Limited 2006

The software disk accompanying this book and all material contained on it is supplied without any warranty of any kind. The publisher accepts no liability for personal injury incurred through use or misuse of the disk.

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licences issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

The use of registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Printed in Germany

9 8 7 6 5 4 3 2 1

Springer Science+Business Media
springeronline.com

Preface

Dimitri Konstantas[1], Jean-Paul Bourrières[2], Michel Leonard[1] and Nacer Boudjlida[3]

[1] University of Geneva, Dept. of Information Systems/CUI, CH-1211 Geneva 4, Switzerland, Dimitri.Konstantas@unige.ch, Michel.Leonard@unige.ch

[2] University of Bordeaux I, Laboratoire d'Automatique et de Productique, Signal et Images, 351 cours de la Liberation, F-33405, Talence cedex, France, bourrieres@lap.u-bordeaux1.fr

[3] University Henri Poincare Nancy I, UHP Nancy 1, Campus Scientifique, Bâtiment LORIA, BP 239, F-54506 Vandœuvre Lès Nancy CEDEX, France, Nacer.Boudjlida@loria.fr

The interoperability in enterprise applications can be defined as the ability of a system or a product to work with other systems or products without special effort from the customer or user.

The possibility to interact and exchange information with internal and external collaborators is a key issue in the enterprise sector. It is fundamental in order to produce goods and services quickly, at lower cost, while maintaining higher levels of quality and customisation.

Interoperability is considered achieved if the interaction can, at least, take place at the three levels: data, applications and business enterprise through the architecture of the enterprise model and taking into account the semantics. It is not only a problem of software and IT technologies. It implies support of communication and transactions between different organisations that must be based on shared business references.

The INTEROP-ESA (I-ESA) conference targets in becoming a major event for both research and industrial development in interoperability, bringing together researchers, users and practitioners dealing with different issues of Interoperability of Enterprise Applications and Software. The conference covers interoperability related research areas ranging from Enterprise Modelling to define interoperability requirements, to Architecture and Platforms, to provide implementation frameworks and Ontologies to define interoperability semantics in the enterprise.

INTEROP-ESA 2005

The First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'2005) was held in Geneva, Switzerland, in February 21 to 25, 2005. It was organized by the INTEROP NoE (IST-508-011) and was Supported by IFIP (International Federation for Information Processing), ACM

SIGAPP (Special Interest Group on Applied Computing) and the Swiss State Secretariat for Education and Research (SER); the conference was hosted by the University of Geneva. and was co-located and organised with the eGOV INTEROP'05 Annual Conference (organized by the Observatory on "Interoperable eGovernment Services").

The INTEROP-ESA'05 conference attracted a total of 85 full paper submissions out of which 35 were retained for presentation. Keynote presentations were given by prominent invited speakers, including Prof. Dennis Tsichritzis Vice-President of the Fraunhofer Institute (Germany)[1], Dr. Hervé Le Guyader, President of *eris@* (France), Dr. Gérald Santucci, Head of the ICT for Enterprise Networking Unit at the European Commission (EU), Dr. Andrew Watson, Vice President of OMG (UK) and Prof. Xiaofei Xu, Dean of the Harbin Institute of Technology (China).

In addition to the 12 sessions of scientific papers, the conference also included two special sessions with invited presentations from industry and from other related research projects funded by the EU.

In the parallel program of the conference 4 workshops and 2 tutorials were organized as well as a doctoral symposium. In the doctoral symposium 23 papers were presented and the PhD students received feedback and directions for their work. The best paper of the doctoral symposium was included in the conference program.

In-spite of the fact that this was the first time that the conference was organised, it received a high attention from both academia and industry. More than 140 persons attended the conference making it a success and confirming the interest of both the industry and the research community in the conference.

Following this first successful conference and in collaboration with the ATHENA consortium, the I-ESA conference will become an annual event. The second I-ESA conference is scheduled to take place in early 2006 in Bordeaux.

[1] Due to illness of Prof. Tsichritzis his talk was presented by Prof. D. Konstantas

Acknowledgements

The conference chairs and organizers would like to thank all the participants of the INTEROP Network of Excellence and the ATHENA project for their support, encouregement and active participation to the conference.

We also want to acknowledge the support and encouragement received from the European Commission and the Swiss State Secretariat for Education and Research (SER) for their support end encouragement to organize the conference via the INTEROP-ESA NoE grant.

The collaboration and encouragement from IFIP and ACM (SIGAPP) is also highly appreciated.

Special thanks go to the local organization committee of the University of Geneva for their excellent work in setting up the conference and ensuring its smooth operation.

Finally and most important, we want to thank all the authors, reviewers and participants of the conference that made this conference a reality and a success.

Table of Contents

Interop-ESA 2005 Program Chairs

General Conference Co-Chairs

Jean-Paul Bourrières, University of Bordeaux I, France
Michel Léonard, University of Geneva, Switzerland

Program Committee Co-Chairs

Dimitri Konstantas, University of Geneva, Switzerland
Nacer Boudjlida, Université Henri Poincaré Nancy I, France

Workshops, Tutorials & Invited Sessions Co-Chairs

Hervé Panetto, Université Henri Poincaré Nancy I, France
François Charoy, Université Henri Poincaré Nancy I, France

Industrial Track Chair

Martin Zelm, CIMOSA Association, Germany

Doctoral Symposium Chair

Giovanna Di Marzo, University of Geneva, Switzerland

Local Organization Chairs

Jolita Ralyté, University of Geneva, Switzerland
Jean-Henri Morin, University of Geneva, Switzerland

Local Organization Commitee

Nicolas Arni-Bloch, University of Geneva, Switzerland
Mehdi Snene, University of Geneva, Switzerland
Michel Pawlak, University of Geneva, Switzerland
Slim Turki, University of Geneva, Switzerland
Abdelaziz Khadraoui, University of Geneva, Switzerland
Jorge Pardellas, University of Geneva, Switzerland
Marie-France Culebras, University of Geneva, Switzerland

Program Committee

Benali, Khalid, University of Nancy 2, France
Berio, Giuseppe, University of Torino, Italy
Berre, Arne, J., SINTEF, Norway
Berztiss, Alfs, University of Pittsburgh, United States of America
Brinkkemper, Sjaak, Utrecht University, The Netherlands
Carvalho, Joao Alvaro, University of Minho, Portugal
Castano, Silvana, University of Milano, Italy
Cellary, Wojciech, Poznan University of Economics, Poland
Chen, David, University of Bordeaux I, France
De Antonellis, Valeria, University of Brescia, Italy
Dietz, Jan L.G., Delft University of Technology, The Netherlands
Di-Marzo-Seregundo, Giovana, University of Geneva, Switzerland
Doumeingts, Guy, Graisoft, France
Ducq, Yves, University of Bordeaux I, France
Eder, Johann, University of Klagenfurt, Austria
Finkelstein, Anthony, University College London, United Kingdom
Fox, Mark, University of Toronto, Canada
Goossenaerts, Jan, Eindhoven Univ. of Tech., The Netherlands
Gruhn, Volker, University of Leipzig, Germany
Ippolitto, Massimo, Fiat Research Centre, Italy
Jaekel, Frank-Walter, IPK, Germany
Krogstie, John, Sintef, Norway
Kusiak, Andrew, University of Iowa, United States of America
Kutvonen, Lea, University of Helsinki, Finland
Lillehagen, Frank, Computas, Norway
Liu, Kecheng, University of Reading, United Kingdom
Mertins, Kai, Technical University of Berlin, Germany
Missikoff, Michele, CNR/IASI, Italy
Molina, Arturo, Technologico de Monterrey, Mexico
Morin, Jean-Henry, University of Geneva, Switzerland
Müller, Jörg, SIEMENS, Germany
Panetto, Hervé, UHP Nancy I, France
Pernici, Barbara, Politecnico di Milano, Italy
Petit, Michaël, University of Namur, Belgium
Pigneur, Yves, University of Lausanne, Switzerland
Poler, Raul, University Politecnica Valencia, Spain
Ralyte, Jolita, University of Geneva, Switzerland
Rolland, Colette, University of Paris I, France
Ruggaber, Rainer, SAP, Germany
Söderström, Eva, University of Skövde, Sweden
Stamper, Ronald, Staffordshire University, United Kingdom
Stevens, Richard, Gruppo Formula Spa, Italy
Vernadat, François, EC Eurostat, Luxembourg
Wangler, Benkt, University of Skoevde, Sweden
Wasserman, Tony, Software Methods and Tools, United States of America
Zelm, Martin, CIMOSA Association, Germany

External Reviewers

Antonia Albani, University of Augsburg, Austria
Aasa Dahlstedt, University of Skoevde, Sweden

Development of a Metamodel to Foster Interoperability along the Product Lifecycle Traceability

Sergio Terzi[1], Jacopo Cassina[2] and Hervé Panetto[3]

[1] University or Bergamo, Department of Industrial Engineering, Viale Marconi 5, Dalmine (BG), sergio.terzi@unibg.it

[2] Politecnico di Milano, Department of Economics, Management and Industrial Engineering, Piazza Leonardo da Vinci 32, Milano, Italy, jacopo.cassina@polimi.it

[3] Centre de Recherche en Automatique de Nancy, CRAN CNRS UMR 7039, BP239 - F54506 Vandoeuvre-les-Nancy, France, Herve.Panetto@cran.uhp-nancy.fr

Summary. The paper summarizes the effort spend by the authors in developing a model enabling the interoperability of systems for managing the product traceability along the entire product lifecycle. The research effort adopts an holonic definition of the traceability problem and, analysing the current most important standards in product representation, proposes a reference data model.

1 Introduction

Within the actual competitive world, enterprises are ever more stressed and subjected to high market requests. Customers are becoming more and more pretentious in terms of products quality and related services. The best product, at the low price, at the right time and into the right place are the success-keys for the modern enterprise. In order to maintain or gain competitive advantages, modern enterprise has to manage itself along two main directions:

- Improve internal and external efficiency, reducing all the not-relevant costs;
- Improve innovation: innovation of product, process, structure, organization.

According to these needs, enterprises have to focus on their core-competences in order to improve the gained efficiencies (managing innovation) and to reduce the inefficiencies. Looking to this research, the product is re-becoming, after the soap-bubble of the new-economy experience, the real enterprise value creator and the whole production process is re-discovering its role. By this way, within the globally scaled scenario, product and production management have been become complex processes where more problems are overlapping each others. Product development might ever more take into account improved customers' tastes and requests in a shorter time-to-market. The related engineering activities are consequently stressed, while inefficiencies in the production and distribution functions not are ever tolerated.

This way, the product and production lifecycle and its related management are be-coming unavoidable key aspects, creating such a "product centric" or "product-driven" problem. The integrated management of all the information regarding the product and its production is one of the related questions.

1.1 Product Lifecycle Management

One of the answers to these questions is already on going and could be advocated as a new emerging paradigm, defined as Product Life Cycle Management (PLM). In fact, listening to the enterprise questions, diverse vendors, coming from the diverse worlds interested into the product and production management, are more and more providing answers, stabling a growing "PLM market". Looking to this market, it is clear as a pletora of "solution-providers" aims to be considered:

- Vendors coming from the digital engineering world (UGS PLM Solutions, Tecnomatix, IBM-Dassault), which start from PD (Product Development) and MSE (Manufacturing System Engineering) processes and are trying to connect Enterprise Engineering and Management processes;
- Vendors coming from the ERP world (Baan, SAP, Oracle), which, at the contrary, start from Enterprise Management processes for turning to connect PD/MSE tools and platforms;
- Vendors coming from the ICT world, dealing with architecture and platforms, which aim to establish such collaborative environments for PLM integration (Microsoft, MatrixOne, Agile), basically using web technologies.

It is important to notice that the needed product and production management is intrinsically related to the management of the information, so it is obvious that the related emerging market is ICT characterized. Nevertheless, PLM is not primary an ICT problem, but at first, is a strategic business orientation of the enterprise ([1],[2]).

From a strategic organization point of view, the adoption of a product and production centric approach signifies a (re-)modeling of the all relations established between the resources (people and equipment) involved into the relevant business processes oriented to a "product" lifecycle direction, with all that it concerns in terms of task allocations and measurement of the obtainable performances.

From an ICT point of view, a centric product and production management is no more than a "database" problem, which physically enables the previous business process modelling. Information about products and processes are dispersed along a variety of information systems, which - until now - have been executed no more than "isolated islands" (e.g. PDM and ERP). The trends and issues currently on going deal with the integration of these "islands" into a larger integrated (even if distributed) repository, in order to provide a wider and more effective use of product and production information.

From a structural point of view, the instantiation of a product and production centric management approach signifies the product centric design and management of several elements:

- An information infrastructure, which concerns with ICT network establishment;
- A resource infrastructure, which concerns with the design and the management of all physical elements involved along a product and production lifecycle (e.g. machines, plants, people, suppliers, warehouses…);
- A product itself "infrastructure" where the same product becomes a resource to be managed and traced into its same lifecycle.

1.2 Interoperability dimensions in the Product Lifecycle Management

The main problem of this "PLM world" is the interoperability dimension: interoperability in terms of product data at ICT levels, but also interoperability in terms of persons, resources and systems that might cooperate and be in contact. The dimension of interoperability with a product-centric point-of-view becomes the problem of managing the tracing of the product along its lifecycle, or in other words, a problem of traceability.

This paper aims to discuss the problem of traceability as a problem of interoperability. In particular the paper aims to present the development of a reference metamodel for product traceability in a PLM scenario. The paper shows the preliminary results of a common effort, conducted by the authors in order to provide an effective and reliable model to trace product and foster interoperability in the whole product lifecycle.

The paper is organized as follows:

- The second paragraph presents the definition of traceability in the lifecycle and introduces an innovative point-of-view (based on holons) as a solution for product traceability and PLM interoperability.
- The third paragraph illustrates the development of the proposed metamodel, realized on a reference model of product lifecycle considering the requirements of a potential user of this model, and taking into account the current status of interoperability in PLM, searched in Enterprise Standards.
- The fourth paragraph l concludes the paper.

2 Product Traceability

The term traceability related to the product or manufacturing has been defined since the 90ies [3], when a series of industrial needs had been highlighted into the establishment of ISO 9000 procedures. Generally, product traceability is the ability of a user (manufacturer, supplier, vendor…) to trace a product through its processing procedures, in a forward and/or backward direction [4]. Physically, product traceability deals with maintaining records of all materials and parts along a defined life-cycle (e.g. from raw material purchasing to finished goods selling) using a coding identification.

Traceability systems are adopted, according to laws, in the food sector, in manufacturing, in the pharmaceutical sector, in distribution, in constructions. Traceability systems can be useful to increase quality and safety of products, for brand protection, and in order to increase efficiency in production and distribution.

Traceability has different meaning in literature: Internal Traceability, which is the traceability inside the factory and the production system, and External, which follows the product into its relationships with customers, maintainers, suppliers, etc. [3]. Another meaning is Backward and Forward Traceability [4]. Backward Traceability records information and data on the past history of the product. Forward traceability explains what will happen to a certain product, in terms of operations and processes; these information are written before the production begins. This kind of traceability could be very useful in automated manufacturing systems [5].

2.1 Towards a Holonic Traceability

The product traceability problem concerns with the identification of a product, even if it is often considered only the type of product, using a coding system (e.g. bar code, EPC code [5]). All the information related to the coded "product" are then stored into one (or more) databases. Then, a merging activity between the product and its information is a mandatory step, also in the most advanced issues (e.g. Auto-Id efforts in [5], or Dialog effort in [7]). This re-merging activity is still not risk-free; even if it could be conducted in an automated manner (e.g. [5], [6]); transactions breakdowns could occur [7] in searching for information into the database. In general, two main problems could be advocated:

- Accessibility. Database could be off-line or unavailable for a short or long period.
- Timing and Costing. Database could become very large and expensive, thus reducing efficient reading time.

A solving attitude could be identified in the concept partly illustrated in [8], where a simple 2D bar-code attached to physical elements had been adopted to translate high-density information (whole plant drawings) from the plant designer to the contractor. Taking into account this example, each product could be provided with an advanced "product information store system" (e.g. RFID based), in order to be (i) from one side tracked into a system (e.g. a plant) and, from another side, (ii) to be able to provide itself the needed information.

In such a vision, the product itself become the medium of the data set, instantiating a kind of "intelligent product" ([7], [9]), able to interoperate in the environment, exchanging information (which is into the product itself) in real-time with different resources (e.g. machines and transporters in a plant, or trucks and inventory databases in a warehouse, or with refrigerators and dishwashers at home…).

Looking to the literature, the paradigm of "product + information" had been already developed and it is defined as "holonic worldview". The word Holon was introduced by Koestler in 1967 [10], as a combination of the Greek Holos (whole) with the suffix –on, which (as in proton and neutron) suggests a particle or individual part. In the 1993, the holonic term was applied to the manufacturing world, creating the Holonic Manufacturing System (HMS) community. For this community a Holon "is an autonomous and co-operative building block of a system for transforming, transporting, storing and/or validating information and physical objects. The Holon consists of an information processing part and often a physical processing part. A Holon can be part of another Holon." [11].

A holonic-based product traceability could be a killer application in the PLM context. Lots of improvements could be gained establishing an intelligent product, sensibly reducing inefficiency in different processes, from manufacturing, to distribution, after sales, quality assessment, till product accounting and selling.

3 Development of a metamodel for product lifecycle traceability

Looking to the Holonic Product Traceability research effort and thinking to the future, in some years a "product holon" could be inserted in more systems (e.g. a plant, a supply chain, a warehouse) where it will have to exchange information with different "resource holons" ([5], [9]). Hence, the problem of information exchange could easily arise and further standardization efforts will be needed, so establishing a kind of barrier to the diffusion of the same holonic traceability.

In order to reduce these further barriers, and in order to improve the currently definition and the study of Holonic Product Traceability, a research effort could be spent since now, looking to the actual situation of enterprise information systems (where product information are resident) and trying to elaborate it in an holonic view, creating a conceptual "HMS product-oriented" architecture.

The current situation of the enterprise information systems could be registered in the analysis of the current accepted standard, which are specifically created for the integration of ICT systems. The analysis of standards is a basic step that could reduce the research effort, avoiding a long state of the art analysis of enterprise ICT systems.

3.1 Product Lifecycle phases

Traceability is an ability needed along the whole product lifecycle. In order to develop a wider reference model, a definition of lifecycle is needed. In literature there are many different "life cycle" models. Trying to merge diverse kinds of product lifecycle models, it is possible to identify the following Product Lifecycle Model, which will be considered in the work. This model standardizes a product lifecycle composed by four different phases:

- Product Development: it deals with the developing phase of the product, starting from product design and ending, through process and plant design.
- Product Production: it comprises both production and distribution activities. Production phase may be very complex and often includes pre-production and prototyping, manufacturing, assembling, finishing, testing,

packaging, etc. Distribution, on the other side, is related with product storage and delivery.

- Product Use: this is the proper product life phase and it represents all activities which take place during product use: they comprise product usage and consumption, maintenance and support.
- Product Dismiss: in this last phase the product is destroyed, or rather disassembled and recycled.

3.2 Analysis of Enterprise Standards

One of the basic requirements for the development of a product traceability model was that of achieving an easy understanding of such a model by people or organizations of different countries, languages and cultures. This suggests avoiding misunderstanding of concepts, ideas or definitions making use, whenever possible, of shared standards. The authors had conducted a literature survey on standards for manufacturing control, product design, and product support. It is a matter of fact that there are many standards, each of one focused on a specific area of the product lifecycle, but none including all the information needed in the whole lifecycle chain, as shown in the next Figure.

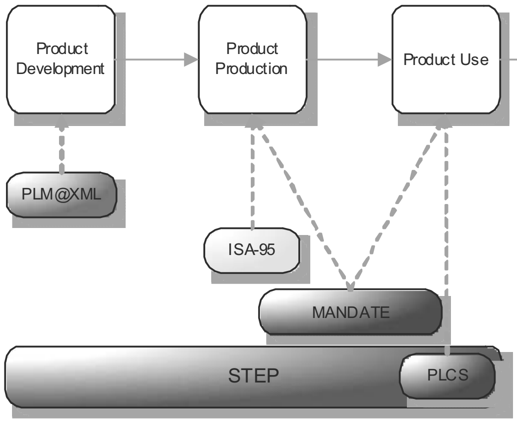


Figure 1. Standards through Life Cycle Phases

Four standard initiatives seem interesting to be studied because they are complementary in the PLM. They are: ANSI/ISA-95 (now ISO/IEC 62264), Mandate (ISO 15531), PLCS (ISO 10303-239), and PLM@XML.

These standards share in common some properties and features, but they are also distinguished by a lot of remarkable differences. First of all, they were designed by different organizations, with different scopes and for different targets.

STEP, PLCS and Mandate can at first sight be grouped together, because each of them is an ISO standard. STEP is an industry standard for product data representation and it's composed of several parts (application protocols) whose aim is to focus on a specific industrial context. There are application protocols for product design, for mechanical and electrical engineering, for sheet-metal manufacturing, for product assembly, for automotive industry and so on. Furthermore, PLCS is an application protocol of STEP. PLM@XML is an open standard developed mainly by EDS (now UGS) and it deals with the product design phase.

ISA-95 is an ANSI standard, but its first part, ANSI/ISA-95.00.01, is also an ISO standard (ISO 62264-1). ANSI/ISA-95 Parts I, II, and III describe the interfaces and activities between an enterprise's business systems and its manufacturing control systems: it focuses, thus, mainly on the area corresponding to product production phase.

Another interesting initiative is the Physical Markup Language (PML), under development at Auto-ID laboratories [5]. The objective of PML is a simple, general language for describing physical objects for use in remote monitoring and control of the physical environment. PLM was thought as a part of a wider structure, built around four major components: electronic tags, Electronic Product Code (EPC), Physical Mark-up Language (PML) and Object Naming Service (ONS).

3.3 Definition of the requirements of the model

In order to develop a comprehensive model, some User Requirements and Main Requirements had been developed analyzing the literature; the first are needs explicitly explained in literature by traceability users, the second are implicit in literature and they are useful for every possible user.

User Requirements are more specific for each single industrial sector, for example specific requirements for pharmaceutical industries exist, others for distribution etc.

Main Requirements are needs implicitly written in literature and not clearly declared. Sometimes, in literature, authors are concerned with some matters of product traceability, especially when referring to a specific context of application. Product traceability is usually dealt by people with different cultural backgrounds, just because traceability is required for agricultural uses as well as for manufacturing industry, or for software development.

The defined Main Requirements are:

- Product Descriptive Power: The model should be able to describe different products.
- Multi-Scenario Descriptive Power: In literature there are many scenarios and many mono-scenario models, but a multi-scenario model is missing [12]. The model has not to fit a special scenario, but it should fit different contexts without modifications.
- Product Lifecycle Scalability: The model should describe different phases of lifecycle.
- Product Detail Scalability: The model should describe different detail levels, comprising final product as well as subcomponents.
- Updatable: The model has to follow the evolution of the product, and keep tracks of information, describing modifications and operations made on it. It shoul include information and data necessary for forward traceability and for recording the product history (backward traceability).

- Shareable: The information should be shared between many users and industries.
- Unambiguously understandable: Many users of different cultures and languages have to access information; the model should avoid misunderstandings.
- Trusted access: To grant information restraint to different kinds of users.
- Being distributable: The information should be stored in different supports (RF tags, barcodes, and databases). Due to technological reasons, for example the amount of free memory on an RF tag for storing information, it could be sometime impossible to keep the whole description of product lifecycle together with the physical product itself; so the information has to be divided on the product and on a remote storage.

3.4 The Holonic Traceability Model

Taking into account the previously presented requirements and the Holon concept defined in [13] (Figure 2), the model for Holonic Product Traceability is hereafter defined. Figure 2 explains that a Holon results from the linking of a Physical Object and some information. There are many information related to the product and they are defined by the ObjectInformation class. If the link between the physical object and its related information is missing, the idea of Holon vanishes and the traceability model miss its target. The link can be established using many technologies (e.g. barcodes and databases, or RFID tags), but this technological implementation is out of the aim of this work, which is more concentrated on the information needed to ensure traceability.

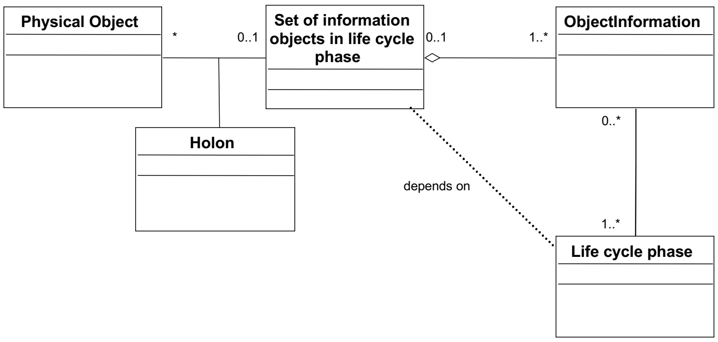


Figure 2. Definition of Holon [13]

ObjectInformation is a group of information that can summarize all the life of the product; it can follow the product during its lifecycle phases. ObjectInformation class (Figure 3) contains general information on the product as the identification, class and batch information, a description of the product, its characteristics and the results of possible tests made on it. The main elements of this class were derived from ISA/ANSI 95 and ISO 10303-239.

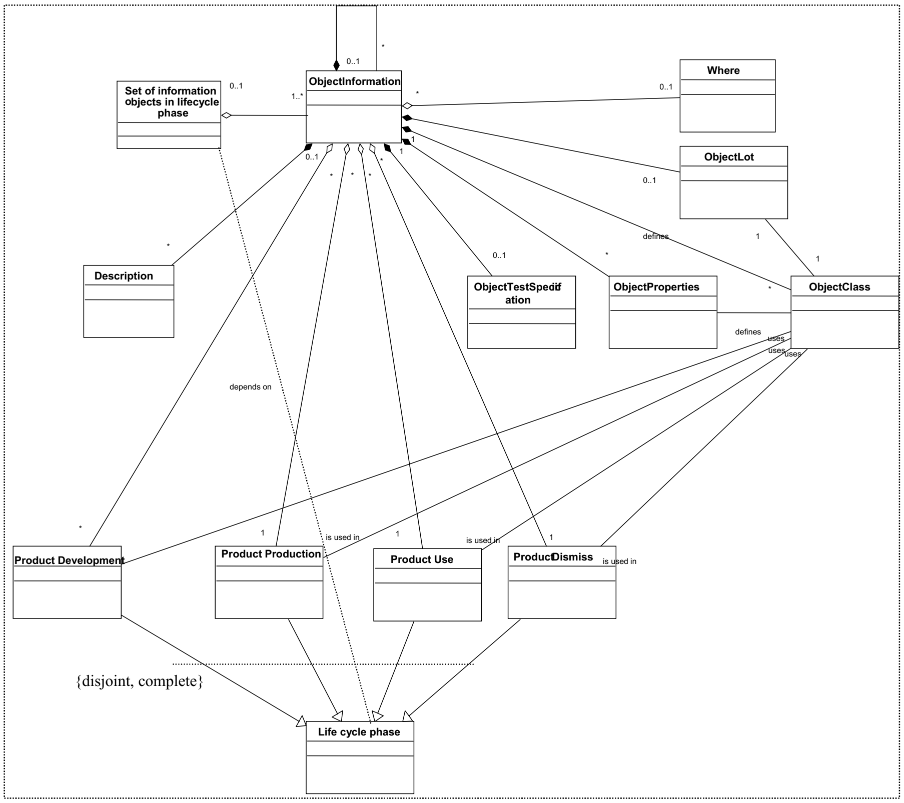


Figure 3. The Object Information schema

It also records information about the lifecycle of the product contained into four specialization of the Life Cycle Phase class (Figure 4). This class describes a generic phase of the lifecycle, using the Event, Activity and Resource classes. An Event causes an Activity that uses Resources. The model can be used both for backward and forward traceability; for the backward, it uses the EventAsOccurs class, which records how and when an event occurs, and the ActivityAsRealized class, which describes how an activity has been fulfilled. It is also possible to implement a forward traceability using the EventAsPlanned class, which describes how and when an event has to occur, and the ActivityAsPlanned, which explains how to do it. When the event and the activity really occur they are recorded in the model as backward traceability, exploiting the EventAsOccurs and the ActivityAsRealized classes.

The model has a fractal structure. It is used recursively to describe the components of the tracked product.

4 Conclusions

With this work it has been proposed a innovative vision that is the Holonic approach, for the traceability as well as the management of lifecycle data. This innovative approach aims to foster interoperability along the diverse enterprise applications, in particular at a manufacturing stage.

We propose a meta-model supporting the informational part for the traceability of products (or Holons-products), along its life cycle. This model was established, re-using, at best, existing work around some standards: PLCS, Mandate, ANSI/ISA-95, PLM/XML and PML-EPC.

The model is technology independent and fits to different application domains. The model can trace the history of the product in details, or can be less detailed, but more compact such a way all the information can be written on a 2D barcode. The model can also use a combination of barcode and remote databases; on the barcode could be written the most frequently used information; the remote database could record less exploited informa-tion. The model could also be implemented on RFID tags that could contain all the necessary information.

The ongoing research particularly relates to the methodological aspects regarding the modeling process of enterprise and manufacturing processes, by taking into account, a priori, the products traceability objectives during the design stage and the integration phase of the productions systems, or when reengineering the existing ones.

Also data management has to be deepened, especially the security and the privacy which is an mandatory for consumers.

Finally, cost estimation on the implementation and the management of traceability systems is needed.

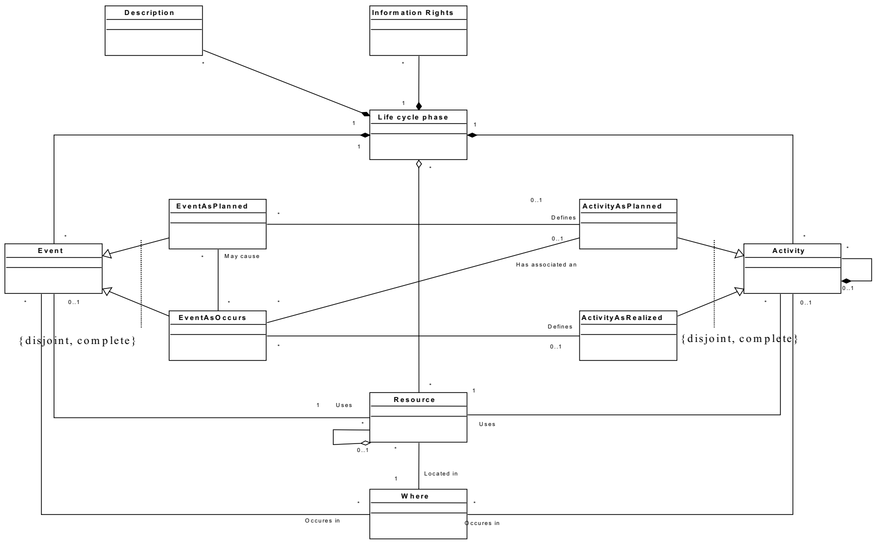


Figure 4. The Life Cycle Phase class

References

1. CIMData, www.cimdata.com
2. Garetti M. Terzi S., (2004), Product Lifecycle Management: definition, trends and open issues, Proceedings at III International Conference On Advances In Production Engineering, 17 - 19 June, Warsaw, Poland
3. Cheng M.L. and Simmons J. E. L., (1994), Traceability in manufacturing systems. Interna-tional Journal of Operations and Production Management, 14, 4-16
4. Jansen-Vullers J., van Dorp A., Beulens B., (2003), Managing traceability information in manufacture, International Journal Of Information Management 23 : 395-413
5. McFarlane D., Sarma J., Chirn G., Wong J., Ashton A., (2003), Auto-ID systrems and intel-ligent manufacturing control, Journal of Engineering Applications of Artificial Intelligence, 16, 365 – 376
6. McFarlane D., Bussmann D., (2000), Developments in Holonic Production Planning and Control, Int. Journal of Production Planning and Control, 11, 6, 522 - 536
7. Kärkkäinen J., Holmström G., Främling J., Artto G., (2003), Intelligent products – A step towards a more effective project delivery chain, Computers in Industry, 50, 141-151
8. Finch F., Flanagan M., Marsh M., (1996), Electronic document management in construction using auto-ID, Automation in Construction, 5, 313-321
9. Morel G., Panetto H., Zaremba A., Mayer G., (2004), Manufacturing enterprise control and management system engineering rationales and open issues, IFAC Annual Reviews in Control, 27, 199-209
10. Koestler A., (1967), The Ghost in the Machine, Arkana Books, London
11. Seidel D., Mey M., (1994), IMS – Holonic Manufacturing Systems: systems components of autonomous models and their distributed control – Vol 0. Also known as The Book of HMS, TC5 project
12. Ramesh B., Stubbs C., Powers T., Edwards M., (1997), Requirements traceability: Theory and practice, Annals of software engineering
13. Gouyon D., Simão J. M., Alkassem K., Morel G., (2004), Work in progress for product driven manufacturing automation, 2004, Proceedings of IFAC INCOM Symposium, April 7th-9th, Salvador de Bahia, Brasil

Business Process Requirements, Modeling Technique and Standard: how to Identify Interoperability Gaps on a Process Level

Boriana D. Rukanova, Kees van Slooten and Robert A. Stegwee

University of Twente, BBT, P.O. Box 217, 7500 AE Enschede, The Netherlands
{b.d.rukanova, c.vanslooten, r.a.stegwee}@utwente.nl

Summary. In a business-to-business setting there is a growing need for organizations to be interoperable. Computer systems involved in automating parts of a business process need to become an integral part of the business process. Before automation is achieved, the part of the business process to be automated needs to be made explicit and then operationalized. Business process models could be used to make the process explicit. Domain standards could be used to make the process operational. However, there is no method available to evaluate the fit between business process requirements, a modeling technique and a standard. In this paper we argue that a set of business process concepts can be useful for the evaluation of the fit. We further propose a method how to perform the evaluation and we illustrate the use of the method. The construction of a complete set of business process concepts remains out of the scope of this paper.

1 Introduction

In business-to-business settings, there is a growing need for different business organizations to work together, in order to achieve the desired business goal. Computer systems have a great potential to automate parts of the business processes. In that respect, the notion of interoperability becomes of great importance. According to the IEEE definition, interoperability is defined as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged”. However, it has been argued that to achieve interoperability between organizations, the implicit notion of a system as a computer-based information systems, needs to be expanded to an organizational system, which is a socio-technical system [16].

Following the ideas from organizational semiotics, it has been argued that to be able to work together, the different organizations need to establish a shared context [15]. When two or more computer systems are involved in automating parts of a business process, they need to be properly embedded into this shared context. If we want to use computer systems to support part of a business process, this requires the disparate information systems to be able to express, and especially interpret a broader range of meanings. Thus, the applications that support the business processes, not only have to be able to express what needs to be communicated, but also to interpret it, and act upon this interpretation in an intelligent manner. Thus the shared context of the business process needs to be captured and made operational. One possibility to make this context operational and embed it in the system is by using domain standards [11].

Although EDI standards promised significant advantages in facilitating the exchange between business partners, reducing errors, increasing speed, cutting cost, and building in competitive advantage [2, 6, 7, 10, 14, 18], EDI standards failed to fully capture the requirements of the shared context. EDI standards were lacking clear and complete lexicon, did not have fully specified grammar, and had nearly no semantics. Furthermore, the focus of many IS professionals on EDI was how to provide technical tools, rather than to support the way people do business [1, 5, 8]. New standards, which strive to allow for interoperability between disparate systems, are currently developed. These standards try to capture elements of the business process context, in order to allow for meaningful communication (for example, in the healthcare domain such a standard is the HL7 standard for clinical data interchange).

Standard development organizations or consortia of companies develop such standards, based on their specific interpretation of the domain. However, in order to have value for a particular business process, a standard needs to be linked to a particular situation, which might be different from what the developers had in mind. Thus, the standard needs to be evaluated (for a specific business process) whether it can cover the context of the particular business process, which needs to be embedded in the system. Failure to cover parts of this context can lead to inter-organizational interoperability problems.

The context of a business process might be to a large extent implicit. Thus, before making it operational (by using standards), it first needs to be made explicit. Models could be used to capture and to make this context explicit. However commonly accepted way of modeling does not exist [17] and different modeling techniques can capture different elements of the context. In that respect, how to make sure that the chosen modeling technique is capable to fully capture the requirements of the business process context.

From the description above, a few issues arise. First, how to evaluate whether a modeling technique has the capability to cover different elements of the context of the business process? Second, how to evaluate to what extent a standard can capture these elements? If we refer to the elements of the context of a business process, which need to be embedded in the system, as business process requirements, we can rephrase the issues described above as follows: How to evaluate the fit between business process requirements, a modeling technique, and a standard?

The rest of the paper is structured as follows. In part two, we discuss the possibility to use a set of business process concepts to evaluate the fit. In part three we propose a method how to use a set of business process concepts to evaluate the fit between business process requirements, a modeling technique and a standard. In part five, we illustrate the use of the method with an example. We end the paper with conclusions and directions for further research.

2 Business Process Concepts as a Prerequisite

The question that we would like to address in this paper is how to identify the fit between business process requirements, a modeling technique and a standard. To evaluate conceptual modeling languages, Wand and Weber [17] use the Bunge-Wand-Weber ontology. Another group of authors [13] proposes a meta model of business process concepts that can be used to evaluate business process modeling languages. In both cases described above, the idea is that one can perform the evaluation of conceptual modeling languages based on a set of reference concepts. Using a similar idea we can say that to be able to identify the fit between business process requirements, a modeling technique and a standard, one can identify a set of reference concepts, which can help us to do this evaluation. As the business process requirement, the modeling technique and the standard, refer in some way to business processes, if one is able to identify a reference set of business process concepts, these concepts can use them to make this comparison (see Figure 1 below).

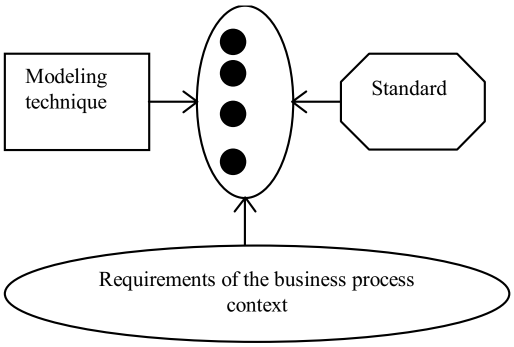


Figure 1. Evaluation of the fit using a reference set of business process concepts

There have been several attempts to identify business process concepts. Lin et al. [9] made an extensive study of a number of methods, tools and techniques and they came to a list of concepts, which they link into a model. The concepts however are not well defined and the logic for grouping these concepts remain implicit. Another study done by Söderström et al. [13] also aims at identifying business process concepts. The authors present a meta model of the business process concepts. The problem is that the resulting set of concepts is not formally defined, which can lead to misinterpretations and ambiguities if one tries to apply them. Further, it is often not clear how this set of concepts is derived and how new concepts can be added.

An alternative way to proceed is to take an existing ontology, like for example the Bunge-Wand-Weber or the FRISCO ontology [3]. The problem is that these ontologies include other concepts apart from the business process concepts.

Rukanova et al. [12] argue that to arrive at a coherent set of business process concepts, a systematic approach needs to be followed. The authors propose a method describing the process of how to arrive at a set of business process concepts. Although Rukanova et al. [12] propose a method on how to arrive at a set of business process concepts, they only illustrate the method and propose a preliminary set of business process concepts, however they do not provide a complete set of business process concepts.

The construction of a complete set of business process concepts is a major task and will require substantial research efforts to perform multiple iterations, to come to a stable set of business process concepts. We will leave the complete elaboration of the business process concepts for further research. However, what we would like to address in this paper is how such a set of business process concepts can be used to evaluate the fit between business process requirements, a modeling technique and a standard.

3 A Method for Comparison

Within this paper we propose a method to compare business process requirements, a modeling technique and a standard. The method consists of several major steps. The Figure below provides a schematic representation of the method.

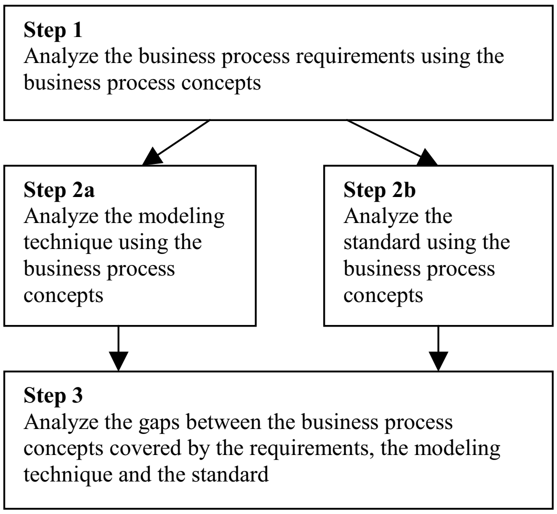

Step 1
Analyze the business process requirements using the business process concepts
Step 2a
Analyze the modeling technique using the business process concepts
Step 2b
Analyze the standard using the business process concepts
Step 3
Analyze the gaps between the business process concepts covered by the requirements, the modeling technique and the standard

Figure 2. Schematic representation of the method for comparison of business process requirements, a modelling technique, and a standard

Each of these steps will be briefly discussed below.

The first step is the analysis of the business process requirements using the set of business process concepts. It is important to notice is that the business process requirements will always be the starting point, and what we need to evaluate is to what extent the business process requirements can be captured by the modeling technique or by the standard. The question that can lead the analysis during step 1 is: “Which of the business process concepts are covered in the requirements?” As a result, only those business process concepts that are relevant for the specific business process situation will be selected. We will call these concepts “business process concepts covered by the requirements”.

Once we have performed the analysis of the business process requirements, we can proceed with the analysis of the chosen modeling technique (step 2a) and the analysis of the chosen standard (step 2b). The questions that can be used to guide the analysis are: “Which of the business process concepts are covered by the chosen modeling technique?” and “Which of the business process concepts are covered by the chosen standard?” respectively.

After performing step 1, step 2a and step 2b, we can proceed to step 3, where the actual comparison between the business process requirements, the modeling technique and the standard can be made. The questions that can guide the analysis at this step can be: “Which of the business process concepts, which are covered by the requirements are not covered by the modeling technique”. By answering this question we will be able to identify problems that can occur due the choice of a modeling technique. A mismatch between the two would mean that certain business process requirements could not be captured by the chosen modeling technique. With respect to the standard, one can ask the question: “Which of the business process concepts, which are covered by the requirements are not covered by the chosen standard?”. By answering this question we will be able to identify problems that can occur due the choice of standard. A mismatch between the two would mean that certain business process requirements could not be captured by the chosen standard. In that way in step three one can identify mismatches between the business process requirements and the modeling technique and the business process requirements and the standard respectively. Each of these mismatches can be further analyzed and if necessary additional measures can be taken to fill these gaps. The handling of mismatches however will remain outside of the scope of this paper. In the next section we illustrate the use of the method.

4 Example

Within this section we will illustrate the use of the method for comparison between business process requirements, a modeling technique and a standard. Below we will first introduce a number of business process concepts that we will use, we will provide a brief description of a business situation (performing a test on a patient) and we will illustrate how the business process concepts can be used to analyze it. We will further discuss a modeling technique (process charting) and analyze its capabilities. After that, we will discuss a standard (HL7 v 3.0 standard) and we will evaluate it as well in terms of the business process concepts. Finally we will analyze the fit between the business process requirements, the modeling technique and the standard.

4.1 Business Process Concepts

For the purpose of the example we will use the preliminary set of business process concepts as identified in Rukanova et al. [12]. As mentioned earlier, this set of concepts is not complete, however we consider that it is sufficient for illustrative purpose. To arrive to this preliminary set of concepts, the authors started with a number of business process definitions, extracted key elements from these definitions and mapped them to the FRISCO ontology. The authors also identify a number of concepts, which need to be defined as an extension to the ontology. For the purpose of this example we will select only a basic set of the business process concepts identified. The concepts, which we will use in the example are: composite action, action, pre-state and post-state of an action, actor, input actand and output actand of an action, basic state-transition structures (sequence, choice, parallel), rule. These concepts are schematically represented in the Figure below.

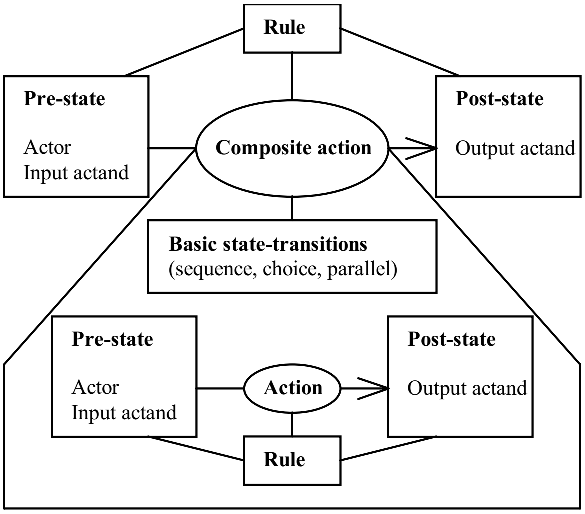


Figure 3. Schematic representation of the FRISCO business process concepts

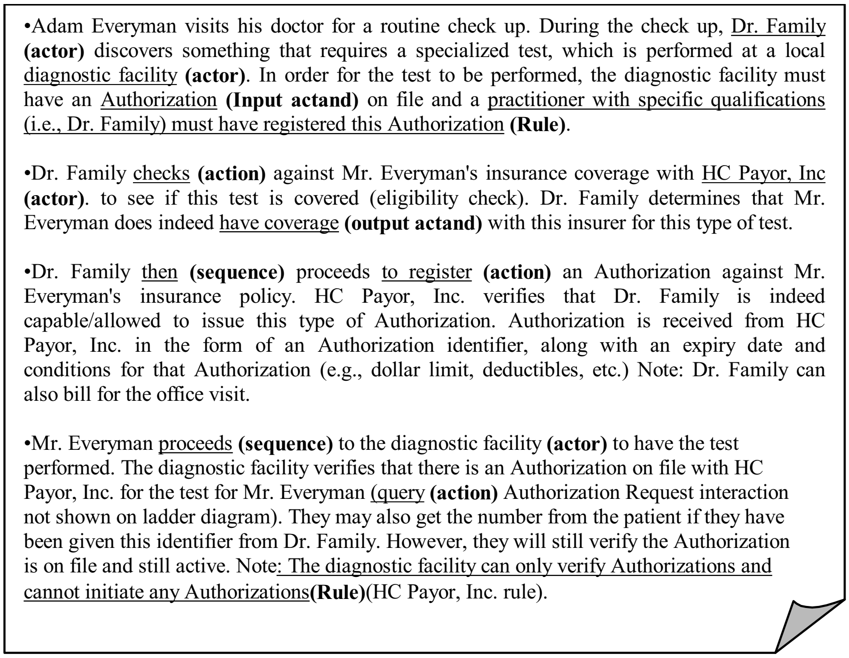
Below these concepts are briefly introduced (the underlined concepts below are FRISCO concepts). For the complete and formal definitions of the concepts, please refer to the FRISCO report itself [3].

A fundamental concept is the concept of action. An action is a transition from a pre-state to a post-state. Actors are a mandatory part of the pre-state of an action and are responsible and capable for causing a transition. If not consumed or destroyed during the transition, they are also part of the post-state. Other things that are part of the pre-state of a action are called input actands. Other things that are part of the pos-state of a action are called output actands. An action can form part of a composite action. Actions can be related via a state-transition structure. Some basic state transition structures are sequence, choice and parallel. The allowed states and transitions are governed by a rule.

Now that we have introduced a number of business process concept and we have briefly discussed them what they mean in terms of FRISCO, we can proceed with the illustration of how to apply them.

4.2 Brief Description of a Business Situation

To illustrate an example of a description of a business situation we will use one of the storyboards that are provided with the HL7 v 3.0 standard. The HL7 standard is a standard for clinical data interchange. In HL7 v 3.0 standard a storyboard describes a type of a business situation. The storyboards are not part of the normative standard, rather they are included in order to provide some background information. As the storyboards provide a rather detailed description, we will use one of them as a description of a business situation (see the Figure below).



•Adam Everyman visits his doctor for a routine check up. During the check up, Dr. Family **(actor)** discovers something that requires a specialized test, which is performed at a local diagnostic facility **(actor)**. In order for the test to be performed, the diagnostic facility must have an Authorization **(Input actand)** on file and a practitioner with specific qualifications (i.e., Dr. Family) must have registered this Authorization **(Rule)**.

•Dr. Family checks **(action)** against Mr. Everyman's insurance coverage with HC Payor, Inc **(actor)**. to see if this test is covered (eligibility check). Dr. Family determines that Mr. Everyman does indeed have coverage **(output actand)** with this insurer for this type of test.

•Dr. Family then **(sequence)** proceeds to register **(action)** an Authorization against Mr. Everyman's insurance policy. HC Payor, Inc. verifies that Dr. Family is indeed capable/allowed to issue this type of Authorization. Authorization is received from HC Payor, Inc. in the form of an Authorization identifier, along with an expiry date and conditions for that Authorization (e.g., dollar limit, deductibles, etc.) Note: Dr. Family can also bill for the office visit.

•Mr. Everyman proceeds **(sequence)** to the diagnostic facility **(actor)** to have the test performed. The diagnostic facility verifies that there is an Authorization on file with HC Payor, Inc. for the test for Mr. Everyman (query **(action)** Authorization Request interaction not shown on ladder diagram). They may also get the number from the patient if they have been given this identifier from Dr. Family. However, they will still verify the Authorization is on file and still active. Note: The diagnostic facility can only verify Authorizations and cannot initiate any Authorizations**(Rule)**(HC Payor, Inc. rule).

Figure 4. Example of a description of a business situation

What is interesting to notice is that the description of the business situation can be provided in lengthy textual documents. By using the system of business process concepts, one can walk through such documents and try to reason which business process concepts are needed to describe it. In the Figure above we have underlined parts of the texts and we have provided some examples to which business process concepts the underlined text refers to (the concepts are put in brackets). The business process concepts that we have identified in the requirements are summarized in column two of Table 1 in section 4.5.

4.3 Modeling Technique

For the purpose of this example we evaluate a modeling technique used in practice called “Process charting”. With the process charting, it is possible to model a business event, a business result, elementary processes, mandatory sequences and optional paths (see Figure 5 below).

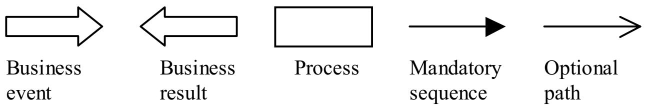


Figure 5. Notation of elements, used in process charting.

A summary of the analysis of the modeling technique can be found in the third column of table 1. Here we will give some examples to illustrate the reasoning in the analysis.

The process charting has elements to model mandatory sequence and optional path. If we look at the business process concepts, we have a concept basic state transition structures, which covers three possible structures, sequence and choice, and parallel. In that respect we can conclude that the process charting covers the basic state-transition structures sequence and choice.

4.4 The Standard

Concerning the chosen standard, we will use HL7 v.3.0. The HL7 standard focuses on defining interactions. In the HL7 v.3 Guide, an interaction is defined as " a unique association between a specific message type (information), a particular trigger event, and the application roles that send and receive a message type. It is a unique, one-way transfer of information." From this definition we can derive, that an interaction is uniquely defined using four components: the sending application role (a system component which sends a message), the receiving application role (a system component, which receives the message), the trigger event (when a certain interaction should occur), and the message type (what message to send). The sending and the receiving application roles are the roles to be fulfilled by the computer systems. Examples of application roles are "eligibility requestor and eligibility manager". Further, the standard describes in detail the content of the message. In some cases, the standard defines also the so called "receiver responsibility". The "receiver responsibility" defines the interaction that the receiving application would need to perform.

We will not provide further elaboration of the standard, but we will go back to the main question, mainly which of the business process concepts are covered by the HL7 standard. A summary of the analysis of the standard can be found in the third column of table 1 and here we will just illustrate the reasoning with respect to several business process concepts. We can say that the standard covers the concept of action, however to a limited extent, as it only addresses communicative actions and not substantive actions (like the actual performance of a test). Further, as it covers only individual communicative actions and in some cases the response to a certain communicative action, it is not capable to cover the concept of a composite action, or how a number of actions are logically related.

4.5 Evaluation of the Fit

The table below is an example of how one can compare the business process requirements, the modeling technique and the standard.

Table 1. Example of a comparison between business process requirements, a modeling technique and a standard

Business process concepts	**Business process requirements**	**Modeling technique**	**Standard**
Composite action	(~)	(X)	(-)
Action	(X)	(X)	(~)
Pre-state			
Actor	(X)	(-)	(~)
Input actands	(X)	(-)	(~)
Post-state			
Output actands	(X)	(-)	(~)
Basic-state transitions			
Sequence	(X)	(X)	(~)
Choice	(X)	(X)	(-)
Parallel	(-)	(-)	(-)
Rule	(X)	(~)	(~)

Table 1 consists of four columns. In the first column, the business process concepts are listed. The list of business process concepts is not complete and is only for illustrative purposes. The second column illustrates which of the business process concepts are covered by the requirements. Column three and four describe which business process concepts are covered by the modeling technique and the standard respectively. Further, we use the following symbols in the table: (X) means that a concept is covered; (~) means that a concept is covered to some extent, and (-) means that a concept is not covered.

We will not discuss the results in the table in detail, however we will discuss several examples. Let us look at the concept actor as part of the pre-state of an action. The concept of actor is covered by the business process requirements. The chosen modeling technique does not cover the concept of actor at all. Within the standard actors are not modeled explicitly, only sending and receiving application roles are described. Further, if we look at the concept input actand of an action, we can see that the requirements specify input actands, the modeling technique is not capable to capture them, and the standard capture only messages as input actands. Further, the requirements clearly cover the concept of a rule. The coverage of the concept of rule, however is limited in both the modeling technique and the standard.

We will not continue further, however we consider that the table illustrates how the business process concepts can be used as a reasoning tool to help compare business process requirements, a modeling technique, and a standard. There is one further remark that we would like to make here. The comparison presented in the table above illustrates how the fit can be identified on a concept level, without looking at the content itself. However, a second level of analysis (content level) is also possible, when using the business process concepts. We will give an examples with the concept of action. The questions that one can ask are: “which are the actions that need to be carried out automatically”, and “Which are the actions that are covered by the standard?”. “Are there actions that need to be automated but are not covered by the standard?” In that way, one can reason about the suitability of the content of the standard with respect to the business process requirements.

5 Conclusions

Within this paper we pointed out the need to evaluate the fit between business process requirements, a modeling technique and a standard. Further we presented a method on how to evaluate the fit and we have illustrated the use of the method with an example. As argued earlier, such an evaluation is important, so that interoperability problems can be identified at an early stage.

Although this paper focuses on describing a method on how to compare the business process requirements, a modeling technique and a standard, a prerequisite for using the method is to have a well-defined system of business process concepts. Within the example presented, a preliminary sample of business process concepts, identified in previous research has been used. However, as a complete system of business process concepts is still missing, the construction of such system can be a subject of further research. A method on how to arrive to such a system has already been developed and can be used as input for the further identification and definition of business process concepts.

References

1. Covington, M.A. (1997). On Designing a Language for Electronic Commerce. International Journal of Electronic Commerce, vol. 1, no 4, Summer 1997, pp. 31–48.
2. Damsgaard J., Truex, D. (2000), Binary Trading Relationships and the Limits of EDI Standards: The Procrustean Bed of Standards. European Journal of Information Systems, vol. 9, no 3, pp. 173-188.
3. Falkenberg, E. D., Hesse, W., Lindgreen, P., Nilsson, B. E., Han Oei, J. L., Rolland, C., Stamper, R.K., Van Assche, F.J.M., Verrijn-Stuart, A. A., Voss, K. (1998), A Framework of Information System Concepts: The FRISCO Report, IFIP (1998), available on- line at: http://www.liacs.nl/~verrynst/frisco.html
4. HL7: Health Level 7 http://www.hl7.org/
5. Huang K. (1998), Organizational Aspects of EDI: a Norm-oriented Approach, PhD thesis, ISBN 90-3651095-3
6. Jelassi, T. and Figon, O. (1994), Competing through EDI at Brun Passot: Achievements in France and Ambitions for the Single European Market, MIS Quarterly, 18(4), pp. 337-352.
7. Kekre, S. and Mukhopadyay, T. (1992), Impact of Electronic Data Interchange Technology on Quality Improvement and Inventory Reduction Programs: A Field Study, International Journal of Production Economics, vol. 28, no. 3, pp. 265-282.
8. Kimbrough S.(1999), EDI, XML, and the Transparency Problem in Electronic Commerce. Available on-line: http://grace.wharton.upenn.edu/~sok/sokpapers/1999-0/indiana-transparency/paper.pdf

9. Lin, F. R., M. C. Yang, et al. (2002). "A Generic Structure for Business Process Modeling." Business Process Management Journal 8(1): 19-41.
10. Mackay, D.R. (1993), The Impact of EDI on the Components Sector of the Australian Automotive Industry, Journal of Strategic Information Systems, vol. 2, no. 3, pp. 243-263.
11. Rukanova, B. D., Slooten, C. van, Stegwee, R.A. (2003).Beyond the Standard Development and the Standard Adoption. In Jakobs, K. (Ed): Proceedings of the 8th EURAS Workshop on Standardization, July 11-12, 2003, Aachen, Germany.Mainz Publishers, ISBN: 3-86073-762-7, pp. 120-138
12. Rukanova, B.D., Aydin, M.N., Slooten, C. van, Stegwee, R.A., (2005). Towards a Meta Model for Business Process Concepts. In: proceedings of ICEIS 2005 (in press)
13. Söderström, E., Andersson, B., Johanesson, P., Perjons, E., & Wangler, B., (2002), Towards a Framework for Comparing Process Modeling Languages", In Pidduck et al. (Eds.), CAISE 2002, LNCS 2348, pp. 600-611, 2002, © Springer- Verlag Berlin Heidelberg, 2002
14. Sokol, P. K. (1995), From EDI to Electronic Comemrce- a Business Initiative, McGraw-Hill, Inc., New York, ISBN 0-07-059512-7.
15. Stamper, R. (1996), Organisational Semiotics. In: Information Systems: An Emerging Discipline, F.Stowell & J. Mingers (eds.), London and New York, McGraw-Hill.
16. Stegwee, R.A., Rukanova, B.D. (2003). Identification of Different Types of Standards for Domain-Specific Interoperability. In: Proceedings of MIS Quarterly Special Issue on Standard Making: A Critical Research Frontier for Information Systems: Pre-Conference Workshop ICIS 2003. pp. 161- 170, available on line at: http://www.si.umich.edu/misq-stds/proceedings/
17. Wand, Y. & Weber, R. (1989), An Ontological Evaluation of System Analysis and Design", In: Falkenberg, E.D. & Lindgreen, P. (Eds.) Information System Concepts: An In-depth analysis, Elsevier Science Publishers B.V. (North-Holand) © IFIP, 1989
18. Wrighly, C.D., Wagenaar, R.W., and Clarke, R. (1994), Electronic Data Interchange in National Trade: Frameworks for the Strategic Analysis of Ocean Port Communities, Journal of Strategic Information Systems, 3(3), pp. 211-234.

An Architecture for Semantic Enterprise Application Integration Standards

Nenad Anicic[1, 2], Nenad Ivezic[1] and Albert Jones[1]

[1] National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD 20899, USA {nanicic, nivezic, ajones}@nist.gov
[2] Faculty of Organization Sciences, 154 Jove Ilica Street, 11000 Belgrade, Serbia and Montenegro, anicic.nenad@fon.bg.ac.yu

Summary. Large, industry-wide interoperability projects use syntax-based standards approaches to accomplish interoperable data exchange among enterprise applications. We are investigating Semantic Web to advance these approaches. In this paper, we describe an architecture for Semantic Enterprise Application Integration Standards as a basis for experimental assessment of the Semantic Web technologies to enhance these standards approaches. The architecture relies on automated translation of the XML Schema-based representation of business document content models into an OWL-based ontology. Based on this architecture, we use Semantic Web representation and reasoning mechanisms to support consistency checking of ontological constructs and constraints specified within the ontology. The proposed architecture is relevant (1) when managing multiple enterprise ontologies derived from, and dependent on, a common ontology and (2) when dealing with model-driven integration using customizable interface models and testing of such integration efforts.

1 Introduction

The scope of the effort reported in this paper is defined partially by the type of industrial problems we identify and partially by the traditional standards usage for enterprise application integration (EAI). Both are discussed below.

1.1 A Prototypical Problem

Two independent but related industry consortia have developed enterprise application integration standards in the form of business document content models. Standards in Automotive Retail (STAR), an automotive retail consortium, has developed XML Schema-based standards to encode business document content models enable message exchanges among automotive manufacturers and their retail houses. Each STAR application adopts and implements the proposed STAR XML-based interface model [1]. Automotive Industry Action Group (AIAG), an automotive supply chain consortium, has developed XML Schema-based standards to encode its own business document content models that enable message exchanges among automotive manufacturers and their suppliers. Each AIAG application adopts and implements the AIAG interface model [2].

Both STAR and AIAG base their interface models on the same 'horizontal' XML document standard – the Open Applications Group (OAG) Business Object Documents (BODs) [3]. The OAG BODs are specifications of general XML Schema components and general aggregations that make up XML Schema-based business document content models from these components. STAR and AIAG independently use the OAG BOD specifications to customize their own business document content models and define rules of usage and constraints. Typically, usage rules and constraints are captured outside of the XML Schema using a syntactic constraint language such as Schematron [6]. A significant manual task is required to identify and reconcile differences among constraints and rules of the two standards [4]. Consequently, major problems can arise whenever a STAR application attempts to exchange automotive parts ordering data with an AIAG application.

In this paper, we describe an approach to enable automated checking of compatibility among rules and constraints that are independently developed within the two (or more) standards that have a common terminology as their bases. Once this approach is implemented, we expect more capable testability of integration efforts and, consequently, more efficient application integration.

1.2 Traditional EAI Standards Architecture

Enterprise application integration (EAI) is being used extensively today. The left portion of Figure 1 shows how a traditional EAI standards architecture could be applied to our STAR-AIAG integration problem assuming, a pure XML Schema-based approach. The following steps are required to translate data and to verify the business document translation:

1. Identify and resolve manually any semantic and syntactic similarities and differences between the interface models.
2. Create two XSLT stylesheet transformations from source to target and vice versa.
3. Based on 2, apply translation to the source XML Schema interface model to obtain a business document conformant to the target XML Schema interface model.
4. Validate translation using equivalence test. This validation may be done by applying an equivalence test between the initial source business document and the final source business document that is obtained through a sequence of two (forward and reverse) translations compatible with XSLT transformations from step 2.

Validation using an equivalence test is not straightforward because issues that require capabilities beyond a simple, syntax-based equivalence checking arise often. Consider the following two examples. First, elements that are ordered differently syntactically may, in fact, be equivalent semantically, if that order is not significant. Second, a time period can be specified either by a start date with an end date or with a start date and a duration. While they are semantically equivalent, they are syntactically quite different.

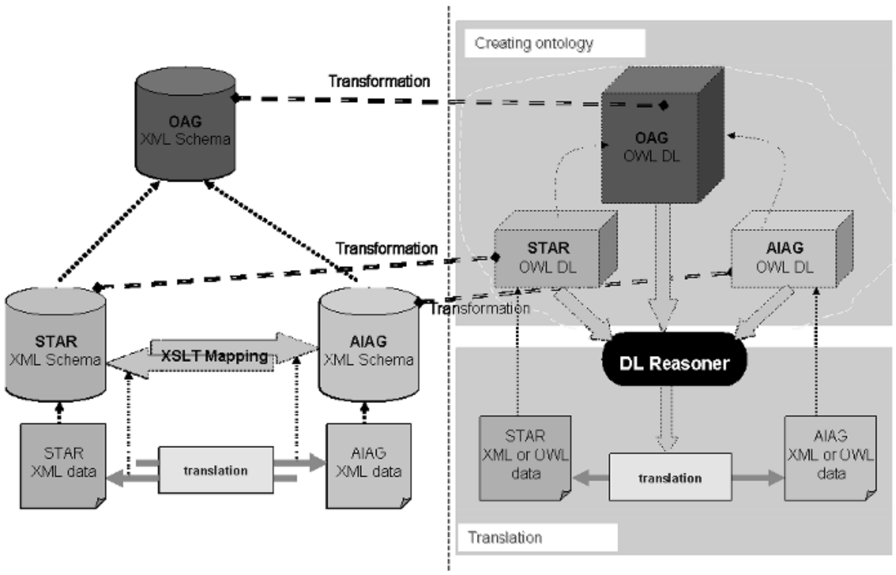


Figure 1. Traditional and Semantic Web-based EAI Standards Architectures.

2 A Semantic Web-Based EAI Standards Architecture

For our approach, which is shown in the right portion of Figure 2, we use the OWL-DL Web ontology language to integrate enterprise applications. The language is based on a subset of the First Order Logic formalism called Description Logics. To do this, we assume that the OAG, STAR, and AIAG business document content models have been rendered into OWL-DL ontologies – a step that will be discussed in detail later in the document. This, in turn, enables us to readily use automated reasoning methods provided by DL reasoners such as Racer [5]. These reasoning methods are fundamental enablers of automated transformations, mapping and translation functions, between OWL-DL interface models that are independently developed but based on a common terminology.

The following steps are envisioned to translate and verify the translations in the proposed architecture. We provide details of executing these steps below.

1. Perform model-based equivalence analysis of STAR and AIAG schemas.
    a. Create ontologies of the common OAG-based general terminology and from respective XML Schemas for STAR and AIAG.
    b. Create a merged ontology from independently developed STAR and AIAG ontologies and check for unsatisfiability.

c. Identify similarity between two schemas based on the comparison of their respective semantic models. (We assume that a high degree of equivalence may be obtained as a result of common usage of core components of the OAG standard.)

2. Apply semantic translation using the merged ontology and an OWL-DL reasoner.
   a. Translate the source (STAR) XML instance to the source (STAR) OWL representation.
   b. Check for consistency and sufficiency w.r.t the merged (source-STAR+target-AIAG) ontology.
   c. Classify the source OWL individual into the target ontology (AIAG) and perform validation and serialization.

3 A Semantic Web-Based Integration Methodology

Figure 2 illustrates the proposed Web-based integration methodology using a scenario-based view of the semantic integration architecture. The top portion shows the ontology-creation activities. These activities, which occur at design time, help us define and test possible interoperable data exchanges. The bottom portion shows translation activities. These activities, which occur at run time, help us reason about concrete transformation from one XML format to another.

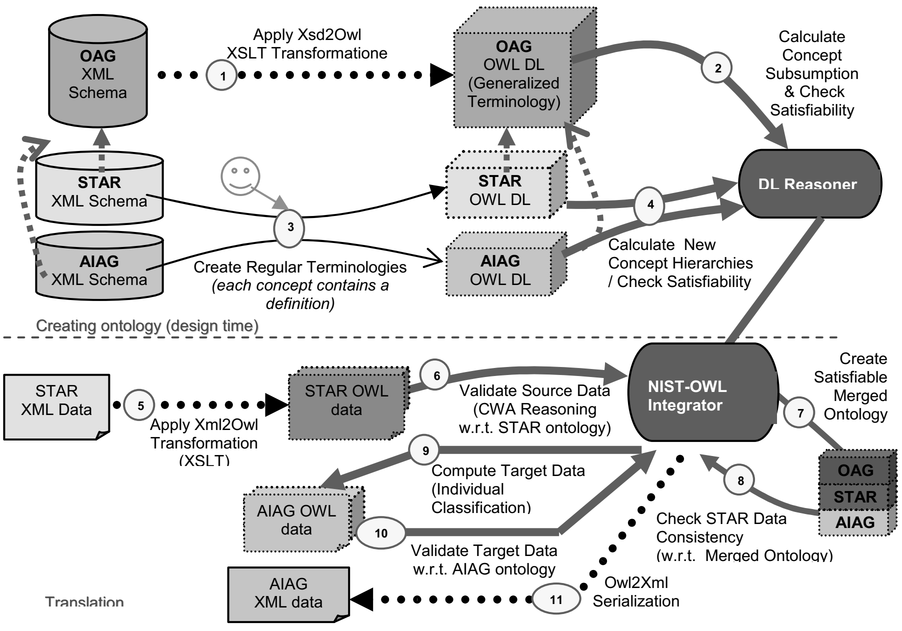

OAG XML Schema
Apply Xsd2Owl XSLT Transformatione
1
OAG OWL DL (Generalized Terminology)
2
Calculate Concept Subsumption & Check Satisfiability
STAR XML Schema
AIAG XML Schema
3
Create Regular Terminologies (each concept contains a definition)
STAR OWL DL
AIAG OWL DL
4
DL Reasoner
Calculate New Concept Hierarchies / Check Satisfiability
Creating ontology (design time)
STAR XML Data
5
Apply Xml2Owl Transformation (XSLT)
STAR OWL data
6
Validate Source Data (CWA Reasoning w.r.t. STAR ontology)
NIST-OWL Integrator
7
Create Satisfiable Merged Ontology
OAG
STAR
AIAG
8
Check STAR Data Consistency (w.r.t. Merged Ontology)
9
Compute Target Data (Individual Classification)
AIAG OWL data
10
Validate Target Data w.r.t. AIAG ontology
AIAG XML data
11
Owl2Xml Serialization
Translation

Figure 2. Scenario-based view of the semantic integration architecture.

We give a brief summary of the sequence of the eleven activities from Figure 2.

1. **Apply Xsd2Owl Transformation.** We began by applying an automated transformation to the OAG XML Schema representation to obtain an OAG OWL-based generalized ontology. This is a generalized ontology that contains concept descriptions only and no definitions [1]. The automated transformation was possible because we took into account the decisions and the rationale that led to the OAG components and document design. The automated transformation is captured in the XSLT transformation rules.
2. **Calculate concept subsumption and check satisfiability of the new OAG ontology.** This results in a new subsumption hierarchy for the OAG generalized ontology and an indication from the reasoner that the new ontology is either satisfiable or not. It can be unsatisfiable for several reasons. For example, an element that is mandatory in the super-type declaration is only optional in the new declaration. All such conditions must be resolved before proceeding.
3. **Create regular terminologies.** Once we have a satisfiable generalized terminology, any individual application integrator, typically a human being, can use the terminology to specify additional constraints or to provide definitions for concepts in a particular context. The original STAR and AIAG Schemas include free-text descriptions of the additional document constraints that need to be 'layered on top' of the OAG generalized terminology. For each such schema, these constraints are used to specify concept definitions based on the original concept descriptions. The outcome of this step are regular STAR and AIAG terminologies.
4. **Check satisfiability of each individual regular ontology.** Similar to Step 2, the outcome of this step is an indication from the reasoner whether each individual ontology (i.e., regular terminology) is satisfiable. All unsatisfiable conditions must be resolved before proceeding to step 5.
5. **Apply automated transformation from source (STAR) XML data to OWL data.** This step initiates the run-time activities required for a successful and interoperable data exchange. We transform XMLSchema instances into OWL-DL individuals that conform to the OWL model-based assumptions used in ontological reasoning. The outcome is STAR OWL data that corresponds to the initial XML data and transformed with respect to STAR ontology. The transormation rules are only dependent on XML Schema to OWL mapping (i.e., the transformation includes annotation of XML data with coresponding ontology (e.g., STAR ontology)).

[1] Concept refers to expressions that define a class in the OWL-DL language, which also provides constructs for establishing relationships between concepts. The meaning of concepts is specified using logical semantics, which distinguishes between concept description and concept definition. Concept description refers to a class with necessary conditions only; concept definition refers to a class with both necessary and sufficient conditions.

6. **Validate source data.** Validation of STAR OWL data includes consistency checking under both Open World Assumption (OWA) and Closed World Assumption (CWA). The outcome of this step, if successful, is an indication from the reasoner that the STAR OWL data are consistent with respect to the STAR ontology. An individual is valid only if it is consistent (belongs to specific concept) in both OWA reasoning and CWA reasoning. Validation is necessary to check the transformation and to check other semantic constraints, which are defined in the corresponding ontology. Examples of such constraints include additional semantic business rules based on Schematron rules and free-text descriptions provided with a schema. Because a DL reasoner makes the open world assumption, if a mandatory property is not present, the reasoner cannot conclude that it is false (as it is wrong to assume it will never be present). For that reason, the reasoner can conclude only contradictory but not insufficient information (i.e., missing properties). In a B2B context, a document being exchanged contains all required information and in order to compute that an instance has all mandatory properties it is necessary to validate instance with CWA.
7. **Create a merged ontology and check satisfiability.** In order to translate from STAR to AIAG OWL data, we need to create a merged ontology from the two individual ones and calculate new, concept-subsumption hierarchy[2]. Because new independently defined ontologies are based on the same generalized OAG terminology, a reasoner may combine axioms and calculate the new, concept-subsumption hierarchy. In the merged ontology one concept might be dependent on some concepts in the other ontology namespace. The merged semantics provides support for inferences over the source data that may yield unexpected results (such as those we discussed in the previous section). However, it is possible that the merged ontology is created at design time. In that case, the merged ontology will be referenced and (for the performance reasons) can be reduced only to include a sufficient set of concepts that is needed during the data transformation step. This step also includes satisfiability checking of merged concepts form source and the target ontology. The tool has to check satisfiability for every concept of the merged ontology and only a satisfiable merged ontology can be used in the next steps.
8. **Check consistency of the source (STAR) data with the new merged ontology.** The successful outcome of this step is an indication from the reasoner that all STAR OWL source data are consistent with respect to the merged ontology. Because the integration tool is a complete reasoner that includes consistency checkers, all axioms of the merged ontology must be loaded.
9. **Compute classification of the source (STAR) OWL data in the target (AIAG) ontology.** Assuming successful completion of the preceding steps, then we can use the individual classification capability of a DL reasoner to

[2] We include this step in the run-time activities, but it could also be done at design time.

compute target data in the AIAG ontology. The result is an assignment of the STAR OWL data to the specific AIAG class(es). At this point, specific STAR XML data may be successfully translated into target AIAG XML data. This, however, doesn't mean that all STAR data may be successfully translated to AIAG.

10. **Apply validation of newly created target (AIAG) OWL data**. The outcome of this step, if successful, is an indication from the reasoner that the AIAG OWL data are consistent with respect to the AIAG ontology. As discussed above, this requires OWA consistency and validation that the same individual is a valid instance of the target concept in the CWA reasoning. The individual consistency checking in OWA is already done with respect to the merged ontology. The OWL individuals classified to the AIAG concept hierarchy have to be checked for sufficiency with respective to target (AIAG) concepts. If an individual is inconsistent in CWA, then translation is not possible. If successful, however, then we can be sure that specific XML source data can be translated into a target OWL data and that the integration will succeed.
11. **Apply serialization of AIAG OWL data into AIAG XML data**. The outcome of this step is an AIAG XML instance that preserves semantics defined in the original STAR OWL data. For serialization into XML format we use concept and property hierarchy. If we use default XSD serialization from our OWL ontology, then the serialization is also provided. If we have a customized mapping to specific XMLSchema syntax (e.g., a sequence of elements defined in separate file), then that serialization is dependent on the mapping rules. The algorithm for serialization takes into account information provided during step 1 where all information about the source XML Schema syntax is captured.

4 Initial Findings and Lessons Learned

In this section, we discuss some initial findings and lessons learned.

4.1 Individual Equivalence Test

When dealing with B2B application integration, it is imperative to determine if two business documents are semantically equal. As mentioned before, during XML to OWL transformation, every new OWL individual is assigned a new URI identifier. That identifier is only necessary for individual classification and its actual value is not significant. That means that the same XML business document instance may be transformed to individuals with different URI identifiers but the same content. For datatypes, "semantically equal" means that the lexical representation of the literals maps to the same value. For individuals it means that they either have the same URI reference or are defined as being the same individual. We have developed a tool that helps us to do this. It is described below.

For every individual, the testing tool creates a temporary concept definition that contains the values constrained to the values specified in the individual properties.

In addition, cardinality constraints on the properties of the temporary concept definition are based on the property occurrence in the particular individual. All the temporary concepts are considered by the reasoner to determine equivalence among the corresponding individuals. Then, for every pair of equivalent concepts, the tool asserts *sameAs* mapping between two individuals. This means that the tool creates an assertion that explicitly defines equality between the two individuals. That new assertion will help the reasoner to calculate any new equivalence. For example, consider two individuals a1 and a2 that belong to the same concept C1 with same occurrence of the property p1 but with different fillers b1 and b2 (i.e., values of the property p1). If we calculate or insert the equality between fillers b1=b2, then corresponding temporary concepts for individuals a1 and a2 will be equivalent and, based on our definition of semantically equal individuals, we can conclude equality between individuals a1 and a2.The process is iterative and will end when no new concept equivalence is identified.

We emphasize that a reasoner can calculate a new subsumption tree and identify new concept equivalence by taking into account both concept definitions and individual assertions.

4.2 Concept Equivalence with Inconsistent Document Instances

We investigated whether two ontologies can facilitate interoperable data exchange and we used reasoner capabilities to perform satisfiability check between them. We determined that a necessary condition for interoperable data exchange is that there are no contradictory concepts in the merged ontology. It is, unfortunately, not a sufficient condition because some individuals may violate business constraints defined for a particular concept. Consider the following example.

Suppose a mandatory property, a necessary condition, is present within the target concept. Since the reasoner uses only definitions to calculate subsumption and equivalence among concepts and since a mandatory property is only a necessary condition, it will not be part of definition. This may give rise to an inconsistent source individual if the source concept specifies that property as optional. In a general case, any logical constraint that is not a part of either target or source concept definition but only their necessary conditions may cause a similar inconsistency and prevent interoperable data exchange.

Human designers face a similar problem every time they create a normalized ontology for a new business document content model specification. This problem deals with establishing whether an axiom is a part of concept definition, which includes necessary and sufficient conditions or only a part of concept description, which includes necessary conditions only. This distinction is critical because a concept definition is the mechanism used to classify concepts.

We are in a position to develop a tool that helps the designer evaluate alternative ways to specify concept description and concept definitions and to determine the potential drawbacks of each. We also plan to investigate ontology design patterns, which may avoid the "concept equivalence-individual inconsistency" type of translation problem. Additionally, we may expand default reasoner satisfiability checking to provide additional testing capability.

4.3 Lessons Learned

Based on our initial examination of Semantic Web technologies, we believe that they are sufficiently mature to allow experimental assessment in a fairly realistic enterprise-application-integration context. Our experiments used production-level XML Schema encoding of OAGIS 9.0 core components. The Xsd2Owl transformation takes a complete set of core component types from OAGIS 9.0 and is capable of creating OWL-DL ontology successfully. We worked successfully with some, but not all, examples of XML Schema definitions for OAGIS Business Object Documents (BODs).

The currently available tools are clearly not sufficiently robust and scalable for risk-free development of specialized add-on tools to support industrial interoperability efforts with all the complexities typically encountered. However, the rate of maturation and adoption of these tools is encouraging and it seems that these issues of robustness and scalability may be addressed in the near future.

When planning for future usage of the Semantic Web technologies within industrial enterprise application integration efforts, it is important to provide a transitioning path for moving from XML Schema-based to OWL-based application integration. While there are potentially other significant issues, we showed that it is possible to translate XML instances into OWL individuals and, in the opposite direction, to serialize OWL individuals as XML instances conforming to a specific XML Schema. This capability is important when presenting these new approaches to industrial communities as it shows that the existing legacy applications do not have to change all their interfaces over night to enjoy the benefits of this new technology. We were encouraged by the initial positive reactions from industry to the initial results from our experimental approach.

5 Conclusions

In this paper, we described a Semantic Web-based approach and a methodology to enhance syntax-based standards for enterprise applications integration (EAI). The methodology contains a number of integration and validation steps that are performed both at design time and run time. During design time, the methodology supports development of generalized and normalized ontologies and allows model-based similarity analysis of these ontological models. During run time, the methodology enables semantic translation of instances of business documents using the previously developed ontologies and automated reasoning tools.

Initial findings in testing the methodology show interesting capabilities such as the ability to perform individual equivalence tests that are content based. Through experimental work, we have gained a significant insight into the issues of necessary and sufficient conditions for achieving interoperable data exchange. The lessons learned so far indicate that the Semantic Web technologies are sufficiently mature to allow experimental assessment in a fairly realistic enterprise application integration context.

Our immediate future work will focus on further assessment of the initial ideas for Semantic Web-based EAI standards. The work will draw from on-going industrial standards-based integration efforts such as the ones going within STAR and AIAG industrial groups. We expect to identify key technical issues for the proposed approach, and through experimental demonstration show how such issues may or may not be addressed using the proposed approach. Our key contribution, we anticipate, will be to provide an increased understanding of whether and how Semantic Web technologies may be applied in a near future to industrial integration efforts.

Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply these products are necessarily the best available for the purpose.

References

1. Standards for Technology in Automotive Retail (STAR) Web Site, accessed November 2004. Available at http://www.starstandard.org/.
2. Automotive Industry Action Group (AIAG) Web Site, accessed November 2004. Available at http://www.aiag.org/.
3. Open Applications Group (OAG) Web Site, accessed November 2004. Available at http://www.openapplications.org/.
4. D.Trastour, C.Preist , and D.Coleman, *"Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches"*. 7th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2003, Brisbane, Australia, Sept 16-19th , 2003.
5. V. Haarslev and R. Moller. Description of the RACER system and its applications. In *Proceedings InternationalWorkshop on Description Logics (DL-2001)*, 2001.
6. R. Jelliffe, The Schematron Assertion Languages 1.5. Web Site, accessed November 2004. Available at http://xml.ascc.net/resource/schematron/Schematron2000.html.

Inter-Organization Interoperability in Transport Chains Using Adapters Based on Open Source Freeware

Paul Davidsson, Linda Ramstedt and Johanna Törnquist

Blekinge Institute of Technology, Department of Systems and Software Engineering, 372 25 Ronneby, Sweden. Paul.Davidsson@bth.se, Linda.Ramstedt@bth.se, Johanna.Tornquist@bth.se

Summary. The significance of electronic information exchange in transport chains and its influence on the performance of the logistics processes is well-known. While much focus is still on the various techniques for information exchange, many Small and Medium-sized Enterprises (SMEs) experience problems that are not primarily caused by an absence of technology. Several advanced solutions exist but investment possibilities within many SMEs are very limited while their needs for electronic data exchange may be very similar as for the large companies. We describe a general "adapter" solution based on open source freeware that make it possible (in principle) for any business system to exchange information with any other business system. It has been successfully applied in a pilot study involving two companies (the transport operator being an SME) using different business systems (Hogia Mobilast and Movex). The solution was evaluated with the result that it met the requirements of the companies and is now planned to be used for future data exchange between the companies.

1 Introduction

A transport chain usually involves a large number of actors, and in order to reach an acceptable level of efficiency it is important to synchronize these actors and their activities. For example, information concerning the status of the cargo needs to be available to the right person, at the right place, and at the right time. Although this is obvious for everyone involved and that advanced technology to collect and distribute information exist, this is seldom achieved. We argue that the main reasons are not due to lack of solutions, but rather based upon difficulties to integrate existing software systems and the high costs associated. Many companies are small or medium-sized enterprises (SMEs) with a limited possibility to invest in the existing solutions, while their needs to perform electronic data interchange to act competitive may not be so much different from those of the large companies. Instead, a large amount of transport-related information, such as bookings and cargo specification, is exchanged manually via fax machines and phone. Problems that arise due to the manual work required are for instance that information may not be accessible at a certain place when needed and also information duplication is complicated. The question that remains to be answered then is; are advanced and expensive systems required to solve these problems, or are there other, simpler and less expensive, solutions?

The project "Transport informatics for increased cooperation between the parts in a logistic chain", also called AIS 42 [1] focused on this question. The project had a reference group of companies located in, or close to, the city of Karlshamn, Sweden. One of the goals was to show that by using only simple and cheap solutions it is possible to establish adequate electronic information exchange between different actors in transport chains that have different prerequisites and preferences. The overall aim of the project was to develop and demonstrate the use of a common platform where the companies are able to exchange the necessary information between themselves as well as their other customers and suppliers. Moreover, the platform should work independently of the existing information systems and routines used by the different actors.

The project was based on two case studies of which we will focus on one. These were studied in detail and the needs for improvements to support the transport activities were identified. The needs were analyzed and classified into categories such as:

- simplification of transport booking and other administrative activities,
- tracing and positioning of goods and carriers,
- deviation detection,
- support for calculations of environmental load, and
- logging of freight-related data,

all of which require (different levels of) system interoperability. The needs of improvements were ranked based on importance which resulted in a requirement specification for the software platform to be developed. The platform and its functionalities were later demonstrated on one of the transport chain case studies and evaluated. Besides the software platform, the project also generated insights in the problems related to technological aspects as well as organizational issues within the freight transportation industry.

The next section describes related work. Section 3 then describes the solution that was developed to support and facilitate information exchange, while Section 4 presents one of the case studies upon which the project is based. Section 5 provides a discussion of the results and pointers to future work.

2 Related Work

The review includes two parts; related research, in particular projects with similar aims, and existing software solutions available on the market.

2.1 Related Research Projects

There are a number of projects that have addressed the problem of information exchange and IT solutions for freight transportation. Many of them are projects funded by the EU that focus on how to create solutions and processes for transport mode-independent information exchange that can provide traceability, as well as, a standardized and smooth way to communicate. Some of these projects are:

- INTRARTIP (Intermodal transport real time information platform) [7]
- ITESIC (Integration of technologies for European short intermodal corridors) [4]
- PISCES (Pipeline intermodal system to support control, expedition and scheduling) [12]
- D2D (Demonstration of an integrated management and communication system for door-to-door intermodal freight transport operations) [3].

Other projects focus specifically on traceability and straight-forward combinations of technologies to achieve this for cross-border transportation. Two examples are SITS (Simple intermodal tracking and tracing) [13] and MULTITRACK (Tracking, tracing and monitoring of goods in an intermodal and open environments) [11].

Even though AIS 42 and the projects mentioned above have some issues in common, AIS 42 is different in several aspects. AIS42 have chosen a pragmatic, bottom-up approach with focuses on the particular situation for SMEs. Furthermore, the size of AIS42 is much smaller than the other projects with fewer resources available for software development and with industrial actors that are comparable smaller and local. Consequently, usability, simplicity, interoperability, and cost-effectiveness are the primary factors in the choice of technology.

2.2 Existing Technology

Support for electronic information exchange is often found in large business system, e.g., ERP (Enterprise Resource Planning) systems. Such systems often focus on a specific type of company, e.g., a transport buyer or a transport operator. The systems do often have some ability to exchange information with other types of business systems, but this is often limited to a standard set of formats and systems. There exist several off-the-shelf TA (Transport Administrative) systems for storage, synthesis and communication of data, such as Movex [10], Hogia MobiLast [6], and many more. These systems and the other systems that support electronic data interchange have varying characteristics, but all of them require substantial investments both to procure and to use. As a consequence, SMEs are able only to invest in one system, if any at all. This, in turn, makes it impossible to carry out electronic business activities with several companies using different business systems due to system interoperability problems.

One existing solution that meets some of the interoperability requirements is Microsoft BizTalk [9]. The main purpose with BizTalk is to facilitate system communication independently of the individual communication formats in the systems by acting as an interpreter between the systems. It is based upon a central server through which all exchanged information pass. It uses XML and supports the main protocols for e-mail and http. However, being a proprietary client-server solution it has several disadvantages, such as, making the actors dependent on third party, being expensive and possible risks for communication bottlenecks.

3 The Adapter Solution

As mentioned in the introduction, some of the most important needs of improvements identified were: simplification of administrative activities, such as, transport booking, tracing of goods and carriers, deviation detection, and calculations of environmental load. Many of these require a complete unbroken process with gathering of data, data processing and information distribution. We decided to focus on the first point that concerns the reduction of usage of fax machines and other manual ways of information exchange (reducing the administrative costs as well as the number of error caused by the human factor) and increase the accessibility of information by making electronic information exchange possible.

Considering the requirement specification, a group of 15 students from Blekinge Institute of Technology developed a software prototype for electronic information exchange. The prototype is built using only on freeware with open source code such as, Enterprise Java Beans, J2EE, JBoss, MySQL, etc., and uses state-of-the-art technology like XML and Web services. The prototype provides the possibility for different business systems to communicate and supports information exchange via web portals, e-mail, fax and SMS. The prototype is an information carrying system meaning that the system acts independent of what type of data that is transferred through it.

The basic idea can be seen as a generalization of the well-known *Adapter design pattern* [5] used within object-oriented software engineering. It is also similar to the concept of *connectors* [2], as well as, *wrapper agents* [8] as used within the area of multi-agent systems [14]. To each (legacy) business system an adapter is built that enables the system to interact with other the other business systems in the transport chain. Such an adapter is mainly composed of three parts; a “bridge”, an “interpreter”, and a “message handler”. (See Figure 1) The Bridge handles the interaction with the business system, and the Interpreter translates the data from the format the sending business is using to the format the receiving business system is using (and vice versa). The Message handler takes care of the communication with the adapters of the other business systems. The bridge typically makes use of the functions for exporting and importing information that most business systems are equipped with. If this is not the case, more sophisticated methods must be used.

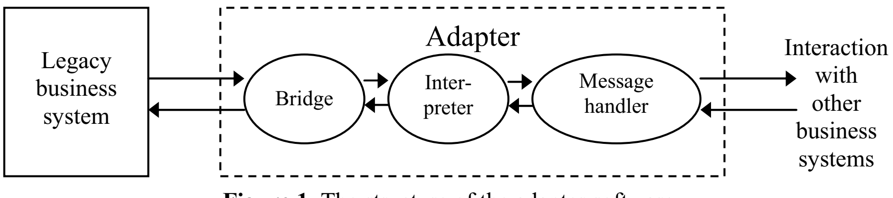


Figure 1. The structure of the adapter software.

The Adapter prototype software is available as freeware and is relatively easy to install. It is structured as a program that includes general message handling components, and shells for the interpreter and the bridge. Much effort was spent on reusability making it easy to adapt the software to arbitrary business system (at least those that have basic data import and export functionality). Also modifiability was considered and since the software is module-based, extensions of the system are facilitated. The system requires “log in” for usage, and encryption and digital labeling is included to increase the security.

4 Pilot Study

The transport chains that were selected as the case studies of the project are not very complicated consisting of just a few actors and little or no cross-border traffic. The focus is on SMEs based in the region of Blekinge, Sweden, with partly very limited resources for investments in IT but still needs the functionalities to stay competitive. Many of the needs of improvements and opportunities identified in the specific case studies can, however, also be found in many other companies in the transportation business of varying size, nationality and resources.

The transport chain we will focus on here consists of a truck transport carried out by Karlshamns Expressbyrå (KE) containing packaged temperature-sensitive cargo from the production facility of Karlshamns AB (KAB) in Karlshamn to the customers of KAB localized in Northern Europe. KE uses a system developed by SA-data for surveillance and booking. History of cargo temperature stored in the cooling systems in the trucks is made available by TermoKing. During the project, KE decided to start using Hogia Mobilast which is a system that supports the planning of transports since available transport resources can be controlled, transport orders can be administered via the system etc. KAB uses Movex for administration of orders, transport prices etc.

Through interviews the following aspects were identified as important for KE:

- Simplification and speeding up of administration of a transport assignment.
- Control of truck space utility.
- Cargo follow-up during and after transport regarding temperature and time. Deviation detection important.
- Calculation and reporting of environmental impact.

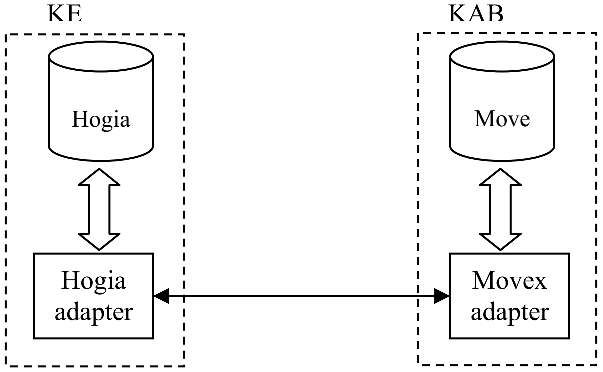


Figure 2. The setting of the pilot study.

Whereas the following aspects were regarded as important by KAB:

- Delivery time precision (punctuality)
- Status of the cargo regarding deviations, i.e. traceability.
- Simplification and speeding up of the administration associated with transport bookings.

Thus, a common aspect of interest for both companies is to minimize the administration connected to transport bookings, and this is what we decided to focus upon in the demonstration experiment.

The prototype platform was tested by having two different major business systems; Hogia Mobilast and Movex, in two different companies, KE and KAB, interacting via the platform (see Figure 2).

In the demonstration experiments the actual business systems (together with the associated Adapters) were executing at the two companies. A number of functionalities were tested (see Figure 3).

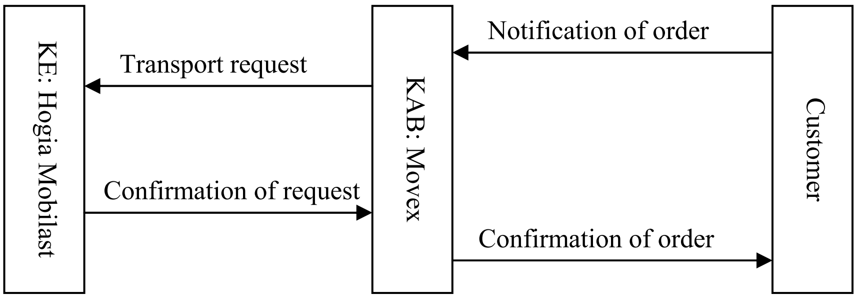


Figure 3. Examples of functionalities tested in the pilot study.

For instance, KAB was able to make transport requests from Movex directly into the Hogia Mobilast system at KE. Similarly, KE was able to confirm the request from inside its system directly to Movex at KAB. Further functionalities include monitoring active requests.

The most complex part of an Adapter is typically the Bridge. As an example, we take a closer look at the Bridge of the Hogia Mobilast adapter which is composed of three threads:

- *Import*, which receives XML messages from the Message handler via a web service client, and saves them as files in the “In” folder.
- *Export*, checks the “Out” folder every fifth minute (this time interval was used in our experiments; it should, of course, be chosen based on the response time requirements for the particular application) and when a new file is detected, it is sent via a web service client to the Interpreter for further processing. When it gets an acknowledgement from the Message handler that the message was successfully sent, the file is moved from the Out folder to a “Sent Orders” folder.
- *Status monitor,* monitors changes regarding the orders that are currently active, by making requests concerning the status of these orders each 20 second (in our experiment) and saves them as XML-requests to the In folder of Hogia

Mobilast. It then reads the reply from the Out folder and sends it to Interpreter for further processing.

Bridges for other business systems with import and export facilities work in a similar way.

5 Conclusions and Future Work

An evaluation of the practical relevance and economical gain of the prototype was made and based on interviews of the participating companies. The prototype provides a direct use since it offers cost-effective (due to the use of freeware and easiness of adaptation) and platform-independent communication. Thus, the prototype work satisfactory in this context at this stage and with respect to attributes such as modifiability and compatibility with several interfaces. These benefits make the prototype useful in several contexts and not only for the project participants. A cost-effective communication solution generates in addition several implicit benefits such as possible reduction of error in documents that are exchanged (due to minimization of manual duplication work) and increased information accessibility. An example of future work is to measure the reduction of errors in exchanged documents as a consequence of implementing the Adapter in companies.

The economical impact is more difficult to approximate partly due to that it is a long-term effect and partly is associated with other factors related to the companies and their adjustments technology-wise. The selection of technology used in the prototype was considered appropriate and in line with needs of the companies and their future business plans.

It is hard to create a fully generic communication solution that does not require any adjustments at all. Even our platform requires some adjustments and further development before it can offer a full-scale solution at KE and KAB. We have, however, demonstrated the possibility to achieve basic and sufficient communication functionalities like the large-scale TA systems offer but to a considerable lower cost. This is something that is of significant importance for the survival of smaller actors with same needs as the larger players but with a much more limited investment possibility.

Compared to existing centralized solutions, such as BizTalk, the proposed solution has a number of advantages, e.g., being independent of third parties and avoiding central communication bottlenecks.

We are currently working with applying the prototype to additional companies and business systems. Moreover, we have investigating the possibilities of developing an ontology, possibly based on [15], so that a common communication language can be used between the adapters. This would significantly reduce the need of developing interpreters, since there for each business system only need to be translation into one language, independent of which business system it will interact with.

6 Acknowledgements

The AIS 42 project was financially supported by VINNOVA (The Swedish Agency for Innovation Systems). We also wish to acknowledge the valuable input from the participants of the project (ABU AB, Carlshamn Mejeri AB, DFDS Tor Line AB, Elektronikcentrum i Svängsta AB, FoodTankers AB, ICA Handlarnas AB, Karlshamns AB, AB Karlshamns Expressbyrå, Karlshamns Hamn AB, Södra Cell AB, TFK, Växjö University), as well as the group of third year students from the Software Engineering program at Blekinge Institute of Technology who developed the software used in the pilot study, namely: Andreas Mattisson, Áron Kelemen, Daniel Hansson, Daniel Hellkert, Johan Gjertz, Jonas Hoverman, Kent Wennerlund, Lars Widmark, Linus Gustavsson, Martin Bengtsson, Martin Skjaeraasen, Pheter Stål, Stefan Hagström, Thomas Sundblom, and Tobias Bergman.

References

1. AIS 42, Project website http://www.ec.se/ais/
2. Bures, T., Plasil, F., Communication Style Driven Connector Configurations, *Software Engineering Research and Applications*, LNCS Vol. 3026, Springer, 2004.
3. D2D, website, http://prosjekt.marintek.sintef.no/d2d/ or http://www.bmt-ts.com/d2d/ for other information
4. EUROMAR EEIG, ITESIC Final report for publication, Deliverable to the European Commission DGVII, 2000 (Available at http://europa.eu.int/comm/transport/extra/)
5. Gamma, E., Helm, R., Johnsson, R., Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented*, Addison-Wesley, 1994.
6. Hogia MobiLast, Company website http://www.hogia.se/godstrafik/
7. INTRARTIP Consortium (2000), Final Report for Publication, Deliverable to the European Commission DGVII (Available at http://europa.eu.int/comm/transport/extra/)
8. Kolp, M. T., Do, T., Faulkner, S., Introspecting Agent-Oriented Design Patterns, In S. K. Chang (Eds.), *Advances in Software Engineering and Knowledge Engineering*, vol. III, World Publishing, 2004.
9. Microsoft BizTalk (http://www.microsoft.com/biztalk/)
10. Movex, product of Intentia http://www.intentia.com/
11. MULTITRACK Consortium (1999), MULTITRACK Final Report, Deliverable to the European Commission (Available via http://www.cordis.lu)
12. PISCES Consortium (2000), PISCES Final Report, Deliverable to the European Commission (Available at http://europa.eu.int/comm/transport/extra/)
13. SITS Consortium (2000), SITS Final Report, Deliverable to the European Commission (Available at http://www.phys.uu.nl/~durr/TransportStuff/sits/)
14. Wooldridge, M., *An Introduction to Multi-Agent Systems*, Wiley, 2002.
15. Zimmermann, R., Butscher, R., Bodendorf, F., An Ontology for Agent-Based Supply Chain Monitoring, *Agent Technologies in Logistics, ECAI 2002 Workshop*, Lyon, France, 2002.

MAFIIA – An Architectural Description Framework: Experience from the Health Care Domain

Ståle Walderhaug[1+2], Erlend Stav[2], Stein L. Tomassen[2], Lillian Røstad[2] and Nils B. Moe[2]

[1] Norwegian Joint Medical Service, Medical R&D, 2058 Sessvollmoen, Norway
[2] SINTEF ICT, 7465 Trondheim, Norway,
{stale.walderhaug, erlend.stav, stein.l.tomassen, lillian.rostad, nils.b.moe}@sintef.no

Summary. Healthcare information systems are characterized by having many stakeholders, roles, complex and diverse information systems, high degree of formalized working practices and an intense focus on quality concerns like interoperability, security and reliability. There is an emerging need for a structured architectural tool for supporting system developers and architects working with this kind of critical infrastructure. This paper presents MAFIIA - an architectural description framework specialized for the health care domain. The framework has been used in the development of three different healthcare information systems: a system for individual care plans, a platform for image-guided surgery and a patient evacuation support system. The experience from the case studies shows that the framework is a useful and flexible tool for creating an architectural description, and assists in keeping the focus on selected quality concerns.

1 Introduction

In health-care organizations a conglomerate of software systems is used. Development and maintenance of such systems are challenging, as special focus is required on security and usability, as well as interoperability with other systems. To handle this challenge, it is essential that the developers and other personnel responsible for the development, maintenance and administration get a good understanding of the system's architecture, its interfaces to environment, and the context in which the system will be used.

For the central concepts of architecture and architectural description we use the following definitions from 1, and for interoperability the definition from 2:

- Architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
- Architectural Description: A collection of products to document an architecture.
- Interoperability: a) The ability of systems, units, or forces to provide services to and accept services from other systems, units or forces and to

> use the services so exchanged to enable them to operate effectively together. b) The condition achieved among communications-electronics systems or items of communications-electronics equipment when information or services can be exchanged directly and satisfactorily between them and/or their users.

Unfortunately, architectural descriptions for Healthcare Information Systems (HIS) vary in structure and content – if they exist at all. They seldom include important information like the stakeholders the system was originally built for, which laws and regulations affected the system, which standards that were applied, and which other systems it was built to interoperate with. From the end users' perspective, successful implementation of a HIS is dependent on the target system's suitability for its purpose. Thus, documentation of both system and user work process interoperability is important.

The lack of formal architectural descriptions and a need for information integration and interoperability was our motivation to develop an architectural description framework called MAFIIA (Model-based Architecture description Framework for Information Integration Abstraction). Collaborative design is encouraged and supported by the framework, to ensure that the systems are built based on real understanding of the needs of the end users and the requirements from environment system interfaces. The framework assures a common structure and formalism of architectural descriptions for an organization's systems. At the same time, it provides the flexibility to focus on the concerns defined by the organization. This will assist developers in maintenance and evolution as well as development and description of new systems.

This paper presents MAFIIA and our experience from three case studies where the framework was used to develop architectural descriptions of HISs with a special concern for usability, reliability and interoperability. For each case study we briefly present the background before we summarize our experience from applying MAFIIA to it. In the discussion, we address the following questions based on the case studies:

- How did the prescribed development process of the framework fit the case, and did it help us document what we needed?
- How did the framework address the interoperability concern?
- Was the framework flexible enough to support the individual differences between the cases?
- Was the framework equally suitable when introduced at the start of the development as when used to continue the development of an existing system?

Before describing MAFIIA we first present the background for development of the framework. MAFIIA is presented in section 3 before the case studies are described in sections 4, 5 and 6. The experiences are discussed in section 7 before we provide some concluding remarks in section 8.

2 Background

SINTEF ICT is a research institute in Norway that works with organizations from many sectors, including defense and health. Both the defense and health sectors are characterized by having many stakeholders, roles, complex and diverse information systems, high degree of formalized working practices and an intense focus on quality concerns like security and reliability. Developing, maintaining and extending such time critical systems, some of which cannot be taken offline, can be very difficult. An explicit system architecture description framework that includes business aspects as well as traditional systems engineering aspects can help addressing these concerns.

Based on the experiences from architecture description frameworks for command-control systems in the Norwegian Armed Forces 34, SINTEF ICT has developed a model based architecture description framework for information systems with a special focus on information integration, called MAFIIA.

3 MAFIIA

MAFIIA is a framework for creating architectural descriptions of software intensive systems. The framework assists the architect by:

- Supporting cooperative design through the definition of a set of views and selection of notation that allow end user involvement in important parts of the work.
- Supporting development and description of the architecture of new systems, as well as documentation of the architecture of existing (legacy) systems.
- Providing use guidelines for architectural patterns applicable to systems that need to integrate information from several heterogeneous environment systems
- Providing a structure that ensures that documentation of different systems developed using the framework will have a uniform structure and content.
- Presenting a list of quality related concerns that the architect should consider when creating the architecture, and instructing how to include description of the concerns of particular importance.

The MAFIIA framework adopts the terminology and conceptual framework defined in IEEE 1471 1. Compared to IEEE 1471, MAFIIA gives further normative guidelines, including the use of UML as notation, a set of predefined viewpoints and a reference architecture.

An architectural description created using MAFIIA is structured around a set of views, each of which describes the system from a certain viewpoint. Views are useful for illustrating different aspects of the same target system, and are also the basis for RM-ODP 5. Concerns that are of special importance to the target system, e.g. security and interoperability, must be identified and described. A set of system assets, e.g. standards and laws to which the system must comply, is included. A reference architecture is defined by MAFIIA, and can be refined for a specific target system, or for a set of related systems.

It should be emphasized that the main purpose of the architectural description is to give the reader an understanding of the fundamental aspects of the system and its context. Thus, the architectural description is kept at a relatively high abstraction level and does not include e.g. full user requirements, complete business process models, or more detailed design information.

In the following subsections, each part of MAFIIA is described in more detail.

3.1 Concerns

MAFIIA defines how to describe concerns of special importance to the system. These concerns will need special attention within all or most of the views described in section 3.4. A concern may require special models or other formal descriptions to be created to ensure that the architecture description is correct and complete.

Functional aspects that are considered to be of such importance that they should be treated separately and be specifically visible in the documentation should be identified and treated as a concern.

For a HIS, security should always be treated as a special concern due to patient privacy issues. Confidentiality, availability and integrity – the key characteristics of information security – are essential in health care information systems. Security should be addressed in a dedicated model in each view of the architectural description.

Interoperability is a special concern for a HIS that must operate in a context where many other critical systems both provide and rely on information from the system being architected. The focus on interoperability require the architects to carefully design the information and operation interfaces to the environment, as well as the distribution and realization of the system components.

3.2 System Assets

System assets are sources of information that can be used when developing an architectural description. System assets can be considered as implicit requirements, which are not necessary to include in the requirement view, however assets may be included in component, deployment and realization views. The most common assets for architectural description of a HIS are dictionary (a reference list of concepts), standards (a formalized model or example developed by a standardization organization or established by general consent), laws and regulations and at last patterns. A pattern is a description of a recurring, well-known problem and a suggested solution. Patterns are identified and can be used on many system levels. The MAFIIA framework includes guidelines for when to apply well-known patterns in the architecture. Summary descriptions of recommended patterns are included, along with references to sources such as 678, where the full pattern description can be found. The MAFIIA framework suggests a number of patterns related to interoperability and information integration. The selected patterns are referenced in the architectural description of the target system, and specialized in the view(s) where they are applied. The use of architectural patterns will facilitate interoperability between systems and sub-systems.

3.3 Reference Architecture

MAFIIA defines an overall reference architecture for information integration systems. This is a high-level, generic architecture which is used as a basis for development of target system architectures, and to compare architectures of existing systems. The MAFIIA reference architecture defines four logical tiers, and the interface to the environment as shown in Figure 1.

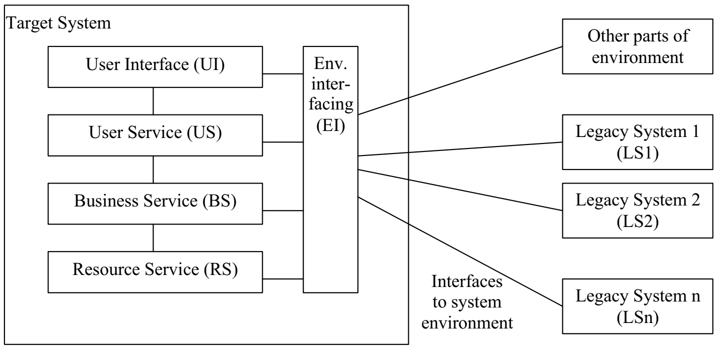


Figure 1. MAFIIA Reference architecture for information integration systems

3.4 Views and Viewpoints

A central part of MAFIIA is its definition of a set of viewpoints. Each viewpoint defines how a specific view of the target system shall be described, and prescribes a set of models that the view shall include. The notation to use for each model is also defined – normally a set of UML diagrams with accompanying textual description is used. Architectural descriptions created with MAFIIA contain the following views:

Context view describes the business-related aspects and stakeholders of the target system and its environment. Environment systems that will be involved in or influence the operation of the target system are identified, and their interfaces and collaborations with the target system are described. The context view should be created in collaboration between end users or domain experts, and software architects. The description in this view is important during the initial development of the architecture, but may be even more valuable during maintenance and integration with other systems, as it provides background motivation for the architecture that may otherwise be forgotten and hard to reconstruct.

Requirement view describes functional and quality requirements that can affect the architecture of the target system. This does not include complete user requirements, but instead generalized versions of each type of user requirement that are of importance to the architecture. The models in this view are based on use case diagrams and tables of prioritized requirements. Interoperability requirements are derived from the interfacing systems described in the context view, and the framework also provides a set of requirement choices guiding the process of eliciting integration requirements.

Component view describes the decomposition of the system into components, including their interfaces, interaction, and the information that is handled. The security model is an important part of this view, and describes security mechanisms and how these are integrated with the rest of the system. The models of this view are kept at a logical and platform independent level, and do not include realization details. For this view, the framework presents a set of architectural design issues for information integration systems, and proposes patterns and other solutions that can be suitable when the issue has specific characteristics.

Distribution view describes the logical distribution of components and roles. It describes which components that must be together on a node, which components that must be distributed to different nodes. The framework includes recommendations for distribution choices based on parameters such as system size, geographical distribution, and communication capacity. The distribution choices can be limited by the current deployment of components in environment systems, as well as their security infrastructure.

Realization view describes how to implement the system described in the other views on a selected target platform. It includes mapping of the architecture to the selected technology platform (e.g. Java or .Net), and also describes the actual deployment of the system on the selected nodes. Both technology platform and deployment choices can be limited by the requirements for interoperability with the environment systems.

3.5 Process

The MAFIIA framework recommends an iterative development process. An iteration of the architectural description work usually starts with describing the context view, and ends with the realization view following the order in which they are presented in the previous section. The work does not proceed in a strict sequence, but frequently returns to previous views when new insight is acquired. Each iteration results in a version of the architectural description that is reviewed. More than one iteration may however be necessary to complete the architectural description.

4 Case 1: Individual Care Plans

In the SamPro project SINTEF ICT cooperated with the local health care region and the company Visma Unique AS to develop a system for individual care plans.

According to the Norwegian health law, patients with long-lasting health problems and with the need for coordinated health services, have the legal right of an individual care plan. The objectives of making an individual care plan are:

- To support a coordinated, unified and individual health service, and ensure a responsible care provider for the patient at all times.
- To map the patient's objectives, resources and needs for different services in different domains and coordinate and evaluate actions to meet those needs.
- To increase cooperation between a service provider, patient and other involved (peer/family), between service providers and other organizations at the same or across levels of service.

A system for individual care plans needs to read, present and report information a variety of health care information systems. To do this it is necessary to identify stakeholders and understand and describe the work processes, user requirements and existing laws and regulations that affect the system, as well as interfaces to the environment systems. MAFIIA was chosen to develop and describe the architecture of the system.

4.1 MAFIIA Applied to Individual Care Plans

The users of an individual care plan system come from different levels of the public health service, and from different health institutions. Some of the users are even outside of the health sector, e.g. patients, relatives and social workers. The main challenge is to ensure that the users of the system at all times can access the information they have the right and need to see. The most important concern was security. The most central system assets were relevant laws and standards concerning health informatics and security.

The users participated in developing the Context and Requirement views. Together with the end-users we described the system stakeholders, their relationship to each other and to the new tool. This information was essential for understanding the complex domain, and gave an overview of all the contributors to the requirements. A process model for the new system was developed as part of the context view, to help understand the work-processes a system for individual care plans needs to support. A model describing environment systems was developed to identify candidate systems for integration. The care plan contains only a short description of the different services a patient receives. The service providers each have a more thorough documentation in their own information systems. The goal was to identify if and how the care plan system should be integrated with these systems. Another integration issue was the use of a common user database. Health care personnel should be able to log on to the care plan system using their regular username and password.

In component view the system was decomposed into sub-systems and their components. The access control components were grouped in a separate sub-system that is used by all the other sub-systems. A model for role based access control combined with information categorization was developed as a part of the architectural description.

The system was designed to be a service in the regional health network. The distribution view described where to place the different logical layers of the system in the network, and the chosen protocols for secure network communication.

.Net was chosen as the implementation platform for the individual care plan system. The realization view therefore included description of mapping to .Net technology components, including mapping of the security mechanisms described in component view to the .Net security model.

One of the main challenges in the project was to communicate the system architecture to the developers. One person from the development team was involved in architectural meetings to help this process, and after the project ended we had an evaluation meeting with the developers to identify any problems or suggestion for improvement. The meeting confirmed that the use of a structured and model-based framework, such as MAFIIA, resulted in an architectural description that was easy to understand and adopt for the developers, with minimal help from the architectural team.

Visma Unique AS is continuing to work on the system, and is planning to release it as a commercial product on the market in 2005.

5 Case 2: CustusX

CustusX is a platform for image-guided therapy developed by the Center of Competence - 3D Ultrasound in Surgery, which consists of St. Olavs Hospital, Norwegian University of Science and Technology (NTNU), and SINTEF. The goal with CustusX is safer, more accurate, and less invasive surgical procedures.
The initial idea behind CustusX was to have a navigation platform for clinical testing of new procedures in the operating theatre. One of the requirements was that the software should be cross platform executable and run on both ordinary personal computers and in large-scale visualization installations.

The development of CustusX has been demo driven – that is, repeatedly new functionality has been requested and then implemented. This had led to a system with no structured documentation of the architecture and consequently very difficult to maintain. It was also acknowledged that the system had become too big and complicated, and consequently did not support the needs of the different target groups very well. It was therefore recognized that the system had to be reorganized to make it both more maintainable and configurable to better meet the needs of the different user groups. The first task in this work was to specify an architectural description of the current system and then for the future system. MAFIIA was chosen for several reasons; it defines a structure for the documentation, it is user centric, and provides support for HIS development.

5.1 MAFIIA Applied to CustusX

CustusX has evolved to be quite an extensive platform for image-guided therapy, which has led to a great number of potential stakeholders that have a strong interest in the use of this system. It was therefore vital to identify all these stakeholders as they may affect the final architectural system description. For instance, some users need the application for telemedicine purposes, e.g. in situations where there is a need of an expert opinion and where the expert is located remotely. This would influence how to describe which components that needs to be distributed.

One important step defined by MAFIIA is to identify all relevant concerns. The concerns identified by the architectural group (including end users) were configurability, performance, reliability, safety, and security. Since CustusX will be used in many different clinical applications like surgery planning, surgery guidance, radiology intervention procedures, radiation therapy, simulation, and training it needs to be highly configurable. The system should be able to seamlessly integrate with environment systems, such as the legacy Picture Archive and Communication System (PACS). Security was important since the system will contain patient information and be distributed.

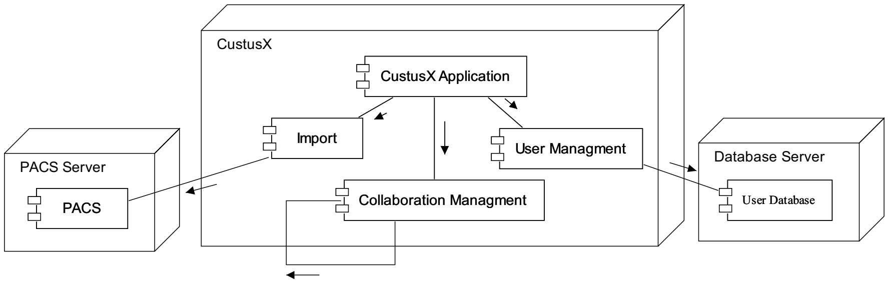


Figure 2. CustusX system distribution model

Figure 2 shows the system distribution model for the CustusX. The Collaboration Management component can be connected to another CustusX system. The PACS server is a legacy system, and is therefore separated in an individual node. Many CustusX systems can access the user database and is consequently located in a separate node.

The architectural group decided not to specify any realization view in the first version of the CustusX architecture. This view was not needed to make estimates and a detailed plan for the implementation work and will be done in the next version of the architecture.

6 Case 3: Evacuation Support System

In 2002, SINTEF ICT and the Joint Medical Service in the Norwegian Armed Forces started a project called Evacuation Support System (EvacSys). The objective was to implement and test a new system to improve information flow between nations and organisational levels in a military medical evacuation chain. Today, when injured soldiers are evacuated from the battlefield, tracking log, medical treatment and observations are documented on a paper-based form that follows the patient. Each nation in NATO has its own paper record format, but should be interoperable between the nations.

EvacSys is a new electronic system for medical and command-control information capture and distribution. Using personal electronic memory tags, wireless military tactical communication, state-of-the art portable computing terminals and a new military Electronic Health Record (EHR) called SANDOK, a complete distributed information system was architected, implemented and tested. EvacSys involved many stakeholders and the system had to be interoperable with both national and other NATO nations' health information systems, as well as the underlying military command and control infrastructure. To handle this concern, MAFIIA was used as a framework for specifying and developing the EvacSys architecture.

6.1 MAFIIA Applied to EvacSys

The first step in the MAFIIA framework is to identify concerns and system assets. Focus was put on functionality, usability, reliability and interoperability. Major system assets were NATO standards and Norwegian laws and regulations for medical documentation and exchange.

The MAFIIA framework's context view was found very useful when describing the context in which the EvacSys will operate. User and business processes were documented to ensure user interoperability. Identifying and documenting the environment information and operational system interfaces required a lot of time and effort, as much of this documentation was hard to find. The Context view models provided an important input to the system requirements models.

As a part of the Components View, the information and processing components were carefully modeled to optimize robustness and interoperability with existing infrastructure. The difference between Norwegian and English/American person identifiers and naming conventions required some additional components to be specified.

The EvacSys Distribution View explicitly models reliability and interoperability related concerns. Critical operational components were distributed onto different logical nodes. The distribution models emphasized that there are no “single point of failure” at the different logical locations (battlefield, mobile aid station, transport etc.) in the EvacSys. For example, all terminals used for information input and output are generic in the way that they have no components that are specialized for one specific user or patient. Information-intensive components such as pulse-oxymetri sensors were put on separate nodes and shared filtered information through standard communication syntax and protocol. The data replication mechanism used to distribute patient data at the field hospitals was implemented according to the push model of the Publisher-Subscriber pattern.

Military standards and regulations were important input to the realization view description. Only a limited set of equipment, platforms and protocols are approved for use in an operational setting.

The first prototype of the EvacSys system was tested and evaluated by the medical battalion of the 6th division during a 5-day military exercise in northern Norway (Setermoen) in December 2003. The evaluation report concludes that the system architecture and information flow worked according to system requirements.

7 Discussion and Related Work

This section discusses experiences from using MAFIIA based on the cases presented and addresses the questions stated in the introduction. It should be noted that MAFIIA was used as a tool in the architectural work of the projects, and were not itself the research focus of the projects. The discussion presented here is thus mainly based on a subjective evaluation from the different researchers that participated in the projects, and to some degree on feedback from other participants in the projects. All of the cases are based on projects of approximately 15-25 person months. There is also great variation in type of systems of the cases, even though all of the cases are within the health care domain.

The feedback from the cases indicates that the development process described in MAFIIA was easy and useful to follow. The combination of a description of what to do, and a checklist of what to include of models, standards etc, helped the developers in the work. The assistance provided by the framework in identifying quality related concerns was reported as important to all the cases, e.g. security in all cases and configurability in CustusX. Also the context view was consistently reported to be very valuable in all the cases.

With respect to interoperability concerns, all three cases have been tested in a real environment. Feedback from all of the cases showed that the context view was an essential tool for understanding the complex domains, and gave an overview of all the contributors to the requirements. The EvacSys prototype was successfully tested in an integrated operational setting. SAMPRO was the first system designed for external access within the Norwegian Health Network.

The flexibility of the framework was essential in supporting the necessary variations in the description which the different cases required. The ability to select different concerns to focus on, and the ability to extend each view with new models were utilized in all three cases. There were some differences between the cases in how the requirements were specified. Some used use cases while other preferred only textual descriptions. There were no reports of problems using either.

The framework was found equally useful in the two cases were it was introduced from the start or early in the development process for a new system, and in the case where used for architectural clean-up and further development of an existing system.

There exist a number of related architectural frameworks that are commonly in use today. RM-ODP (Reference Model of Open Distributed Processing) 5 is a framework that provides the developers a standard for creation of systems that support distributed information processing services to be realized in heterogeneous environments. The method uses five different viewpoints to describe the system. The framework is neutral in the selection of tools for describing the architecture.

TOGAF (The Open Group Architecture Framework) 9 is an enterprise architecture framework that "consists" of a subset of architectures: business, data, application, and technology respectively. TOGAF consists of a set of tools and methods for developing different architectures. The goal of TOGAF is to become an industry standard method that is neutral to both selection of tools and technologies.

ATAM (The Architecture Tradeoff Analysis Method) 10 is an analysis method used to understand the various tradeoffs that have to be made when creating architecture for software intensive systems.

NATO has started a Multilateral Interoperability Program 11 that focuses on interoperability between member nations' command and control systems.

9 Conclusion

The findings from the case studies indicate that the use of the MAFIIA facilitates development of systems that will operate in a complex environment. Despite the individual differences of the case studies presented here, the framework has proven to provide good assistance for the architectural work, and results in a well-structured architecture description. The method gives excellent support when developing architecture with a strong focus on specific selected concerns, and security in particular.

Applying an architectural description framework like MAFIIA will have best effect if it is used as a standard for all software intensive systems developed within and for the organization. We believe that large organizations, e.g. public health care within a state or country, is in a position where they can require that at least all new system that they acquire or developed are described in a standard of their choosing.

References

1. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Std 1471-2000. ISBN 0-7381-2518-0.
2. Federal Standard 1037C, Department of Defense Dictionary of Military and Associated Terms in support of MIL-STD-188.
3. J.Ø. Aagedal, A-J. Berre, B.W. Bjanger, T. Neple, C.B. Roark, “ODP-based Improvements of C4ISR-AF”. 1999 Command and Control Research and Technology Symposium, U.S. Naval War College, Rhode Island, USA. June 29 - July 1, 1999.
4. B. Elvesæter, T. Neple, J.A. Aagedal, R.K. Rolfsen, “MACCIS 2.0 – An Architecture Description Framework for Technical Infostructures and their Enterprise Environment”, Submitted to 2004 Command and Control Research and Technology Symposium.
5. Basic Reference Model of Open Distributed Processing – Part 1: Overview and guide to use the Reference Model. ITU-TS, Rec. X901 (ISO/IEC 10746-1), Standard 1995
6. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns. Elements of Reusable Object-Oriented Software, Addison Wesley, 1995.
7. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. Pattern-Oriented Software Architecture. A System of Patterns. John Wiley & Sons, Ltd, 1996.
8. D. Schmidt, M. Stal, H. Rohnert, F. Buschmann. Pattern-Oriented Software Architecture. Patterns for Concurrent and Networked Objects. John Wiley & Sons, 2000.
9. TOGAF, Open Group, http://www.opengroup.org/togaf/, accessed 2004-11-30
10. P. Clements, R. Kazman, M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison-Wesley, 2001.
11. Multilateral Interoperability Program, http://www.mip-site.org/, accessed 2004-11-30

ISO Standards for Interoperability: a Comparison

Kurt Kosanke

CIMOSA Association e.V., Stockholmer Str. 7, D-71034 Böblingen, Germany, kosanke@cimosa.de

Summary. The lack of interoperability between ICT systems is becoming more and more a bottleneck in the collaboration and co-operation of enterprises. Co-operating parties have to exchange business information and have to have the same understanding of the meaning of the exchanged information and to trust both the communication itself and the validity of its contents. The paper reviews two ISO standards, which are aiming on the interoperability issue comparing the different aspects of these standards and introduces a new work item in standardisation, which expects support from two current European initiatives on interoperability and others. The paper is intended to guide further standard developments by identifying current solutions and their relation with one another.

1 Introduction

Operating in the global business environment requires worldwide co-operations between enterprises sharing their core competencies in order to exploit short-term market opportunities. Besides the timely availability of products and services, the real time exchange of related business information between the co-operating partners is even more important. Such exchanges are needed for both operational control and to an even larger extend for the decision-making processes during the establishment of the cooperation like market opportunity exploration and co-operation planning and implementation. Therefore, both the easy communication between the people involved and the quality of interoperation between the supporting systems of information and communication technology (ICT) play a key role in such co-operations. The urgent need for organisational interoperation and decision support on all levels of the enterprise operation is recognised in the communities of business and academia as well as in standardisation [1].

Major issues in global collaboration and co-operation of enterprises are the problems in communication between people, between people and ICTs and between different ICTs. Whereas the first one is due to different cultures, languages and even professional jargons and can only be addressed by either a common language, which is very unlikely, or through translations between the different languages and meanings. The other two problem areas originate from the different software and hardware implementations of the information and communication technology. These areas require both human and machine understanding of the exchanged information.

But what is the meaning of interoperability? First of all interoperability is domain specific. Besides the three domains of people, people and ICT, and ICT itself identified above, there are the different business domains like industry, finance, health, each one having sub-domains like categories of humans (managers, experts, operators), of devices (controllers, actuators, sensors), and of systems (computers, machines, communication networks) with their specific needs for interoperation.

But the ability to interoperate within the ICT supported business arena is a rather complex task. The heterogeneity of ICT implementation is such that there exist different solution spaces depending on the combination of existing systems and in many cases such solutions are not transferable to other cases. According to Chen and Doumeingts [2], who reported on the results of a European initiative on a development for road maps to interoperability, there exist several levels of interoperability. They identified interaction between two systems can at least take place at the three levels: *data, resource* and *business process* and interoperability may therefore be achieved on multiple levels: inter-enterprise coordination, business process integration, semantic application integration, syntactic application integration, and physical integration.

The recommendations in Chen and Doumeingts are to address the subject of interoperability through three main research themes or research domains: enterprise modelling, ontologies, and architectures and platforms. These three areas deal with a) the representation of the inter-networked organisation to establish interoperability requirements; b) address the semantics necessary to assure interoperability; and c) define the implementation solution to achieve interoperability. General state of the art reports have been issued by both the European Network of Excellence (NoE) initiative INTEROP [3] and the EU Integrated Project (IP) ATHENA [4]. Standardisation in these areas has been and is continuously addressed by both international standards organisations and industry consortia. A multiplicity of standards exists in the three fields identified above.

2 Definitions on Interoperability

There exist numerous definitions on interoperability; e.g. a very careful chosen web search produced 22 entries on interoperability. Selected examples are:
Interoperability:

1. achieved only if the interaction between two systems can, at least, take place at the three levels: *data, resource* and *business process* with the semantics defined in a business context (Chen, Doumeingts) [2]
2. ability of two or more systems or components to exchange information and to use the information that has been exchanged (IEEE) [5].
3. ability to communicate with peer systems and access their functionality (Vernadat) [6]
4. ability of different types of computers, networks, operating systems, and applications to work together effectively, without prior communication, in order to exchange information in a useful and meaningful manner. There

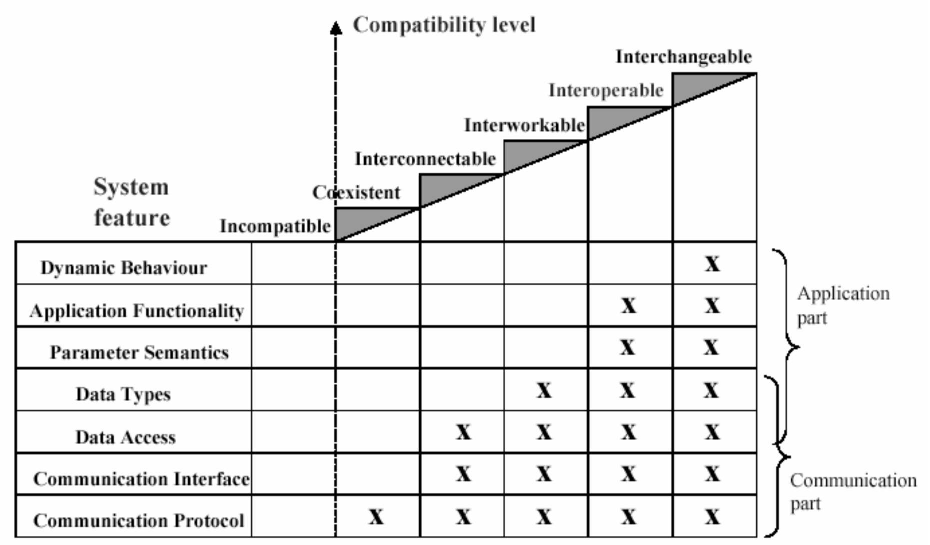
are three aspects of interoperability: semantic, structural and syntactical (from www) [7]

5. (Computer Science) ability to exchange and use information (usually in a large heterogeneous network made up of several local area networks) (WordNet) [8]

Related definitions are:

1. Interoperable: (computer Science) able to exchange and use information (WordNet) [8]
2. Interoperation: implies that one system performs an operation on behalf of another (Chen, Doumeingts) [2]
3. Interoperation may occur between two (or more) entities that are related to one another in one of three ways (ISO 14258) [9]:
    1. *Integrated* where there is a standard format for all constituent systems
    2. *Unified* where there is a common meta-level structure across constituent models, providing a means for establishing semantic equivalence
    3. *Federated* where models must be dynamically accommodated rather than having a predetermined meta-model

IEC TC 65/290/DC [10] identifies degrees of compatibility depending on the quality of communication and application features (see Figure 1).



System feature	Incompatible	Coexistent	Interconnectable	Interworkable	Interoperable	Interchangeable
Dynamic Behaviour						X
Application Functionality					X	X
Parameter Semantics					X	X
Data Types				X	X	X
Data Access			X	X	X	X
Communication Interface			X	X	X	X
Communication Protocol		X	X	X	X	X

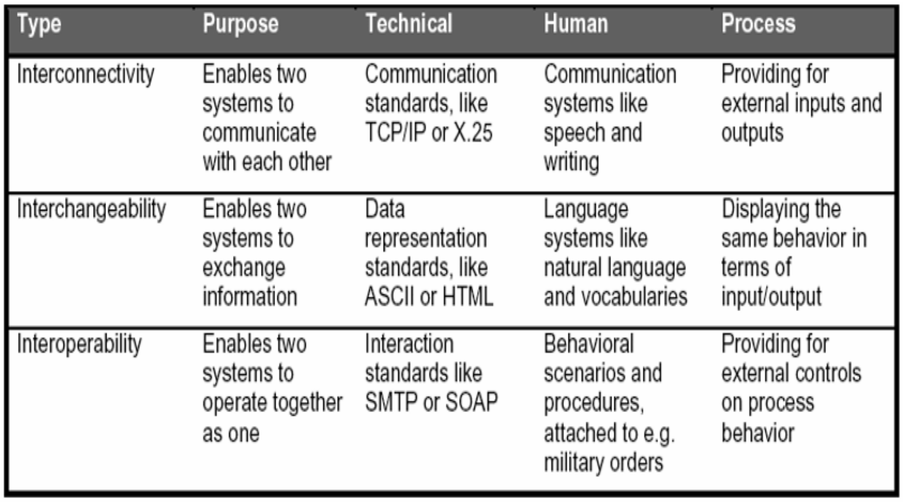
Figure 1. Compatibility levels (adapted from IEC TC 65/290/DC) [10]

The related definitions of the compatibility levels in Figure 1 are:

1. Incompatibility: inability of two or more devices to work together in the same application
2. Coexistence: ability to of two or more devices operate independently of one another in the same communication network

3. Interworkability: ability of two or more devices to support transfer of device parameters
4. Interoperability: ability of two or more devices to work together in one or more applications
5. Interchangeability: ability of two or more devices to replace each other in working together in one or more application

Another attempt to categorise interoperation has been published by Stegwee and Rukanova [11]. Starting from the assumption of communication as the main concept in interoperation and using the IEEE definition of interoperability, the authors have defined a framework, which is shown in Figure 2. The framework identifies three types of communication and enables the identification of relevant standards to support communication between the different components of organisational interoperation. The authors favour the ISO - OSI layer model as a means to support interoperation.



Type	Purpose	Technical	Human	Process
Interconnectivity	Enables two systems to communicate with each other	Communication standards, like TCP/IP or X.25	Communication systems like speech and writing	Providing for external inputs and outputs
Interchangeability	Enables two systems to exchange information	Data representation standards, like ASCII or HTML	Language systems like natural language and vocabularies	Displaying the same behavior in terms of input/output
Interoperability	Enables two systems to operate together as one	Interaction standards like SMTP or SOAP	Behavioral scenarios and procedures, attached to e.g. military orders	Providing for external controls on process behavior

Figure 2. Identifying types of interoperation and related of standards (from [11])

The related definitions of the types of communication in Figure 2 are:
Interconnectivity: ability to exchange information at a network, syntactical level
Interchangeability: ability to use information at a presentation, semantic level
Interoperability – ability to use information at an application, pragmatic level

The mapping of the definitions of interoperation given in the two frameworks is presented in table 1, which shows the difference in terminology as well as the difference in scope of the term interoperation. Especially the term interchangeablity is used with quite different meaning. Whereas it is used for an intermediate level of communication in [11] it identifies the ultimate interoperation in [10].

However, it seems very unlikely that in reality interoperability or interoperation on a larger scale will occur in any one of the three ways identified in ISO 14258 (see above), but in a mixture of those. Assuming a global environment there will be neither possibility for global unification nor for global integration and even federation in the dynamic mode as identified above seems very hard to achieve without any a priory knowledge about the entities that have to interoperate. The two standards analysed in the following try to provide this a-priory knowledge by creating profiles of their entities – applications and manufacturing software units, respectively.

Table 1. Categories of interoperation

IEC TC 65/290/DC) [10]	Stegwee and Rukanova [11]
	interconnectivity
interworkability	interchangeability
interoperability	interoperability
interchangeability	

3 ISO 15745 -Industrial automation systems and integration — Open systems application integration frameworks [12]

The standard consists of four parts: Part 1: Generic reference description; Part 2: Reference description for ISO 11898-based control systems; Part 3: Reference description for IEC 61158-based control systems; Part 4: Reference description for Ethernet-based control systems.[1]

This standard outlines an Application Integration Framework (AIF) - a set of elements and rules for describing application interoperability profiles, which will enable a common environment for integrating applications and sharing life cycle information in a given application domain. The generic elements and rules are providing for developing templates for Application Interoperability Profiles (AIPs), and their component profiles - process profiles, resource profiles, and information exchange profiles.

Such profiles may describe profiles based upon particular technologies and therefore makes this standard applicable to application integration frameworks developed for industrial automation environments. Environments such as discrete manufacturing, process automation, electronics assembly, semiconductor fabrication, wide-area material handling, and other automation and control sectors such as utility automation, agriculture, off-road vehicles, medical and laboratory automation, and public transport systems.

[1] Note. Application here means industrial automation applications, which comprise several types of interconnected devices, which perform a variety of functions.

4 ISO 16100 Industrial automation systems and integration — Manufacturing software capability profiling for interoperability [13]

The standard consists of four parts as well: Part 1: Framework (relating to ISO 15745); Part 2: Profiling Methodology; Part 3: Interface services, protocols and capability templates; Part 4: Conformance test methods, criteria, and reports.

ISO 16100 specifies a framework for the interoperability of a set of software products used in the manufacturing domain and to facilitate integration into manufacturing applications. The framework addresses models for information exchange, software objects, interfaces, services, protocols, capability profiles, and conformance test methods. The standard specifies a methodology for constructing profiles of manufacturing software capabilities and requirements for interface services and protocols used to access and edit capability profiles and associated templates used in the capability profiling method. In addition, conformance test method and criteria for the capability profiling of a manufacturing software unit are specified.

5 Relations between the two standards

The main content identified in each standard is shown in Table 2. It identifies subtle differences owing partly to the difference in scope of the two standards. Whereas Framework elements in ISO 16100 are solely concerned with the interfacing requirements for interoperability of manufacturing software units (MSUs), their roles and the entities they have to support, ISO 15745 identifies a larger set of such elements needed to support interoperability between the components of applications.[2] Similarly for framework rules where again ISO 15745 provides a larger set. However, some of the missing items in ISO 16100 are listed under framework aspects, an entry that does not exist in ISO 15745.

The latter uses the concepts of life cycle and model to classify the requirements in terms of a set of interfaces, services, components and configurations intended to guide the developers of industrial automation applications.

The three integration model types identified in Table 2 correspond to three different profile classes and a complete integration model of an application corresponds to a set of application interoperability profiles (AIPs), which identify the particular interfaces, services, protocols, and timing used by the different components within the application.

[2] Note. Template is seen differently in the two standards. Being an element in 15745 it is identified as a rule in 16100.

Table 2. Content of ISO 15745 and ISO 16100

ISO 15745 Open systems application integration frameworks	ISO 16100 Manufacturing software capability profiling for interoperability
Application Integration Framework (AIF) - framework elements: diagram, model, profile, template, specification - framework rules: diagram notation and semantics, model composition, profile template and taxonomy, template conformance and exchange syntax	Mfg. software interoperability framework - framework elements: roles, activities, artefacts associated with software entities when dealing with manufacturing process, information, and resources - framework rules: relationships, templates, and conformance statements needed for constructing a capability class - framework aspects: - syntax and semantics shared between MSUs; - functional relationships exist between MSUs; - services, interfaces, and protocols offered by MSUs; - ability to provide MSU capability profiling
Concepts and constructs used: - Application life cycle - AIF integration models: - process integration model (PIM) presents views of process control, material and information flow - resource integration model (RIM) identifies devices, communication networks equipment, material/product, and operators (humans) needed in the PIM - information Exchange integration model (IEIM) identifies syntax, semantics, structure, and sequences of information produced, consumed, organized, stored, and retrieved by the resources involved in the PIM	Concepts and constructs used: - manufacturing software unit (MSU)

Table 2. Content of ISO 15745 and ISO 16100 (continued)

ISO 15745 Open systems application integration frameworks	ISO 16100 Manufacturing software capability profiling for interoperability
Application interoperability profiles (AIPs) consists of: - one process profile - one or more resource profile(s) - one or more information exchange profile(s)	Software capability profile - taxonomy - capability classes and rules: types of mfg. domain, activity, computing system, services, protocols, supplier, others
AIP concepts and rules: - combine the interface specification option selections as required by the application - shall be a single specification aimed at providing a defined function - shall comprise a specific combination of profiles	Capability templates and rules: - common part contains general information about SMU - specific part contains SMU specific lists of attributes, methods, resources, constraints, others
AIPs shall be constructed by: 1) documenting the functional requirements as noted by a PIM 2) selecting the appropriate base specifications for the object interfaces denoted in the integration models 3) selecting (conforming) sets of options, or subsets, in the base specifications 4) combining references to a set of compatible base specifications in order to meet the identified application functional requirement 5) describing it in terms of an interface definition language	capability profiling process 1) analyse software requirements 2) identify/create template 3) enter profile data

ISO 16100 provides only one concept and construct: the manufacturing software unit (MSU). Again to support interoperability a software capability profile is defined. Capability classes and rules as well as templates and rules provide the elements for constructing the capability profiles of the software units.

The actual process of profile creation is shown as Application Interface Profiles (AIP) construction (ISO 15745) and capability profiling process (ISO 16100) in the lower part of the Table 2. The number of steps identified in the two standards varies from 3 for ISO 16100 to 5 for ISO 15745. However, the process itself is rather similar considering the difference in scope of the two standards.

6 Summary and Conclusions

The two standards presented in this paper address the issue of interoperability between ICTs. With their focus on interoperability within manufacturing applications and between manufacturing software units, respectively, the two standards are both using the concept of profiles to capture the information needed to identify the capabilities of the entities, which have to interoperate.

In respect to the three types of interoperation identified in ISO 14258 [9] the two standards contribute to a semi federated approach of interoperation at the ICT level, providing means for identifying a priory information that can be mapped or matched at run time to/with the profile of the partner entity.

However, the comparison between the two standards also shows the need for standards harmonisation. Both terminology and structure of the two standards differ and any potential user, who has to employ both standards, will be confused by their difference. Certainly a more thorough analysis of the state of the art in standardisation and an adoption of already established structures, rules and terminologies would reduce such differences as identified for the two standards.

In addition, the two standards do not yet address sufficiently the human aspects of interoperation, which have been identified in Stegwee and Rukanova [11]. For the communication between people and between people and machines, information about the internal structure and the dynamics of the application may be even more important than the information about the potential exchange itself. Business process models can provide such information with their focus on semantic unification and orientation on end-user needs.

Work on standards for interoperability has been started in ISO. In TC 184 SC5/WG1 a new work item 'Requirements for establishing information interoperability in manufacturing-enterprise-processes and their models' will address this aspect of interoperability. New standards have to improve the ICT interoperability as well by providing a base for unifying the needed information exchange between the parties involved, may it be between operational processes during enterprise operation or between their models during decision support. Inputs to this work will also be provided besides others by the two European initiatives ATHENA and INTEROP.

Standardisation for enterprise interoperability is considered an important subject. However, the current state of standardisation is not yet sufficient to allow easy implementation at the operational level. Many of the standards are still on the conceptional level and more details are still required to make them truly useable in the operation. Work is required in areas of languages and supporting platforms, especially for the business process model creation and execution. To enable cross-organisational decision support especially the subject of 'common' semantics has to be addressed. In ISO/CEN 19440 [14] modelling constructs are defined using a meta-model and accompanying text (sufficient to define an intuitive semantics as well as to define a model repository database). However, the capture of finer details of these meanings requires even more detailed formal semantics. Ontologies will play an important role in the area of semantic unification.

References

1. Kosanke, K. (1997), Enterprise Integration and Standardisation, in Kosanke, K. & Nell, J.G. (Eds.). Enterprise Engineering and Integration: Building International Consensus *Proceedings of ICEIMT'97 (Intern. Conference on Enterprise Integration and Modelling Technology),* Springer-Verlag, pp 613-623
2. Chen, Doumeingts, (2003), European initiatives to develop interoperability of enterprise applications—basic concepts, framework and roadmap, Annual Reviews in Control 27, pp 153–162
3. INTEROP (2004), Knowledge map of research in interoperability in the INTEROP NoE, Deliverable D1.1, EU-NoE Project IST-508 011, www.interop.noe.org
4. ATHENA, (2004), Document Deliverable D.A1.1.1, Project A1: Enterprise Modelling in the Context of Collaborative Enterprises ATHENA, EU IP- Project - No 507849
5. IEEE (1990), IEEE (Institute of Electrical and Electronics Engineers): Standard Computer Dictionary- A Compilation of IEEE Standard Computer Glossaries
6. Vernadat, F.B. (1996) Enterprise Modelling and Integration: principles and applications; Chapman & Hall, ISBN 0 412 60550 3
7. www (2001) library.csun.edu/mwoodley/dublincoreglossary.html
8. WordNet Browser 1.7.1, Princeton University Cognitive Science Lab
9. ISO 14258 (1999), Industrial automation systems and integration – Concepts and rules for enterprise models, ISO TC 184 SC5
10. IEC TC 65/290/DC (2002), Device Profile Guideline, TC65: Industrial Process Measurement and Control
11. Stegwee, R.A., Rukanova, B.D. (2003). Identification of Different Types of Standards for Domain-Specific Interoperability. In Proceedings of the Workshop on Standard Making: A Critical Research Frontier for Information Systems, John L. King and Kalle Lyytinen, (eds.), Seattle, WA, December 12-14, 2003, pp. 161- 170. available on line at: http://www.si.umich.edu/misq-stds/proceedings/139_161-170.pdf Standard Making: A Critical Research Frontier for Information Systems, MISQ Special Issue Workshop
12. ISO 15745 (2000), Industrial automation systems and integration — Open systems application integration frameworks, ISO/TC/184/SC5
13. ISO 16100 (2002), Industrial automation systems and integration — Manufacturing software capability profiling for interoperability, ISO/TC/184/SC5
14. CEN-ISO 19440 (2003), Constructs for Enterprise Modelling *CEN TC 310/WG1* and *ISO TC 184/SC5/WG1*

ONAR: An Ontologies-based Service Oriented Application Integration Framework

Dimitrios Tektonidis[1], Albert Bokma[2], Giles Oatley[2] and Michael Salampasis[3]

[1] ALTEC S.A., Research Programmes Division, M.Kalou 6, GR – 54629, Thessaloniki, Greece dte@altec.gr
[2] Centre for Electronic Commerce, University of Sunderland, Sunderland, SR6 0DD, United Kingdom Albert.Bokma@sunderland.ac.uk, Giles.Oatley@sunderland.ac.uk
[3] Department of Informatics, Technological Educational Institute of Thessaloniki, Thessaloniki, Greece, cs1msa@it.teithe.gr

Summary. The evolving technologies of Semantic Web and Web services are providing new means for application integration frameworks. The need for semantically enriched information exchange over the flexible environment of the internet provides a valuable enhancement to traditional methods and technologies for Enterprise Application Integration. However the utilization of the Semantic Web and Service Oriented Architecture (SOA) is not as straightforward as it appears and has specific limitations and inefficiencies due to the fact that is was not originally designed for that purpose. This paper presents a methodology that aims at the exploitation of these two technologies and the definition of an ontologies based enterprise application integration framework (ONAR).

1 Introduction

The application integration between heterogeneous applications such as ERP systems between companies as well as B2B environments in general is a challenge that has been confronted with a plethora of technologies and methodologies. In our research [1] we have seen that the key aspect in from simple integration to more complex cases [2] is the semantics. When we refer to the semantics we signify the meta-information that describes the exchanged information.

Service oriented application integration (SOAI) as it is presented in [3] exploits the capabilities for the functional description of web services that are used for the actual integration. This paper presents our work on the creation of an integration framework based on SOAI that utilizes Semantic Web Technologies [4] in order to enrich the semantics of the exchanged information.

The Ontologies Based Enterprise Application Integration (ONAR) Framework [5] utilizes web ontologies to create semantic conceptualizations of the business concepts that exist inside an application. This conceptualization is used for the creation and the registration of the Web services in a UDDI based registry.

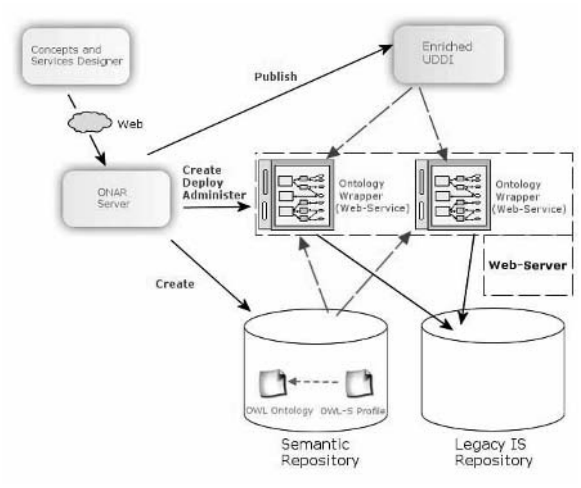


Figure 1. Ontologies Based Enterprise Application Integration (ONAR) Framework.

2 Conceptualizing Information System Using OWL Ontologies

Ontologies are used for the formal definition of concepts in a given domain. However the utilization of ontologies for software component definition is not as simple as the definition of concepts alone. Research on this domain as presented in [6] and [7] shows that the utilization of the ontologies requires some modification in the principals of the frame-based ontologies languages. The state of the art ontologies languages are presented evaluated in [8] and [9], but even a purely frame-based logic is still insufficient as there are important differences principally with respect to object-oriented modelling. Of course ontologies were not designed for software engineering, and thus their utilization for this purpose requires creating a semantic layer that will describe and define the technical infrastructure of an application.

Our work has focused on creating associations between the conceptual models and the application logic. To facilitate this we have developed a graphical tool (Figure 1) that enables the creation of the conceptual models using ontologies and associating these models to the actual system resources.

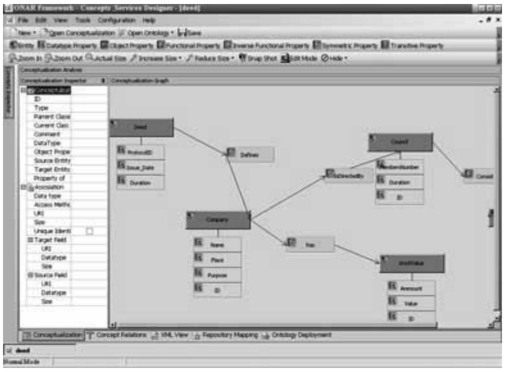


Figure 2. Ontologies based Information System conceptualization using ONAR Concepts and Services Designer

2.1 Definition of Logical Entities

Following the OWL [10] frame based logic, the concepts that represents a logical entity inside the application are defined by an OWL class. The entities inside the ONAR conceptualization methodology inherit all the features of the OWL classes. However their semantics changes in order to be adapted to the semantic description of a system resource.

Inheritance is permitted allowing some class features to be inherited from one class to another. The relation between the parent class and child classes follows the same rules that both object oriented engineering and frame-based logic supports, however multiple inheritance is not permitted. This constraint is due to the fact that in complex information system polymorphism of some concepts will unduly increase the complexity of conceptualization.

In our approach, the conceptualization of the system is not application centric but integration centric. This means that the creation of a conceptualization is developed and the need of the integration and not regarding the structure of the system. This increases the reusability of the concepts definition that can be reused in different cases.

Entities are allowed to have properties that correspond to OWL datatype proper-ties. A property can belong to more than one entities and is contrary to the usual principles of object oriented modelling. This principle derives from the frame based logic of OWL ontologies and extends the object oriented engineering where a property belongs to only one class.

Finally both entities and properties are enriched with notional properties like lexical descriptions, the ability to define the inverse of a concept and the ability that a property can potentially be derived from another property. These enrichments are used to increase the inference for the semantics of the concepts and their properties inside a model following the OWL principles.

2.2 Defining Relations Between Entities

Apart from the definition of entities and their properties, our work has focused on creating semantics in the relations between concepts. The existing semantics of relations that are expressed in object oriented class diagrams in UML or Entities Relations (E-R) diagrams pose limits to the semantic two concepts may have at the logical level. Therefore using the four types of relations (object relations) that are defined in OWL we have adjusted these relations types in order to increase the inference capabilities of the conceptualizations. We have followed the OWL logic and syntax having the two (or more) entities that participate in a relation divided into:

a) Domains of the relation: that is consisted of the concepts that consist of the area of definition of the relation.
b) Ranges of the relation: that the domains concepts of the relation can receive values from.

More precisely we have used and adjusted the following relations between entities:

- **Functional relation:** where the domain concepts instance can be related to more than one instance of the range concepts.
- **Inverse Functional relation:** where the range concepts instances can be related to more than one domain concepts instances.
- **Symmetric relation:** 1 to 1 relation between the instance of the domain and the range concepts.
- **Transitive relation:** we have defined transitive property as an inferential symmetric property to facilitate the logical allocations of concepts.

However we have to limit the syntax of OWL which permits the existence in a relation of more than one concepts as ranges and domains. In our approach (in a relation) if the domains contain more than one concept then the range should be defined from only one concept and vice versa. This ensures that in the association of the logical model to the system resources will not create indeterminism.

3 Associating Logical Entities to Information System Repository

One of the most important results of our work is the association between the logical model and the repository of the system. In our early attempts we have tried to extend OWL in order to include mapping mechanisms. However we have discovered that the new “extended OWL” syntax will be too complicated and it will mix up semantic with syntactic issues. This of course would decrease the reusability of the conceptualizations.

Therefore we distinguish the conceptualization into two ontologies that are:

a) The conceptualization ontologies based on OWL that define semantics aspect of the conceptualizations.
b) The association ontologies that association the conceptualization ontology to the repository of the application.

The disassociation between the semantic and the syntactic description has as the following results:

a) To increase the reusability of the conceptualizations ontologies: The same conceptualization ontologies can be used to different information system.
b) To reduce cost of maintainability when changes happened in to the application repository.
c) To enable the creation of common vocabularies that can be shared between heterogeneous applications.

In our work so far we have achieved the application of the ONAR framework to relational databases but where the goal is to support any type of repository including application servers.

4 SOAI Using Semantic Conceptualizations

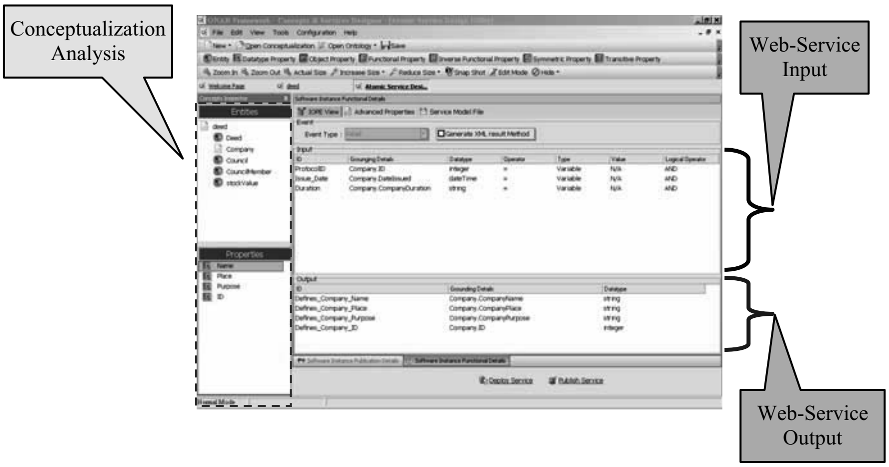
The creation of a Service Oriented Architecture that will implement the integration between the applications is also based on the semantic conceptualizations. In fact the ONAR framework software components (Web services) are based on the conceptualization schema (the set of the conceptualizations) of the application. This strong relation between the conceptualizations and the web services server is the actual innovation compared with other existing approaches [11][12].

The ONAR integration process is the methodology that should be followed in order an integration case to be defined and implemented using the ONAR framework. The process consists of the following phases:

a) **Conceptualization phase** where the entities that need to participate in the integration case are conceptualized to shared conceptualizations (OWL ontologies). The set of the conceptualizations create the conceptual schema of the integration.
b) **Association phase**. The conceptual schema is associated to the information systems repositories that participate in the integration.
c) **Design phase**. In this phase using the conceptual schema the definition of the software instances takes place.
d) **Deployment phase** where the software instances are implemented as software components (Web services) and deployed.
e) **Publication phase**. The framework uses a UDDI registry enriched with OWL-S profile features (from semantic invocation) to publish its software in-stances. Using the semantic meta-information that the conceptual schema provides the framework publish information regarding the utilization of the web services.

4.1 Designing web Services Based on the Conceptual Schema

In our work, the whole web services design creation and their registration to the UDDI registry is related to the conceptual schema. The design of the software instance is the process where elements of a conceptualization are used to define the input and the output of the Web service. The design of a web service consists (Figure 3) of its syn-tactic definition and its semantic description.



Input

ID	Grounding Details	Datatype	Operator	Type	Value	Logical Operator
ProtocolID	Company.ID	integer	=	Variable	N/A	AND
Issue_Date	Company.DateIssued	dateTime	=	Variable	N/A	AND
Duration	Company.CompanyDuration	string	=	Variable	N/A	AND

Output

ID	Grounding Details	Datatype
Defines_Company_Name	Company.CompanyName	string
Defines_Company_Place	Company.CompanyPlace	string
Defines_Company_Purpose	Company.CompanyPurpose	string
Defines_Company_ID	Company.ID	integer

Figure 3. Designing Web services using a conceptualization Ontology

The syntactic definition defines the input and the output as well as the behavior of the Web service. This definition is based one conceptualization of the conceptual schema. Starting from one basic entity the Web service designer can use all the concepts that are necessary for his definition. The syntactic definition (ONAR Service Model) of the service contains all the semantic and syntactic relations between the basic entity (primary concept) and the secondary concepts. The ONAR Service Model [5] is an XML document that contains a conceptualization ontology and an association ontol-ogy of the functionality of a Web service.

The reason of having a new definition document for the syntactic definition of the service, apart from maintenance purposes, is to increase the portability of the web services. We have implement the necessary functionality that enable us to automati-cally generate the source code (C#) of a Web service based only to the conceptualiza-tion ontology the association ontology and the ONAR Service Model. Therefore if two applications share the same conceptualization ontologies but have different struc-tures (different association ontologies) they can exchange Web services.

4.2 Creating Semantic Description for Supporting SOAI

The Web Service Definition Language (WSDL) is used to describe web services at the technical level. However the use (or consumption) of a web service requires first the technical implementation and an understanding of its functionality. The capabilities of the WSDL at this level are limited as it was not designed to serve this purpose. Instead other protocols have been produced in order to support Web services discovery and description.

Our work has been based on M.Paolucci attempts [13][14] to integrate a powerful UDDI discovery and categorization mechanisms with the semantics of OWL-S [15]. We have extended UDDI using OWL-S elements in order to include semantic description of the web services. Apart from the ONAR service model that a service designer defines with the definition of the behaviour of the web service (Inputs, Outputs and effects), the designer should describe the web service behaviour at the semantic level.

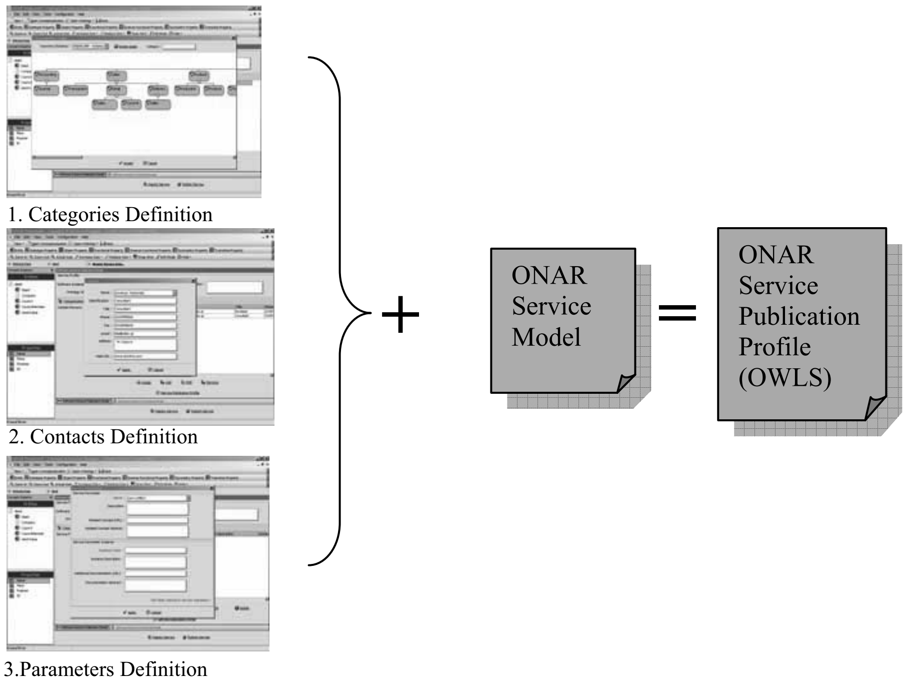


Figure 4. ONAR Service Publication Profile

This Service Publication Profile defines the semantic of the web services. The Web service designer defines:

1. The categories of the service: Using the UDDI categorization schemas the web service designer defines the categories that apply to the web service.
2. The contact list of the service: A list of persons that are associated with the service and their type and responsibilities
3. The parameters of the service: Special features of the services that do not belong to a particular categorization schema.

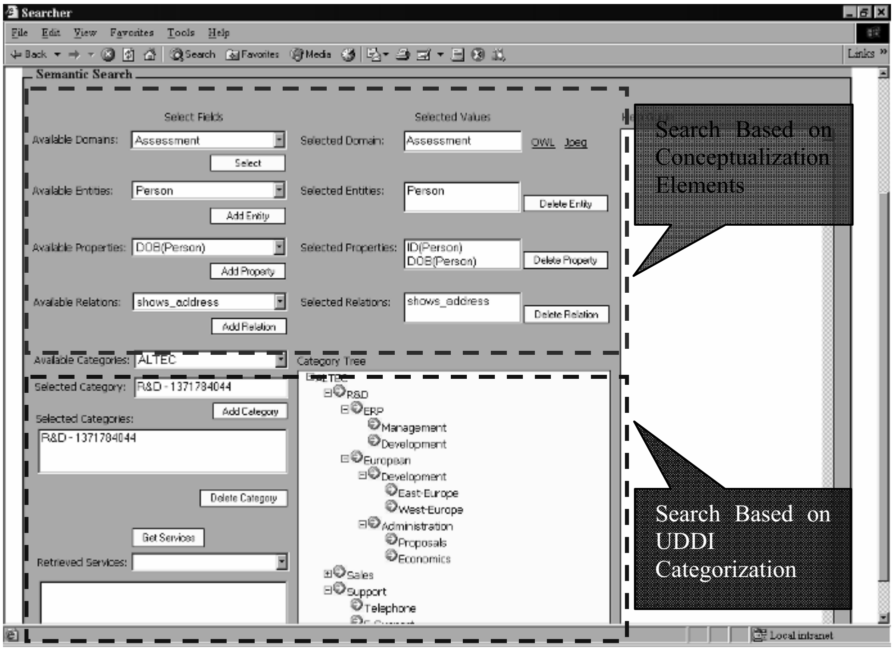
Inside the Service Publication Profile we have also included the elements of the Service Model (Inputs, Outputs and Effects). These elements that are parts of certain conceptualizations carry also semantic descriptions that are also registered to the UDDI registry.

These semantic descriptions of the web service leverage the semantic discovery of the web services reducing the errors of incorrect usage of the web services. This is considered essential for effective SOAI solutions regarding the number of the web services required to support complex integrations and the complexity of each Web service.

5 Extending UDDI Registries to Support Semantics

In order to include semantic meta-information to UDDI registries we have extended both the UDDI and OWL-S profiles. Our work was based on M.Paolucci work in this domain and also some recommendations from [16]. The ONAR Service Publication Profile is based on OWLS profile syntax with some additions that can be summarized (further information can be found on [5]) as follows:

a) Enhancement of the OWL-S input and OWL-S output class in order to define the related classes and relations.
b) Enhancement of the OWL-S Service Category and Service Parameter is order to exploit all the UDDI description capabilities
c) Additional Models to support OWL-S concepts like Input, Output, Effect, Actor and Service Parameter



Available Domains: Assessment
Selected Domain: Assessment OWL Jpeg
Available Entities: Person
Selected Entities: Person
Available Properties: DOB(Person)
Selected Properties: ID(Person) DOB(Person)
Available Relations: shows_address
Selected Relations: shows_address
Available Categories: ALTEC
Category Tree
Selected Category: R&D - 1371784044
Selected Categories: R&D - 1371784044
Retrieved Services:

```
ALTEC
  R&D
    ERP
      Management
      Development
    European
      Development
        East-Europe
        West-Europe
      Administration
        Proposals
        Economics
  Sales
  Support
    Telephone
```

Figure 5. ONAR Service Publication Profile

This integration between the UDDI and OWL profile leverage the search capabilities that can be based on concepts of conceptualizations. We have developed a search facility (Figure 5) that enables the user to choose semantic filters in the level of class properties or relation between classes apart from the typical UDDI inquiry. The search can use both of the techniques lead to more accurate results.

6 Conclusions and Future Work

The large number and the complexity of the web services that support SOAI create the necessity for semantic description in order to facilitate the integration process. Web services and WSDL should not be considered as panacea just because they are more descriptive that older RPC (remote procedure call) methods.

The web services that support an integration case should be handled by a frame-work, as otherwise the maintenance of the web services will be inefficient. We are currently working on a solution to the problem of SOAI that will utilize the technologies of the semantic web and modify and apply them to the field of Application Integration.

In our future plan we intend to enrich even more the expressiveness of the descriptions of the web service architecture and integrate it more efficiently with the conceptual schema. We will continue to extend the elements of OWL and adjust them to Application Integration in order our framework to fully exploit the capabilities of Semantic Web.

We have implemented some test cases in order to evaluate the capabilities both of our methodology and framework. Our work future results and achievements will continuously update the content of our site[5].

References

1. D. Tektonidis, A. Vontas, U. Hess, J. Meschonat “Handling the Shop-Floor of an Industry through ERP systems – A functional Integration Model”, International Conference on E-business & E-Work 2002, Prague 2002
2. M. Themistocleous, Z. Irani, R. O’Keefe & R. Paul “ERP Problems and Application Integration Issues: An Empirical Survey” Proc. of the 34th Hawaiian International Conference on Systems Science, Hawaii Big Island 7-10 January 2001, USA, IEEE 0-7695-0981-9/01
3. David Linthicum “Next Generation Application Integration: From Simple Information to Web Services”, Addison-Wesley 2003, ISBN: 0201844567, USA
4. W3C, Semantic Web Description, 2001, available at http://www.w3.org/2001/sw/ , as accessed in 10 Nov. 2002
5. ONAR, Ontologies based Enterprise Application Integration Framework, 2004, available at http://research.altec.gr/ONAR , as accessed in 10 Oct. 2004
6. Li min Liu & M. Halper - “Incorporating semantic relationships into an object oriented database system” Proc. of the 32th Hawaiian International Conference on Systems Science, Hawaii Big Island 5-8 January 1999, USA, IEEE 0-7115-1405-

9/99Baldonado, M., Chang, C.-C.K., Gravano, L., Paepcke, A.: The Stanford Digital Library Metadata Architecture. Int. J. Digit. Libr. 1 (1997) 108–121

7. Chia-Chu Chiang - "Wrapping legacy systems for use in heterogeneous computing environments", Information and Software Technology, Volume 43, Issue 8, July 2001, Pages 497-507
8. O. Corcho, M. Fernández-López and A. Gómez-Pérez "Methodologies, tools and languages for building ontologies. Where is their meeting point?" , Data & Knowledge Engineering, Volume 46, Issue 1, July 2003, pages 41-64
9. V. Ratnakar, and Y. Gil, "A Comparison of (Semantic) Mark-up Languages", Proceedings of the 15th International FLAIRS Conference, Special Track on Semantic Web, Pensacola, Finland , May 2002.
10. OWL 2003, W3C (2001b) OWL 1.0 Syntax Specification, available at http://www.w3.org/TR/owl-ref/ , as accessed in 15 May, 2003
11. 11. M. Mecella & C.Batini "Cooperation of Heterogeneous Legacy In-formation Systems: a Methodological Framework" Proceedings of the 4th International Enterprise Distributed Object Com-puting Conference, Makuhari, Japan, 2000
12. 12. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. "Ontology-based Integration of In-formation - A Survey of Existing Approaches", In: Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, USA, pages 108-117.
13. M. Paolucci, T. Kawamura, T. R. Payne, K. Sycara. "Semantic Matching of Web Services Capabilities", International Semantic Web Conference (ISWC), 2002, pp 333-347
14. M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Importing the Semantic Web in UDDI. In Proceedings of E-Services and the Semantic Web Workshop, 2002
15. DARPA, DARPA Agent Markup Language Web Services (DAML-S) Version 0.7 Specification, 2002, available at http://www.daml.org/services/daml-s/0.7/ , as accessed in 10 Nov. 2002
16. Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", 14th International Workshop on Research Issues on Data Engineering, Boston, USA , March 28-29, 2004.

A Framework for Semantic Enterprise Integration

Saïd Izza, Lucien Vincent and Patrick Burlat

Laboratoire G2I, Ecole des Mines de Saint-Étienne, 158 Cours Fauriel, Saint-Étienne, France, izza@emse.fr, vincent@emse.fr, burlat@emse.fr

Summary. Nowadays, Enterprise Integration (EI) constitutes a real and growing need for most of enterprises, especially for large and dynamic ones. It constitutes the main approach to deal with heterogeneity, and particularly to deal with semantic heterogeneity, which it forms the crucial problem of EI. This last problem is not correctly addressed by today's EI solutions that focus mainly on the technical and syntactical integration. Addressing the semantic aspect will promote EI by providing it more consistency and flexibility. Some efforts are suggested to solve the semantic problem at different level, but they are still not mature. Here, solutions based on service-oriented architectures and semantic web are promoted as nirvana of the integration problem, and they are being actively researched. The paper will outline the potential role that semantic web technologies can offer in solving some key challenges such as semantic mediation. In particular, the paper will propose a flexible approach called ODSOI (Ontology-Driven Service-Oriented Integration) that aims to extend the state-of-the-art in EI, especially in EAI (Enterprise Application Integration), in order to address the semantic problem.

1 Introduction

Today change, which constitutes the only certainty in business must be dealt with in order to provide more agility and flexibility for enterprises, especially for large and dynamic ones. For this purpose, several solutions, typically known as enterprise application integration (EAI) technology, have emerged as a sub-field of enterprise integration (EI). In essence, EAI technologies provide tools to interconnect multiple and heterogeneous enterprise information systems (EIS) such as ERP (Enterprise Resource Planning) systems, CRM (Customer Relationship Management) systems, SCM (Supply Chain Management) systems, and legacy systems. The most difficulty of this interconnection is due to the fact that the integrated systems were never designed to work together.

More recently, Web Services (WS) have emerged with the advent and the evolution of the Internet and they provide a set of standards for EAI. They constitute the most important instantiation of the SOA (Service-Oriented Architecture) model. They can be deployed inside (in the context of EAI) and also outside (in the context of B2B) of the enterprise. Even if WSs are not fully mature, they seem to become the linga franca of EAI. This will notably make integration simpler and easier through using web protocols and standards. Today, some new integration products based on WS standards exist and will certainly replace in the near future the proprietary solutions that are the traditional EAI systems.

Despite the whole range of available tools and widespread standards adoption, the main goal of EAI, which is the semantically correct integration of EISs, is not yet achieved [3]. Indeed, EAI still provides technical and syntactical solutions but does not address correctly the semantic problem, even if there exist some current initiatives for semantic integration, especially those around SOA (Service-Oriented Architecture) such as OWL-S, METEOR-S and WSMF that are still immature. So, dealing with semantic application integration problem is still open and it constitutes a big challenge that must be addressed, especially in the specific and complex context of our industrial project, which is called MiMIS (Microelectronics Manufacturing Information System).

Semantic integration and interoperability constitutes the main issue in order to overcome the semantic problem. It constitutes the main topic of the research work presented here. This latter is motivated by the observation that we can not rely on the existing integration and interoperability approaches to solve the application integration and interoperability problems in the context of large and dynamic enterprises, where scalability and flexibility are needed. EISs must adapt at deployment and run-time in order to integrate their functionality and also in order to dynamically interoperate with other EISs.

This paper will propose an approach in order to address the semantic integration in the context of EAI in general and in the context of MiMIS project in particular. Our approach is called ODSOI (Ontology-Driven Service-Oriented Integration) and is ontology-based and service-oriented. In the rest of this paper, we will firstly present the integration problem; secondly, we will briefly review the related works. Finally and before concluding, we will describe some important aspects of our work attempting to provide a flexible solution for the integration problem.

2 Integration Problem

Enterprise information system (EIS) is defined as an enterprise application system or an enterprise data source that provides the information infrastructure for an enterprise. Typically, an EIS can take many different types including batch applications, traditional applications, client/server applications, web applications, application packages, relational databases and so on. These systems are often materialized in enterprise reality in form of relational databases, ERP (Enterprise Resource Planning) systems, CRM (Customer Relationship Management) systems, SCM (Supply Chain Management) systems and legacy systems.

An appropriate characterization of EISs in context of EAI is that these systems are HAD (heterogeneous, autonomous, distributed) [3]:

- *Heterogeneous* means that each EIS implements its own data and process model.
- *Autonomous* refers to the fact that EISs run independently of any other EIS.
- *Distributed* means that EISs locally implement their data model, which they generally do not share with other EISs.

The consequence of the characteristics above is that EISs are standalone software entities which constitute what it is often labeled “islands of information” or “islands of automation”. Of course, none of the islands contain enterprise-wide processing and information, which makes it then necessary to integrate the islands in order to create unified enterprise information and process system. In this case, any form of integration of these islands (EISs) must happen outside of the involved islands (EISs), by using integration systems such as EAI systems. This integration consists then in interconnecting the interfaces of each EIS using technologies supported by the integration systems such as queuing systems, file systems, databases or remote invocations.

Despite the importance of all the problems described above, we focus only on the heterogeneity problem, precisely on the semantic heterogeneity problem, which is the hard problem of EI in general, and EAI in particular [3].

Furthermore, the integration problem is more complicated by our industrial context, which is called MiMIS (Microelectronics Manufacturing Information System) project. This project concerns a large and complex enterprise in the multidisciplinary microelectronics area. This industrial context is mainly characterized by several and heterogeneous knowledge domains with many heterogeneous and autonomous EISs, and that needs sophisticated semantic mediation in order to achieve the semantic integration.

3 Related Research

The integration problem is not new and it still constitutes a hard and complex domain addressed by several communities such as database community, software engineering community, data mining community, enterprise information system community and so on. There have been many standards and technologies that have been proposed for data and application integration and interoperability. In this section, we will briefly and without any exhaustiveness present some important the related works.

Semantic interoperability aims to solve all the semantic heterogeneities that can be occurring between EISs. Goh identifies three main categories of semantics conflicts in the context of data integration that are confounding conflicts, scaling conflicts and naming conflicts [6]. Traditionally, such semantic conflicts have typically been dealt with at design time through careful schema design, through the handcrafting of interoperability gateways (direct translator), or by imposing a centralized database, a neutral (canonical model) or a standard mechanism for interchange (e.g., STEP, IGES, PSL, ebXML, RosettaNet). Although these traditional approaches have been successful in well-understood/homogenous/static interchange environment, each of these approaches are inappropriate in the context of large and dynamic enterprises [4]: the schema design solution fails due to the rapidly changing nature of the needed exchanges; the handcrafting translators solution fails because it does not scale to large number of EISs; and the last solution fails due to the impossibility to agree on some standards, neutral format or a unique database.

These semantic conflicts concern not only the data, but concern also behavior of EISs. So, we have also semantic behavioral heterogeneities that can be occurred when EISs invoke each other, and that are referred as invocation heterogeneity by [3]. There have been many traditional approaches for developing semantic behaviour integration, ranging from centralized integration and client-sever integration to distributed integration. Only the last approach can be appropriate in the context of dynamic and large enterprises due to the fact that it is scalable. Traditionally, solutions based on workflow or BPM engines are used to implement the distributed integration. However and with the advent of Internet, the current trend for this behavioural integration approach is web-based integration [15]. This latter is based on web services and provides loose integration of applications with the help of some standards such as XLANG, WSFL, BPEL4WS [2].

A solution to the problems of semantic heterogeneity should equip HAD systems with the ability to share and exchange information and services in a semantically consistent way. Ontologies can be used to capture the semantics of application systems. They seem to be a suitable technology of semantic web in order to solve the semantic integration and interoperability problem. According to Gruber, an ontology is defined as an explicit and formal specification of a conceptualization [7]. The role of ontologies is becoming important in order to realize the vision of semantic web. For this purpose, several languages (e.g., RDF, DAML, DAML+OIL, OWL) and methodologies (e.g., Methontology, Uschold) have emerged. OWL which is XML-based, constitutes a recommendation of W3C and can be applied in multiple situations relyed to EI [19].

The use of ontologies as a possible solution to semantic integration and interoperability has been studied over the last decade. Wache reviewed several ontology-based approaches and architectures that have been proposed in the context of data integration and interoperability [20]. A good example of such architectures and systems in the context of data integration are COIN (Context Interchange) project [6], OBSERVER (Ontology Based System Enhanced with Relationships for Vocabulary Heterogeneity Resolution) project [11], and so on. All these work are not concerned about the mediation in the context of SOA.

In addition to the listed related works above, there are some other works that are addressing the WS viewpoint such as Active XML from GEMO project [1] and SODIA (Service-Oriented Data Integration Architecture) from IBHIS (Integration Broker for Heterogeneous Information Sources) project [[17]]. Active XML extends XML language by allowing embedding of calls to WSs. SODIA is an implementation of Federated Database System in the context of WSs. These work do not support any mediation services.

Furthermore, in the context of application and process integration, some important initiatives and works exist, especially the initiatives around the concept of semantic web services [16]. These initiatives are mainly OWL-S [19], WSMF [5] and METEOR-S [12]. OWL-S provides an ontology markup language in order to semantically describe capabilities and proprieties of WSs. WSMF and METEOR-S are initiatives that provide frameworks in order to support the concept of semantic web service which are defined WSs with a formal description (semantics) that can enable a better discovery, selection, composition, monitoring, and interoperability. WSMF are based on the extension of DAML-S and OWL-S whereas METEOR-S is an effort, which provides semantic web services through the extension of existing standards such as WSDL and UDDI. But, most of these efforts do not provide mature concepts for mediation in the context of EAI.

Finally, none of the related works provide a flexible and unified framework in the context of SOA to integrate semantically data, application and processes. The approach we propose in the following is based on the use of semantic WSs and ontologies. It is based on some principles that improve the flexiblity of integration process in the context of large and dynamic enterprises in general, and in the context of MiMIS project in particular.

4 ODSOI Approach

This section succinctly describes some important characteristics of our approach called ODSOI (Ontology-Driven Service-Oriented Integration), where some initial aspects were described in [8]. It relies on the use of both ontologies and WSs technologies, and that aims to extend the state-of-the-art in EAI in order to address the semantic problem.

4.1 General Principles

First of all, ODSOI approach is a solution to the information system integration problem. This means that our approach addresses the heterogeneity problem by providing an *ontology-driven* integration solution that is based on ontology mediation. Indeed, our approach is *service-oriented* (based on SOA) since it uses WSs for integrating EISs (unification principle). The result architecture integration that we suggest is called ODSOA (ODSO Architecture). This latter extends SOA with a semantic layer that aims to enhance service mediation in the context of EAI. Furthermore, our approach can support *dynamic oriented integration.* This latter means that the binding services of target EISs can be performed at run time according to special binding rules, contrarily to a static integration solution where an integration scenario predefines every detail of potentially connectable EISs. The reasons of this choice are that the dynamic solution is sometimes more flexible and scalable form of integration than static one. In addition to this, the integration scenario becomes simpler and the configuration EIS interfaces can be changed and new systems can be added easily, even while the EAI system is running.

In addition to this, our approach is based on the urbanization principle, which urbanizes the exposed services and also the used ontologies in the sense that these latter are regrouped in districts and areas according to some urbanization principles inspired from those introduced by [10].

4.2 Global Architecture

The ODSOA concept provides a unified framework in order to integrate EISs. In this framework, three main types of services are defined: data-services, functional-services and business-services. These different types can respectively address data, application and process integration.

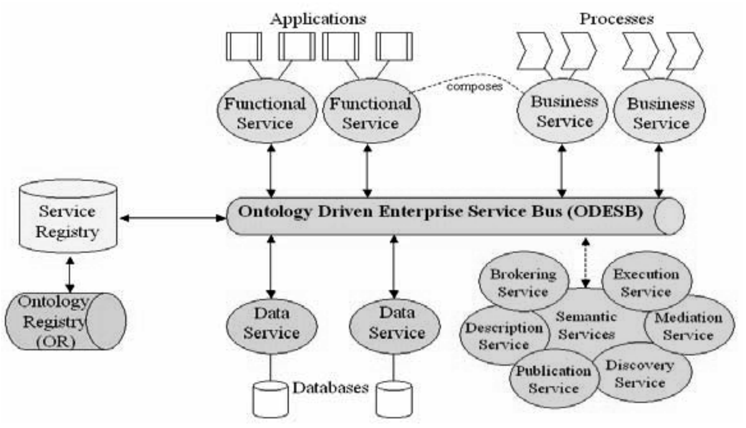


Figure 1. Global view of ODSOA Architecture

Data-Services (DS) are services that expose data sources (EDS – Enterprise Data Sources) as services. Functional-Services (FS) are services that expose application systems, fundamentally functional systems (EAS – Enterprise Application Systems) as services. Business-Services (BS) are defined as the combination of the above services in order to expose business processes (EPS – Enterprise Process Systems) as services.

Figure 1 which is a particular SOA recapitulates these important types of services. Indeed, there are, of course, some other important semantic services that are mainly brokering service, description services, mediation services, publication services discovery services and execution services. Some of them will be described below.

A cross section of the integration bus (also called ODESB – Ontology-Driven Enterprise Service Bus) (Figure 2) shows many concentric standard layers such as Transport layer, Exchange layer, Description layer, Registry layer and Transversal layer. In addition to these standard and currently existing layers, we suggest to adopt, in a similar way as semantic web service concept [16], an other layer, called semantic layer and which contains two sub-layers that are ontology layer and integration layer.

The ontology layer aims to describe the three fundamental types of services described above using specific descriptions such as DSD (Data Service Description) for DSs, FSD (Functional Service Description) for FSs, and BSD (Business Service Description) for BSs. All these descriptions exploit an ontology that is a specialization of OWL-S (referred to as OWL-S+) and which referes to some specific domain ontologies. The use of OWL-S+ is motivated by the fact that it constitutes a semantic orientation to the description of WSs, contrary to WSDL (Web Service Description Language) which provides only a description based on a syntactic orientation.

The integration layer provides mechanisms to perform the resolution of semantic differences (semantic mediation). In the next sub-section, the services of semantic layer, which are the important ones in our approach, will be developed.

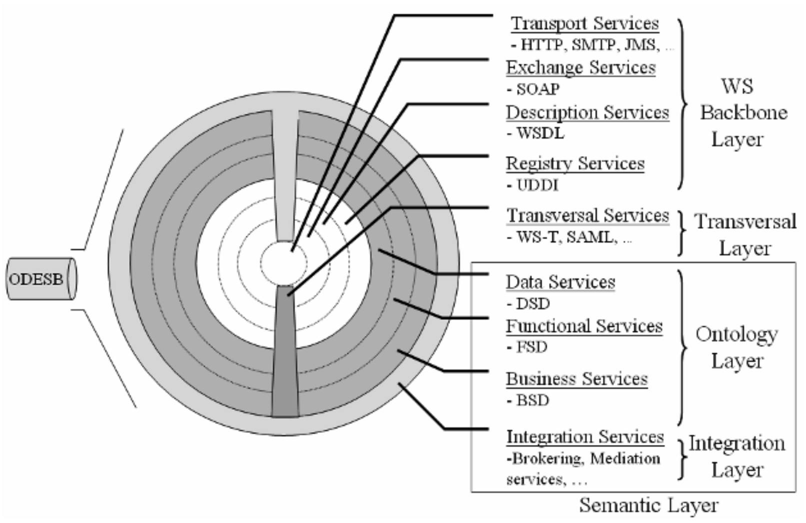


Figure 2. Cross Section of the ODESB Bus

4.3 Main Semantic Services

Semantic services are the main services that address the semantic problem. They include description services and mediation services that will be described below.

4.3.1 Description Services

The principle of ODSOA is based on the semantic services that exploit some formal ontologies in order to define the semantic description of services. For this purpose, two kinds of ontologies are used and are: OWL-S+ ontology and specific ontologies. Here, we will focus only on the latter, and we will show below how they are structured.

In the context of information integration, two fundamental approaches can be selected to achieve semantic integration: shared ontology approach and non-shared (local) ontology approach [20]. In general, we use an hybrid approach that combines the two approaches and that consists to map local ontologies onto a shared ontology [20]. For our purpose, we have chosen a variant of the hybrid approach. This can be motivated by the fact that in a large enterprise of autonomous EISs, the definition of a single set of ontology standards that are suitable for everyone is nearly impossible. Indeed, this can also be motivated by the difficult leveraging when adopting multiple independent ontologies. In our approach, we have defined three major types of specific ontologies that are: information or data-based ontologies, behavior or functional–based ontologies and process or business-based ontologies (Figure 3). These ontologies are used in service descriptions (precisely refered by OWL-S+) such as DSD (Data Service Description), FSD (Function Service Description) and BSD (Business Service Description) to explicit the semantics of respectively DSs, FSs and BSs.

Data-based ontologies are the most basic ones. They provide semantic description of the data (DSs). These ontologies are required in all cases, no matter if we leverage functional-based or business-based ontologies. This means that, data integration is a precondition for any other integration type.

Functional-based ontologies define semantic description around functions that provided by the multiple EISs (FSs) and that can be remotely invoked. These ontologies are generally required in order to provide a better reuse of functionalities.

Business-based ontologies define semantic description around coordinating business processes. These ontologies are generally required in order to take into account business processes and workflows.

In addition to this, description services are based on the context of a service (service-context) which is defined by a set of ontologies related to the concerned service. This service-context is also called local ontology which means that there are several ontology levels. This is generally appropriate within a large and dynamic enterprise with several different business domains such as the studied society. For our purpose, three ontology levels have been identified: local level, domain level and global level (Figure 3).

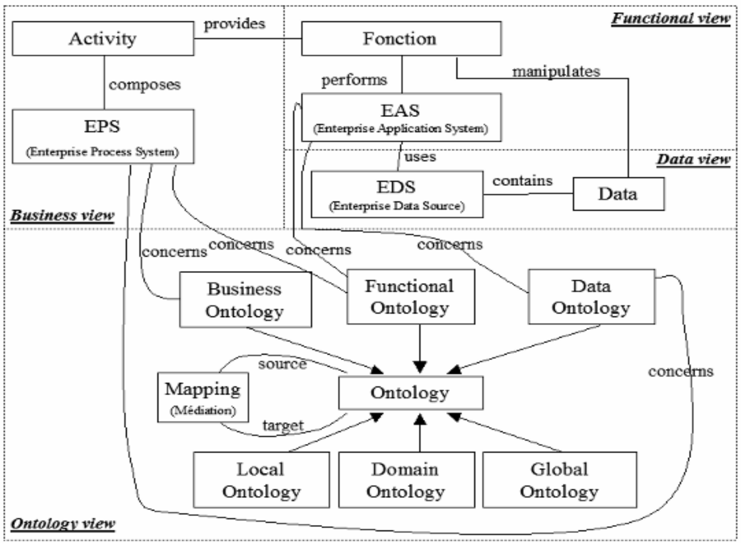


Figure 3. ODSOI ontology taxonomy

In essence, local ontologies concerns services, whereas domain ontologies concern the generalization of local ones that belong to the same domain (Production, Metrology, Packaging, etc.) and they can serve in aligning some local ontologies. At last, global ontology is considered as generalization of domain ontologies. It is the root of the ontology hierarchy, and it can serve in aligning domain ontologies and may also be used in B2B integration that can constitute a natural extension of our approach.

The ontology taxonomy defines somewhat a clustering process which is firstly used in [18]. This clustering is an important concept which defines the ontology structuration within an enterprise and can be used in order to master the ontology construction and evolution. We call this clustering: ontology urbanization and it takes an important role in our integration approach.

4.3.2 Mediation Services

Mediation services are generally invoked by brokering service in order to perform matching or resolution of semantic heterogeneity between services. They exploit the descriptions provided by the description services described above.

Since we use an hybrid ontology approach, this require the integration and mediation of ontologies which are performed by Ontology Mediation Services (OMS) and that are based on ontology mapping [9]. This latter is the process whereby two or more ontologies are semantically related at conceptual level. According to the semantic relations defined in the mappings, source ontology instances can then be transformed (or matched with) into target ones.

In addition to OMS and according to the above different fundamental types of services, we can distinguish mainly three other types of mediation services: Data Mediation Service (DMS), Functional Mediation Service (FMS), Business Mediation Service (BMS). These mediation services aims to mediate respectively between DSs, FSs, BSs and they are based on OMS that match and mediate between different ontologies. To be performed, Mediation Services can exploit two particular utility services that are inference service and matching service. These particular services can be respectively supported by academic or commercial inference engine and matching tool. For the initial prototype that is ongoing, we decide to use Racer engine [14] and OLA (OWL Lite Alignment) matcher [13] that seems be appropriate to our approach.

In order to illustrate our mediation process using the ODSOI approach, we will focus in this paper only on one simple type of instance mediation, which is data mediation. Our data mediation approach is based on an OWL inference, and precisely on OWL instance reasoning, which reasons on both concepts (TBox) and instances (ABox) and which performs automated transformations of data. In our approach, the data that is exchanged between services must be XML document with XSD schemas. These XML schemas must be mapped to our OWL ontologies (specific ontologies) from the initial stages, precisely at the service description process (performed by Description Service, which is described above). The transformation of data from one context (source service context) to another context (target service context) is accomplished with an inference service, which is called Data Mediation Service. This latter, is based on an inference engine such as Racer [14], and is able to dynamically reclassify data in OWL format and then transform the source data to the target data with the help of the informations contained in the ontologies. A similar behavior mediation is also defined. It concerns both function and process mediation and it is based on an OWL instance reasoning tool in order to integrate dynamically FSs and BSs. This inference-based mediation approach provides more flexibility due to the fact that it is based on the semantics.

4.4 Overview of the Integration Process

ODSOI approach provides a flexible and uniform framework to integrate EISs at different levels: data, application and process integration level that are provided by three specific services or sub-frameworks that are respectively ODSODI (ODSO Data Integration) framework, ODSOAI (ODSO Application Integration) framework and ODSOBI (ODSO Business Integration) framework. Figure 4 illustrates the general principle of the three components of the global framework (ODSOI framework).

DSs can be integrated with the ODSODI framework by using data ontology mediation, whereas FSs can be integrated in similar way by using both data and functional ontology mediation. At last, in case of process integration, BSs can be integrated by using data, functional and business ontology mediation. These three sub-frameworks define a stratification so that each level include its inferior levels. Since we can not detail the different sub-frameworks in this paper, so we just give a generic synopsis about the integration process.

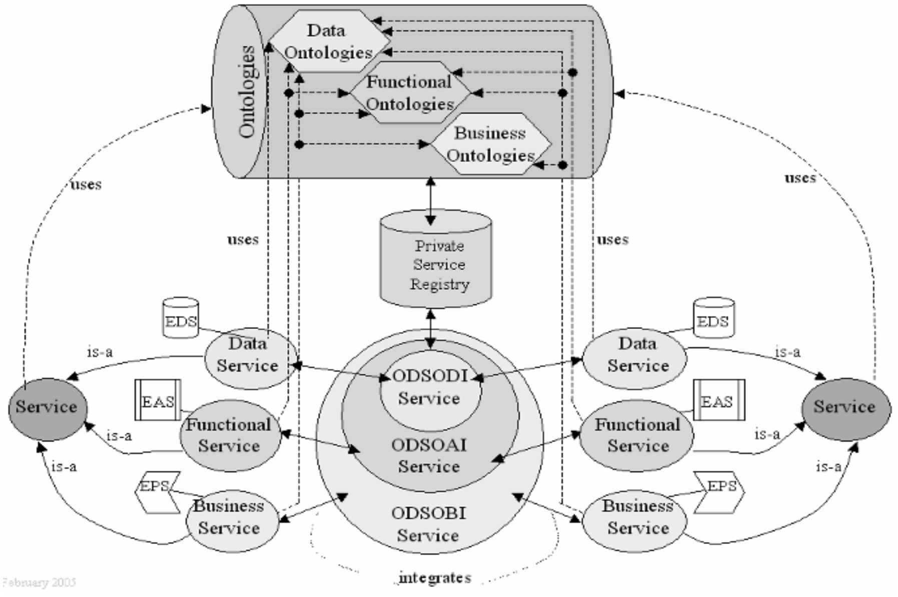


Figure 4. General vision of ODSOI framework

The heart of integration process is realized by mediation services with the help of a Brokering Service (service broker). This latter orchestrates different semantic services such as integration service, discovery service, mediation and execution service. The generic scenario of the integration process is shown on Figure 5.

The integration process starts once the WSs have been described (semantic description) by using description services. After that, they are published (semantic publication) in enhanced private UDDI registries (by description services) and then they can be discovered (semantic discovery) by discovery service in order to carry out the realization of a given task modeled as a service query (that corresponds to a user or to a specific service query) by the integration service. The discovery service can use mediation service in order to perform the semantic matching. The invoked mediation services exploit an inference engine that calculate rapprochement between ontology concepts and then between the involved characteristics of services. Once the desired WSs have been discovered, they are also mediated in order to resolve the semantic heterogeneity differences by the mediation service. Finally, the mediated services are executed by the execution service and can invoke the integration service, which can then perform another similar loop of the integration process. Furthermore, the above process includes dynamic integration, which is made possible by some binding rules contained in the integration rule base. These rules, which are extracted from business ontologies, exploit mainly the characteristics of services such as preconditions, effects and so on.

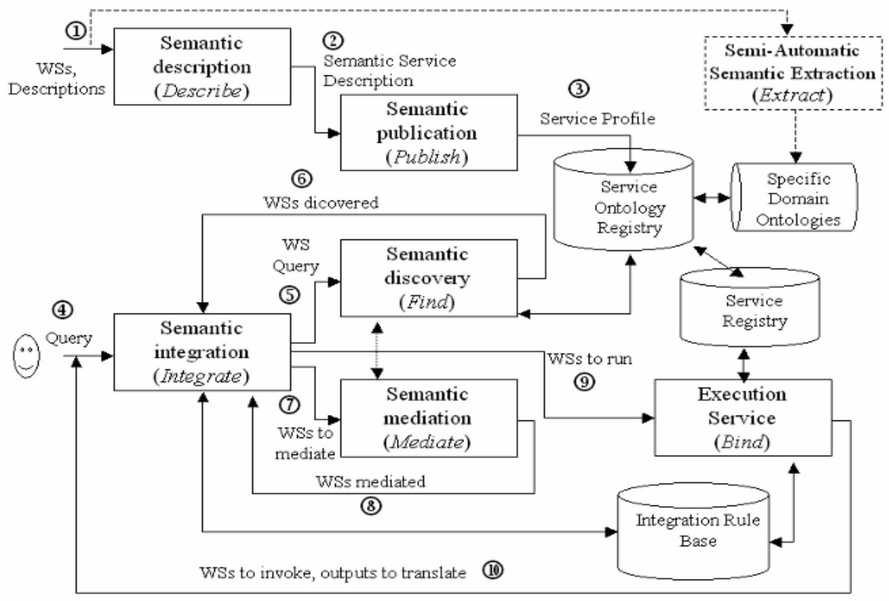

①
WSs, Descriptions
Semantic description (*Describe*)
② Semantic Service Description
Semantic publication (*Publish*)
③ Service Profile
Semi-Automatic Semantic Extraction (*Extract*)
Service Ontology Registry
Specific Domain Ontologies
⑥ WSs dicovered
WS Query
⑤
Semantic discovery (*Find*)
Service Registry
④ Query
Semantic integration (*Integrate*)
WSs to run
⑨
⑦ WSs to mediate
Semantic mediation (*Mediate*)
Execution Service (*Bind*)
WSs mediated
⑧
Integration Rule Base
WSs to invoke, outputs to translate ⑩

Figure 5. Generic integration process

5 Conclusion

The semantic integration of enterprise information systems is a crucial problem that can concern data, applications and processes. This problem needs semantic mediation and is best resolved in the context of service oriented architectures.

This paper has focused on proposing a flexible and unified approach for enterprise application integration that exploits both ontology mediation and web services. This approach called ODSOI (Ontology-Driven Service-Oriented Integration) aims to extend the current web services stack technology by a semantic layer offering semantic services that can define the service semantics and also perform semantic mediation in the context of EAI. A typology of services and also of ontologies have been suggested, and a global vision of the integration framework is described. We intend to materialize in the future our work with an initial prototype addressing the data aggregation problem within MiMIS project, which provides us an operational environment in order to verify the pertinence and validity of our approach.

References

1. Abiteboul S., Benjelloum O., Milo T., Web Services and Data Integration. INRIA, (2002).
2. BPEL4WS, http://www-128.ibm.com/developerworks/ library/ws-bpel/, (2004).
3. Bussler C., The Role of Semantic Web Technology in EAI. In the Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, (2003).
4. Cui Z., Jones D., O'Brien P., Issues in Ontology-based Information Integration, Proceedings of the IJCAI'0147 (141-146), (2001).
5. Fensel D., Bussler C., The Web Service Modeling Framework WSMF. In Electronic Commerce Research and Applications, 1(2). Elsevier, Sciences B. V, (2002).
6. Goh C. H., Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources. PhD Thesis, MIT, (1997).
7. Gruber R. T., Towards Principles for the Design of Ontologies Used for Knowledge Sharing. KSL, Stanford University, (1993).
8. Izza S., Vincent L., Burlat P., A Unified Framework for Application Integration. ICEIS' 05 USA, (2005).
9. Kalfoglou Y., Schorlemmer M., Ontology Mapping: The State of The Art. The Knowledge Engineering Review, Vol. 18(1), 1-31. Cambridge University Press, (2003).
10. Lonpégé C., Le projet d'urbanisation du système d'information, Dunod, (2002).
11. Mena E., Kashyap V., Sheth A., Illarramendi A., OBSERVER: An approach for query processing in global information systems based on interoperability across pre-existing ontologies. In Proceedings 1st IFCIS International Conference on Cooperative Information Systems, Brussels, (1996).
12. METEOR-S, http://lsdis.cs.uga.edu/Projects/METEOR-S, (2004).
13. OLA, www.iro.umontreal.ca/~owlola/alignment.html., (2004).
14. Racer, http://www.racer-systems.info, (2004).
15. Roszko A., Kontogiannis K., Revolutionizing EAI with Enterprise Web Services. STEP'03, IEEE International Conference on Software Maintenance, ICSM'03, (2003).
16. SWSI. Semantic Web Services Initiative, www.swsi.org, (2004).
17. Turner M., Zhu F., Kotsiopoulos I., Russel M., Bennett K., Brereton P., Keane J., Rigby M., Using Web Service Technologies to Create an Information Broker: An Experience Report. 26th International Conference on Software Engineering (ICSE'04). IEEE Computer Society. pp 552-561, (2004).
18. Visser P., Tamma V., An experience with ontology clustering for information integration. In Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration, Sweden, (2004).
19. W3C, www.w3.org, (2004).
20. Wache H., Vögele T., Visser U., Stuckenschmidt H., Schuster G., Neumann H., Hübner S., Ontology-Based Integration of Information - A Survey of Existing Approaches. Intelligent Systems Group, Germany, (2001).

A Framework to Support Interoperability among Semantic Resources

Celson Lima[1], Catarina Ferreira Da Silva[1,2], Chan Le Duc[1] and Alain Zarli[1]

[1] CSTB, Centre Scientifique et Technique du Bâtiment, Routes de Lucioles BP209, 06904 Sophia Antipolis, France {c.lima, catarina.ferreira-da-silva, leduc, alain.zarli}@cstb.fr
[2] LIRIS, Laboratoire d'InfoRmatique en Image et Systèmes d'information, CNRS/Université de Lyon 1, 43 boulevard du 11 novembre 1918, F-69622 Villeurbanne cedex, France. catarina.ferreira.da.silva@liris.cnrs.fr

Summary. This paper presents the framework to support the interoperability among *Semantic Resources*[1] (SRs) proposed by the FUNSIEC project. FUNSIEC aims at studying the feasibility of building and maintaining an Open Semantic Infrastructure for the European Construction Sector (OSIECS). FUNSIEC adopted a layered approach where the SRs are mapped at the meta-schema and schema levels. This process produces the respective OSIECS meta-schema and schema, which support the design of the OSIECS kernel – the heart of the OSIECS infrastructure. This paper presents and discusses the elements involved in FUNSIEC's work (e.g. framework, architecture, methodology). Some conclusions and future work complete the paper.

1 Introduction

The FUNSIEC project touches a dynamic, constantly evolving subject: the meaning of things or in a word, their **semantics**. Manufacturing industry has faced a worthwhile challenge in the last two decades, finding ways to communicate precisely (i.e. without ambiguity) and indeed good technical results have been achieved (e.g. STEP-related works, EDI, and more recently the ebXML initiative). The construction industry faced the same challenge and found similar tools to cope, aided by the development and promotion of the Industry Foundation Classes (IFC model & tools) established by the IAI collaborative initiative [13]. Other projects and initiatives have worked in the same domain and produced related contributions to raise awareness in the whole sector, such as the LexiCon from Stabu [14], the e-COGNOS IST project [5], the CEN/ISSS eConstruction Workshop [14], to name just a few.

[1] Semantic Resource is an expression coined in the SPICE project which refers all ontology-similar entities, such as taxonomies, dictionaries, thesauri, etc.. Two arguments supported this choice: there is no consensus about what ontology really is and there is a myriad of expressions currently available to define ontologies and similar resources.

FUNSIEC is a continuation of some of these efforts aiming at to evaluate the feasibility of building and maintaining an Open Semantic Infrastructure for the European Construction Sector (OSIECS), intended mainly to support the e-business needs of Construction. FUNSIEC targets the current situation where standards (both official and de facto), classification systems, taxonomies, ontologies, and related technologies and tools are used in a complementary or even in a completely isolated way to produce the "desirable but feasible" framework where the previously described elements can be combined in an inter-related and holistic fashion.

OSIECS has been created with the SRs currently available that support e-business practices. Following the recommendation provided by the CEN/ISSS eConstruction Workshop [14] the SRs were selected and the most relevant have been mapped among themselves at meta-schema and schema levels respectively. This mapping is expected to produce OSIECS meta-schema and schema.

This paper focuses on the development of OSIECS, describing the FUNSIEC approach to achieving the goal. The document is structured as follows: section 2 describes the context in which the work is carried out; section 3 presents the FUNSIEC approach to develop OSIECS; section 4 discusses the architecture of the OSIECS kernel; section 5 concludes the paper and points out what is next in FUNSIEC.

2 Context of Work

As previously mentioned, the FUNSIEC approach to developing OSIECS is based on the selection and mapping of the most relevant SRs. The mapping is made at two levels in a semi-automatic way, with experts and supporting software tools creating the *liaisons* among the SRs that form the essence of OSIECS. This is, indeed, a kind of *pivot* providing the necessary bridges to handle semantic queries concerning all the SRs mapped through OSIECS.

2.1 The FUNSIEC Project

The FUNSIEC project focuses on the interoperability of semantic resources through the design of the OSIECS infrastructure. FUNSIEC is not a classical R&D project in the sense that it targets mainly the feasibility of building and maintaining OSIECS at a technical, organisational and business level. As such, the answer must be pragmatic and take advantage of the resources currently available, including public results produced by international initiatives and EC-funded projects. The feasibility study also evaluates the alternatives to *maintain semantic mappings* amongst the available SRs in order to foster complementarities amongst them and favour the emergence of new electronic services (especially those supporting multiple languages). The development of OSIECS is expected be an effective mechanism to provide answers to the business needs related to the use of SRs.

The main elements involved in the work of FUNSIEC are e-business, standards, and semantic resources (the inputs) and the OSIECS infrastructure and the education dimension (the major outputs). All are evidenced in the FUNSIEC

Semantic Experience Centre (FSEC) a kind of hands-on learning experience. The semantic resources are evaluated and mapped among themselves. Standards and recommendations provided by standardisation bodies are included in a thorough way in FUNSIEC. The FSEC is, of course, the key element supporting the education axis in FUNSIEC. The (e)business side of the story is related to the very essence of the feasibility study on the use of SRs. How they can be used effectively and how they can provide business advantage are the questions FUNSIEC seeks to answer.

FUNSIEC intends to be an instrument that promotes e-business practices to the construction community. It aims to become a useful and extensible aid relating to the semantic resources theme, a place where people can obtain educational support. Such support will be provided by the creation of a showroom of SRs available and potentially usable by the construction sector in Europe, including usage scenarios, on-line tutorials and demonstrations. FUNSIEC is not about reinventing the wheel; rather it is oriented towards reuse and adoption/adaptation of what seems best in helping the construction sector to be educated in the meaning, development and use of SRs.

2.2 The FUNSIEC Framework

The FUNSIEC framework contains three domains, namely User, Application, and Resource (Figure 1). The User domain contains the actors involved in the e-business arena. The Application domain holds the e-business areas of application. Finally, the Resource domain offers the SRs used by the application areas and brings in two new elements, namely OSIECS meta-schema and schema. Together these provide the cornerstone supporting the mapping (at different levels) between SRs.

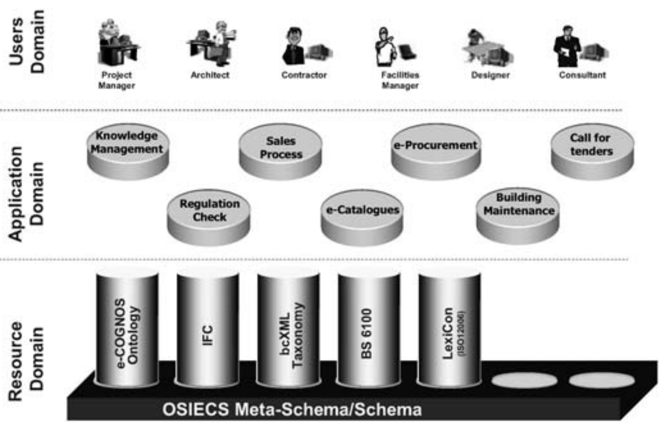


Figure 1. The FUNSIEC framework

The approach to conceiving OSIECS started with the selection of the SRs that form it. Since they are not described in a uniform language, their respective meta-schemas were converted to a single format adopted by FUNSIEC (i.e. the Ontology

Web Language – OWL) allowing (semi)automatic detection and validation of commonalities amongst the meta-schemas. This process produces the OSIECS meta-schema that supports the mapping amongst the SRs in OSIECS. The OSIECS meta- schema is then used to guide the mapping of the entities (i.e., concepts, relations, properties, etc.) defined in each schema of the OSIECS components. The final end result is the OSIECS schema.

3 The FUNSIEC Approach

Beyond the "simple" pooling of linguistic resources, FUNSIEC aims at creating a harmonised environment where each resource is clearly characterised (in terms of content, scope, usage, etc.) and situated in the overall map of resources, where semantic links are created among them. Currently, the lack of visibility together with the absence of agreed standards on a large scale are two of the main factors explaining why end-users and service providers are not using exploiting resources more intensively. Moreover, the creation of consistent linguistic resources is most often a very tedious task for content providers with no guarantee of return for the effort. Certainly a major challenge for FUNSIEC is to overcome these problems and create "critical mass" interest and confidence needed to encourage content and service providers. Therefore, the potential risks of failure, along with the success factors, have been carefully evaluated during this feasibility project.

The FUNSIEC approach to developing the OSIECS infrastructure consists of the following steps: (*i*) identification and selection of the SRs to provide the input for the creation of the OSIECS meta-schema; (*ii*) analysis of structure, syntax, and semantics of the respective meta-schema of the SRs selected in step *i*; (*iii*) design of the OSIECS Kernel, a software tool to help creating the OSIECS meta-schema/ schema; (*iv*) semi-automatic conversion of the meta-schemas for the same format; (*v*) semi-automatic mapping of the converted meta-schemas producing the (*unified*) OSIECS meta-schema; (*vi*) production of the OSIECS schema from the meta-schema; and (*vii*) design the OSIECS infrastructure and putting in place a demonstration scenario. These steps are described in more detail in the next sections.

3.1 The Three Layers Vision in FUNSIEC

All semantic resources follow some underlying meta-schema and schema, even if not always stated explicitly. In order to enable interoperability among different Semantic Resources it is essential to understand beforehand the meta-schemas they follow. By way of illustration only, Figures 2, 3 and 4 show the bcXML meta-schema, part of *bcBuildingDefinitions* taxonomy and a catalogue (bcXML compliant) of products.

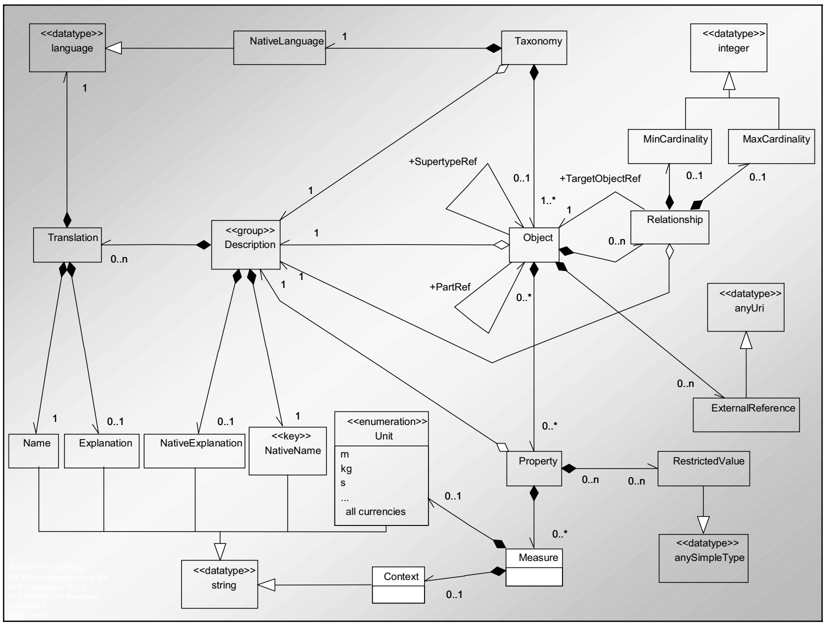
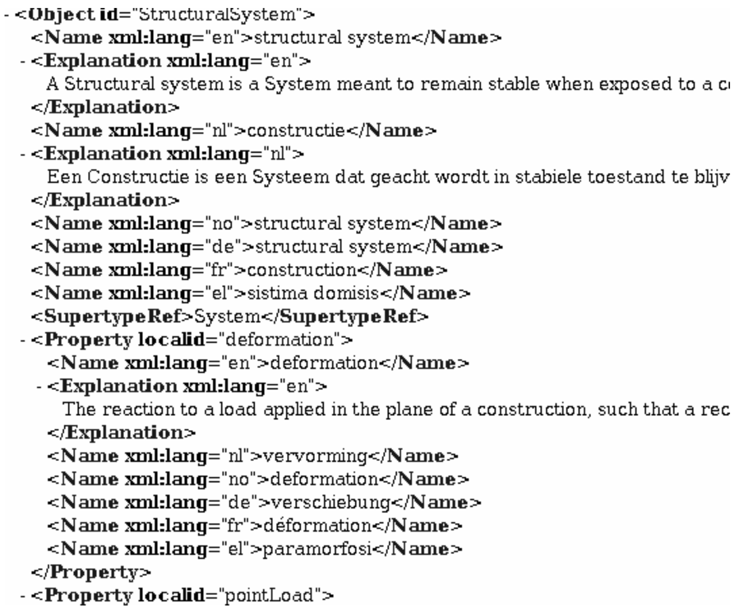


Figure 2. The bcXML Meta-schema

The CEN/ISSS eConstruction Workshop[2] recommends the use of frameworks structured in two levels, namely *meta-schema* and *schema*. FUNSIEC considers a third level in this framework, the *instances*. At the highest level, there are meta-schemas that describe very general and abstract concepts that are necessary when structuring an information model and those that describe broadly sharable ideas usually being specific to building construction. From a top-down perspective, specialisation is the axis transforming a meta-schema into instances and generalisation supports the other way round. The lower levels are more detailed and real things whilst higher levels are more abstract and intangible things.

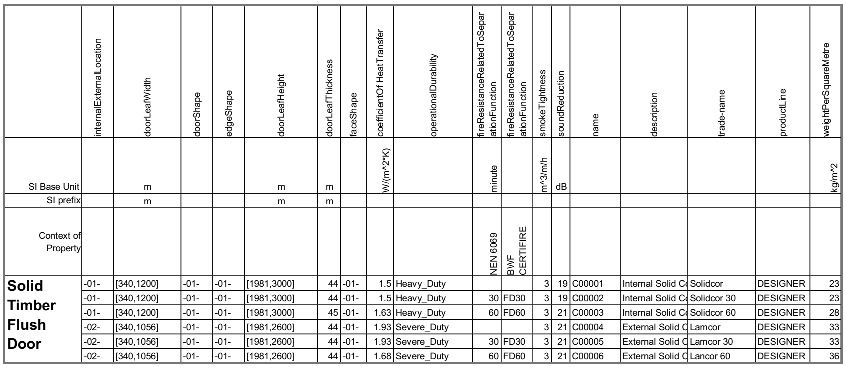
In the second level, the schema represents an agreed structure expressed in some suitable (mostly formal) language able to describe things on different ‘description levels’ (e.g. a taxonomy or an ontology). It should be able to handle both definitions and specifications (types and occurrences) of construction-related products and services. Finally, at the bottom level, **instances** are very specific representations of a schema; for instance a catalogue of products where all the properties that define a given product have the correct values defined (e.g. catalogues of products).

[2] The CEN/ISSS Workshop on eConstruction was a mechanism to facilitate “e-volution” in European Construction by providing an open discussion forum to achieve the required consensus on a number of interrelated specifications needed for outworking of eConstruction.



```
- <Object id="StructuralSystem">
   <Name xml:lang="en">structural system</Name>
  - <Explanation xml:lang="en">
     A Structural system is a System meant to remain stable when exposed to a c
   </Explanation>
   <Name xml:lang="nl">constructie</Name>
  - <Explanation xml:lang="nl">
     Een Constructie is een Systeem dat geacht wordt in stabiele toestand te blijv
   </Explanation>
   <Name xml:lang="no">structural system</Name>
   <Name xml:lang="de">structural system</Name>
   <Name xml:lang="fr">construction</Name>
   <Name xml:lang="el">sistima domisis</Name>
   <SupertypeRef>System</SupertypeRef>
  - <Property localid="deformation">
     <Name xml:lang="en">deformation</Name>
    - <Explanation xml:lang="en">
       The reaction to a load applied in the plane of a construction, such that a rec
     </Explanation>
     <Name xml:lang="nl">vervorming</Name>
     <Name xml:lang="no">deformation</Name>
     <Name xml:lang="de">verschiebung</Name>
     <Name xml:lang="fr">déformation</Name>
     <Name xml:lang="el">paramorfosi</Name>
   </Property>
  - <Property localid="pointLoad">
```

Figure 3. Partial view of bcBuildingDefinitions taxonomy



	internalExternalLocation	doorLeafWidth	doorShape	edgeShape	doorLeafHeight	doorLeafThickness	faceShape	coefficientOf HeatTransfer	operationalDurability	fireResistanceRelatedToSeparationFunction	fireResistanceRelatedToSeparationFunction	smokeTightness	soundReduction	name	description	trade-name	productLine	weightPerSquareMetre
SI Base Unit		m			m	m		W/(m^2*K)		minute		m^3/m/h	dB					kg/m^2
SI prefix		m			m	m												
Context of Property										NEN 6069	BWF CERTIFIRE							
Solid Timber Flush Door	-01-	[340,1200]	-01-	-01-	[1981,3000]	44	-01-	1.5	Heavy_Duty			3	19	C00001	Internal Solid Co	Solidcor	DESIGNER	23
	-01-	[340,1200]	-01-	-01-	[1981,3000]	44	-01-	1.5	Heavy_Duty	30	FD30	3	19	C00002	Internal Solid Co	Solidcor 30	DESIGNER	23
	-01-	[340,1200]	-01-	-01-	[1981,3000]	45	-01-	1.63	Heavy_Duty	60	FD60	3	21	C00003	Internal Solid Co	Solidcor 60	DESIGNER	28
	-02-	[340,1056]	-01-	-01-	[1981,2600]	44	-01-	1.93	Severe_Duty			3	21	C00004	External Solid C	Lamcor	DESIGNER	33
	-02-	[340,1056]	-01-	-01-	[1981,2600]	44	-01-	1.93	Severe_Duty	30	FD30	3	21	C00005	External Solid C	Lamcor 30	DESIGNER	33
	-02-	[340,1056]	-01-	-01-	[1981,2600]	44	-01-	1.68	Severe_Duty	60	FD60	3	21	C00006	External Solid C	Lancor 60	DESIGNER	36

Figure 4. Typical Door Catalogue Content

3.2 Mapping the Semantic Resources

Mapping semantic entities, even if they are formally defined, is a considerable challenge involving several aspects. Quite often it requires identifying pair-wise similarity between entities and computing the best match for them [2]. There are many different possible ways to compute such similarity with various methods designed in the context of data analysis, machine learning, language engineering, statistics, or knowledge representation. FUNSIEC relies on *semantic methods* [3] to deal with the problem. The basic assumption behind semantic methods is that they aim at discovering relations between (pairs of) entities belonging to different schemata *based on the meaning of the two entities* (their semantics!).

OSIECS is formed by SRs from the following: *bcBuildingDefinitions* taxonomy, e-COGNOS, ISO 12006, and IFC. The first step in the development of OSIECS is mapping their meta-schemas. The *bcBuildingDefinitions* is the taxonomy developed by the eConstruct project in order to demonstrate the power of bcXML[3], an XML-based language tailored to the representation of products/services in the construction sector. The *bcBuildingDefinitions* relies on bcXML meta-schema [4].

The e-COGNOS ontology [5] focuses on construction concepts related to the *consistent knowledge representation of (construction) knowledge items.* The e-COGNOS ontology comprises two taxonomies (concepts and relations). Those concepts and relations are grounded in the IFC entities, which form the highest level taxonomies.

Table 1. Similarities among entities of several meta-schemas

Meta-schema / *Item*	*eCognos*	*bcXML*	*ISO 12006-3*	*IFC*
Semantic Resource identification	Object (abstract)	Taxonomy	XtdRoot (ABS)	IfcRoot (ABS)
Concept	eCognosConcept	Object	XtdObject (ABS)	IfcObject (ABS)
Relation	Relation	Relationship	xtdRelationship	IfcRelationship
Property	Attribute	Property	xtdProperty	IfcProperty
Assign properties to objects	Attribute	Property	xtdRelAssignsProperties	IfcRelDefinedByProperties
Description	ObjectConceptDefinition	Description	xtdDescription	Description (attribute)
Reference to an external resource / entity	-	ExternaReference	xtdReference	IfcExternalReference

The primary target of the IFC model is interoperability of software applications in the building and construction sector. IFC classes are therefore defined according to the scope and the abstraction level of software systems dealing with Construction specific content. The entities of the IFC model are grouped in layers where kernel and core extension layers deal with general, abstract concepts whilst the shared elements and domain layers deal with specialised concepts of the real world [6].

The ISO 12006-3 [7] is a Construction specific standard that defines a schema for a taxonomy model, which enables concepts to be defined by means of

[3] bcXML primarily supports simple eCommerce communication of products (materials, components, equipment, documents) and services within or across borders.

properties, the grouping of concepts and defining relationships between concepts. *Objects*, *collections* and *relationships* are the basic entities of the model [8].

Despite of the differences of the SRs described above found among the meta-schemas, the preliminary results produced by analysis in FUNSIEC brought positive signs for the possibility of building a *pivot* meta-schema bridging (semantically) those SRs. The table 1 presents the partial results of such analysis.

4 The Architecture of the OSIECS Kernel

Based on the framework described above, the preliminary version of the OSIECS Kernel is depicted in Figure 5. Essentially it covers both meta-schema and schema levels. In the former, the Kernel is formed by: the Syntax Converter, the Semantic Analyser, the Converter, the Detector of Similarities, and the Validator. In the latter, the Kernel relies on the Entities Matcher.

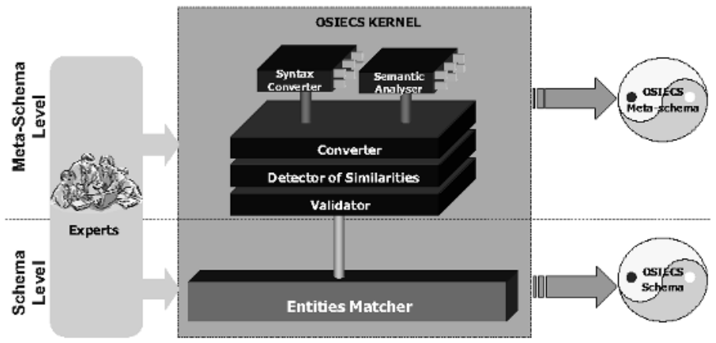


Figure 5. The OSIECS Kernel

The formalism adopted to represent both OSIECS meta-schema and schema is the OWL language - the ontology representation recommended by the Semantic Web group - for two main reasons: *i*) the expressiveness; and *ii*) the explicitness of its semantic representation which allows some automated deduction.

The Syntax converter and the Semantic analyser work together using the meta-schemas of the respective SRs as input in order to produce the conversion rules to be used by the Converter for guiding the production of the OWL meta-schemas for each of the SRs in the Kernel. The Detector of Similarities (which is, indeed, the ONDIL system briefly introduced in section 4.1) works with the OWL-converted meta-schemas in creating the OSIECS meta-schema. This meta-schema is then analysed and assessed by the Validator. Then moving one level down, the schemas of the OSIECS components are matched by the Entities Matcher. The output of this process is the OSIECS schema.

With the help of software tools, the experts play a very important role in this process. They act in both levels taking care of: *(i)* the manual analysis of the SRs and their respective meta-schemas/schemas; *(ii)* analysis of the rules of conversion; (*iii*) the assessment of the detection of similarities; *(iv)* checking of the validation process; and finally *(v)* the assessment of the output of the Entities Matcher.

4.1 The ONDIL System

As previously refereed, within the OSIECS Kernel the ONDIL system is responsible for the detection of similarities among meta-schemas. Broadly speaking, ONDIL provides inference services for Description Logic-based ontologies [9]. The expressiveness of such ontologies allows semantics of modelling languages to be formalised (*e.g* UML [11], EXPRESS [12]) and to make the semantics as explicit as possible. It is worth emphasising that formal and explicit semantics are crucial to automated deduction.

The ONDIL system comprises three modules, namely *ontology management*, *mediator*, and *inference engine*. The heart of ONDIL is an inference engine that uses the structural algorithms developed in [9] for non-standard inferences and the optimised algorithms for standard inferences. ONDIL uses the inference engine to deduce new knowledge using the ontologies (one or many) as the primary source of knowledge. The knowledge deduced is essentially new relations among the ontological concepts. The relationships among the several semantic resources (if they exist) are usually only implicit. These relationships can be viewed more as knowledge to be detected rather than knowledge to be predefined in the semantic resources. Therefore, in FUNSIEC the ONDIL system is used to establish mappings among the semantic resources.

4.2 Syntax Conversion and Semantic Analysis

As previously explained, the mapping process involves three main elements of SRs, namely the structure, the syntax, and the semantics. The solution most recommended for syntax and semantics problems is to represent (through a conversion) the original SRs in a *neutral* formal format. These converted versions are then free of syntactical problems. Both structural and semantic-related problems are solved through a semi-automatic process where the experts are aided by software tools.

The meta-schemas forming OSIECS are originally represented in different formalisms: EXPRESS in ISO 12006-3 and IFC, and UML in e-COGNOS and bcXML. The *Converter* works with the meta-schemas in their original formalisms and produces the correspondent OWL versions (Figure 6). The experts play a very strategic role in this phase. They analyse the SRs meta-schemas and create a set of conversion rules (written in Java) that allows converting each entity from their original format (in a given meta-schema) in OWL. This transformation must preserve the semantics of the entities. The set of rules is used by the JavaCC [10] tool to generate a compiler capable of translating any meta-schema written in the original formalism automatically into OWL. As part of the OSIECS development, two compilers were generated to support the translations of both EXPRESS and UML to OWL. Now the inclusion of new SRs represented in EXPRESS or UML no longer require human intervention.

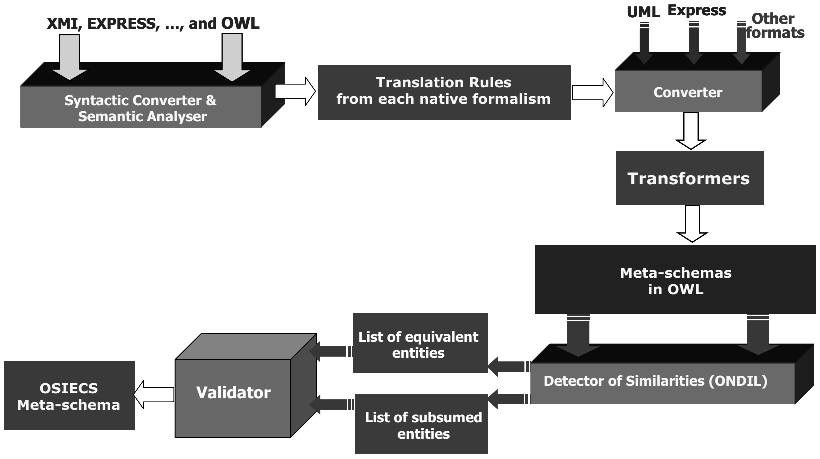

XMI, EXPRESS, ..., and OWL
Syntactic Converter & Semantic Analyser
Translation Rules from each native formalism
UML
Express
Other formats
Converter
Transformers
Meta-schemas in OWL
Detector of Similarities (ONDIL)
List of equivalent entities
List of subsumed entities
Validator
OSIECS Meta-schema

Figure 6. Creating the OSIECS Meta-schema

Table 2 shows the translation of an ENTITY ISO-12006 (in EXPRESS) into OWL. The EXPRESS ENTITY is firstly represented in description logic and this representation is translated into OWL. The class XtdRelAssociates defines a set of instances that satisfy the following property: each instance of class *XtdRelAssociates* is associated with an instance *I0* of class *XtdObject* via the attribute *RelatingObject* and with *n* instances *I1,…,In* of class *XtdObject* via the attribute *RelatedObjects*. The constraint WR1 states that the instance *I0* is different from all instances *I1,…, In*.

Table 2. An example showing the conversion EXPRESS-OWL

EXPRESS	DL	OWL
ENTITY XtdRelAssociates;	relatingObject ⊑ relationship	<owl:ObjectProperty rdf:ID="relatingObject">
RelatingObject: XtdObject		<rdfs:subPropertyOf rdf:resource="#relationship" />
RelatedObjects:	relatedObjects ⊑ relationship	</owl:ObjectProperty>
SET[1 :?] OF XtdObject;		<owl:ObjectProperty rdf:ID="relatedObjects">
WHERE	XtdRelAssociates ⊑∀relatedObjects. XtdObject	<rdfs:subPropertyOf rdf:resource="#relationship" />
WR1:		</owl:ObjectProperty>
SIZEOF(QUERY(	XtdRelAssociates	<owl:ObjectProperty rdf:ID="invRelationship">
Result<*RelatedObjects |	⊑ ≥n	<owl:inverseOf rdf:resource="#relationship"/>
RelatingObject:=:Result))=0;	relatedObjects.⊤	</owl:ObjectProperty>
END_ENTITY;	XtdRelAssociates	<owl:ObjectProperty rdf:ID="invRelatedObjects">
	⊑ ≤n	<owl:inverseOf rdf:resource="#relatedObjects"/>
	relatedObjects.⊤	</owl:ObjectProperty>
		<owl:ObjectProperty rdf:ID="invRelatingObject">
		(…)

4.3 Detection of Similarities

The *Detector of Similarities* uses the ONDIL inference engine to detect similarities between each of two concepts belonging to different meta-schemas. The similarity between two concepts is defined in four levels according to its granularity.

Consider two concepts *C1* and *C2* belonging to different meta-schemas. Firstly, the inference engine verifies whether they are equivalent according to OWL semantics. If so, the result is sent to the *Validator*. If not, they are sent to the *Subsumption Detection* component to check if one concept is subsumed by (i.e. part of) the other. If that is not the case, similarity is evaluated by the *Intersection Detection and Difference Detection* components. This allows more precise detection of similarities between the two concepts. The similarities among the meta-schemas are validated in order to produce the OSIECS meta-schema.

4.4 Matching the Entities

The similarities found in the previous stage are used over the SRs schemas following a specialisation approach. Therefore, the *Entities Matcher* applies a similarity detected between two entities (at the meta-schema level) to the entities at the schema level. For instance, if *A* is an entity from the e-COGNOS meta-schema and *B* an entity from the ISO 12006-3 meta-schema, then *S(A, B)* represents a similarity between those entities. This similarity is matched to the entities of the corresponding e-COGNOS and LexiCon schemas, *S'(a, b)*. All entities matched in the schema of the selected SRs comprise the OSIECS schema.

5 Conclusions and Future Work

The FUNSIEC framework supporting the interoperability of SRs for the construction sector has been presented with emphasis on the development of OSIECS infrastructure. The heart of such infrastructure is the OSIECS Kernel, which relies strongly on the OSIECS meta-schema/schema, both developed from the respective meta-schemas/schemas of the SRs used in OSIECS. The OSIECS Kernel and the OSIECS Meta-schema/schema, to be released by the end of 2004, are the major results of the FUNSIEC project. Both OSIECS meta-schema and schema are created semi-automatically by experts in a process aided by specially developed software tools. The FUNSIEC approach began with the analysis of the meta-schemas/schemas of the SRs chosen to form OSIECS. The selected SRs, represented in EXPRESS (ISO-12006-3 and IFC) and UML (e-COGNOS and bcXML), were converted to OWL, the neutral representation language adopted in FUNSIEC. For that, rules of conversion from EXPRESS/UML were manually generated to feed the JavaCC tool, which in turn generated two *compilers* to automatically transform any SRs from EXPRESS/UML to OWL. The converted meta-schemas were semantically compared and mapped. The final output is the OSIECS meta-schema, which is used afterwards to produce the respective OSIECS schema. The OSIECS infrastructure, now being developed, aims to prove the usefulness of these components.

The future of FUNSIEC remains full of challenges. The project is in its crucial phase where the concepts and ideas have to be demonstrated and assessed properly. Most of the OSIECS infrastructure will be in operation by January 2005 and by that time the underlying assumptions behind FUNSIEC should be confirmed. Additionally, in order to guarantee the formal basis of the conversion its technical and semantical soundness and completeness are required to be studied. Finally, the inclusion of fuzzy logics to provide a *degree* of equivalence when mapping the meta-schemas is another challenge facing FUNSIEC.

References

1. Lima, C., Fiès, B., Ferreira-da-Silva, C.: Setting up the Open Semantic Infrastructure for the Construction Sector in Europe – the FUNSIEC Project. In: 5th European Conference on Product and Process Modelling in the Building and Construction Industry – ECPPM 2004, Istambul, Turkey (2004).
2. Euzenat J., Le Bach T., Barrasa J., Bouquet P., De Bo, J., Dieng R. et al.: D2.2.3: State of the art on ontology alignment – Knowledge Web project, realizing the semantic web, IST-2004-507482 Programme of the Commission of the European Communities (2004)
3. Benerecetti M., Bouquet P., Zanobini S.: Soundness of Semantic Methods for Schema Matching. Workshop on Meaning Coordination and Negotiation at the 3rd International Semantic Web Conference, Japan (2004).
4. Lima, C. P., Stephens, J., Bohms, M.: The bcXML - Supporting eCommerce and Knowl-edge Management in the construction industry. Itcon Journal, v. 8, (2003) p. 293-308.
5. Lima, C. P., Fiès, B., Lefrancois, G., Diraby, T. E. (2003). The challenge of using a domain Ontology in KM solutions: the e-COGNOS experience. In: 10TH ISPE 2003, Funchal, Por-tugal. International Conference on Concurrent Engineering: Research and Applications, (2003) p. 771-778.
6. The IAI web site, http://www.iai-international.org/iai_international/.
7. ISO/DIS 12006-3 – Building construction Organization of information about construction works - Part 3: Framework for object-oriented information exchange (2002).
8. European eConstruction Meta-Schema (EeM), CEN Workshop Agreement – CWA3. Documents produced by the CEN/ISSS eConstruction Workshop, Brussels (2004) .
9. Le Duc, C.: Transformation d'ontologies basées sur la Logique de Description – Application dans le Commerce Electronique. PhD Thesis, Université de Nice, France (2004) .
10. JavaCC, https://javacc.dev.java.net/.
11. UML: Unified Modeling Language, Object Management Group, http://www.uml.org/.
12. EXPRESS-G - annex A of ISO 10303-11 - Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual (1994) .

Virtual Breeding Environment: Working and Sharing Principles

Jaime Irigoyen, Nathalíe Galeano, David Guerra and Arturo Molina

Centre for Innovation in Design and Technology, Eugenio Garza Sada 2501, 64849 Monterrey, México, atairigoyen@itesm.mx, ngaleano@itesm.mx, david.guerra@itesm.mx, armolina@itesm.mx

Summary. Collaboration is one way to achieve competitiveness in a global market. Virtual Organization Breeding Environments (VBE) motivate the creation of Virtual Organizations (VO) as organizations that respond with high flexibility to rapid changes in market needs. VBEs define (implicit or explicit) main working and sharing principles in order to foster collaboration between members and guarantee long-term benefits. This paper presents the results of a first stage of an action-research effort focused on identify VBE working and sharing principles that will define main elements for a VBE Framework, such as actors, activities, knowledge sharing, organizational structure and culture.

1 Introduction

Nowadays companies face the need to strongly use collaboration mechanisms and share valuable knowledge due to globalization and market aggressiveness. Diverse collaborative forms arise as an alternative for companies that wish to achieve flexibility and fast market respond in changing environments. Collaborative Networks (CN) and their forms, such as a Virtual Organisation (VO, also known as a network) created out of a Virtual Breeding Environment (VBE, also known as a cluster), provide accepted benefits, such as capitalizing on knowledge and market power existent in the partners, which give them the edge in a competitive situation [1].

Clusters (VBE) that are dynamic have moved beyond simple hierarchical networks to become characterized by numerous repeated connections between individuals, firms and institutions that constantly shift and expand (VO creation process) [2]. In order to replicate these effective VBEs, a model that describes its internal dynamics is needed. An approach to develop this model is through the identification and characterization of actors that have developed readiness for collaboration inside a VBE throughout networks and relationships [3]. The aim of this paper is to present first results of an action-research effort, in which actors, organizational forms, activities, culture, policies and supporting ICT tools of VBEs were identified in order to understand its working and sharing principles.

2 VBE Research Context

Diverse forms of Collaborative Networks (CN) have emerged during the last years as a result of the challenges faced by business and scientific worlds [4]. This research focus on two manifestations of CN:

- Virtual Organization (VO), which is a set of independent organizations that share resources and skills to achieve its mission or goal [4], and.
- VO Breeding Environment (VBE), which is an association (also known as a cluster) or pool of organizations and their related supporting institutions that have the potential and the will to cooperate with each other through the establishment of a "base" long-term cooperation agreement and interoperable infrastructure. When a business opportunity is identified by one member (acting as a broker), a subset of these organization can be selected and thus form a VO [4].

Cortada et. al. [5] pointed that the success starts always on the environment comprehension. Following this guideline, the beginning of successful generation of a VO is the understanding of the environment in which it is breed (the VBE model). An effective VBE Model requires: well identified actors, roles explicitly described, and interrelations between them outlined.

3 Research Methodology

Action Research (AR) [6] it the methodology used. Three action research cycles were planned (see Figure 1). First, a review of the state of the art in actual VBE models is carried out, the output of this stage is a draft Model that identifies main VBE working and sharing principles. Then, two spirals or cycles of the action research methodology will be performed: The first cycle will apply the draft principles in a small pilot case analysis (a group of 3 or 4 enterprises that collaborate for a specific purpose in a VBE), and the second cycle will apply the redefined principles in a second industry case (automotive VBE in Mexico).

The research results presented in this paper are related to the first action research spiral. The activities performed during the research are explained above.

Plan Activities:

1. Planning: A plan for review the state of the art of actual VBE is done including resources allocation.
2. Development of a template: A format is developed to extract specific information from VBE cases. The VBE template is a data sheet that includes eight areas: General information (name, location, age, brokers' name, mission and objectives), impact, actors, intellectual property, financial, products/services, ICT used and CNO cases. In each area of the template there are several blanks to fill, i.e. broker's name, VBE location on its life cycle, development stages of the VBEs, value creation sources and integrators' names.

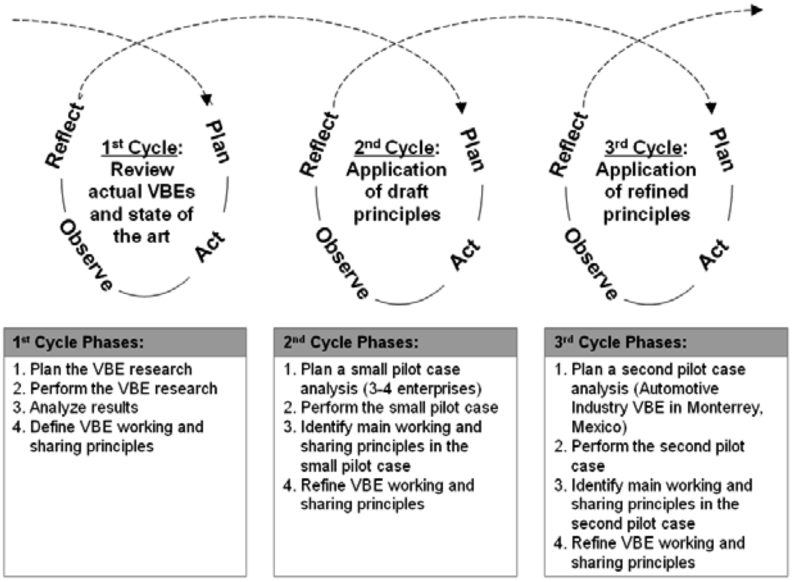

Reflect
Plan
Observe
Act
1st Cycle: Review actual VBEs and state of the art
2nd Cycle: Application of draft principles
3rd Cycle: Application of refined principles
1st Cycle Phases:
1. Plan the VBE research
2. Perform the VBE research
3. Analyze results
4. Define VBE working and sharing principles
2nd Cycle Phases:
1. Plan a small pilot case analysis (3-4 enterprises)
2. Perform the small pilot case
3. Identify main working and sharing principles in the small pilot case
4. Refine VBE working and sharing principles
3rd Cycle Phases:
1. Plan a second pilot case analysis (Automotive Industry VBE in Monterrey, Mexico)
2. Perform the second pilot case
3. Identify main working and sharing principles in the second pilot case
4. Refine VBE working and sharing principles

Figure 1. Action Research Cycles.

Act Activities:

1. Identification of VBE cases.
2. Sample selection: Figure 2 represents the sample selected based on the documented representative collaboration in the cases. The sample aims to have industrial heterogeneity, economic variety, and geographical diversity.
3. Collect Information: the VBE template is used to document each case.

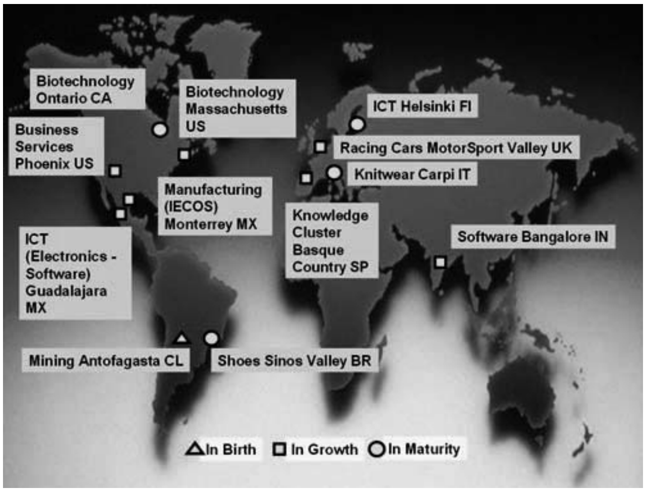

Biotechnology Ontario CA
Biotechnology Massachusetts US
ICT Helsinki FI
Business Services Phoenix US
Racing Cars MotorSport Valley UK
Knitwear Carpi IT
Manufacturing (IECOS) Monterrey MX
Knowledge Cluster Basque Country SP
Software Bangalore IN
ICT (Electronics - Software) Guadalajara MX
Mining Antofagasta CL
Shoes Sinos Valley BR
In Birth
In Growth
In Maturity

Figure 2. Sample of VBE Cases.

Observe Activities:

1. Identification of VBE key factors. A trigger question for identify key factors was used: What make successful a given VBE? Table 1 (columns 2 - 16) show the factors observed in the sample.
2. Classification of VBE cases. Table 1 lists the ordered sample (on first column) regarding to technology applications in processes and products: High tech VBE, medium tech VBE and low tech VBE.
3. Identification of the factors presence in each VBE. There are three main types of presence: "Strong Basis" for indispensable factors that will guarantee VBE success, "Secondary Basis" for supporting factors and "Complementary Basis" for factors with low impact on VBE success.
4. Grouping key factors. Key factor are ordered according to the type of VBE that is more connected: high-tech VBE, medium-tech VBE, low-tech VBE or common to all VBE cases.

Table 1. Success Bases on VBE Cases.

	A	A	A	A	A	L	L	M	HM	HM	HM	H	H	H	H	H
The success is based on ... ->	Brokerage Activities	Operational Support	ICT Use	Innovation	Richness Generation (Profitability)	Flexibility	Horizontal Cooperation	Planning (to Grow)	Education	ICT Availability	Integral Solutions	Information Culture	Knowledge Sharing	Vertical Cooperation	Research Industry Link	Technology Push Strategy
Applied Technology																
Biotechnology, Ontario, CA	.	O	O	O	O	o	.	O	O	O	o	O	O	O	O	O
ICT, Helsinki, FI	O	O	O	O	O	o	.	.	O	O	O	O	o	O	O	O
Biotechnology, Massachusetts, US	o	o	o	o	o	o	o	O	O	.	o	o	O	.	O	.
Racing Cars, Motor sport Valley, UK	o	O	o	O	O	.	.	O	.	o	.	.	.	o	O	o
Business Services, Phoenix, US	O	o	o	o	O	o	.	O	.	o	O	o	.	o	.	.
Software, Bangalore, IN	o	O	O	o	O	O	o	o	O	o	O	.	.	o	.	.
Knowledge Cluster, Basque Country, SP	o	O	o	.	o	O	O	O	.	o	.	o	O	.	.	.
ICT (Electronics to Software), Guadalajara, MX	O	o	o	o	o	O	O	o	O	o	O	o	.	.	.	.
Manufacturing (IECOS) Monterrey MX	O	o	o	o	O	O	o	o	o	.	o	o	.	o	o	.
Knitwear, Carpi, IT	O	O	.	O	o	O	O	.	.	.	.	.	o	.	.	.
Mining, Antofagasta, CL	O	O	o	.	o	O	O	o	o	.	.	.	.	.	.	.
Shoes, Sinos Valley, BR	O	O	.	o	o	O	o	.	.	.	.	.	o	O	.	.

H: Common in High Tech Cases
M: Common in Medium Tech Cases
L: Common in Low Tech Cases
A: Common in All Cases

O Strong Basis
o Secondary Basis
. Complementary Basis

Reflecting Activities:

1. Analyze results. VBE key factors identified and its relation with the type of VBE (high, medium and low tech) are analyzed.
2. Identify and model VBE actors. The key factors help to discover actors that participate in VBE life cycle processes. The connections between these actors define the VBE dynamic and interoperation.
3. Analyze the influence of VBE structures and VBE cultural issues.

VBE key factors are present in different cases, but it is important to mention that their connection with each other is what makes synergy in a VBE. Synergy allows the creation of a new and unique value propositions in network collaborative enterprises through complementing, integrating and leveraging its members' capabilities and competencies [7].

From the reflecting activities is pointed out that, the success bases in high-tech VBEs cases are focused in intellectual activities (e.g. design integral solutions, knowledge sharing) and proximity between research and industry sectors. Factors like VBE planning, education and integral solutions are strong presented in medium-tech VBE cases. Moreover, flexibility, horizontal cooperation and brokerage activities are strong presented in low-tech VBE cases.

4 VBE Model

This section proposes a VBE Model based on actors' interactions and focused on VO creation. Actors should play roles that support the development of the VBE key successful factors identified from the previous analysis.

In analogy with neurons and thoughts inside the brain, enterprises are interconnected in temporal events called Virtual Organizations. The brain should be in healthy conditions to generate networks of neurons (thoughts), the VBE should also be properly operating to generate networks of enterprises (VOs). Actors identified inside the VBE should provide the conditions for an effective collaboration.

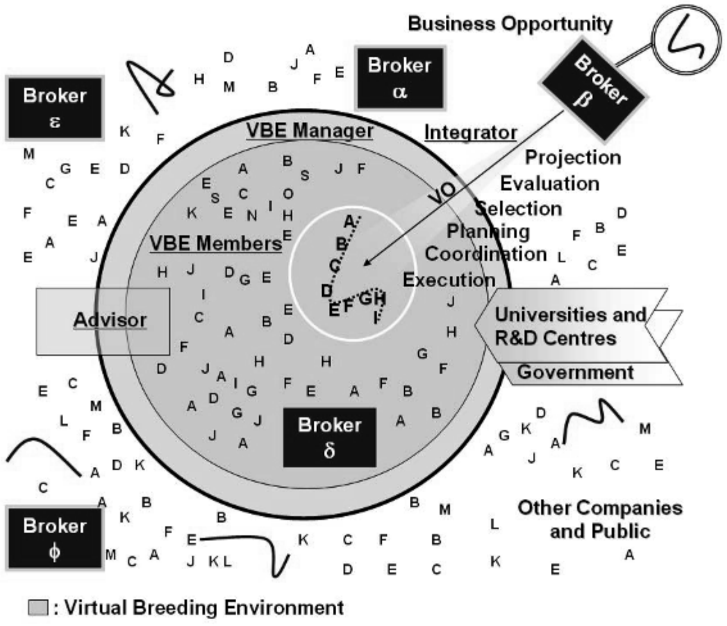


Figure 3. VBE Model based on Actors' Interaction.

Figure 3 represents the seven actors identified in a VBE and depicts VO generation upon a business opportunity. These actors are: Brokers, Integrators, Manager, Advisor, Members, Government, Universities, and, Research and Development (R&D) Centres. The letters in the model represents organizations with different competences, which will be aligned in a VO, according to a Business Opportunity found by a Broker. This alignment is coordinated by an Integrator that operates the VO. The environment needed (e.g. tools, documentation, system, procedures) is managed by the VBE Manager, while an Advisor monitors and reports VBE improvements areas to the actors. Due to an unavoidable impact on society, Government, Universities and R&D Centres are involved supporting VBE operation (e.g. legal issues, economic issues, ontology and models development), which are indispensable for a well structured VBE system. Each VBE actor is briefly described in the paragraphs below.

Broker: Brokers are specialized in business opportunities search, which is the starter element for VOs creation. Broker should also guide the communication with Integrators to ensure the right understanding of business opportunities. VBE members' capabilities and competences are an important information source for the Broker.

Integrator: Is the project manager in the VO Operation. Integrators develop the ideas (business opportunities) selected by the broker. Broker and integrator roles, could be performed by the same entity. The success in VOs is based on an effective communication and coordination between Integrator and VBE Members. Integrator coordinates VO Members according to the Broker guidelines.

VBE Manager: Its main responsibility is to provide the best medium to foster collaboration between VBE members (i.e. availability of Information and Communications –ICT- tools, infrastructure and legal framework). VBE resources should be managed by this actor. Manager should interact with all VBE actors to have a complete knowledge and vision of the VBE as a system.

Advisor: Advisor performs monitoring and control activities, required to enhance productivity. This actor needs an effective metric system for VBE evaluation. His highest interaction is basically with VBE manager, but this actor should establish connections with all VBE actors, who should have performance measures defined.

VBE Members: Collaboration and productivity are their main responsibilities (mainly achieved by horizontal and vertical cooperation [8]). These companies are the core of the environment; their effective collaboration represents the VBE strength. A VBE Member interacts with: other VBE Members, VO Broker, VO Integrator, VBE Manager and VBE supporting institutions (government, universities and R&D centres).

Government, who should facilitate the access to public infrastructure, stimulate growth, and avoid gradual deterioration of the industrial infrastructure [9]. Due to the social and economic VBE impact in a region, society must support VBE, via its government. Government institutions interact with all VBE members supporting the VBE operation, providing real welfare states.

Universities and Research and Development (R&D) Centres: The goal for these actors is to raise intellectual capital, its creation encompasses the whole spectrum of knowledge-based activities from replication to innovation [10]. Innovation in products, processes and models is result of an intensive research in these centres. The interaction of these centres is with all VBE actors for the development of theoretical and practical basis that will support the creation of new services, products and tools to enhance competitiveness.

An effective communication through VBE community is an elemental requirement that the VBE actors should have. ICT can catalyze effectiveness in several key processes using the following tools: a Semantic Web (to capture and browse information about core competences in companies), a Distributed

Workflow System (to guide process collaboration), architectures for applications integration and a set of specific applications for brokerage activities.

4.1 Organizational Structure in VBE

Work, Individuals, Organizational Formal Agreements and Informal Organization are the four components needed to define an organization [11]. Regarding to a VBE, *Individuals* are the seven actors identified; *Work* is represented by the VBE operational activities; *Organizational Formal Agreements* are the roles and responsibilities of the actors; and, the *Informal Organization* is presented on the interactions between VBE members. Three main types of structures in VBE Cases have been identified during the research: Cellular, Clusters and Chain (names given by the authors). See Figure 4 and Table 2.

Cellular: This is a common type of organization to create temporal VOs with high speed. It is present in environments with members that have high entrepreneurship; lightness to incubate SMEs for satisfying VBE's customer needs is one advantage.

Clusters: This is a structure that enhances the creation of more structured VOs oriented to large projects. It's used in environments with well-identified members, who have a structured profile that certifies their capabilities and that are grouped in specialized areas.

Chain: It presents less interaction between members; their interaction is limited to the relationships in a plain value chain during VO Operation. VO partners work according to a plan already developed and under well-defined roles. This is common for product development projects. A disadvantage of this structure is the low knowledge sharing potential because of limited interaction between partners in the design and planning activities.

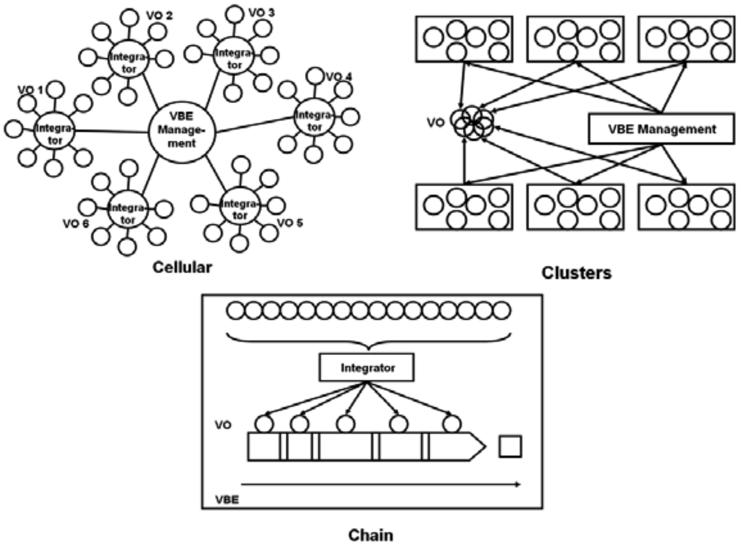


Figure 4. Types of Structures used in VBE Cases for creating VOs

Table 2. Map of Structures Identified in VBE Cases

VBE	Cellular	Clusters	Chain
Biotechnology, Ontario, CA	√		
ICT, Helsinki, FI	√		
Biotechnology, Massachusetts, US		√	
Racing Cars, Motor sport Valley, UK		√	
Business Services, Phoenix, US	√		
Software, Bangalore, IN	√		
Knowledge Cluster, Basque Country, SP	√		
ICT (Electronics to Software), Guadalajara, MX	√		
Manufacturing (IECOS), Monterrey, MX		√	
Knitwear, Carpi, IT		√	
Mining, Antofagasta, CL			√
Shoes, Sinos Valley, BR			√

From the three major categories of organizational structure presented, two constants can be identified about their elements: members should be closely communicated, so a horizontal and plain architecture fits very well; and someone needs to coordinate different efforts in order to gain synergy between the VOs. This means that in successful VBE there should be well-developed support industries in such a way that VOs can be easily developed. These support industries can form a "Supporting VBE" due to their integration with the main VBE. This integration between the value chain and the organizational structure should facilitate the development of the supporting VBE, as happens in Singapore VBE which has focused the entire country's organizational structure to support its growth [12].

An appropriate VBE should have few levels in its hierarchical organization in order to enhance VOs creation through a rapid and flexible process. Proof of functionality of this size of structure are the cases of Aportia in Guadalajara, Mexico [13], ICT Cluster in Helsinki, Finland [14], the Industrial Districts all around Italy [15] and the Software Manufacturing Industry in Bangalore [16]. In these cases small and medium enterprises (SMEs) are organized by simple teams with no more than two levels of hierarchy.

4.2 Culture in VBE

The environment in which a VO is created has a common collaboration and trust culture. Trust between organizations is a competitive requirement. The build-up of trust between organizations is founded upon the inter-personal bonding via trust developed between the individuals in the different organizations (or indeed between different units within a single organization) [17]. This means that a successful VO can be generated with partners that present a collaborative culture inside their own organizations.

Collaborative culture involves the ability of team work. Team work is an optimal organizational solution to solve a complex problem, analysis of strategic elements and planning are key processes during the problem solution processes [18].

Among other factors, the primary requirements for a proper collaboration culture in a VBE can be associated to the following guidelines: *Long-Term and*

Global Vision: A VBE should have a defined long term vision, which will guide its activities according to the strategy defined by the organization. This vision should consider the global market aspect, especially for the development of dynamic VOs. *Innovation:* The ability to be creative in products, process and services is the base of innovation. VBE actors should involve innovation in the execution of its activities. Innovation should be also present during VO creation, operation and dissolution. *Specialization*: Organizations should have their core competences completely developed, but they should also have the ability to integrate with their complementary partners. Another cultural aspects found in VBE Actors are: *Initiative, Commitment, Leadership, Self-Learning, Effective Communication, Entrepreneurship, Team Work, Continuous Training, Knowledge Sharing, Information Culture, Specialization* and *Negotiation.*

4.3 Knowledge Sharing in VBE

The seven actors defined in VBE Model are focused on generation of a long term competitive environment, through complementation of competencies between them. Complementary competencies are an integral part of knowledge-based collaboration [19]. A VO can be temporal but it should share the knowledge acquired in its life cycle to the VBE in order to improve other VOs. In a more general view, the entire VBE can evolve into a better one by learning about itself (knowledge sharing).

To describe knowledge sharing in a VBE is necessary to define common knowledge concepts. First it is important to differentiate data, information and knowledge: Data is only text that does not answer questions to a particular problem [20], Information can be defined as data with meaning [21] and, Knowledge is information with added detail relating to how it may be used or applied [22].

An important requisite for knowledge usage is adequate knowledge structure definitions, describing the role of each knowledge type in overall context [23]. *Explicit* knowledge, is knowledge that has been articulated and very often captured in the form of text, tables, diagrams, product specifications and so on. *Tacit* knowledge is highly personal and hard to formalize, thus making it difficult to communicate or share with others. *Implicit* Knowledge is the type of knowledge that can be articulated but has not [24].

The types of knowledge described (explicit, implicit and tacit) are mapped on the actors of the VBE Model. Table 3 indicates these relations and depicts in what type of knowledge are largely based the activities of given actor. The description of the knowledge type found in Broker, Integrator and VBE Members activities is detailed below.

Table 3. Type of Knowledge largely based for activities of Actors in a VBE

		Knowledge Type		
		Explicit	Implicit	Tacit
VBE Actors	VBE Member	o	O	O
	Broker	.	O	O
	Integrator	O	.	o
	VBE Manager	O	o	o
	Universities and R&D Centres	O	O	o
	VBE Advisor	.	o	O
	Government	O	o	o

O Strong Basis o Secondary Basis . Complementary Basis

The *broker* is certainly an important source of tacit knowledge because there is a "feeling" in the ability to find the best business opportunities. This is the result of the acquired experience in a broker (see Figure 3). A broker compiles many data about market and trends and processes it to produce information that use in the business opportunities search. The vision of VO as a solution to market needs is developed via applied knowledge and is not formerly registered.

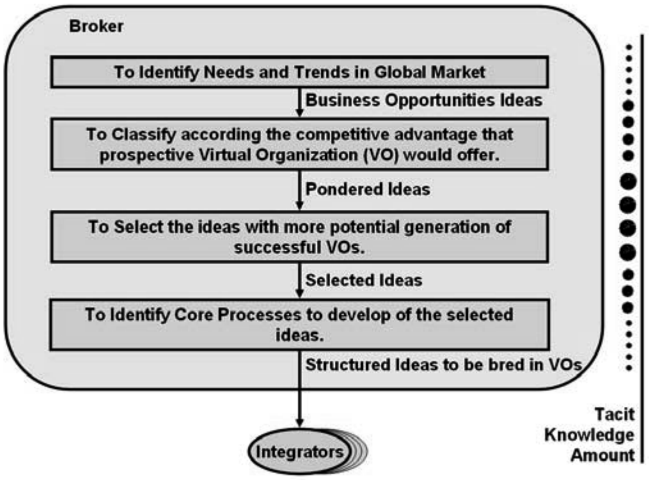

Broker
To Identify Needs and Trends in Global Market
Business Opportunities Ideas
To Classify according the competitive advantage that prospective Virtual Organization (VO) would offer.
Pondered Ideas
To Select the ideas with more potential generation of successful VOs.
Selected Ideas
To Identify Core Processes to develop of the selected ideas.
Structured Ideas to be bred in VOs
Integrators
Tacit Knowledge Amount

Figure 5. Tacit Knowledge in Broker Activities within VBE model

The knowledge in the *integrator* is focused to the project management, because the VO is a project that needs to be managed. The main thrust of Process Management is the concept of knowledge management [25].

In *VBE members*, the knowledge that is important to develop for enhance the collaboration between them, is the know-how needed to create effective partnerships. An important issue in this ability is trust, a complex factor that is related to tacit knowledge. This knowledge has potential value in a VBE if it were spread over many other members.

When collaboration is the objective, a holistic vision of the actors highlights the value of knowledge sharing in the environment as a source of competitiveness. A combination of knowledge from different perspectives impels abilities to create new opportunities and to respond to challenges in innovative ways [19].

5 Conclusions and Further Research

The results of a first cycle of an action-research effort focused on finding VBE working and sharing principle have been presented. The research methodology has been described and the main elements for a VBE Framework have been depicted: 1) Model of Interoperability of 7 Actors for a VBE, 2) Organizational Structures inside VBE Cases, 3) Cultural Aspects, 4) Knowledge Sharing in VBE Key Actors, and 5) ICT tools that catalyze effectiveness in key processes inside a VBE.

Interoperability between actors is presented through VBE network where each node (VBE Member) is connected directly or indirectly between them. Interoperability of systems and processes is a characteristic of the effective communication required in a VBE for its right operation.

Next stages in this research will focus on second and third cycles of the action research methodology presented, considering knowledge management as the medium for achieving a sustainable development in VBE, a team based as the optimum organizational structure for an efficient operation, and a culture of collaboration as the social environment required for accomplish the main goal in a VBE: competitiveness and VO creation.

6 Acknowledgements

The research presented in this document is a contribution for the European Collaborative Networked Organizations Leadership Initiative (ECOLEAD) project, funded by the European Community under the "Information Society Technology" Programme. A special thankfulness to the Centre for Innovation in Design and Technology (CIDYT) of ITESM University and PYME-Creativa project for its support and trust in this research.

References

1. Noran, Ovidiu (2004) Towards a Meta-Methodology for Collaborative Networked Organisations in *Virtual Enterprises and Collaborative Networks*. Kluwer Academic Publisher, 71-78
2. Brown, Peter and McNaughton, Rod B. (2002) Global Competitiveness and Local Networks: A Review of the Literature, Global Competition and Local Net-works, Ashgate.
3. Kandampully, Jay (2003) B2B relationships and networks in the Internet age. Management Decision , Vol. 41 No. 5 p. 443
4. Camarinha-Matos, Luis M. and Afsarmanesh, Hamideh (2004) The Emerging Discipline of Collaborative Networks in *Enterprises and Collaborative Networks*. Kluwer Academic Publisher .

5. Cortada, James W.; Hargraves, Thomas S. And Wakin, Edwards (2000) Into the Networked Age, How IBM and Other Firms are Getting There Now. *Translated by Fernando Martínez and Othon Juárez.* Oxford University Press.
6. Kemmis, S., & McTaggart, R. (Eds.). (1988). The Action Research Planner (3rd ed.). Geelong: Deakin University
7. Bititci, U; Martinez, B; Albores, P and Parung, J (2004) Creating and managing value in collaborative networks. International Journal of Physical Distribution & Logistics Management, Vol. 34, No. 3 p. 251
8. Johansson, Ch.; Svenningsson, I. and Kaplan, A. (2004) Achieving advantages of scale in small companies through horizontal networking in *Proceedings of the 10th International Conference on Concurrent Enterprising*. ICE
9. Castells, M. and Himanen, P. (2002) The Information Society and the Welfare State: The Finnish Model. *Translated by Jesús Albores*. Alianza Editorial.
10. Pöyhönen, A. and Smedlund, A. (2004) Assessing intellectual capital creation in regional clusters. Journal of Intellectual Capital, Vol. 5 No. 3, p. 351 – 365
11. Nadler, D. A. and Tushman, M. L. (1999) El diseño de la Organización como Arma Competitiva, El poder de la Arquitectura Organizacional. Oxford, p. 31.
12. Kelly, M. and Boulton, W. (1997) Electronics Manufacturing in the Pacific Rim, World Technology Evaluation Center.
13. Donoso, J. (2004) Cluster Fénix, Industria Electrónica, MANUFACTURA, July 2004.
14. Tukianen, J. (2002) ICT Cluster Study Helsinki Region, University of Helsinki, Department of Economics.
15. Ferrari, A. (2004) El surgimiento de la pequeña y mediana empresa en Italia y el desarrollo de los distritos industriales, ICE (Instituto nacionale per il Commercio Estero) October 2004.
16. NASSCOM: Indian SMEs Poised for Growth, Indian SME Fact Sheet, National Association of Software & Service Companies: SME FORUM, August (2004)
17. Edwards, J.S. and Kidd, J.B. (2003) Knowledge management sans frontiers, The Journal of the Operational Research Society. Oxford, Feb 2003, Vol.54, Iss. 2; pg. 130
18. Sheard, A.G. and Kakabadse, A.P. (2004) A process perspective on leadership and team development, The Journal of Management Development, Volume: 23 Number: 1 Page: 7 –10.
19. Skyrme, David J. (2003) Knowledge Networking, Creating the Collaborative Enterprise. ButterWorth Heinemann.
20. Quigley, E. J. and Debons, A. (1999) Interrogative Theory of Information and Knowledge in *Proceedings of SIGCPR '99*, ACM Press, New Orleans, LA., pp. 4-10.
21. Spek, R. v.d. and Spijkervet, A. (1997) Knowledge Management: Dealing Intelligently with Knowledge, CIBIT, Utrecht
22. Harding, J.A. (1996) A Knowledge Representation Model to Support Concurrent Engineering Team Working. Loughborough University of Technology, Loughborough
23. Guerra, David (2004) A Manufacturing Model to Enable Knowledge Maintenance in Decision Support Systems. Loughborough University of Technology, Loughborough
24. Guerra D., Young R. (2005) A Manufacturing Model to Enable Knowledge Maintenance in Decision Support Systems, 33 Annual Conference of North American Manufacturing Research Institution of Society of Manufacturing Engineering (NAMRI/SME), Columbia University, NY, May 24-27..
25. WIKIPEDIA Home Page, consulted on November 2004, http://en.wikipedia.org/

Modeling and Using Business Collaborations

Giorgio Bruno

Dip.Automatica e Informatica, Politecnico di Torino, Torino, Italy,
giorgio.bruno@polito.it

Summary. Business processes interact with each other by sending and receiving messages; how such interactions take place (i.e. in which order and with which timing constraints) has to be defined in advance with a collaboration model, so the parties can be developed separately in conformity with it. With orchestration languages, however, the role of a collaboration model is that of an interface, so it is the run-time responsibility of the activities of the business process to guarantee that messages are exchanged in proper order and time. This paper motivates the need of run-time collaboration instances, which, by keeping the state of ongoing collaborations, relieve business processes of the burden of checking the conformity of the actual operations with the collaboration model.

1 Introduction

At the basic level of e-business, organizations interact by sending and receiving messages, however in order to attain a given common goal, the parties have to exchange a number of messages with appropriate content and order. For this reason it is important to focus the attention on all the messages exchanged for a particular purpose rather than on single interactions. The term collaboration (or conversation) is used to refer to the flow of messages taking place between two parties for a given purpose; it is a particular view on the whole set of the messages exchanged in a multi-party system.

The parties involved in a collaboration are often referred to as services or business processes or simply software applications. Usually one party takes the role of the "provider", the other that of the "requester".

A well-known example of collaboration is the purchasing of goods or services, which can be described as follows: organization A's purchasing service (the requester) sends a request for quote (rfQ) to organization B's selling service (the provider), which responds with a quote; if the purchasing service accepts the quote, it will then send an order to the selling service. This example will be used throughout this paper.

Although, from an external point of view, a collaboration is a flow of messages (such as the flow consisting of an rfQ, a quote, and an order), it is perceived by each party as an ordered flow of message-oriented activities. For the selling service the above-mentioned collaboration implies that first it has to receive an rfQ, then it has to send a quote, and finally it might receive an order.

The actual flow of messages can vary from case to case. In fact the selling service might not reply or the quote might not be accepted and in that case the order won't be sent. Therefore, in one case the collaboration might consist of only an rfQ message, while in another case it might be made of all the three messages (rfQ, quote and order).

Anyway an actual collaboration can take place only because the parties in advance have agreed on how to put it into practice. Such an agreement is based on an abstract representation of the collaboration, i.e. on a collaboration model.

There is growing interest in modeling collaborations as processes as a process denotes an ordered flow of activities. Moreover the parties involved in collaborations can be modeled as processes as well since they, too, consist of ordered flows of activities.

Although similar in structure, the processes used to model collaborations are different from those related to services or applications; in this paper the former will be called collaboration processes, the latter business processes.

The activities in a business process fall into three major categories: manual activities, automatic activities and control-flow ones. Manual activities require human intervention and they are usually performed through a web-based user interface, so their implementation is outside the scope of the business process. Automatic activities and control-flow ones, instead, are in charge of the process (either directly or indirectly) as will be shown later on. For example the selling business process might consist of the following activities: receiving an rfQ, preparing a quote, sending it to the requester, receiving an order, and processing the order; the activity of preparing a quote and that of processing the order probably require some human intervention (hence they are manual), while the others could automatically be performed by the system.

A collaboration process, instead, is meant to be a contract between the parties, therefore it is mainly made up of message-oriented activities, whose purpose is to show which messages have to be sent or received.

As an example WSCL [1] presents collaboration processes as UML activity diagrams in which activities fall into four categories: ReceiveSend, Receive, Send, and SendReceive. Messages carry XML documents which are defined in appropriate schemas, and the actual protocols are defined using WSDL. The collaboration is presented from the point of view of the provider, the requester's one being easily obtainable if sending activities are turned into receiving ones and vice versa.

While business processes are intended to be operational, collaboration processes are meant to specify the behavior, in terms of the messages to be exchanged, the business processes have to conform with.

In fact with orchestration languages, such as BPEL [2] and BPML [3], business processes are executable, so they exist at run-time in the form of business process instances, just as types exist in the form of variables. Collaboration processes, instead, do not have real-time counterparts, since they are interpreted as interfaces, which are defined in a choreography providing the global view of all the interactions in the system. Special languages, called choreography languages, such as WSCI [4] and WS-CDL [5], have been defined for that purpose. As an interface a collaboration process is to be implemented by a provider and used by a requester.

However, since it is extremely difficult, in general, to statically prove the conformity of a business process to the collaboration process it has to implement or use, it is the responsibility of the activities of the business process to guarantee proper behavior at run-time. As an example assume that an rfQ has a deadline after which the quote won't be taken into account by the requester; then the burden of deciding whether it is worth sending the quote or not is entirely on the provider. If there were a collaboration instance (i.e. a run-time representative of the collaboration process, which knows the state of the collaboration) that decision could automatically be taken by it, so the activities of the provider's business process would be simplified. A collaboration instance can be thought of as a channel through which a party sends and receives messages belonging to a given collaboration.

The purpose of this paper is to motivate the need of run-time collaboration instances, which, by keeping the state of ongoing collaborations, relieve business processes of the burden of checking the conformity of the actual operations with the collaboration model. In addition collaboration processes can be made more flexible with the introduction of parameters, which can be tailored to actual collaborations; such parameters need to be kept in collaboration instances.

Moreover this paper presents a software environment, called bProgress, which we have developed so as to experiment with business processes and collaboration ones; it consists of modeling tools and of services for implementing business processes and collaborations.

This paper is organized as follows. Section 2 describes how collaboration processes are modeled with bProgress, while section 3 shows how business processes use collaboration instances. Then the bProgress environment is illustrated in section 4 and finally a comparison with related work is presented in section 5.

2 Collaboration processes

Basically a collaboration process consists of message-oriented activities placed within a control structure providing for sequential, alternative, and parallel paths as well as for timeout-related paths. In bProgress processes are represented as UML activity diagrams.

The model shown in Figure 1 is the selling collaboration process introduced in the previous section.

A message-oriented activity is a receiving activity (such as receiveRfQ) if it is connected to an input signal, it is a sending activity (such as sendQuote) if it is connected to an output signal. For simplicity we consider asynchronous messages only. Messages are represented by signals, and each message has a name and a type (not shown in Figure 1). The types of the messages are defined in an XML schema associated with the collaboration process. Each message-oriented activity has a deadline (such as t1), which appears in the label of the outgoing timeout transition. When a deadline has been reached, a timeout exception occurs and the activity fails.

Unlabelled transitions represent precedence constraints; timeout transitions (i.e. those labelled with keyword "timeout") show the effects of timeouts; in this example timeouts end the collaboration. The structure of a collaboration process can get much more complicated if alternative paths or parallel ones are needed.

Collaborations can have parameters; in this case the three deadlines are parameters.

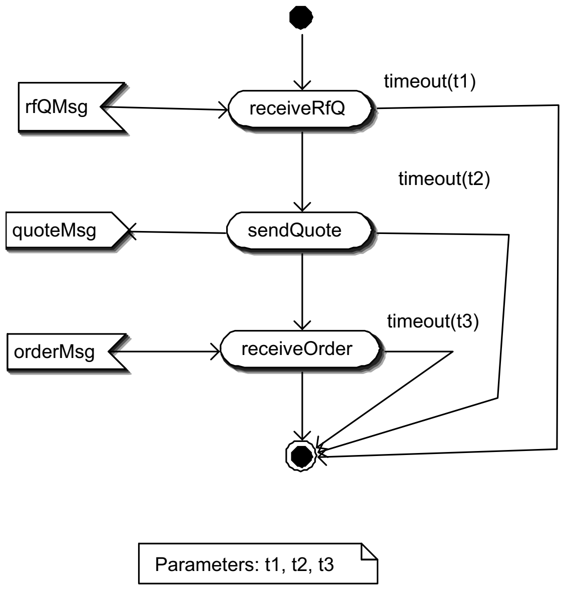


Figure 1. The selling collaboration process

Parameters have to be agreed upon by the parties before an actual collaboration is made effective; so there is the need of a preliminary phase in which parameters are set, this phase being itself a special case of collaboration. If the preliminary phase is successful, the collaboration is started.

The model in Figure 1 shows that, after the collaboration has been started, the provider is ready to receive an rfQ message (rfQMsg) sent by the requester; if that message does not arrive before instant t1, a timeout will occur and the collaboration will end. After the rfQ message has been received, the provider has to reply with a quote message; if it doesn't so before instant t2, a timeout will occur and the collaboration will end. After the quote message has been sent, the provider is ready to accept the order message but, if it is not received before instant t3, a timeout will occur and the collaboration will end.

The three deadines, t1, t2 and t3, are parameters, so they can be tailored to the actual collaboration.

From an operational point of view, a collaboration process requires two representatives, a requester collaboration process instance, rcpi, on the sender's side and a provider collaboration process instance, pcpi, on the provider's side. A pcpi operates in conjunction with a provider business process instance (pbpi), whilst an rcpi operates in conjunction with a requester business process instance (rbpi).

Activating a new collaboration requires a standard protocol at the end of which parameters have been agreed upon and both the rcpi and the pcpi have been started (and they know of each other).

From the point of view of the provider the selling collaboration takes place as follows. After the collaboration has been activated, the provider (i.e. the pcpi) can receive an rfQ message; when it receives it, it will pass it to the business process (i.e. the pbpi). Then the provider can get a quote message from the pbpi; when it gets it, it will send it to the requester (i.e. the rcpi). Finally the provider is able to receive an order message; if it receives it, it will pass it to the pbpi. The purpose of a receiving activity is to pass the related business process instance the message received from the requester, whilst the purpose of a sending activity is to send the requester the message got from the business process instance.

The requester interprets the same collaboration process by turning sending activities into receiving ones and vice versa.

Collaboration process instances accomplish several tasks: they perform the actual sending and receiving of messages and maintain the state of the collaboration (by keeping local states aligned with each other) as well as the history of the messages exchanged. They prevent a party from sending a message in the wrong order or at the wrong time; for example, if the provider tries to send a quote after deadline t2 has expired, its collaboration process instance will reject it. In addition, their presence automatically guarantees the correlation of messages to business process instances, thus relieving the process activities of that burden.

3 Using collaborations

Collaborations in bProgress are managed by business processes and this section illustrates how this is done.

A business process is made up of a number of activities, which fall into three major categories (exception handling is not considered, for simplicity): manual activities, automatic activities and control-flow ones.

Manual activities require human intervention and they are usually performed through a web-based user interface, so their implementation is outside the scope of the business process. Automatic activities and control-flow ones, instead, are in charge of the process, in the sense that the process will use software components (called process-support components) able to perform such activities. In bProgress we can associate actions (i.e. blocks of code) with the model's activities so as to give a complete description of the business process; a code generation tool of the bProgress environment is able to produce process-support components incorporating those actions within suitable contexts.

There are two business processes interested in managing selling collaborations, the purchasing business process and the selling business process; the former is shown in Figure 2, the latter in Figure 4.

The purchasing business process operates as follows. The first activity, as it consists in preparing an rfQ, is a manual activity to be performed by a purchaser. A manual activity is connected to the role required. For example activity prepareRfQ requires a purchaser; this means that it has to be performed by a person playing that role. The next activity, sendRfQ, has to send the rfQ to all the suppliers invited, i.e. the suppliers that have been chosen in the previous activity. Quotes will be received, one at a time, by receiveQuote. When all the quotes expected have been received, or the deadline of receiveQuote has expired, manual activity selectQuote can be performed. If a quote has been accepted, an order to the supplier will be sent by activity sendOrder. The deadline of receiveQuote is a parameter, so it can be tailored to the specific purchasing process instance.

It results from the above description that business processes cannot be described separately from the business entities they are related to; so it is necessary to explain the nature of such inter-relations before the details of the activities are presented.

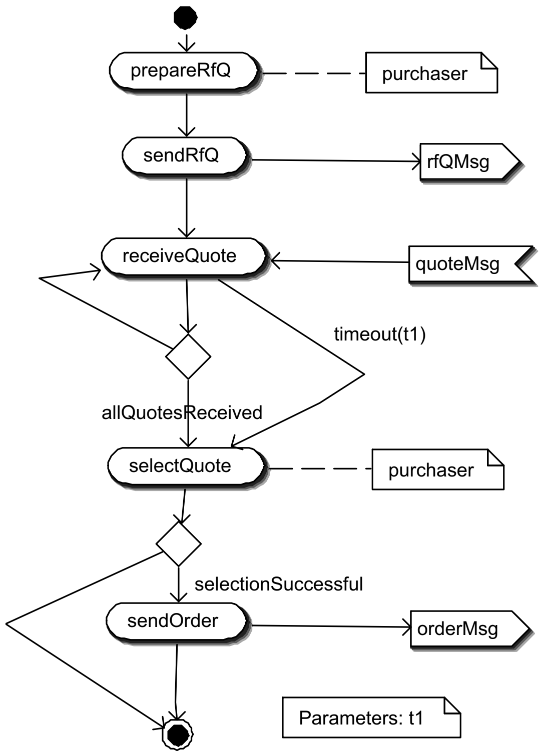


Figure 2. The purchasing business process

Business entities are related to each other and such relationships are usually presented in an entity-relationship (or class-relationship) model, such as the data model of the purchasing organization shown in Figure 3. From this model it is evident that an rfQ is connected to the suppliers invited (i.e the organizations invited to the purchasing auction), as well as to the quotes received.

In bProgress business process instances are business objects, too, and they are represented by an entity called BPI (business process instance). Likewise, actual collaborations are business objects and they are represented by entity RCPI (requester collaboration process instance) if they refer to a requester-side collaboration, or by entity PCPI (provider collaboration process instance) if they refer to a provider-side collaboration. A BPI can be involved in multiple RCPIs or PCPIs; however a given RCPI (or PCPI) is connected to just one BPI (the one involved in the collaboration).

Of course a process instance is related to the corresponding process model but such connections belong to a different level and are outside the scope of this discussion.

In general in bProgress a business process is meant to take care of a given business entity and, by extension, of those associated with it, as in the case of the purchasing process which manages the lifecycle of an rfQ. For this reason a business process instance (BPI) can be connected to any business object implementing interface WFItem (workflow item); entity RfQ implements that interface, so an rfQ is process-driven (in other words it is a controlled business object).

In general, the actions associated with automatic activities and control-flow ones operate on the business objects through a set of classes providing an object/relational mapping.

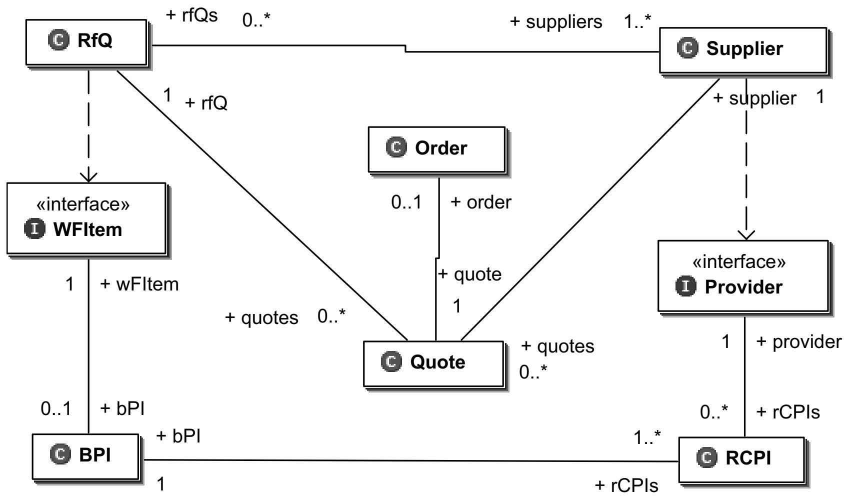


Figure 3. The business entity model of the purchaser

In fact, while business objects are mapped onto records in a relational database, such records are not acted on directly as it would be too awkward and error-prone. They are, instead, operated on through an object/relational interface consisting of a set of classes, which provide methods for generating, retrieving and modifying the records and also methods for navigating among the records along the relationships defined in the business entity model.

The actions are presented below as blocks of pseudo-java code. Each action must be considered to be placed in a context including a reference (bpi) to the current business process instance and a reference to the controlled business object; the name of the second reference is that of the class of the controlled business object with the initial in lower case (such as rfQ). At run-time those references will be properly initialized by the bProgress process engine (cf. next section). The actions of the activities of the purchasing business process are as follows.

```
sendRfQ:
for Supplier s in rfQ.getSuppliers() {
  RCPI rcpi = new RCPI("SellingCP", s, bpi);
  rcpi.addParameter("t1", rfQ.getT1());   //idem for t2 and t3
  rcpi.start();
  if (rcpi.getState().equals("accepted"))
            rcpi.sendMsg("rfQMsg", rfQ.getMessage());
  else rcpi.remove();
}
receiveQuote:
Quote q = new Quote(quoteMsg, rfQ);
allQuotesReceived:
return rfQ.getQuotes().size() == bpi.getRCPIs().size();
selectionSuccessful:
return rfQ.getQuote("state = 'accepted'") != null;
sendOrder:
Quote q = rfQ.getQuote("state = 'accepted'");
Order o = new Order(rfQ, q);
for RCPI rcpi in bpi.getRCPIs() {
  if (rcpi.getProvider().equals(q.getSupplier()) {
            rcpi.sendMsg("orderMsg", o); break;
  }}
```

Activity sendRfQ invites suppliers to submit quotes; the suppliers to be invited are those that have been associated with the current rfQ in manual activity prepareRfQ. For each supplier a new collaboration is activated and an rfQ message is sent.

Method getSuppliers is a navigational method, which retrieves all the suppliers (business objects) related to the current rfQ. A supplier implements the Provider interface: this is a synthetic way of indicating that a supplier business object has all the information (e.g. the url of the service) needed to establish a collaboration with the corresponding organization.

For each supplier to be invited a new requester collaboration process instance (rcpi) is generated; the constructor of class RCPI takes as parameters the name of the collaboration process, the provider with which the collaboration has to be established, and the current business process instance. For simplicity we assume that each supplier supports the selling collaboration process shown in Figure 1. The values of the three parameters are then added to the rcpi; they are taken from the rfQ business object in which they have been defined by the purchaser during activity prepareRfQ. Next the collaboration is started with a synchronuos interaction with the provider. If the collaboration is accepted, the rfQ message will be sent; method sendMsg is provided by class RCPI and takes two parameters, the name of the message and its content (which is provided by method getMessage of class RfQ). Method sendMsg performs all the checks required, in particular it won't send the message if it is late, and in that case it will return a negative result.

Activity receiveQuote is in charge of receiving a quote message; then it transforms the message into a quote business object and connects the latter to the supplier and to the rfQ (this is actually done by the constructor of Quote).

Activity allQuotesReceived returns true if all the quotes expected have been received (i.e. there are as many quotes as the number of collaborations started).

Activity selectionSuccessful returns true if there is a quote whose state is "accepted". The selection is made by manual activity selectQuote; the best quote, if any, has been put into state "accepted", while the others remain in state "received".

Activity sendOrder sends the order to the winner: it searches the collaboration process instances to find the one related to the supplier of the winning quote.

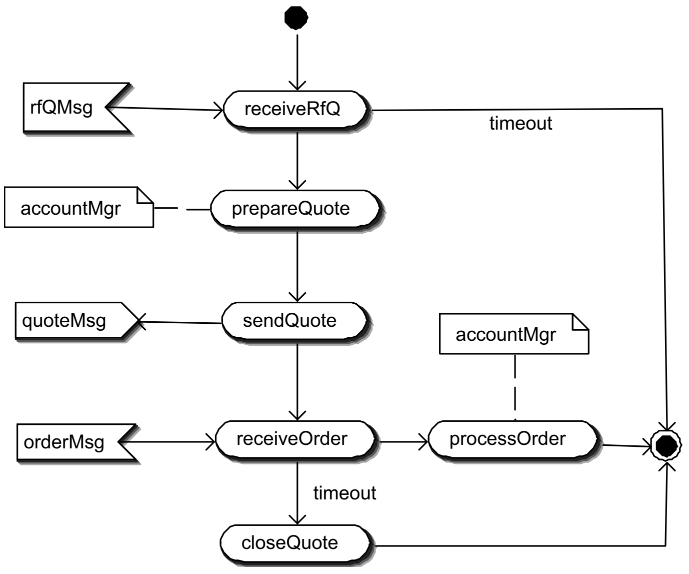


Figure 4. The selling business process

The code of the selling business process is similar and is not shown. A short account of its activities is as follows. After the rfQ has been received, manual activity prepareQuote is enabled (and scheduled for an account manager). The quote is then sent. If the order is received, it will be processed otherwise the quote will be closed. The timeouts of the receiving activities are those signalled by the provider collaboration process instance; for this reason the values of timeouts are not specified in the labels of the corresponding transitions. When a provider accepts a request for a new collaboration, it starts a provider collaboration process instance and it also normally activates the corresponding provider business process instance (and associates the two).

4 The bProgress environment

We have developed a java environment (based on Eclipse), called bProgress, in order to prototype business processes and collaborations.

The three types of models presented in this paper, i.e. business entity models, business processes and collaboration processes, have graphical UML representations as well as XML representations. XML representations can be directly written or can be automatically produced from the XMI versions of the UML models.

There are two code generators in bProgress. One takes the XML representation of a business entity model and produces: a) the scripts for generating the corresponding relational tables; b) a set of classes providing an object-oriented interface to those relational tables. The other code generator takes the XML representation of a business process (including the code of automatic activities) and produces the corresponding process-support software component.

Business and collaboration processes are loaded into the database by the process loader. The environment includes a process engine, which manages process instances by interpreting the corresponding business processes. Collaborations are managed by specific bProgress components which use the SOAP protocol and run on top of the JBoss application server. Collaboration processes rely on WSDL as to the definition of messages. Transactional aspects in collaborations is of primary importance as shown by the OASIS ebXML [6] and BTP initiatives [7]; and bProgress is meant to support experimentation in that research field.

5 Comparison with related work

Business process languages fall into two major categories, orchestration languages and workflow languages, each category emphasizing a particular point of view.

Orchestration languages, such as BPEL [2] and BPML [3] are mainly devoted to supporting interactions between services described in WSDL and ignore manual activities (as first-class constructs); on the contrary, workflow languages such as XPDL [8] support manual activities but are weaker in message processing.

As to the control flow, XPDL has a graph structure where activities (i.e. the nodes) can have multiple incoming/outgoing transitions; although the language has an XML representation, it is difficult to manually write a program made up of nodes and links. Orchestration languages, on the other hand, enforce structured programming through the use of nested blocks.

Collaboration types, in general, are considered to be different from business processes. WSCI [4] describes collaboration types as interfaces and it is meant to provide a global view (i.e. a choreography) of all the parties involved in a complex interaction. WS-CDL [5] provides more features than WSCI while basically following the same approach.

We believe that collaborations are inherently binary, so, in our opinion, a global picture describing several collaborations provides too much information.

Furthermore we consider collaboration types as abstract processes consisting of message-oriented activities placed within a suitable control structure; deadlines, and hence timeouts, are of primary concern. Parameters are also important as they allow a generic process to be tailored to specific situations.

Collaboration types are shown from the provider's point of view although we envisage a peer-to-peer relationship between the parties. Alternatively it is possible to take a neutral position, as it is done in [9], if the model directly focuses on the message flow rather than on message-oriented activities; however as the direction of messages has to be specified, it is not really different to take one direction, i.e. that of the provider, as the reference one.

Collaboration types are public processes acting as standards the interested parties have to conform to. The notion of public process has been introduced in [10] within workflow languages, as a Petri net showing the interactions between two organizations, one acting as the contractor, the other as the subcontractor. Each organization is responsible of a distinct portion (public subflow) of the public workflow and it can then develop a private workflow which has to comply with the respective public subflow. Rules are given so conformity can be automatically checked.

In bProgress we use run-time collaboration instances, which, by keeping the state of ongoing collaborations, are able to prevent a party from sending/receiving a message in the wrong order or at the wrong time. In bProgress a business process is definitely a workflow process, so manual activities are first-class activities. The process control structure should be as rich as possible yet amenable to direct writing, so we have selected a number of constructs and represented them with XML tags and UML activity diagrams. Such constructs meet the programming patterns required in workflow applications as shown in [11].

6 Conclusion

This paper has addressed the issue of modeling cross-enterprise collaborations and has presented an approach based on collaboration processes.

Basically a collaboration process consists of message-oriented activities placed within a control structure providing for sequential, alternative, and parallel paths as well as for timeout-related paths.

A collaboration process is a special case of business process and this uniformity entails several advantages both in conceptual and in practical terms.

This paper has also presented the bProgress environment whose tools make the models of business and collaboration processes operational.

References

1. W3C (2002) Web Services Conversation Language Version 1.0. http://www.w3.org/TR/2002/NOTE-wscl10-0020314/
2. IBM (2003) Business Process Execution Language for Web Services Version 1.1. http://www-106.ibm.com/developerworks/library/ws-bpel/
3. BPMI.org (2002) Business Process Modeling Language. http://www.bpmi.org/
4. W3C (2002) Web Services Choreography Interface Version 1.0. http://www.w3.org/TR/2002/NOTE-wsci-20020808/
5. W3C (2004) Web Services Choreography Description Language Version 1.0. http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/
6. UN/CEFACT, OASIS (2001) ebXML Business Process Specification Schema Version 1.01. http://www.ebxml.org/specs/ebBPSS.pdf
7. OASIS (2004) Business Transaction Protocol Version 1.0.9.1. http://www.oasis-open.org/committees/workgroup.php?wg_abbrev=business-transaction
8. WfMC (2002) Workflow process definition interface - XML Process Definition Language. http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf
9. Dubray JJ (2001) OAGIS implementation using the ebXML CPP, CPA and BPSS specifications Version 1.0. http:// xml.coverpages.org/OAGI-ebXML-WhitePaper-103.pdf
10. van der Aalst WMP (2002) Inheritance of interorganizational workflows: how to agree to disagree without loosing control? Information Technology and Management Journal: 195-231
11. Russell N, Hofstede AHM ter, Edmond D, van der Aalst WMP (2004) Workflow data patterns. QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane

Establishing Interoperability of Coordination Protocols in ad hoc Inter-Organizational Collaborations

Bettina Bazijanec, Johannes Maria Zaha, Antonia Albani and Klaus Turowski

Business Informatics and Systems Engineering, University of Augsburg, Universitätsstraße 16, 86135 Augsburg, Germany, {bettina.bazijanec, johannes.maria.zaha, antonia.albani, klaus.turowksi} @wiwi.uni-augsburg.de

1 Introduction

In inter-organizational collaboration business processes that span multiple partner organizations have to be efficiently supported while preserving autonomy of each partner. Due to the fact that no internal process details should be disclosed only messages containing relevant information can be exchanged between business partners for coordination purposes. It is crucial that messages reach business partners on time and that they contain exactly the information necessary to complete the next internal process step. Therefore, collaboration partners agree on coordination protocols that define valid messages and a chronological order for the information exchange. Every partner is then able to implement a binding of these protocols to internal business processes and applications. In ad hoc collaborations the agreement on coordination protocols is much harder to achieve, because in such a scenario it is not guaranteed that every partner knows each other's capabilities. Hence, at the beginning of every ad hoc collaboration there has to be a setup phase where partners agree upon basic facts regarding their joint business activities. This may also include providing trust between partners as this is a crucial factor in inter-organizational collaboration. The focus of this paper is set on establishing technical interoperability assuming that a necessary level of trust between business partners already exists.

In the aforementioned setup phase two main activities have to be performed: agreeing on *collaboration content* as well as on the *chronological order* of collaboration tasks. Agreement of collaboration content includes the definition of desired collaboration goals and necessary tasks that will lead to these goals e.g. tasks like *order* or *quote* are relevant to the goal *supply of material*. The assignment of roles (e.g. buyer or seller) based on identified tasks and the partition of tasks into sub-tasks are also included in this phase (see Figure 1).

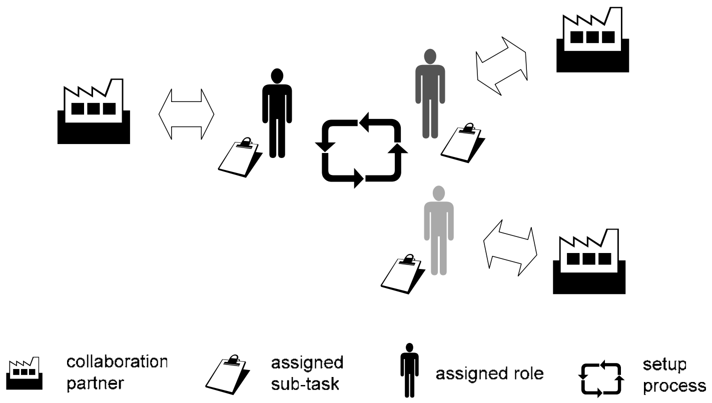


Figure 1. Agreement on collaboration content

As sub-tasks have to be performed in a coordinated way to achieve the given goal their chronological order based on causal connections and technical capabilities of the participating partners has to be defined by the means of coordination protocols.

In this paper a setup process for ad hoc collaborations will be introduced that describes these main setup activities in detail and shows how to obtain an interoperable, overall coordination protocol. At first, section 2 introduces tasks that are relevant to inter-organizational collaboration and how transition systems can be used to formally define coordination protocols. These concepts are then used in section 3 to describe and explain the process steps. In section 4 related work is discussed and in section 5 conclusions are drawn and an outlook on future work is given.

2 Collaboration Tasks and Coordination Protocols

In the following sections two important concepts are introduced that will later allow to specify necessary steps in the setup phase of an inter-organizational collaboration: *tasks* and *coordination protocols*.

2.1 Types of Collaboration Tasks

In a single collaboration scenario several tasks that address different aspects of collaboration have to be coordinated at the same time. Hence, different types of collaboration tasks can be identified [3] that fall into one of the following categories:
Setup related tasks: As already outlined above, these tasks have to be performed before an ad hoc collaboration can start and include search for potential collaboration partners that are capable to perform certain tasks (e.g. in a central registry), negotiation of collaboration content and adjustment of remaining incompatibilities.

Business related tasks: These tasks define the collaboration subject i.e. those steps that are necessary to achieve the desired goal. Since every new collaboration scenario may have other goals than previous ones partners have to agree on these tasks each time again.
Context related tasks: Communication in inter-organizational collaboration is performed between two or more partners over open networks. Therefore, additional tasks have to be carried out in order to assure correct, secure, and consistent delivery and storage of business-related information. Hence, they constitute the execution context of business-related tasks.

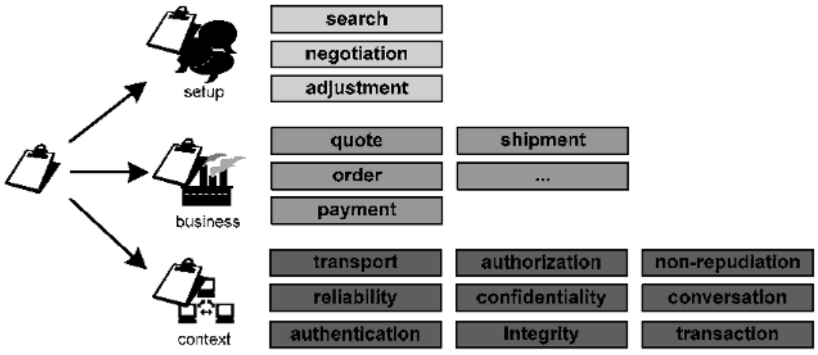


Figure 2. Setup-, business- and context-related coordination tasks

Figure 2 shows the identified task categories and corresponding task types. Whereas the number of setup and context related tasks is more or less limited there are a lot of business related tasks that partners can choose from. However, during the setup phase a set of business-related tasks will be fixed for the scope of the collaboration scenario (see section 3). Every collaboration task generates a coordination effort between the partners that has to be mastered by the means of message exchange, e.g. the task order implies the exchange of a purchase order by the buyer role such that the seller role can process it and then acknowledge or deny. Hence, for each task a protocol can be defined. As setup-related tasks have to be performed in order to agree on business-related and context-related tasks it is assumed that the setup process as it will be introduced in this paper can be understood and performed by each potential partner.

2.2 Definition of Coordination Protocols

As already mentioned, coordination protocols define message exchanges that are necessary when performing inter-organizational collaboration tasks. A *protocol* can be formally defined following the definition of transition systems [10] as a 5-tuple $p=(Q, \Sigma, \delta, s0, F)$ where Q is a finite set of states and Σ is a finite set of valid labels that represent protocol messages which are exchanged during an interaction. Messages are marked *out* if they represent outgoing messages and *in* if they represent incoming messages. $\delta \subseteq Q \times \Sigma \times Q$ is transition relation, $s0 \in Q$ is the

initial state, and F is a set of final states. The resulting transition system is specific for a particular role in an interaction. Each outgoing message in the protocol description corresponds to an incoming message of another role's protocol description. As an example, a simple purchase protocol from a seller's viewpoint can be defined as follows:
$Q = \{q0, q1, q2, q3, q4\}$, $\Sigma = \{RFQ, QUOTE, PO, POA\}$, $F = \{q4\}$, $s0 = q0$
$\delta = \{(q0, in:RFQ, q1), (q1, out:QUOTE, q2), (q2, in:PO, q3), (q3, out:POA, q4)\}$

First, the seller receives a request for quote (RFQ) which leads to a protocol state change from q0 to q1. In this state he is able to send out a QUOTE-message that contains current conditions. After sending out the message he again changes his state where he is then able to receive a purchase order (PO) and finally to send out a purchase order acknowledgement (POA).

This kind of protocol definition only uses a set of message names that denote the types of corresponding messages. In order to also define these types a data structure is used describing the respective message, i.e. if a purchase order message and its corresponding purchase order type are defined, then every message with the same data structure is considered to be a purchase order [4]. Data structures have a specific information content and can be described on three semiotic layers (see Figure 3): syntax, semantics, and pragmatics [11].

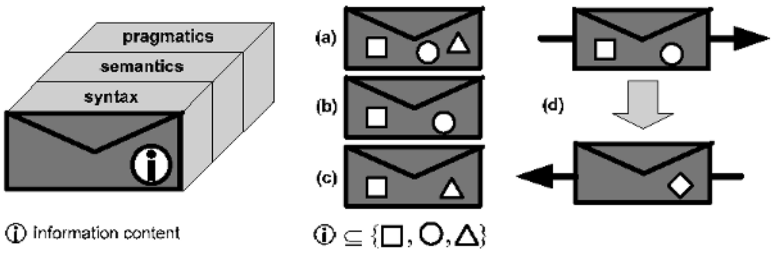


Figure 3. Messages: syntax, semantics and pragmatics

The *syntax* describes which expressions are valid and which rules are used to build up the message's data structure from these expressions. A schema definition can provide this kind of description. Since the mentioned expressions and also data structures that are build up from expressions, refer to real world objects they have a specific meaning which is called *semantics*. Although two expressions are different they may have the same meaning e.g. addr. and mailaddress can both refer to a postal address. Another aspect of semantics is the information content that is captured in a data structure [14], because different partners may have defined separately a type with the same name that should denote the same real world object but nevertheless they are different because one type includes more information than the other, e.g. an address may be defined with or without providing country information. If, for example, an organization is only acting within the US then this information is not relevant. Therefore is has to be clear which information content is minimal to define the shared semantics of a message. Figure 3 gives an example: if square and circle define one real world object then (a) and (b) are representations of this object although (a) provides more information. (c) does not refer to the same object (even if it has the same name as described above). If only the square defines an object then all three messages would represent the same object. Ideally, two separately defined message types with the same name should have the same information content. Finally, the *pragmatics* of a message provides causal relations between a received message and the following action. With regard to protocol definitions a received message can enforce the sending of a message with a specific type, e.g. a request for quote requires to be followed by a quote (see also Figure 3 on the right). It is also possible that a message requires some particular action to be taken by the receiving actor that are not obviously linked to the next following message, e.g. a message requires a specific processing or it has to be decoded by a specific algorithm so that the content can be read out.

3 Establishing Coordination Protocol Interoperability

A collaboration scenario is a combination of various coordination steps supporting several tasks. These tasks have to be identified and agreed upon by the participating parties. Necessary coordination steps for each task are defined by the means of so called protocol definitions as described in the previous section. As every protocol introduces its own messages and sequences an overall collaboration protocol has to be composed from single coordination protocols in a way that it leads to the desired collaboration outcome. This process of setting up an ad hoc relation between collaboration partners is shown in Figure 4 and will be described in the following sections:

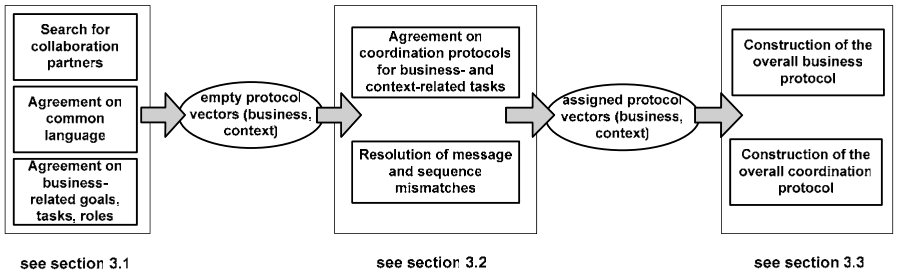

Search for collaboration partners
Agreement on common language
Agreement on business-related goals, tasks, roles
empty protocol vectors (business, context)
Agreement on coordination protocols for business- and context-related tasks
Resolution of message and sequence mismatches
assigned protocol vectors (business, context)
Construction of the overall business protocol
Construction of the overall coordination protocol
see section 3.1
see section 3.2
see section 3.3

Figure 4. Collaboration setup process (overview)

3.1 Agreement on Collaboration Content

In order to identify relevant tasks for an inter-organizational collaboration goals have to be defined and partners have to be found. In an ad hoc scenario there is typically one organization that has some kind of demand and tries to satisfy this demand by requesting some value-adding activities from one or more other organizations. This can be the shipment of physical goods as well as the computation of driving directions on the Internet. Since potential partners are not known in advance some kind of directory is needed to search for them. This is a common approach also used in the field of Web Service discovery [1]. The directory provides contact information and a classification of enterprises based on industry sector or even based on a service classification. After having identified a set of potential partners (assuming the general willingness to collaborate) negotiation in form of direct interaction can begin in order to gain knowledge about different task types which need to be performed. In this negotiation, first a common language has to be determined. There could be a different understanding of task identifiers e.g. a task *purchase* could only include the exchange of order information or it could also include prior exchange of price information. A possible way to overcome such problems is to agree on a taxonomy that contains a hierarchy of business related tasks. Today, taxonomies are well-known in the field of material classification (e.g. eCl@ss [6]). After having established a common understanding of task types the agreement on tasks themselves can start. Figure 5 shows a possible outcome of this task agreement step.

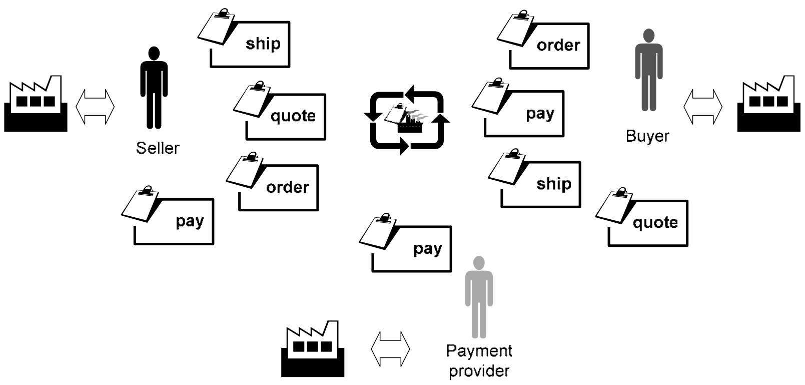


Figure 5. Agreed business tasks and roles

Three roles have been identified and for each role tasks have been defined. At this time no sequencing of tasks is expected, only the capability to perform these tasks in the given role is relevant. Since each task execution relies on the usage of coordination protocols, as introduced in section 2.2, two so-called *protocol vectors* can be defined, namely one protocol vector V_B for business-related and one protocol vector V_C for context-related tasks. A protocol vector represents one possible protocol combination to coordinate given tasks and is defined as a tuple $V \in P_1 \times P_2 \times \cdots \times P_n$ where P_i is the set of protocols $p_i^1, \ldots, p_i^n$ that coordinate task *i*. In order to be able to agree on a common protocol vector it is first necessary to determine the set of possible coordination protocols for each of the positions in both of the vectors i.e. all partners have to provide information about their supported protocols.

3.2 Adjustment of Coordination Protocols

This process step begins with the publication of supported coordination protocol standards (for V_B as well as for V_C) by each business partner. These can be standardized protocols like for example described in a RosettaNet PIP or in an OpenTrans document exchange. If no standard can be referenced a protocol description has to be provided. These descriptions can be specified with the help of standards like BPEL or ebXML/BPSS which allow to define view-specific coordination protocols [1]. Since both references to standardized protocols and self defined protocols can be found in this negotiation step they have to be made comparable. As formal protocol descriptions like transitions systems (see section 2.2) or Petri Nets [15] allow reasoning about correctness or compatibility of definitions [17] a mapping from protocol standards to transition systems has to be provided. For example, compatibility of protocols can be tested by computing the intersection of two transition systems as described in [17]. If a protocol is defined according to a reference standard then additionally a formal description has to be provided for this protocol. If a description standard is used then a mapping of description language constructs to the formal notation is necessary. If protocols are defined in an aggregated way (e.g. a complete purchasing protocol including task like quote, order, and payment) these have to be split so that a sub-protocol for each task can be provided. The goal here is to be able to compare protocols and reason about their compatibility. This is necessary because there can be protocol mismatches that hinder partners to interoperate through given protocols [18], [5]. Those problems are:

Message mismatch: Data structures representing message types may differ in their syntax or semantics. Ideally, two separately defined types with the same name should have the same schema and the same information content, but that is dependent on the actors that defined these types. In an inter-organizational scenario with many participants it is desirable to agree upon a data schema in order to avoid problems with differently defined types. Especially in ad hoc collaborations this can not be achieved so these problems may occur.

Sequence mismatch: Mismatches can also occur regarding message exchange orders. If patterns of interaction do not fit together [8] problems like deadlock or unspecified reception may arise [18]. Although partners support coordination of the same task they are not able to interact. Sequence problems may be caused by different message pragmatics (i.e. different reactions of partners are expected after receiving a message e.g. because of internal processing restrictions), or different semantics of same message types (i.e. types of the same name do not describe the same information content).

Some message and sequence mismatches can be resolved. This is called *mediation* and is based on the construction of a mediator [16], also named proxy or adapter, that interposes message exchange. This mediator is able to transform messages i.e. to map a source to a target schema which includes for example rearrangement and aggregation of expressions [13]. Mediators may therefore filter out unnecessary information or add information from external sources. Mediators may also resolve sequence mismatches by collecting or splitting information and generating new messages [18]. For those coordination protocol mismatches that remain unsolved *human intervention* is necessary to resolve these problems if possible. At best, all partners use the same reference standard for a certain coordination task e.g. RosettaNet PIP 3A4 for the order task. In this case a simple *adaption* of technology can achieve interoperability as long as all parties can agree on essential protocol parameters. If not, then for each task their protocol descriptions have to be analyzed with regard to protocol compatibility i.e. if potential message or sequence mismatches can be resolved.

3.2 Construction of an Overall Collaboration Protocol

If an agreement on one coordination protocol is possible for every position in vector V_B then the overall sequencing of business related tasks has to be defined i.e. an overall business protocol on a higher aggregation level has to be constructed. Here again, potential problems due to sequencing mismatches may be encountered. For example, the buyer wants to receive the goods before payment but the seller only ships goods after payment. The techniques used to check compatibility of single coordination protocol can again be used in this process step. After having finished this first construction step an overall collaboration protocol has to be constructed where protocol information stored in vector V_C is added to the overall business protocol. As context-related tasks may be performed at each single step of a business protocol one context protocol may appear more than once in the overall coordination protocol (e.g. if multiple transactions have to be implemented). Also, context protocols may affect the construction of a business messages (e.g. if encryption is necessary). Figure 6 shows an example of a simple purchasing protocol generated from four business tasks (*quote*, *order*, *ship* and *pay*). This protocol has to be extended with context related tasks, namely *transport*, *confidentiality*, *authorization*, and *transaction*.

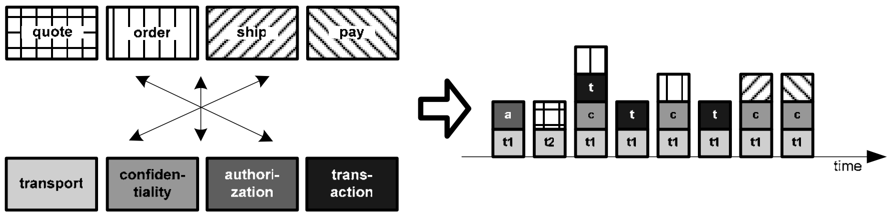


Figure 6. Construction of overall collaboration protocols (example)

Collaboration partners have to specify where each context related task has to be performed, so that an overall protocol can be constructed. In this example two transport protocols are used for different business protocol steps e.g. t1 represents HTTP and t2 represents SMTP. So, all message exchanges except the exchange of quote information, which is done via SMTP, will be performed using HTTP. For the other three context-related tasks one particular protocol standard is used.

The transition definition part of each protocol is given in Figure 7. It is shown how an overall protocol definition (right column) can be generated using single protocols (left column). Sometimes it is only necessary to concatenate a context related protocol like the authorization protocol in this example. Other protocols are inserted between business protocol steps like the part of the transaction protocol that coordinates the two-phase commit (PREPARE/CONFIRM; CANCEL). Then there are protocols that change business protocol messages. This can simply be additional processing information in a header (brief X[<…>]) or a message encoding like ciphering (brief Y(<…>)). In this case placeholders are used in the protocol descriptions that are specified by the partners.

(q0, in:RFQ, q1)
(q1, out:QUOTE, q2)
(q2, in:PO, q3)
(q3, out:POA, q4)
(q4, out:SHIP, q5)
(q5, in:PAY, q6)
overall business protocol

(s0, in:T[<req>], s1)
(s1, out:ENROLL, s2)
(s2, in:ENROLLED, s3)
(s3, out:T[<resp>], s4)
(s4, in:PREPARE, s5)
(s5, out:PREPARED, s6)
(s6, in:CONFIRM, s7)
(s7, out:CONFIRMED, s8)
(s4, in:CANCEL, s9)
(s9, out:CANCELLED, s8)
transaction protocol

(r0, in:USR_PW, r1)
(r1, out:AUTH_OK, r2)
(r1, out:AUTH_DENY, r3)
authorization protocol

(t0, in:C(<req>), t1)
(t1, out:C(<resp>), t2)
confidentiality protocol

(u0, in: USR_PW, u1)
(u1, out:AUTH_OK, u2)
(u1, out:AUTH_DENY, u17)
(u2, in:RFQ, u3)
(u3, out:QUOTE, u4)
(u4, in:C(T[PO]), u5)
(u5, out:ENROLL, u6)
(u6, in:ENROLLED, u7)
(u7, out:C(T[POA]), u8)
(u8, in:PREPARE, u9)
(u9, out:PREPARED, u10)
(u10, in:CONFIRM, u11)
(u11, out:CONFIRMED, u12)
(u8, in:CANCEL, u13)
(u13, out:CANCELLED, u14)
(u12, out:C(SHIP), u15)
(u15, in:C(PAY), u16)
overall collaboration protocol

Figure 7. Construction of the overall transition system (example)

4 Related Work

This paper introduces a process model for establishing interoperability in ad hoc collaborations. The approach to publish capabilities and select intersecting capabilities for configuration of the partners' integration systems can also be found in ebXML. There, a so-called collaboration protocol agreement (CPA) is built up from the partners' collaboration protocol profiles (CPP) but the specification of single supported business tasks is very limited because it is only possible to reference whole business processes specified as ebXML artifacts [9] what simplifies the automatic negotiation of CPAs. However, a definite negotiation process is still not specified in the ebXML standard although the OASIS/ebXML CPPA Technical Committee is currently working on it. [2] describe requirements for the process of e-contracting including consistency rules. The proposed framework can be used to check correctness and consistency when constructing a negotiation process itself but gives no guidance how to specify or negotiate business protocols [2]. The problem of business protocol interoperability is addressed by several authors (cf. [17], [12]). The goal is to automatically decide if protocols are compatible which allows building architectures for ad hoc collaboration. Approaches in the field of component interoperability, e.g. [18], [7], refer to similar problems so that their results, e.g. automatic generation of mediators form protocol descriptions, can be used with business protocols as well.

5 Summary and Outlook

This paper presents the definition of a setup process that has to be performed before an ad hoc inter-organizational collaboration can start. Within this process agreement on collaboration content can be achieved as well as an overall coordination protocol can be constructed which defines a chronological order of tasks out of single protocols that describe the information exchanges that are necessary to coordinate given tasks. Our goal is to define a component-based architecture that implements this process and uses the outcomes to dynamically configure itself. Using this architecture it will be possible to technically support a wide range of ad hoc collaboration scenarios as long as mismatches can be resolved. Due to the extensibility of component-based architectures it will be possible to easily deploy additional components implementing for example a new coordination standard for transactions which then can be used as one possible assignment to a protocol vector position. Future work therefore is twofold: the presented setup process has to be refined and implemented, and the component model as well as the composition model for the architecture has to be defined. For negotiation, compatibility testing, and mediator generation previous work in the fields of agent communication and component interoperability will be further examined and integrated.

References

1. Alonso, G., Casati, F., Harumi, K., Machiraju, V. (2004), Web Services - Concepts, Architectures and Applications, Springer-Verlag, Berlin.
2. Angelov, S., Grefen, P. (2002), An Approach to the Construction of Flexible B2B E-Contracting Processes, University of Twente.
3. Bazijanec, B., Winnewisser, C., Albani, A., Turowski, K. (2004), A Component-based Architecture for Protocol Vector Conversion in Inter-organizational Systems, International Workshop on Modeling Inter-Organizational Systems, Larnaca, Cyprus, Springer, LNCS 3292, pp. 556-567.
4. Bussler, C. (2003), B2B Integration, Springer, Berlin.
5. Bussler, C. (2002), Public Process Inheritance for Business-to-Business Integration, Technologies for E-Services, Third International Workshop, TES, pp. 19-28.
6. eCl@ss - Standardized Material and Service Classification, http://www.eclass-online.com/.
7. Farias, A., Südholt, M. (2002), On components with explicit protocols satisfying a notion of correctness by construction, DOA/CoopIS/ODBASE 2002 Confederated International Conferences, pp. 995-1012.
8. Garlan, D., Allen, R., Ockerbloom, J. (1995), Architecture Mismatch: Why Reuse is so Hard, IEEE Software, 12, pp. 17-26.
9. Hofreiter, B., Huemer, C., Klas, W. (2002), ebXML: Status, Research Issues, and Obstacles, 12th Int.l Wrkshp on Research Issues in Data Engineering: Engineering e-Commerce/ e-Business Systems (RIDE.02), pp. 7-16.
10. Hopcroft, J. E., Motwani, R., Ullman, J. D. (2001), Introduction to Automata Theory, Languages, and Computation, Addison Wesley
11. Liu, K. (2000), Semiotics in Information Systems Engineering, Cambridge University Press, Cambridge, UK.
12. Mallya, A. U., Singh, M. P. (2004), A Semantic Approach for Designing Commitment Protocols, Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04).
13. Rahm, E., Bernstein, P. A. (2001), On Matching Schemas Automatically, VLDB Journal, 10, pp. 334-350.
14. Sheth, A. (1996), Data Semantics: What, Where, and How?, in R. Meersman and L. Mark: Data Semantics (IFIP Transactions), Chapman and Hall, London, pp. 601-610.
15. van der Aalst, W. M. P., Weske, M. (2001), The P2P Approach to Interorganizational Workflows, Proceedings of the 13th International Conference on Advanced Information Systems Engineering, pp. 140-156.
16. Wiederhold, G. (1995), Mediation in information systems, ACM Computing Surveys, 27, pp. 265–267.
17. Wombacher, A., Fankhauser, P., Mahleko, B., Neuhold, E. (2004), Matchmaking for Business Processes Based on Choreographies, IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), pp. 359-368.
18. Yellin, D. M., Strom, R. E. (1997), Protocol Specifications and Component Adaptors, ACM Transactions on Programming Languages and Systems, 19, pp. 292-333.

Ontology-based Interoperability Services for Semantic Collaboration in Open Networked Systems *

Silvana Castano, Alfio Ferrara, and Stefano Montanelli

Università degli Studi di Milano
DICo - Via Comelico, 39, 20135 Milano - Italy
{castano,ferrara,montanelli}@dico.unimi.it

Summary. In this paper, we describe an ontology-based collaboration model for supporting semantic interoperability in open networked systems. We characterize discovery and matchmaking *semantic interoperability services* for retrieving information resources semantically related to a target request, to enable a coordinated and virtualized access to distributed heterogeneous information resources.

1 Introduction

Semantic interoperability is a crucial problem in open networked systems where many different and independent enterprise parties need to cooperate and share heterogeneous information resources often in response to opportunities or challenges that cannot be anticipated in advance and require a rapid response. Accessing heterogeneous and distributed informational resources in a coordinated and virtual way requires appropriate semantic interoperability techniques to enable a seamless access and retrieval of the right information resources, while preserving the information representation and management requirements of each single party involved in the coalition [1]. In addition, a further requirement for effective semantic interoperability techniques regards the availability of semantically rich descriptions of information resources in use by an enterprise party [10]. Ontologies are generally recognized as an essential tool for allowing communication and knowledge sharing among distributed users and applications, by providing a semantically rich description and a common understanding of a domain of interest. Due to the Semantic Web, a large body of research has been recently devoted to ontologies and ontology language standard proposals [2]. In this context, a crucial role is related to the availability of matchmaking techniques based on Semantic Web technologies (e.g., OWL [12]) in order to discover information resources based on available ontology descriptions. Work related to this topic has been addressed in [3, 7], where intelligent techniques based

* This paper has been partially funded by NoE INTEROP, IST Project n. 508011 - 6th EU Framework Programme.

on a Description Logics approach are described, which compare the knowledge contained in different concept ontologies, by looking for semantic mappings denoting similar concepts. Recent research in P2P systems focuses on providing techniques for evolving from basic P2P networks supporting only file exchanges using simple filenames as metadata, to more complex systems like schema-based P2P networks, capable of supporting the exchange of structured contents (e.g., documents, relational data) by exploiting explicit schemas to describe knowledge, usually using RDF and thematic ontologies as metadata [8, 9].

In this paper, we focus on the information resource discovery problem, and we propose a semantic collaboration model for open networked systems, where autonomous enterprise parties require a coordinated access to heterogeneous and distributed information resources. We rely on ontologies for representing the structure and the semantics, including interdependencies and relationships, of the information resources required and in use by a given enterprise. We characterize three ontology-based *interoperability services*, namely, the *matching service* for performing semantic affinity evaluations on ontology elements, the *discovery service* for query composition, propagation, and processing, and the *acquisition service* for information resource access.

The paper is organized as follows. In Section 2, we discuss semantic interoperability requirements and a collaboration model for open networked systems. In Section 3, we present ontology-based semantic interoperability services. In Section 4, we describe an application example of the proposed collaboration model. Finally, concluding remarks are presented in Section 5.

2 Semantic Interoperability Requirements in Open Networked Systems

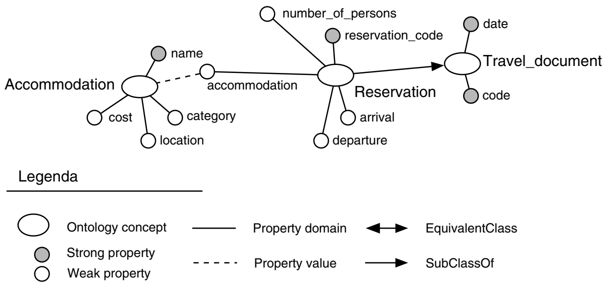
The following features affect collaboration in open networked systems, and need to be addressed by appropriate techniques: (i) *dynamism of the system*, regards the fact that enterprise parties are allowed to join and leave the networked organization at any moment; (ii) *autonomy of enterprises*, in that each enterprise is responsible for its own information resource management and representation; (iii) *absence of a-priori agreement*, about ontology specification vocabulary and language to be used for knowledge specification; (iv) *equality of responsibilities*, no centralized entities with coordinating tasks are recognized and each party enforces interaction facilities with other parties for resource sharing and collaboration.

In an open networked system, each involved enterprise party is autonomous and acts as a node (peer) like in typical open distributed systems (e.g., P2P, Grid). In particular, each enterprise party takes part to the networked system by exposing the information resources to be shared and by providing an ontological representation for them.

2.1 Ontology-based Information Resource Description

In order to support the discovery of relevant information with respect to a target request, information resources need to be described in a way that is understandable and usable by the networked organization. To this end, each enterprise party provides an ontological description of its information resources, according to a Semantic Web-compatible language for its specification (e.g., OWL [12]). In order to describe the interoperability services in an ontology language-independent way, we refer to a reference ontology model, in terms of concepts, properties and semantic relations between concepts. A concept is characterized by a name and a set of properties that represents its features. We distinguish between *strong* and *weak* properties, to denote mandatory properties (i.e., properties with minimal cardinality ≥ 1), and optional properties (i.e., properties with minimal cardinality $= 0$), respectively. Each property is associated with a name and a value, which can be a datatype or a reference to another concept. Semantic relations are defined between concepts, to express the most appropriate relations existing between them. In particular, semantic relations that are specified include the typical relations provided by the Semantic Web languages (e.g., equivalentClass, subClassOf in the OWL language). For example, a detailed description of how to map OWL on the reference model is provided in. [6].

As an example, we consider the enterprise party EP_1 which provides information resources related to the travel domain. As shown in Fig. 1(a), the EP_1 ontology



(a)

```
<owl:Class rdf:ID="Reservation">
  <rdfs:subClassOf rdf:resource="#Travel_document"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#accommodation"/>
      <owl:someValuesFrom rdf:resource="#Accommodation"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="reservation_code"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="&xsd;int">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:DatatypeProperty rdf:about="#reservation_code">
  <rdfs:domain rdf:resource="#Reservation"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="arrival">
  <rdfs:domain rdf:resource="#Reservation"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="departure">
  <rdfs:domain rdf:resource="#Reservation"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="number_of_persons">
  <rdfs:domain rdf:resource="#Reservation"/>
  <rdfs:range rdf:resource="&xsd;int"/>
</owl:DatatypeProperty>
```

(b)

Fig. 1. (a) An example of enterprise party ontology; **(b)** a fragment of OWL code contains the Accommodation, Reservation, and Travel_document concepts; the Reservation concept is a specialization (i.e., subClassOf) of the Travel_document concept. Furthermore, the value of the property Accommodation of the Reservation concept refers to the Accommodation concept. In Fig. 1(b), the fragment of OWL code specifying the Reservation concept is reported. In particular, strong property constraints are defined with a property restriction by setting the minCardinality clause to the value 1 (e.g., reservation_code).

2.2 The Peer-based Collaboration Model

The peer-based collaboration model enforces resource sharing and discovery in open networked systems by exploiting a number of internetworked ontologies. The effectiveness of the knowledge discovery process depends on the capability of retrieving the information related to a target resource request, by exploiting semantic features of enterprise ontology descriptions.

Each enterprise party in the system acts both as a client and as a server in the networked organization interacting with other parties directly, by submitting queries containing a request for one or more resources. The architecture of the peer-based collaboration model is shown in Fig. 2. Two different query types are supported in

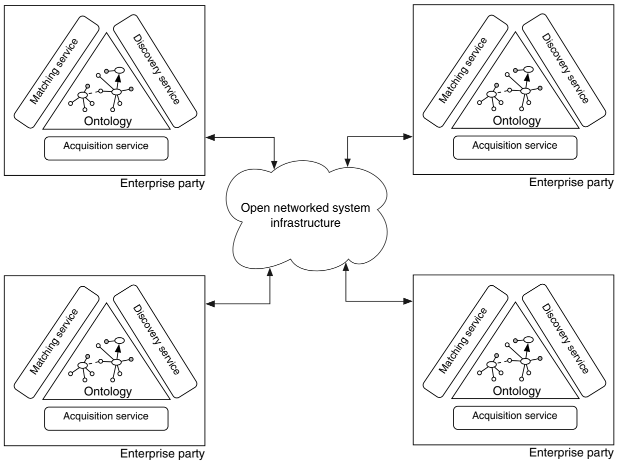


Fig. 2. Architecture of a open networked system for peer-based collaboration

the peer-based collaboration model, namely the *probe query* and the *search query*. A probe query is used for discovering potential collaborating enterprises, and contains specifications of target concepts describing the resources of interest. A search query is used in order to acquire resource data related to one or more target resources, once a collaborating enterprise has been identified. In order to support ontology-based resource sharing and discovery, each enterprise party realizes three *semantic interoperability services*:

- The *discovery service*: it is responsible for performing the activities related to probe query composition and propagation, and of invocation of probe query processing. The discovery service is invoked by an enterprise party in order to discover which collaborating parties within the networked organization can provide relevant information resources with respect to a target request, based on ontology descriptions.
- The *matching service*: it performs probe query processing against the ontology of an enterprise party. The matching service is based on a set of flexible matchmaking techniques which enable each enterprise party to compare the incoming requests against its ontology in order to identify whether there are information resources (i.e., concepts) matching the target.
- The *acquisition service*: it is related to the accomplishment of the collaboration between two enterprise parties by realizing the acquisition of resource data. The acquisition services of two collaborating parties are responsible for managing search queries and for retrieving resource data by exploiting the internal services (e.g., Web services) for data access.

3 Semantic Interoperability Services

In this section, we describe the interoperability services by showing how they exploit ontology knowledge for supporting inter-enterprise semantic collaboration.

3.1 The Matching Service

The matching service is responsible for performing comparison of ontology concept descriptions for probe query processing. According to the ontology model presented in Sect. 2, the meaning of ontology concepts depends basically on the names chosen for their definition and on their contexts, that is, the properties and the relations they have with other concepts in the ontology. Different ontologies can describe the same resources using different modeling choices. For this reason, the matching service is based on a set of ontology matchmaking techniques capable of coping with different levels of detail in modeling the resources of interest, by considering various ontology elements separately or in combination. With the goal of providing a wide spectrum of metrics suited for dealing with many different matching scenarios, ontology matchmaking techniques support four different matching models: surface, shallow, deep, and intensive matching models. Each model calculates a semantic affinity value $SA_{c,c'}$ of two concepts c and c' which expresses their level of matching. $SA_{c,c'}$ and is produced by considering linguistic and contextual features of concept descriptions [6].

- *Linguistic features*. Linguistic features refer to names of ontology elements and their meaning. To capture the meaning of names in an ontology, the matching service refers to a thesaurus Th of terms and terminological relationships among them. Th is automatically derived from the lexical system WordNet [11]. In order to express the implication of terminological relationships for semantic affinity, a weight $W_{tr} \in [0, 1]$ is associated with each terminological relationship tr derived from WordNet.

- *Contextual features*. Contextual features refer to properties and concepts directly related to a given concept (i.e., adjacents) in an ontology. A weight $W_{sr} \in [0, 1]$ is associated with each semantic relation sr to denote the strength of the connection expressed by the relation on the involved concepts for semantic affinity evaluation purposes. The greater the weight associated with a semantic relation, the higher the strength of the semantic connection between concepts. Furthermore, a weight W_{sp} is associated with strong properties sp, and a weight W_{wp} with weak properties wp, respectively, with $W_{\mathsf{sp}} \geq W_{\mathsf{wp}}$ to capture the importance of the property in characterizing the concept for matching.

With respect to the matching models available in the matching service, the surface matching considers only concept names. The shallow matching considers concept names, concept properties, and information about the presence of cardinality constraints on properties. The deep matching model is defined to consider concept names and the whole context of concepts, in terms of properties and semantic relations. In addition to the previous model, the intensive matching includes also property values in the matching process. In each matching model, the semantic affinity evaluation is calculated as a weighted sum of linguistic and contextual features, whose relevance in the semantic affinity evaluation process can be properly established, by setting the weight of the linguistic affinity $W_{la} \in [0, 1]$.

We have implemented and tested such matchmaking techniques in the framework of HELIOS, our peer-based system where knowledge sharing and evolution is based on interactions among peer and on peer ontologies for resource description. A more detailed description of these matchmaking techniques is given in [4].

3.2 The Discovery Service

The discovery service performs all the activities related to composition, propagation, and processing of probe queries. The discovery service is based on a query/answer paradigm in which there is not a centralized authority managing the collaborations, and the involved parties are dynamically selected based on the semantic affinity of the information resources with the given target. Given a target request, the discovery service is invoked in order to query a set of prospective collaborating parties and evaluate the affinity with respect to the target using ontology descriptions. The receiving parties, compare the request against their ontologies and reply whether they can provide relevant information resources (i.e., resources matching the target). In particular, query management requires an expressive representation capable to support the description of target resources in terms of ontology concepts searched (target concept(s)), with possible properties and semantic relations. To this end, a reference query template for information resource discovery is provided in Fig. 3(a) and it is composed of the following clauses:

- *Find:* list of target concept(s) names.

- *With:* (optional) list of properties of the target concept(s).
- *Where:* (optional) list of conditions to be verified by the property values, and/or (optional) list of concepts related to the target by a semantic relation.
- *Matching model:* (optional) specification of the matching model asked by the requesting peer to process the query.
- *Matching threshold:* (optional) specification of the threshold value t, with $t \in (0, 1]$ to be used for the selection of matching concepts based on the semantic affinity value determined by the matching process. If a matching threshold is not specified in the query, the answering peer adopts its own default threshold.

At the same, the discovery service has to provide an expressive representation of query answers. A query answer can be considered as a list of matching concepts. As described in Fig. 3(b), the structure of the answer template contains the following clauses:

- *Concept:* name of the matching concept.
- *Properties:* (optional) list of properties of the matching concept.
- *Adjacents:* (optional) list of concepts related to the matching concept by a semantic relation.
- *Matching:* set of pairs $\langle$target concept, affinity value$\rangle$, specifying the target concept with which the matching concept matches, together with the corresponding affinity value.
- *Matching model:* specification of the matching model applied to process the query.
- *Matching threshold:* (optional) specification of the threshold value t, with $t \in (0, 1]$ used for the selection of matching concepts based on the semantic affinity value determined by the matching process.

3.3 The Acquisition Service

The acquisition service is responsible of the effective collaboration establishment between two enterprise parties. When a requesting enterprise has identified a relevant partner for the collaboration, it sends a search query in order to access and acquire data about shared information resources. Search queries contain the *Find*, *With* and *Where* clauses of the probe query template. An enterprise party enforces appropriate techniques to access its repositories containing resource data, in order to support search query processing.

- The *Web Service-based* approach. Each enterprise party provides a standard access to its shared information resources by means of a Web Service. Standard protocols (e.g., SOAP, WSDL) can be adopted by the acquisition service in order to interact with the Web Service and provide a seamless access to the underlying information resources. The acquisition service interacts with the Web Service by means of the SOAP protocol which supports well-defined XML-based message communications. The WSDL document provides the specification of the set of

Probe query template

Find	Target concept name [, ...]
[*With*	⟨Property name⟩ [, ...]]
[*Where*	Condition, ⟨related concept, semantic relation name⟩ [, ...]]
[*Matching model*	Matching model to be used]
[*Matching threshold*	$t \in (0, 1]$]

(a)

Probe answer template

{*Concept*	Concept name
[*Properties*	⟨Property name⟩ [, ...]]
[*Adjacents*	⟨related concept, semantic relation name⟩ [, ...]]
Matching	⟨Target concept, affinity value⟩[, ...]
Matching model	Matching model name
[*Matching threshold*	$t \in (0, 1]$]}

(b)

Fig. 3. Reference template for (a) probe query and for (b) probe answer

methods which can be invoked by the acquisition service as well as the structure of the returned data extracted from the information resources.

- The *Wrapper-based* approach. The access to the shared information resources is provided by means of a wrapper service. The acquisition service interacts with the wrapper service by submitting target queries for information resource access. The wrapper service manages mapping rules defining the concepts of the enterprise ontology and the underlying information resources in order to reformulate the target query in terms of queries over specific structures of the repository where the information resources are stored (e.g., relational database structure). Finally, the answer to a search query sent back to the requesting party is an XML document containing the information about the shared information resources.

4 Exploiting Interoperability Services for Semantic Collaboration

In this section, we describe the use of interoperability services in the peer-based collaboration model with an application example.

4.1 The Peer-based Semantic Collaboration Model

In Fig. 4, we outline the main interactions, together with service invocations, between two collaborating parties in the peer-based model. An enterprise party invokes its discovery service when it intends to find potential collaborating enterprises of the networked organization capable to provide information semantically related to one or more target concepts of interests. According to the query template of Fig. 3(a), a

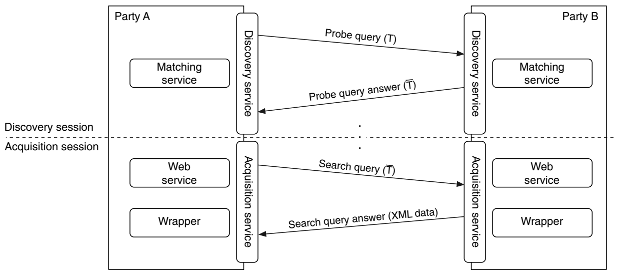


Fig. 4. Interactions and services in the peer-based collaboration model

probe query is composed by exploiting the discovery service containing the specified target concepts. Such a request is submitted to the other enterprise parties (i.e., B in Fig. 4). In the context of a peer-based collaboration, many enterprise parties can be involved in the organization, and if each probe query is propagated to all the nodes of the network, performance and efficiency of the overall system can drop dramatically. For this reason, the discovery service of the requesting enterprise party defines semantic routing rules in order to restrict the query propagation to a subset of nodes following semantic criteria: the probe query is sent to the parties whose ontologies are expected to contain relevant concepts with respect to the target query. Preliminary results on this topic are described in [5].

Receiving a probe query, the discovery service of a receiving enterprise party interacts with the matching service in order to identify if there are concepts matching the target request. In particular, the discovery service provides to the matching service an ontological description of the target concept(s) (extracting such information from the Find, With, and Where clauses), as well as the matching model and the threshold to apply (derived from the Matching model and the Threshold clauses, respectively). As a result, the matching service returns a (possibly empty) ranked list of concepts semantically related to the target, and, for each entry, the corresponding semantic affinity value in the range $(0, 1]$, computed by the ontology matchmaking techniques. Finally, the results of the matching service are organized according to the probe answer template in Fig. 3(b) by the discovery service, and such an answer is replied to the requesting enterprise party (i.e. the answer $\overline{T}$ in Fig. 4).

Collecting query replies from answering parties, the requesting enterprise evaluates the results and decides whether to establish a collaboration with the enterprise parties found to be relevant by the discovery service (collaborating enterprise(s)). To this end, the acquisition service is invoked to send an appropriate search query formulated over the matching concepts of each collaborating enterprise (i.e., the concepts provided in the probe query answer during the discovery session). The acquisition service retrieves data for a search query according to the information resource access approach supported by the answering party, and sends back resource data in an XML format (i.e., search query answer in Fig. 4).

4.2 Application Example

As an example of discovery in a peer-based collaboration, we consider networked organization composed of four enterprise parties, namely EP_1, EP_2, EP_3, and EP_4. As shown in Fig. 5, the ontology of each enterprise party contains concepts related to the travel domain. We suppose that EP_1 intends to discover whether any enterprise

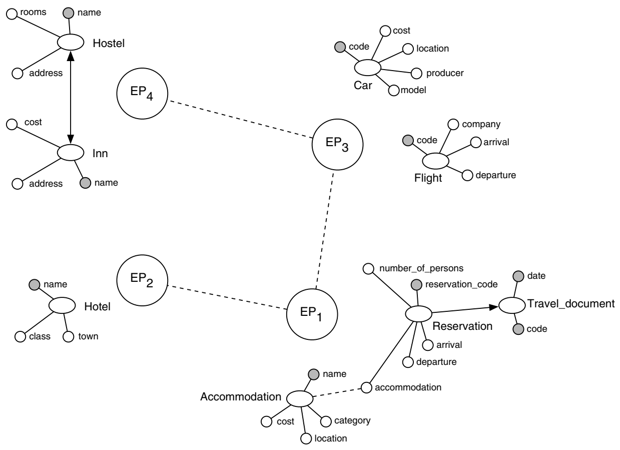


Fig. 5. An example of networked enterprise ontologies

party in the networked organization can provide relevant resources with respect to the Accommodation concept. To this end, EP_1 invoke the discovery service and submits to the system a probe query with the following clauses: *Find* Accommodation; *With* name, category, location; *Matching model* deep; *Matching threshold* 0.5.

Exploiting its semantic routing rules, EP_1 sends the query to EP_2 and EP_3. The discovery service of EP_2 and EP_3 catches the incoming query and invokes the matching service which performs ontology matching using the deep model to evaluate the semantic affinity between the incoming query and the concepts contained in each enterprise ontology. According to the matching service results, the discovery service of EP_2 replies to EP_1 with the following answer: *Concept* hotel; *Properties* name, class, town; *Matching* $\langle$accommodation, 0.75$\rangle$; *Matching model* deep; *Matching threshold* 0.5.

Following the same procedure, EP_3 does not identify matching concepts over the specified threshold in its ontology, since the matching results between the concepts flight and car and the target concept accommodation are 0 and 0.125, respectively. Nevertheless, by exploiting its semantic routing rules, EP_3 forwards the query to EP_4 which is expected to provide semantically related concepts. According to its matching service results, the discovery service of EP_4 replies directly to EP_1 with the answer shown in Fig. 6.

Concept	hostel	Concept	inn
Properties	name, address, rooms	Properties	name, address, cost
Adjacents	⟨inn, equivalentClass⟩	Adjacents	⟨hostel, equivalentClass⟩
Matching	⟨accommodation, 0.625⟩	Matching	⟨accommodation, 0.625⟩
Matching model	deep	Matching model	deep
Matching threshold	0.5	Matching threshold	0.5

Fig. 6. The answer provided by EP_4

5 Concluding Remarks

In this paper, we have presented an ontology-based collaboration model and semantic interoperability services for supporting discovery and sharing of heterogeneous information resources. Future research issues will regard the implementation of the semantic interoperability services. For what concern the matching service, we have developed H-MATCH, an algorithm for ontology matching and we are now testing performance and effectiveness of such an algorithm. In the context of the discovery service, we are working on the development of a semantic routing protocol, in order to enhance the discovery functionalities by taking into account information about semantic neighborhood among nodes for semantic query forwarding [5]. Finally, we are working on the definition of flexible techniques for the acquisition service capable to deal with different scenarios in information resource access.

References

1. H. Afsarmanesh, C. Garita, and L.O. Hertzberger. Virtual Enterprises and Federated Information Sharing. In *Proc. of the 9th Int. Conference on Database and Expert Systems Applications (DEXA 1998)*, Vienna, Austria, August 1998.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
3. P. Bouquet, B. Magnini, L. Serafini, and S. Zanobini. A SAT-based Algorithm for Context Matching. In *Proc. of the 4th Int. and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2003)*, pages 66–79, Stanford, CA, USA, June 2003. Springer Verlag.
4. S. Castano, A. Ferrara, and S. Montanelli. H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In *Proc. of the 1st VLDB Int. Workshop on Semantic Web and Databases (SWDB 2003)*, Berlin, Germany, September 2003.
5. S. Castano, A. Ferrara, S. Montanelli, E. Pagani, G. P. Rossi, and S. Tebaldi. On Combining a Semantic Engine and Flexible Network Policies for P2P Knowledge Sharing Networks. In *Proc of the 1st DEXA Workshop on Grid and Peer-to-Peer Computing (GLOBE 2004)*, Zaragoza, Spain, September 2004.
6. S. Castano, A. Ferrara, S. Montanelli, and G. Racca. From Surface to Intensive Matching of Semantic Web Ontologies. In *Proc. of the 3rd DEXA Int. Workshop on Web Semantics (WEBS 2004)*, Zaragoza, Spain, September 2004.

7. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to Map between Ontologies on the Semantic Web. In *Proc. of the 11th Int. World Wide Web Conference (WWW 2002)*, pages 662–673, Honolulu, Hawaii, USA, May 2002.
8. J. Broekstra et al. A Metadata Model for Semantics-Based Peer-to-Peer Systems. In *Proc. of the 1st WWW Int. Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID 2003)*, Budapest, Hungary, May 2003.
9. W. Nejdl et al. EDUTELLA: a P2P Networking Infrastructure Based on RDF. In *Proc. of the 11th Int. World Wide Web Conference (WWW 2002)*, Honolulu, Hawaii, USA, May 2002.
10. L. Li, B. Wu, and Y. Yang. An Ontology-Oriented Approach for Virtual Enterprises. In *Proc. of the 6th Asia-Pacific Web Conference (APWeb 2004)*, pages 834–843, Hangzhou, China, April 2004. LNCS.
11. G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM (CACM)*, 38(11):39–41, 1995.
12. Michael K. Smith, Chris Welty, and Deborah L. McGuinness (eds.). OWL Web Ontology Language Guide, 2004. W3C, http://www.w3.org/TR/owl-guide/.

Interoperability through integrating Semantic Web Technology, Web Services, and Workflow Modeling

John Krogstie[1], Csaba Veres[2] and Guttorm Sindre[2]

[1] SINTEF and Department of Computer and Information Science Norwegian University of Science and Technology (NTNU) `krogstie@idi.ntnu.no`
[2] Department of Computer and Information Science Norwegian University of Science and Technology, `veres@idi.ntnu.no, guttors@idi.ntnu.no`

Summary. A number of technologies are mentioned under the rubric of "The Semantic Web", but good overviews of these technologies with an eye toward practical applications are scarce. Moreover, much of the early focus in this field has been on the development of representation languages for static conceptual information, while there has been less emphasis on how to make semantic web applications practically useful in the context of knowledge work. To achieve this, a better coupling is needed between ontology, service descriptions and workflow modeling. This paper reviews all the basic technologies involved in this, and outlines what can be achieved by merging them in the context of real world workflow descriptions.

1 Introduction

"The Semantic Web" [1] is seen as the next generation of web systems, providing better information retrieval, better services, and enhanced interoperability between different information systems. The Semantic Web initiative is currently overseen in the semantic web activity of the W3C (http://www.w3.org/2001/sw/) and includes a number of core technologies. Some core technologies that will be relevant to this overview are XML, RDF, RDF/S, OWL, and Web Services (SOAP, WSDL, UDDI). While these technologies are promising, it can still be argued that alone, they are not sufficient to achieve interoperability in the business domain, allowing for a smooth integration between different information systems within and between organizations. For this to be accomplished, it is not enough to describe ontological metadata about the information and services available – one also needs to know the work context in which the different types of information and services are requested. Hence there is a need to integrate ontologies and service descriptions with models of workflows and business processes. The purpose of this paper is as follows:
To provide a survey (not a review) of the relevant technologies (ontology, service models, workflow models).

To show how these technologies fit together, both in theory (presented as something called “The interoperability pyramid”) and in practice (illustrated by an example of how the technologies could be used).

The rest of this paper is structured as follows: Sections 2-4 survey ontologies, service models, and workflow models, respectively. Section 5 then presents an integrated approach to enterprise and IS development, where interoperability among the various systems (and enterprises) would be a major focus. Finally, section 6 provides some concluding remarks.

2 Ontology

The lower level technologies XML, RDF, and RDF/S are less relevant to us due to their shortcomings for semantic annotation. Also, we assume that they are well known (or otherwise easily investigated by the reader looking at other sources), so we jump directly to the level of ontology languages.

A good starting point for understanding what ontology entails, is to consider Figure 1, adopted from [2], which places a number of knowledge models on a continuum. As you go from the lower left corner to the upper right, the richness of the expressible semantics increases. This is shown on the right side of the arrow with some typical expressions that have some sort of defined semantics for the particular model. The names for the knowledge models are given on the left of the arrow. It is important to note that all of the terms on the left hand side have been

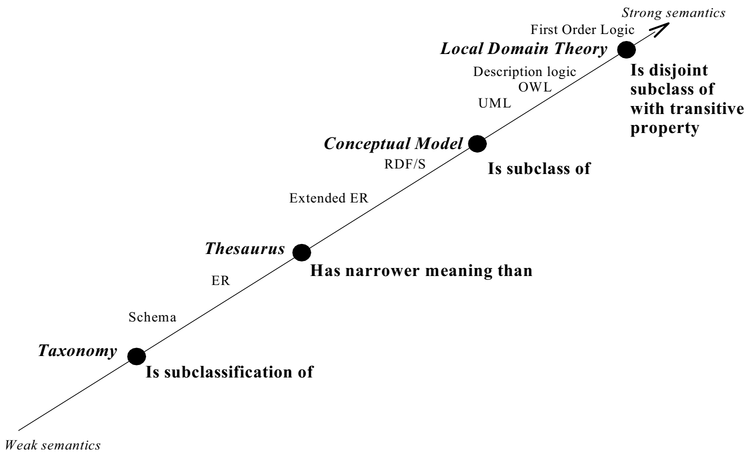


Figure 1. The ontology spectrum

called “ontology” by at least some authors, which is part of the source for confusion about the word.

Models based on the various points along the ontology spectrum have different uses [4]. In the simplest case, a group of users can agree to use a controlled vocabulary for their domain. This of course does not guarantee that they will use the terms in the same way all the time, but if all the users including database designers chose their terms from an accepted set, then the chances of mutual understanding are greatly enhanced.

Perhaps the most publicly visible use for simple ontologies is the taxonomies used for site organization on the World Wide Web. This allows designers to structure information and users to browse and search. Taxonomies can also help with sense disambiguation since the context of a term is given by the more general terms in the taxonomy.

Structured ontologies provide more sophisticated usage scenarios. For instance, they can provide simple consistency and completeness checks. If all *products* must have a *price* then web sites can automatically be checked for missing or conflicting information. Such ontologies can also provide completion where partially specified information can be expanded automatically by reference to the terms in the ontology. This expanded information could also be used for refining search, for instance. Ontologies can also facilitate interoperability, by aligning different terms that might be used in different applications [4].

Now we are in a position to see why the ontologies on the most formal end of the spectrum are often taken as the default interpretation in the context of the semantic web, providing the conceptual underpinning for " ... making the semantics of metadata machine interpretable" [5]. But for the semantics of a domain model to be machine interpretable in any interesting way, it must be in a format that allows automated reasoning in a flexible manner. Obviously, taxonomies can specify little in this sense. Database schemas are more powerful, but limit the interpretation to a single model. The only automated reasoning that can be performed is what is allowed by the relational model, and the semantics can only be understood through complex inferences supplied by humans. Formal logic based ontologies provide multiple possible models allowing machine based inferences, but still limit the set of formal models to the set of intended meanings. They are at the same time more formally constrained and more semantically flexible than database schemas. Ontologies based on different logical models can support different kinds of inference, but a minimal set of services should include reasoning about class membership, class equivalence, consistency, and classification [3].

The ontology representation language adopted by the Web Ontology Working Group of the W3C[1] is the Web Ontology Language (OWL). OWL is a response to a number of requirements [6] including the need for a language with formal semantics that enables automated reasoning, and to address the inherent limitations of RDF/S.

2.1 OWL

According to the original design goal, OWL was to be a straightforward extension of RDF/S, guaranteeing downward compatibility such that an OWL aware processor could also understand RDF/S documents without modification.

[1]http://www.w3.org/2001/sw/WebOnt/

Unfortunately this did not succeed because the generality of some RDF/S elements (e.g. the semantics of *class* as "*the class of all classes"*) does not make RDF/S expressions tractable in the general case. In order to maintain computational tractability, OWL processors include restrictions that prevent the interpretation of some RDF/S expressions. The OWL specification defines three sublanguages: OWL Full, OWL DL, and OWL Lite: OWL Full is upward and downward compatible with RDF but OWL DL and OWL Lite are not.

The names of the three sub languages of OWL describe their expressiveness, keeping in mind a fundamental tradeoff between expressiveness, efficiency of reasoning, and support for human understanding. OWL Full has constructs that make the language undecidable. Developers should therefore only use OWL Full if the other two sub languages are inadequate for modeling the relevant domain, or if they wish to maintain full compatibility with RDF. Similarly, OWL DL should be used if OWL Lite is not sufficient. Details of the syntax and semantics can easily be obtained from the technical documentation web site of the W3C.

3 Web Services

There is a great deal of interest about web services and service oriented architectures in general. A useful definition can be found in [2]: "Web services are software applications that can be *discovered, described,* and *accessed* based on XML and standard Web protocols over intranets, extranets, and the Internet." This definition exposes the main technical aspects of web services, to do with discovery and description, as well as the role of WWW (e.g. XML) technologies for data exchange and communication. Also the definition is abstract enough to exclude low level protocols like RPC as web services. These core concepts along with the associated technologies are shown in Figure 2 below.

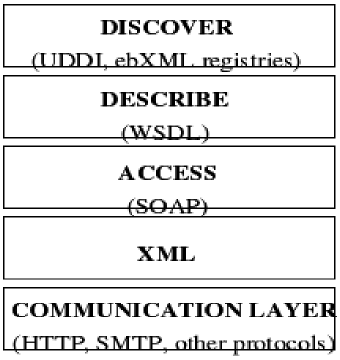


Figure 2. The basic layers of Web services [2]

It is important to situate the role of Web services in the real world. Daconta, Obrst and Smith [2] argue that the most important factor for determining the future of a new technology is not "... how well it works or how "cool" it is ..." but on business adoption. Along this line they see a bright future for Web services which is being promoted by Microsoft, IBM, Sun, as well as the open source community.

But why such widespread support? One reason is the promise of interoperable systems. Once businesses adopt standardized web service descriptions, the possibility of exchanging data and sharing the cost of services increases. In addition, the open standards prevent monopolization of applications, preventing the dreaded "vendor lock-in" associated with proprietary solutions. Finally, a widespread adoption of Web service protocols means that existing applications can be leveraged by turning them into Web services. As an example, it is even possible for .NET clients and servers to talk to J2EE servers using SOAP.

The point of all this is that Web services enable interoperability at the level of business processes without having to worry about interoperating between different applications, data formats, communication protocols, and so on. We will see in the next section that this influences the way workflows and knowledge based work processes are modeled and instantiated in particular work environments.

4 Workflow and Enterprise Process Modeling

The unprecedented flexibility of web services provides a particular challenge for how to integrate their use in enterprise work practices. On the one hand demand based service provision promises to be a blessing for facilitating problem solving; on the other hand, the instance based variability provided through the relatively free range of solutions offered in service composition could result in a serious challenge to established workflow modeling paradigms.

Process modeling implicates a family of techniques used to document and explicate a set of business and work processes in a way that allows their analysis at various levels, and for various purposes. Our specific interest in the current project is to use workflow modeling to analyze the work contexts that are likely to be involved in the day to day activities of an enterprise, with the aim of improving the timely delivery of appropriate information related resources and services. The purpose is to integrate workflow modeling with the potential of web services, to capture the likely usage scenarios under which the services will need to operate and to model this integrated use. The aim is that the model of work practices will allow better specification of actual information needs, which will in turn allow for richer requirements for the service descriptions expected from a web service, which will facilitate service composition and interoperability. The research problems therefore complement one another: workflow modeling helps web service design, but the availability of these services in turn improves workflow modeling techniques.

The challenge for us is to construct modeling approaches that maintain sufficient expressive power for the required points of view as well as to allow flexibility for changing situations. Jørgensen [7] argues that static workflow models cannot handle the changing demands of real world situations, and that adaptive models, while providing greater flexibility, still cannot adequately handle the instance based user driven modifications needed in many situations. He argues for interactive models that can be dynamically configured by users.

4.1 Workflow and Process Modeling Languages

Workflow modeling has been used to learn about, guide and support practice in a number of different areas including software process improvement [8], [9]; enterprise modeling [10], [11]; process centric software engineering [12]; and workflow systems [13]. The process modeling languages employed in these areas have been usefully categorized into one of the following types: transformational, conversational, role-oriented, constraint-based, and systemic [14]. In addition, [7] argues that UML based approaches need to be considered as a sixth category. A summary of each type is given in [7] where they are considered for their suitability as interactive modeling paradigms.

Transformational languages represent the majority of process modeling languages in use, adopting an input-process-output approach. Some well known languages adopting this approach are Data Flow Diagrams (DFD), Activity diagrams, Event-driven Process Chains (EPC), and Petri nets. While there are clear differences between the formalisms, it is possible to generalize in terms of their basic expressive commitments and therefore suitability for modeling dynamic, flexible workflows [15], [16], [17], [18]. The standards defined by the Workflow Management Coalition (WfMC) [13], the Internet Engineering Task Force (IETF) [19], and the Object Management Group (OMG) [20] are all predicated on a common perspective. We consider a few languages from this perspective.

The WfMC standards for process definition interchange between systems [21] include a large portion of the primitives involved in transformational languages. Processes are modeled with hierarchical decomposition, control flow structures for sequences, iteration, AND and XOR branching. Activities can be associated with organizational roles and actors, tools and applications. The core terminology of the WfMC is shown in [13]. Importantly, there is a distinction between process definition (idealized process) and instance (actual work),

The Business Process Modeling Language (BPML) [22] defines a *web service* interface description language, which presents obvious promise concerning the present requirements. BPML emphasizes low-level execution and contains several control flow primitives for loops (foreach, while, until), branching (manual choice or rule based switch, join), decomposition (all, sequential, choice), instantiation (call, spawn), properties (assign), tools (action), exceptions (fault), and transactions (compensate). The ability to define manual as well as rule based branching is promising for use in flexible systems. Unfortunately the promise is only partially realized since different primitives are used for the two cases, implying that the automation boundary must be defined during process design. Additionally, BPML has weak support for local change and unforeseen exceptions. A visual notation, for BPML, BPMN has been developed lately and is getting increasing attention.

Event-driven Process Chains (EPC) is a process language used in industrial systems, being part of the SAP ERP system and ARIS modeling tool [23]. EPC models units of work as *functions* enabled by pre-events and causing post-events. Relationships between events and functions are modeled by arcs, AND, XOR, and OR connectors. Some attempts have been made to map EPC to Petri nets [24] and UML activity diagrams [38]. EPCs some degree of freedom to represent alternative events, though the alternatives once again need to be determined at design time.

There is some recognition for the need to separate design components from run-time components for increased flexibility. This is realized in the WfMC's XML Process Definition Language (XPDL) [39]. But even here, the separation is focused mainly for facilitating the reuse of design components across different workflow engines and design tools. There is little support for user driven interaction at run-time.

It appears that current approaches are not designed with the flexibility required to accommodate the adaptive workflows that are enabled by Web Services technologies. One approach worth pursuing is the use of interactive models [7].

5 Integrating Enterprise and IS Development

The different approaches outlined above can be combined with more traditional model-driven system development to support a number of problem situations from very static, to very dynamic, even emergent processes. The different process types decide the extent to which the underlying technology can be based on hard-coded, predefined, evolving or implicit process models. This gives a number of development approaches. On one extreme; systems are manually coded on top of a traditional runtime environment, and on the other enterprise process models are used directly to generate solutions. In between these, we have the approaches typically described in MDA, namely the development of Platform Independent Models (PIMs) for code-generation (e.g. on top of a UML Virtual Machine), or for Platform Specific Models (PSMs) for more traditional code-generation.

In Figure 3, we outline the different types of interoperability across these different levels, being illustrated as a pyramid of different interoperating levels [27].

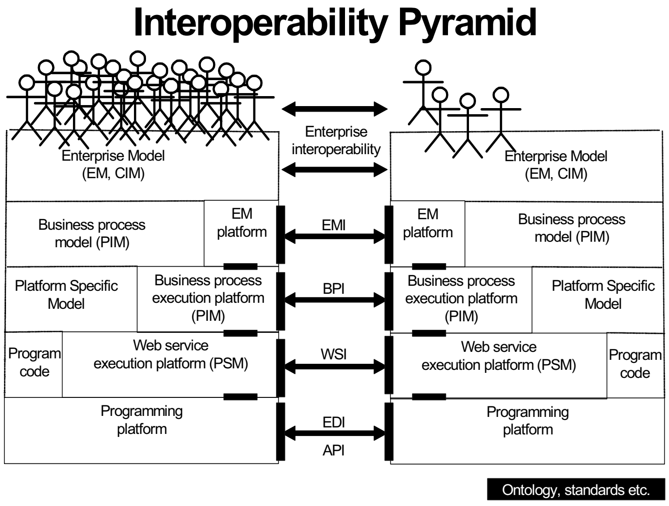

Interoperability Pyramid
Enterprise Model (EM, CIM)
Enterprise interoperability
Enterprise Model (EM, CIM)
Business process model (PIM)
EM platform
EMI
EM platform
Business process model (PIM)
Platform Specific Model
Business process execution platform (PIM)
BPI
Business process execution platform (PIM)
Platform Specific Model
Program code
Web service execution platform (PSM)
WSI
Web service execution platform (PSM)
Program code
Programming platform
EDI
API
Programming platform
Ontology, standards etc.

Figure 3. Interoperability between different platforms

Whereas traditional systems use special APIs and approaches such as EDI for interchange of data, on the next level (PSM), we can identify Web Services Interfaces. Above this level, there is a lot of work on specific business process execution platforms, with a possibility to exchange directly using a BPI. Finally, projects such as EXTERNAL (http://www.external-ist.org) have provided solutions for how to interoperate on the enterprise model level, using different modeling languages and different tools in the process. Standards and ontologies can be used across all levels to make the interoperation happen more smoothly.

We will try to clarify this picture with an example which originally had focus on the enterprise level and cross-organizational business processes to support dynamically networked organizations. The infrastructure to support networked organizations originally developed in the EXTERNAL IST project (which is currently further developed in the ATHENA (http://www.athena-ip.org), MONESA and WISEMOD projects) can be described as consisting of three layers. These layers are identified as:

Layer 1, the *information and communication technology* (ICT) layer: – defining and describing the execution platform, software architectures, tools, software components, connectivity and communication.

Layer 2, the *knowledge representation* layer: - defining and describing constructs and mechanisms for modeling, including ways of annotating models and meta-models with content from ontologies.

Layer 3, the *work performance and management* layer; - modeling and implementing customer solutions, generating work environments as personalized and context-sensitive user interfaces available through web-portals.

5.1 The ICT Layer

The ICT-infrastructure is an integration of the enterprise and process modeling tools, such as:

- METIS [28], a general purpose enterprise modeling and visualization tool,
- XCHIPS [29], a cooperative hypermedia tool integrated with process support and synchronous collaboration,
- SimVision [30], a project simulator used to analyze resource allocation, highlighting potential sources of delay and backlogs.
- WORKWARE [7] a web-based emergent workflow management system with to-do-lists, document sharing and process enactment.
- The architecture has 3-tiers, clients, application servers, and data servers. The implementation is web-based, utilizing HTTP both for control and data integration, and XML for data exchange. Three types of integration mechanisms are provided :
- Data-centered integration: based on a common EXTERNAL XML DTD, XML importing/exporting utilities are implemented in each of the enterprise modeling tools for data exchange between the tools or with an XML repository.
- Control-centered integration: uses the APIs provided by the tools and the repository to be integrated.

- Worktop-based integration: this is a service-oriented integration at the user-interface level, utilizing both data-centered integration and control-centered integration to access shared models, information objects, and tools.

New components can be developed on a need-driven basis, using the approaches of service oriented architecture (SOA) or model-driven architecture (MDA).

5.2 The Knowledge Representation Layer

The knowledge representation layer defines how models and meta-models are represented, used and managed. A version of Action Port Modeling (APM) [7] constitutes the core of the modeling language (EEML). The process logic is mainly expressed through nested structures of *tasks* and *decision points*. The sequencing of the tasks is expressed by the *flow* relation. *Roles* are used to connect resources of various kinds (people, organizations, information, and tools) to the tasks. Hence, modeling in EEML captures an extensive set of relationships between the goals, organizations, people, processes and resources. This is particularly useful considering the dynamic nature of networked organizations. For new partners joining the network, the rich enterprise models provide a valuable source of knowledge on how to "behave" in the network. We plan to further enable the annotation of such models with OWL-ontologies, improving model matching and model interoperability across organizations.

Moreover, the interactive nature of the models, meaning that the users are free to refine them during execution, increases their potential as sources of experience and knowledge. As such they *document* details on how the work was actually done, not only how it was once planned. EEML can be extended to link into for formal process modeling (E.g. BPMN) as well as languages used in the SOA and MDA approaches, particularly UML, through the meta-modeling features of METIS.

From a knowledge management perspective, process models are carriers of process knowledge; knowledge of how to do things. But through the possibility in EEML of attaching information resources to the tasks at any level, such a model also imposes a structure upon the set of information resources relevant for the work described by the process model. That way, the process models themselves form the basis for information management. Further extensions with semantic annotations based on OWL would enhance this even further, combining process ontologies with more traditional structural ontologies.

5.3 The Work Performance and Management Layer

Users access their solutions through project portals. A project portal for a networked organization must have support for methodology adaptation, communication, co-ordination and collaboration. Project management, reporting and other services must be offered, and finally project work must be performed with possibilities for repetition, providing security and privacy mechanisms.

In our infrastructure, the web-based portal registers and qualifies users, and invokes other tools through WORKWARE, an example of what we term a model-generated workplace (MGWP). The modeled tasks are also executed through the invocation of tools and applications from the web based user environment comprised of the portal and WORKWARE. WORKWARE sets up the context for each task, giving access to the knowledge and resources needed to perform the task. The actual work performance is done by invoking appropriate web services. The task performers may access desktop tools, organizational information systems, web services, or automated processes through this user environment.

User environments are generated dynamically based on the definition of tasks using EEML. Forms and components for interacting with different model objects are selected and composed based on user interface policies.

The dynamically generated work management interface includes services for work performance, but also for process modeling and meta-modeling. The *worktop* is the main component in this interface. In addition to the services for performing and managing the task, it contains links to all knowledge in the process models that is relevant for the task. Since the worktop is dynamically generated, subject to personal preferences, the skill levels of task performers can be taken into account,. Similarly, customized worktops for project management can support the project management team. The contents may include an overview of the project, adopted management principles, applicable methodologies, project work-break-down structure, results, plans and tasks, technologies and resources, status reporting and calculations. For further details see [7], [31].

6 Conclusion

This paper has provided a unified survey of relevant technologies for achieving semantic interoperability in the context of enterprise information systems, namely ontologies, service descriptions, and workflow models including both automated and interactive tasks. The Interoperability Pyramid, together with the example explanations of this, illustrates how the combination of these technologies can provide more advanced interoperability that with current systems. We suggest that "interoperability" in the abstract may be an untenable goal, at least in the immediate future. But interoperability in the context of dynamic and interactive workflows, as the next best thing, is very much within our reach.

Acknowledgements

Csaba Veres is funded by through the WISEMOD project (NFR IKT-2010). John Krogstie has been funded by the projects ATHENA (http://www.athena-ip.org) and MONESA. The ideas of this paper are pursued in the context of the EU NoE project INTEROP, and we thank other partners of INTEROP for valuable inspiration. The paper does not represent the view of E.C. or the ATHENA or INTEROP consortium, and authors are solely responsible for the paper's content.

References

1. Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web., Scientific American, 2001, May
2. Daconta, M.C., Orbst, L.J., Smith, K.T The Semantic Web, Wiley, 2003
3. Grigoris, A. & v. Harmelen, F. Web Ontology Language: OWL, in Handbook On Ontologies, Eds. International Handbooks on Information Systems, Springer, 2004, Berlin, Germany
4. McGuiness, D. L. Ontologies Come of Age, in Spinning the Semantic Web, Fensel, D., Hendler, J., Lieberman, H. Wahlster, W, Eds., MIT Press, 2003, Cambridge, MA.
5. Staab, S. & Studer R., Handbook On Ontologies, International Handbooks on Information Systems, Springer, Berlin, Germany, 2004
6. OWL Web Ontology LanguageUse Cases and Requirements, 2004, http://www.w3.org/TR/webont-req/, Accessed Aug. 23, 2004
7. Jørgensen, H. D., 2004, Interactive Process Models, PhD-thesis, NTNU, Trondheim, Norway, ISBN 82-471-6203-2
8. Bandinelli, S., Fuggetta, A., Lavazza, L., Loi, M., and Picco, G.P. Modeling and Improving an Industrial Software Process, IEEE Transactions on Software Engineering, vol. 21, no. 5, 1995.
9. Derniame, J. C. Software Process: Principles, Methodology and Technology, LNCS 1500, Springer, Berlin, Germany, 1998
10. Brathaug, T. A., and Evjen, T. A. Enterprise Modeling, STF 38 A96302, 1996, Trondheim, Norway
11. Fox, M. S. and Gruninger, M. Enterprise Modeling, AI Magazine, 2000
12. Ambriola, V., Conradi, R., and Fuggetta, A. Assessing Process-Centered Software Engineering Environments, ACM Transactions on Software Engineering and Metho, 6, 3, 1997
13. WfMC Workflow Handbook 2001, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2000
14. Carlsen, S Conceptual Modeling and Composition of Flexible Workflow Models, Norwegian University of Science and Technology, 1997
15. Conradi, R. and Jaccheri, M. L. Process Modelling Languages, in Software Process: Principles, Methodology and Tech, Eds. Lecture Notes in Computer Science, Springer LNCS 1500, 1998
16. Curtis, B., Kellner, M. I., and Over, J. Process Modeling, Communications of the ACM, 35, 9, 1992
17. Green, P. and Rosemann, M. Integrated Process Modeling: An Ontological Evaluation, Information Systems, 25, 3, 2000
18. Lei, Y. and Singh, M. P. A. A Comparison of Workflow Metamodels, in ER Workshop on Behaivoral Modeling, Eds., Springer LNCS 1565, 1997
19. Bolcer, G. and Kaiser, G. SWAP: Leveraging the Web to Manage Workflow, IEEE Internet Computing, 3, 1, 1999
20. OMG Workflow Management Facility v. 1.2, OMG, 2000
21. WfMC Workflow Management Coalition, Interface 1: Process Definition Interch, Draft 5.0, Workflow Management Coalition, 1996
22. Arkin, A. Business Process Modeling Language - BPML 1.0 Working Draft, BPML.org, 2002
23. Scheer, A.W. and Nuttgens, M., ARIS Architecture and Reference Models for Business Process Management, in Business Process Management, W.v.d. Aalst, J. Desel, and A. Oberweis, Eds., Springer LNCS 1806, 2000, Berlin
24. Aalst, W. M. P. v. d. Formalization and Verification of Event-driven Process Chains, Information and Software Technology, 41, 10, 1999.

25. Loos, P. and Allweyer, T. Process Orientation and Object Orientation - An Approach for Integrating UM, University of Saarland, Germany, 1998
26. Workflow Process Definition Interface - XML Process Definition Language, 2002, http://www.wfmc.org/standards/docs/TC-1025_10_xpdl,(Aug. 19, 2004)
27. Krogstie, J. Integrating Enterprise and IS-development Using a Model-Driven Approach. in 13th International Conference on Information Systems Development. 2004. Vilnius, Lithuania: Kluwer
28. Lillehagen, F., 1999, Visual extended enterprise engineering embedding knowledge management, systems engineering and work execution, *Proceedings of IEMC '99, IFIP International Enterprise Modelling Conference,* Verdal, Norway
28. Haake, J. M., and Wang, W., 1997, Flexible support for business processes: Extending cooperative hypermedia with process support, *Proceedings of GROUP '97*, Phoenix, Arizona USA
30. Kuntz, J. C., Christiansen, T. R., Cohen, G. P., Jin, Y., and Levitt, R. E., 1998, The virtual design team: A computational simulation model of project organizations, *Communications of the ACM*, vol. 41, no. 11
31. Krogstie, J., and Jørgensen, H. D., 2004, Interactive models for supporting networked organizations. *Proceedings of CAiSE'2004*, June 9-11, Latvia, Riga

An M3-Neutral Infrastructure for Bridging Model Engineering and Ontology Engineering

Jean Bézivin[1], Vladan Devedzic[2], Dragan Djuric[2], Jean-Marie Favreau[1], Dragan Gasevic[2] and Frederic Jouault[1]

[1] ATLAS Group (INRIA & LINA), University of Nantes, 2, rue de la Houssinière - BP 92208 44322 Nantes Cedex 3, France, bezivin@univ-nantes.fr, jeanmarie.favreau@free.fr,frederic.jouault@univ-nantes.fr

[2] GOOD OLD AI Group, FON – School of Business Administration, University of Belgrade, Jove Ilića, 11000 Belgrade, Serbia and Montenegro, devedzic@fon.bg.ac.yu, dragandj@gmail.com, dgasevic@acm.org

Summary. In this paper we report on some research activities in the ATLAS team in Nantes and the GOOD OLD AI research group in Belgrade in the domain of model-based engineering. We start from the idea that a convenient organization of various technical spaces in three "metamodeling" layers offers a convenient working environment. The main message of this paper is that it is possible to consider software engineering and ontology engineering as two similarly organized areas, based on different metametamodels (M3-level). Consequently, building bridges between these spaces at the M3-level seems to offer some significant advantages that will be discussed in the paper.

1 Introduction

Model driven engineering (MDE) is being considered as an important departure from traditional techniques in such areas as software engineering, system engineering and data engineering. In software engineering, there are two well-known supporting approaches: the MDA approach proposed by OMG and Microsoft's software factories [1].

Technical spaces have recently been introduced as a means to structure the solution space and to figure out how to work more efficiently by using the best possibilities of different technologies, including MDA [2] Nowadays, using only a single technology in solving different engineering problems is usually not enough. For example, software engineers can benefit from ontological engineering, or database developers can find useful improvements in using the XML technology.

In this paper we propose the idea that it should be possible to establish generic coordination between different technical spaces by making explicit the M3-level properties and providing domain-independent transformation facilities at this level. This would be more efficient than providing ad-hoc, case-by-case transformation between various DSLs belonging to the same or different technical spaces.

In section 2 we introduce some general considerations on the three-layer conjecture. Section 3 presents the domain of ontology engineering. In Section 4, we show how the idea of defining bridges between these spaces at the M3-level may bring a lot of significant economies and other advantages. Section 5 provides more in-depth thinking about mappings between ontology engineering and model engineering. In section 6 we compare the proposed M3 mapping techniques to more conventional M2-mappings. Finally, we summarize the project goals and sketching possible extension paths.

2 The 3-Layer Conjecture

In this section we recall the main characteristics of the three-layer conjecture and introduce the term of technical space. We are also discussing Model Driven Architecture (MDA), an important technical space widely accepted by software industry.

Model Driven Architecture (MDA) defines three viewpoints (levels of abstraction) from which a certain system can be analyzed. Starting from a specific viewpoint, we can define the system representation (viewpoint model). The representations/models/viewpoints are Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM) [3]. MDA is based on a four-layer metamodeling architecture and several complementing OMG standards (Figure 1). These standards are Meta-Object Facility (MOF) [4], Unified Modeling Language (UML) [5] and XML Metadata Interchange (XMI), and the layers are: meta-metamodel layer (M3), metamodel layer (M2), model layer (M1), and the real world layer (M0).

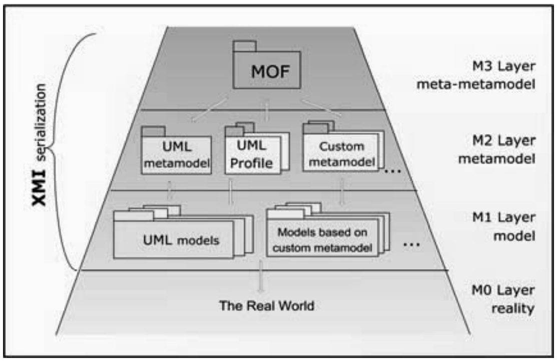


Figure 1. The four-layer Model Driven Architecture

The purpose of the four layers with common meta-metamodel is to support multiple metamodels and models and their scaling – to enable their extensibility, integration and generic model and metamodel management. All metamodels (both standard and custom) defined by MOF are positioned at the M2 layer. One of these is UML. The models of the real world, represented by concepts defined in the corresponding metamodel at the M2 layer (e.g. UML metamodel) are at the M1 layer. Finally, at the M0 layer are things from the real world. Although the "traditional" approach in MDA is to consider real-world things as instances of model elements, this was changed in newer approaches to a more natural approach, where model is a "snapshot" of reality [6] [7].

Technical spaces have been recently introduced as a means to figure out how to work more efficiently by using the best possibilities of different technologies [2]. A technical space is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities. Although some technical spaces are difficult to define, they can be easily recognized (e.g. XML, MDA, and ontology technical spaces in the case of approaching MDA and OWL).

In order to get a synergy of different technical spaces we should create bridges between them, and some of these bridges are bi-directional. The bridges can be created in a number of ways. Note that technical spaces can be classified according to their layers of abstraction (e.g. MDA and ontological engineering are high-level spaces, whereas XML and databases are low-level spaces). The Semantic Web integrates XML and ontological engineering technical spaces.

3 Ontologies, ontology languages and their architecture

Ontologies have been around for quite some time now. Nowadays, ontologies and ontological engineering span such diverse fields as qualitative modeling, language engineering, database design, information retrieval and extraction, knowledge management and organization, ontology-enhanced search, possibly the largest one, e-commerce (e.g., Amazon.com, Yahoo Shopping, etc.), and configuration [8].

Several ontology languages have been developed during the last few years [9]. In the first time, the most used ontology languages were Lisp-like ones, like Knowledge Interchange Format (KIF), Loom, Algernon, etc. By combining Semantic Web and XML technologies, many XML-based ontology languages have been defined like Ontology Exchange Language (XOL), SHOE, and Ontology Markup Language (OML). On the other hand, we have the languages Resource Description Framework (RDF) and RDF Schema as general languages for the description of metadata on the Web [10]. The Web Ontology Language (OWL) is a current semantic markup language for publishing and sharing ontologies on the WWW adapted by W3C [11]. OWL is developed as a vocabulary extension of RDF and is derived from the DAML+OIL Web Ontology Language. OWL has three variants: OWL Lite, OWL DL, and OWL Full. OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF and RDFS by providing additional vocabulary along with a formal semantics. More on ontological engineering can be found in [12].

Although there is no standard ontology architecture as the case is with MDA, ontology languages can be analized in the similar fashion – the architecture is shown in Figure 2. Of course, this figure could be unfamilliar to some ontology practitioners, as some of ontology languages do not make differences between meta-layers, while in some of them we can define that a class is an instance of the other class. However, developers mainly see ontology languages as we do, so we can say that the architecture in Figure 2 represents a pragmatic architecture of ontologies languages that allow us to find similarities to the MDA architecture.

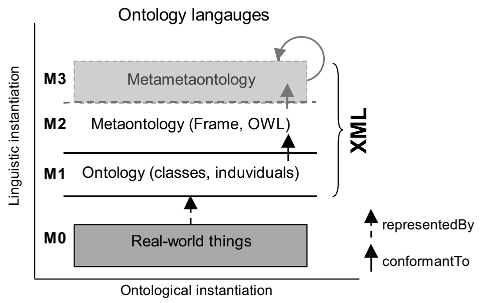


Figure 2. Meta-layers in terms of ontological engineering

Since most of practitioners are unfamiliar with ontology development techniques, many researches have proposed using well-known tools and techniques (e.g. UML) for ontology development as a solution of this problem [13]. The problem of using UML for ontology development has been firstly addressed in [14].

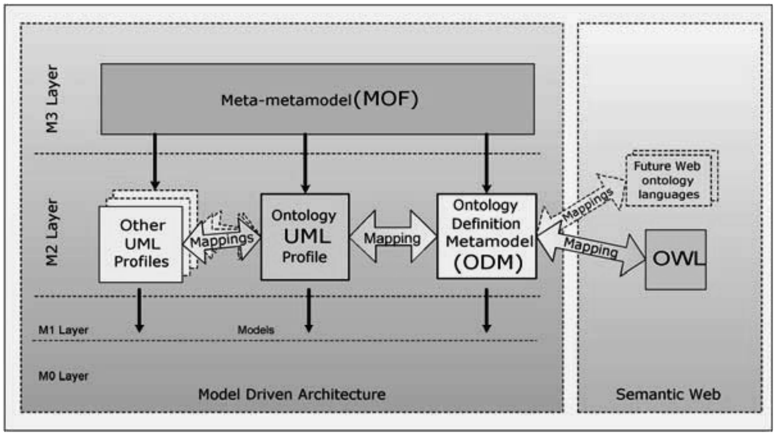


Figure 3. Ontology modeling in the context of MDA and the Semantic Web

Currently, there is an OMG initiative aiming to define a suitable language for modeling Semantic Web ontology languages in the context of MDA [15]. In the context of that initiative we present in Figure 3 our proposal for such an architecture[16]. The key components in the architecture are:
Ontology Definition Metamodel (ODM);

Ontology UML Profile (OUP) – a UML notation for ontology definition;
Two-way mappings between OWL and ODM, ODM and OUP, and from OUP to other UML profiles.

Starting from the main intention of ontologies to be used for sharing knowledge we can say that ontologies do not exist without any knowledge about surrounding technical spaces. According to the Semantic Web initiative ontologies are based on XML and RDF(S), and thus we use XML technologies for sharing ontologies. This also assumes that ontologies are related with grammarware and EBNF as MDA is. There is another relation between ontologies and grammarware – APIs for manipulation of ontologies (e.g. Protégé's API, Jena, etc.), or repositories of ontology-aware knowledge in the form of databases (e.g. DBMS technical space). On the other hand, the presented ontology metamodels (ODM and OUP) are MOF-compliant languages defined in the context of the MDA's metamodeling architecture. In that way, ontologies are connected with MDA technical space. Since we can use JMI for accessing to MOF-based repositories we have an alternative way that connects ontology technical space with grammarware technical space. Of course, we have not listed all possible technical spaces the ontology technical space has connection with. Our idea is to show the importance of defining relations between ontologies and other different technical spaces. In the rest of the paper we discuss the transformations between model engineering and ontological engineering.

4 Relations between different technical and modeling spaces

Generally, each technical space has different purposes. We can easily recognize some overlaps between different technical spaces. That means, that the same real-world thing can be captured by different technical spaces.

Additionaly, each technical space include one or more modeling spaces. Each modeling space is a context of one meta-metamodel. A technical space usually consists of one main modeling space and a number of additional modeling spaces which can be shared with other technical spaces. For example, MDA technical space has MOF modeling space as its main modeling space, but also deals with XML, which is a part of XML and EBNF modeling spaces. We can classify modeling spaces into two parallel groups:
Conceptual modeling spaces – are about conceptual (abstract or semantic) things, like models, ontologies, mathematical logics, etc. They are not interested in techniques for representation of sharing their abstractions. However, we must have some techniques to materialize (or serialize) those modeling spaces.
Concrete modeling spaces – have techniques that allow us to have more material (i.e. physical, syntactical) representation of conceptual things. The examples of these modeling spaces are different kind of grammars (e.g. KIF and XML for ontologies, XMI for MOF-based models).
According to the previous classification we give the following conclusions regarding mappings between different conceptual modeling spaces:

Mappings between two conceptual modeling spaces. We can conceptually define mappings at the conceptual level. The mappings defined at the conceptual level give recommendation for their implementation.
A physical implementation of mappings between two conceptual modeling spaces always must be done through a concrete modeling space.
From the fist statement we recognize logical relations between two technical spaces. That means, we find out epistemological equivalences between their central modeling spaces. Since we are trying to have mappings at the M3 layer we need to represent each modeling space as a three-layer architecture. Afterwards we specify their mutual mappings at the M3 layer. Those mappings should be propagated to lower meta-levels (i.e. M2 and M1) of the three-layer architecture.

The second conclusion states that we always must use concrete modeling space for implementing mappings between conceptual spaces. For example, implementation of mappings between ontological engineering and model engineering can be through the EBNF modeling space as both of these modeling spaces have XML bindings. However, the statement about implementation is also valid for mappings inside the same modeling space. For example, in case of mappings between ODM and UML (i.e. inside the model engineering technical space) we could implement through the EBNF modeling space. The EBNF modeling space we use in a case we are dealing with XMI documents. In this case we can use either XSLT or a programming language as an implementation tool. The DBMS technical space we use when we have a MOF-based repository stored in a DBMS. We can use a MOF2 QVT language in this case if the repository supports one of them.

5 Mappings between ontological engineering and model engineering

In this section we try to identify ways for connecting different technical spaces in the main emphasis on the relations between ontological engineering and model engineering. This problem is mainly being solved partially by defining a pair of transformations between languages from two different technical spaces we are going to connect to . That mainly means developing transformations according to M2 layer, so that we can transform models at M1 layer [17]. However, we believe that if we can achieve mappings between technical spaces at the M3 layer we can get many significant economies and other advantages. The most important one is the decrease of the number of necessary transformations for bridging different technical spaces.

5.1 Epistemological relations between technical spaces

In order to clarify mappings between ontological engineering and model engineering we start from the organizational architectures of their languages

(Figure 1 and Figure 2). From those figures we can deduce that either of them consist of three layers. Epistemologically, the corresponding layers are equivalent:

In order to illustrate this equivalence we use the MOF-based ontology language (ODM) as a language from the model engineering technical spaces (Figure 5). It is important to note that ODM is defined at M2 layer. On the side of ontological engineering we have a definition of an ontology language (e.g. OWL) at the M2 layer. Concrete real world models are at M1 layer and they consist of classes and their instances for both model engineering and ontology engineering technical spaces. That means, we must not have one ontological layer at M1 layer according to [6], and we have two ontological layers: one for classes and one for class instances (i.e. objects).

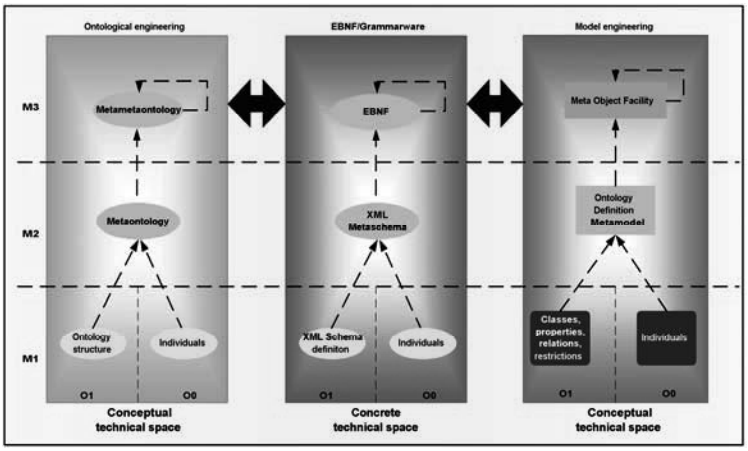


Figure 4. Mappings between ontological engineering and model engineering through the EBNF technical space

5.2 Pragmatic relations between technical spaces

We see here the high potential impact of considering these technical spaces as explicit and semi-formal entities. In most of these spaces we have internal transformation tools (e.g. XSLT and XQuery for XML, QVT for MDA, etc.). Some of these internal transformation tools are general and other are specialized (a compiler can be seen as a specialized transformation tool of the EBNF/Grammarware space). These transformation tools have evolved in their own context to fit with specific objectives and the main representation system of the corresponding space and there is no reason to change that. Now we have to consider another kind of transformation: across technical space boundaries. Let us call these transformation projectors in order to distinguish them from other transformations internal to one technical space.

The responsibility to build projectors lies in one space. The rationale to define them is quite simple: when one facility is available in another space and that building it in a given space is economically too costly, then the decision may be taken to build a projector in that given space. There are two kinds of projectors according to the direction: injectors and extractors. Very often we need a couple of injector/extractor to solve a given problem.

In order to illustrate this situation, let us look at the MDA technical space. The main entity there is a model (a metamodel may be considered as a kind of model). A model contains very useful and focused information, but by itself it is very dull and has no much capability. If we want MDA models to be really useful we have to give them these capabilities. There are two ways to do this: either to build them in the MDA space or to find them in another space. In the latter case what we'll have to provide is some set of projectors.

An MDA model is a graph (non directed graph with labelled edge ends). Since there was no possibility to exchange MDA models, the OMG started a RFP called SMIF (Serial Model Interchange Format). The objective of SMIF was to find a serialization scheme so that any kind of MOF model could be exchanged by simple means. Short after the time they come up with the solution, many people realized that it would be economically much more interesting to define standard serialization in XML. Instead of serializing graphs on text flow, the XML approach serializes graphs as trees and the let the remainder of the work being handled in the XML space. As a consequence a bidirectional projector was defined by the XMI convention.

Each MDA projector has a specific goal, to provide new facilities to models that are available in other technical spaces. XMI brings the capability of global model exchange to the MDA space and this capability is found in the XML space. Global model exchange means only the possibility to have batch-style of communication between tools. This is an interesting facility, but in many occasions it is not sufficient because we have to provide a fine grain access to model elements. XMI is of no use to do this. Here again the problem of adding new capabilities to models arose. Building intra-MDA tools for doing this was considered as very costly. So, as part of the Java community process program, a standard projector with the Java technical space was defined under the name JRS #40. The capability to access models elements in MDA was given with the help of the Java technical space. This projector is known today under the name JMI.

As we may see, every projector has a specific purpose. In the UML 2.0 initiative, the diagram interchange part deals partially with the separation of content and presentation for MDA models [18]. In order to help model presentation, specific tools could have been added to the MDA space, but with a high implementation cost. Here again a solution was found in the XML space, by using the SVG standard for scalable vector graphics. Although the solution is limited to only certain kind of models, here again we see the interest of using important investments of other technical spaces to bring economically and rapidly functionalities to a given space (here the MDA) with the help of projectors.

Many other examples could be found showing the need for a very precise definition of the goal of any projector. For example, after the introduction of XMI, it was rapidly found that this projector was not bringing the facility of easy textual reading to the MDA space. Many solutions were possible, including applying

XSLT transformation to XMI-serialized models to make them more usable for human operator (considering that XMI is sufficient for computer operators). Then the OMG decided to address this problem separately and a solution involving the EBNF space was defined under the name HUTN (Human Usable Textual Notation). HUTN offers three main benefits: (1) It is a generic specification that can provide a concrete HUTN language for any MOF model; (2) the HUTN languages can be fully automated for both production and parsing; (3) the HUTN languages are designed to conform to human-usability criteria. Later it was found that HUTN was not sufficient and that the set of projector should be completed in a bidirectional way with the anti-Yacc proposal [19] by DSTC. In the same spirit, SINTEF is today studying more general kinds of projectors between the MDA and the textual flat-file technical space [20].

So we can see all the gain that could be reaped from the homogeneous consideration of bridges between technical spaces with the help of generic projectors. There are many activities presently going on in this area with technical spaces like data base (SQL projectors, E/R diagram projectors), in the OS technical space (Unix projectors), in the legacy technical spaces (Cobol, ADA, PL/1 projectors to name only a few of them), in the natural language processing technical space for requirement engineering applications, etc.

One goal of the collaboration between the ATLAS group in Nantes and the GOOD OLD AI group in Belgrade is to define and build a set of generic projectors between the model engineering and the ontology engineering technical spaces.

6 Comparing the proposed M3 bridging to present ways of mappings

In this paper we have advocated the interest of factoring out the technical bridge work by considering relations at the M3 level. This is based on the conjecture that each space could exhibit such a M3 level. Much evidence can be found to support such a conjecture.

In this three-space conjecture, we assume the existence at level M3 of some form of representation ontology with a set of associated facilities. There is not infinity of possibilities at level M3 like there are at level M2. The representation systems are based on known algebraic structures with well defined properties. Very often these structures are based on trees, graphs, hypergraphs or categories. Mapping between these algebraic structures have been studied and their properties are known. For example transforming a graph into a tree is a reasonably known operation, even if there are many ways to perform it. Hypergraphs have been found very convenient to express some situations in visual languages, but mapping between graphs and hypergraphs have also been studied.

So, in some cases we can find entities in various technical spaces that nearly correspond semantically. In this case the gain is very high because we can apply generic M3-based projectors to do automatic conversion. At the other extreme of the spectrum, if we have two M2-based entities with no direct correspondence,

there is no more we can do. For example, if we want to match *MusicML* and *MathML*, there is no obvious way to do this.
So we see that there are really three levels of technology involvement:
Specialized: someone works only inside a given technology space, with a M2 commitment. For example a given team only works with the UML modeling language or the Java programming language considering that these offer sufficient expression capabilities for their work.
Generic: someone works inside a given technical space, with several M2-based commitments. This is happening mainly in the XML space, the software factories MDE space (with different DSLs), etc.
Universal: someone works with the possibility of mapping his/her problem on several technical spaces. A given information system problem for example may be more easily and generally solved by an XML solution, a Java solution, an MDA solution, an ontology solution. This level provides the best level of agility to solve computer engineering problems. This is the solution we are advocating here. It presupposes the existence of well designed generic projectors between the various spaces so that a given user may never become captive of a given technology.

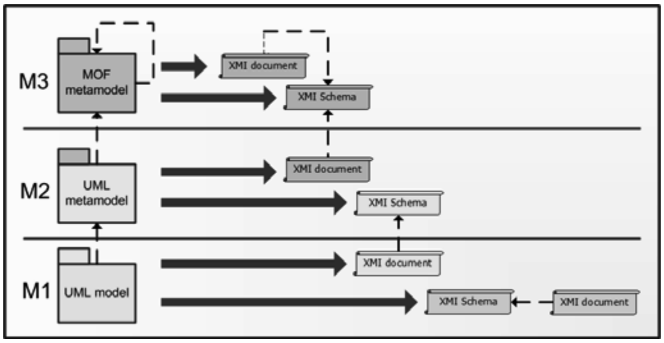


Figure 5. Mapping MDA metametamodel, metamodels, and models to XMI

Figure 5 can illustrate that current techniques understand that mappings from conceptual space to concrete space can be at different meta-levels. According to our approach we will have only one transformation between conceptual TS (model engineering) to a concrete TS (XML), so we will avoid complicated procedures.

It is obvious from the previous descriptions that we cannot provide direct mappings between the MDA technical space and the OWL technical space. In fact, this transformation can only be defined through the XML technical space. It is important to define a *pair* of transformations in order to enable two-way mapping (one transformation for either direction) between all OWL ontologies and all ontologies represented in an MDA-based ontology language. The transformations can be based on "meta-definitions" of OWL (i.e. on its meta-ontology) and an MDA-compliant language (i.e. a metamodel). This transformation principle is compliant with the principle of metamodel-based model transformation [17].

Table 1. Transformation approaches

	Target language					
		ODM		OUP		OWL
Source langua ge	ODM	–		XML TS	MDA TS	XML TS
				XSLT	QVT	XSLT
	OUP	XML TS	MDA TS	–		XML TS
		XSLT	QVT			XSTL
	OWL	OWL TS	XML TS	OWL TS	XML TS	
		RDQL	XSLT	RDQL	XSLT	

Table 1 gives some guidelines on how to make transformation between each pair of the languages discussed above.

7 Conclusions

We have presented here some initial work on the ways of bridging model engineering and ontology engineering. The main idea is that the M2-based solution seems quite costly and may not scale-up. We thus suggest using M3-based projectors to provide a general solution to this problem. Many benefits may be reaped from this approach. This paper has only presented our first investigations in this area, with some documentation evidence. We are planning to go ahead with the practical study of this increasingly important engineering problem.

The notion of technical space has been found to be essential in this investigation. However we must bring more conceptual formality to this study. A technical space is defined by a three-level organization, by a top-level representation ontology sometimes called a metametamodel, by a collection of related M2-level domain specific languages (metamodels or schemas or grammars, etc.) and by a number of facility implemented by widely available tools.

Building generic bridges at the representation level (i.e. the M3-level) seems a very promising engineering practice. We have provided some illustrations in support of this hypothesis. There is still much work to be done in this area. However if the general framework is shown feasible in these areas of model and ontology engineering, it may probably also be applied to many other areas as well. Several of the ideas presented in this paper are supported by initial prototypes developed in the research groups in Belgrade and Nantes. Work is going on in developing these prototypes, learning from previous experiments and building up a collaborative research on the subjects discussed in this paper.

Acknowledgments

Parts of this work has been supported by the "Interop" European network of excellence and by the "Modelware" IST European project 511731.

References

1. Greenfield, J., Short, K. , Cook, S., Kent, S. Software Factories : Assembling Applications with Patterns, Models, Frameworks and Tools. Wiley Publishing, September 2004, ISBN 0-471-20284-36
2 C. Atkinson and T. Kühne, Model-Driven Development: A Metamodeling Foundation, *IEEE Software*, Vol. 20, No. 5 (2003) 36-41.
3. Kurtev, I., Bézivin, J., Aksit, M.: Technological Spaces: An Initial Appraisal. Int. Federated Conf. (DOA, ODBASE, CoopIS), Industrial track, Irvine, 2002.
4. Miller, J., Mukerji, J. (eds.), "MDA Guide Version 1.0.1," OMG Document: omg/2003-06-0, http://www.omg.org/docs/omg/03-06-01.pdf, 2003.
5. Meta Object Facility (MOF) 2.0 Core Specification version 2.0 Final Adopted Specification, OMG Document ptc/03-10-04, http://www.omg.org/cgi-bin/apps/doc?ptc/03-10-04.pdf, 2003.
6. Unified Modeling Language: Superstructure, version 2.0, Final Adopted Specification, OMG Document ptc/03-08-02, http://www.omg.org/cgi-bin/apps/doc?ptc/03-08-02.zip, 2003.
7. C. Atkinson and T. Kühne, Model-Driven Development: A Metamodeling Foundation, *IEEE Software*, Vol. 20, No. 5 (2003) 36-41.
8. Bézivin, J.: In search of a Basic Principle for Model Driven Engineering, Novatica/Upgrade, Vol. V, N°2, (April 2004), pp. 21-24, http://www.upgrade-cepis.org/issues/2004/2/upgrade-vol-V-2.html
9. D. L. McGuinness, Ontologies Come of Age, *In D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster (eds.) Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential* (MIT Press, Boston, 2002) 171-194.
10. A. Gómez-Pérez and O. Corcho, Ontology Languages for the Semantic Web, *IEEE Intelligent Systems*, Vol. 17, No. 1 (2002) 54-60.
11. Corcho, O., Fernández-López, M. and Gómez-Pérez, A., “Technical Roadmap v1.0,” OntoWeb Consortium Deliverable D1, http://www.ontoweb.org/download/deliverables/D11_v1_0.pdf, 2001.
12. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider, L. A. Stein, OWL Web Ontology Language Reference, *W3C Recommendation* (2004) [Online]. Available: http://www.w3.org/TR/2004/REC-owl-ref-20040210/
13. V. Devedžić, Understanding Ontological Engineering, *Communications of the ACM*, Vol. 45, No. 4 (2002) 136-144.
14. Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M. and Smith, J. UML for Ontology Development. The Knowledge Engineering Review, 17(1), 2002. 61-64.
15. Cranefield, S. Networked Knowledge Representation and Exchange using UML and RDF. Journal of Digital information, 1(8), 2001. http://jodi.ecs.soton.ac.uk
16. Ontology Definition Metamodel Request for Proposal, OMG Document: ad/2003-03-40, http://www.omg.org/cgi-bin/doc?ad/2003-03-40,(2003)

17. D. Djurić, D. Gašević, and V. Devedžić, Ontology Modeling and MDA, *Journal on Object Technology*, Vol. 4, No. 1 (2005)
18. J. Bézivin, From Object Composition to Model Transformation with the MDA, *In Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems*, Santa Barbara, USA (2001) 350-355.
19. UML 2.0 Diagram Interchange, January 6 2003,
20. David Hearnden, Kerry Raymond, Jim Steel, Anti-Yacc: MOF-to-textD. Hearnden, K. Raymond, J. Steel. AntiYacc: MOF-to-text. In Proceedings 6th IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2002), pp 200-211.
21. Jon Oldevik, Tor Neple, Jan Øyvind Aagedal, "Model Abstraction versus Model to Text Transformation," In Proceedings of the Second European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations, Canterbury, England, September 7th-8th 2004
http://www.jeckle.de/files/UML2DIRevSub.pdf

Contract-based Interoperability for E-Business Transactions

Sergei Artyshchev and Hans Weigand

Infolab, Tilburg University, PO Box 90153,5000 LE Tilburg, The Netherlands sergei@uvt.nl, h.weigand@uvt.nl

Summary. Due to the heterogeneous nature of web services, interoperability is a crucial aspect, and this interoperability has not only a data aspect (ontologies) but also a process aspect. To ensure correctness of e-business processes, some kind of transaction support (e.g. WS-TXM) is necessary but not sufficient. In this paper, we define correctness in terms of contract compliance. Being contract-guided, e-business processes require a contract-dependent transaction protocol that can be represented as a set of formally specified agreements and obligations between participants. In current standards and frameworks, contract-based interoperability is only referred to, but not specified in details. This paper contributes to contract-based interoperability in the following aspects: for the first time it classifies characteristics for various degrees (levels) of contract-based interoperability, and it provides formal definitions of basic support operations - check and lock.

1 Introduction

Interoperability of web services is a crucial aspect of the enterprise integration. Interoperability can be loosely defined as the ability of enterprise software and applications to interact. True interoperability is more than connectivity and communication. It includes the possibility that one role performs some activity for the other role, and so it assumes that there is shared understanding of what the meaning of the request is: both the content semantics (activity name, parameters) and the pragmatics (the intended effect, e.g. that the other role executes the request or sends a reject message). This "shared understanding" can be implicit in the code, or be more explicit in an agreed-upon protocol definition, "collaboration agreement" (ebXML), or "contract". In this paper, we are interested in contract-based interoperability, defined as: "the ability of applications to interact and work together on the basis of a contract", where a contract is defined as: "an agreement between two or more roles that governs their interaction in terms of obligations and permissions". A contract need not be explicit, although this does have certain advantages. In principle, every interaction is contract-based, as every interaction assumes certain semantics/pragmatics of the communication to be in place, but typically these pragmatics are implicit in the standard protocol that is imposed from the beginning. Several frameworks (ebXML, WS-Coordination) provide the participants with the possibility to define new or extended protocols (agreements) with specifically defined semantics. When we use the term “contract” in this paper, we mainly refer to these user-defined agreements, but sometimes we also use it in the more general sense of the agreement underlying any interaction.

We will define different levels of contract-based interoperability, depending on the level of support of various contract manipulation procedures. Common procedures for contract manipulation include but are not limited to: contract establishing, contract verification, contract evolution during transaction execution, contract monitoring, sub-contract handling and inclusion, etc.

This paper is organized as follows: first we introduce the contract-based interoperability levels; then, section 3 introduces basic concepts to achieve contract-based interoperability, such as locking; section 4 provides operational semantics for these basic concepts, and the final section contains conclusions and future work guidelines.

2 Levels of Contract-based Interoperability

In the context of collaborative business development, contract-based interoperability of web services can be divided into six categories (see Figure 1). Both external and internal contract-related functionalities are the basis for classification. Each higher-level category includes functionality of lower level.

Level 0 indicates that no contract-based interoperability features are supported. Internal functionality of participants of this level allows execution of pre-designated operations, however, only final results (and, sometimes, intermediate status) are externalized. Other participants could locate its profile in the registry, but no communication except for service requests is possible. The web service van execute advertised functionality, but no interoperation protocol support is defined.

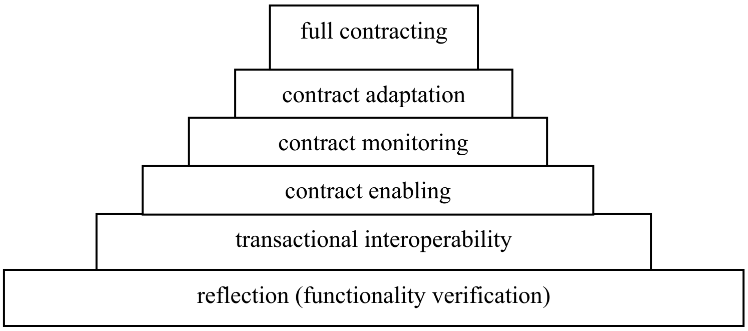


Figure 1. Six levels of contract-based interoperability

Level 1. A participant can not only **advertise,** but also **confirm (verify)**, if requested, its functionality and make a choice for the most appropriate operation (of the same type) to be employed at a run-time. Internally a participant should have a request processor which performs communication with other parties and, if necessary, a self-test mechanism that can generate latest status/performance relevant characteristics upon request. Externally, such a participant allows request processing (with appropriate authentication, if necessary), basic two-way communication, and a check operation (to be defined in section 3).

Level 2. A participant supports one or more transactional protocols. Support of level 2 interoperability indicates **transactional interoperability** – the capability to be engaged in transactions with some kind of (relaxed) ACID properties. To correspond to this level, a web service should support at least one transaction protocol and this capability should be advertised in a registry.

Level 3. The basic functions are in place that allow participants to make commitments and fulfill them. This assumes message-processing capabilities (to accept, process and respond to incoming messages), transactional interoperability, but, most importantly, what is supported at this level is outcome reservation (locking, to be defined in section 3), outcome determination, and other basic **contract-enabling** operations.

Level 4. Participants can **monitor a contract**. Monitoring contract assumes that participants are capable to understand, execute and verify compliance of other parties' activities to contract clauses. *Understanding* the contract means the capability to interpret contractual clauses (expressed in some XML-based contract definition language), and support the operations defined in the contract. *Execution* refers to the internal functionality to fulfill obligations assumed as part of a contract. Finally, *monitoring* itself refers to the capability to verify other parties' activities against contract clauses and response with contract-defined corrective actions. Contract monitoring has been the subject of several recent research projects [6,10,12].

Level 5. Participants cannot only execute a given contract, but also **adapt a contract** by means of negotiation and refinement. This level requires rather developed conversation capabilities and support of obligation-based contract composition. At this level participants are not yet assumed to establish a complete contract from scratch, rather, they should reuse already existing contacts (or templates), compose a contract from other contracts (as in supply chain scenario) or refine already existing contract with clauses and parameters relevant to the concrete business scenario. This level has been explored in [11] in the context of agent societies.

Level 6 Participants are able not only to refine contract templates, but also to **setup a new contract,** typically on the basis of explicit *goals* and preference structures from each participant. This functionality implies the use of goal-based negotiations and the ability to extract or compose contractual clauses from sources other then pre-defined templates and samples.

Those proposed interoperability levels could be utilized to characterize the level or degree to which a participant is interoperable as part of the architecture. A possible implementation might be to use these levels in the WSDL description in a registry thus facilitating the discovery of business partners with appropriate characteristics.

The contract-based interoperability framework can be used to assess current state-of-the-art technology. If we look at web service standards such as BTP

[14] and WS-Transaction [15], we can characterize them as level2. They do provide transactional interoperability by means of which parties can synchronize a certain event, but the business semantics of these synchronized events (in some cases called “BusinessAgreement”) are not specified. Therefore, there is also no monitoring of obligations, the only monitoring, if any, concerns the transaction protocol execution. If we look at current agent models, mainly confined to research labs, dealing with contracts, we can characterize them as level 4 to 5. Level6 technology doesn’t exist yet. However, contract drafting is a research topic in the area of negotiation support and e-commerce, see e.g. [7]. If we look at the ebXML framework, we can observe that in principle, it supports all levels of contract interoperability. However, in practice collaboration profile agreements (CPA) are still composed manually, and the notion of commitment or obligation is not explicit, so ebXML is better characterized as level2.

The objective of this paper is to close the gap between web service technology and agent theory by addressing precisely the level that is still not supported by current web service standards and, at the same time, is largely implicit in the agent models – that is, the basic level3 contract functions. We do not have a complete list yet of what these basic functions are, but we claim that the check and lock operations introduced in section 3 are essential. The formal semantics are given in section 4.

3 Locking as an Interoperability Mechanism

In the previous section we defined locking as a level3 contract-enabling function. In this section, we list the main requirements on e-business transactions [13] and then introduce our locking model. For a more elaborate discussion of the model, see [2].

3.1 Locking Requirements

The following requirements of e-business transactions distinguish them from advanced transaction models (ATM) defined in the ‘90s [3]:

- Participants are dynamic (volatile) – while initially advertising their functionality in UDDI-type registry, they might change as a whole or some of its characteristics only – therefore locks should address actual functionality, not assumed or initial;
- Participants are autonomous entities, sometimes exhibiting opportunistic behavior – locks could be removed not only by Requestor, but also by Provider itself;
- The transactions’ technical infrastructure (media, protocols, etc.) might be unreliable, creating issues of recovery and preservation of locks;
- Rather then being focused on execution speed, e-business transactions use schedules and timeouts (specified duration). While almost insignificant for

traditional transaction models, process execution here isn't instantaneous and is subject to coordination and scheduling.

- Participants have functional and capacity restrictions on their operations. Functional reflect internal business capabilities of the participants that are hard to change, capacity addresses participants' characteristics at the execution time.

Although their semantics have shifted since the time they were introduced in the context of database transactions, we claim that *locks* are still a valuable mechanism to arrange mutual reservation on participant's capacity. E-business locking guarantees that either a functionality is exercised as requested or the party, incurred losses due to another party' obligation un-fulfillment, is compensated, as it is agreed upon when the lock is applied or as specified in contract. The traditional business equivalents of e-business locking are (binding) quotations, contracts and frame contracts.

There is principal difference between database and e-business transactions based on participants' behavior. A database is a passive entity and the owner is reactive only. In business transactions the participants are dynamic and owners are active agents that can decide whether and how they will execute a certain request. In complex scenarios, there might be even a negotiation about behavior of locked resource before lock's application is attempted.

3.2 E-Business Locking Model

In order to cope with the requirements listed in section 3.1, e-business locks should be modified compared to those of ATM. Transaction preparation might be split into two sub-phases: prepare and locking [4]. Technically, e-business locking is an asynchronous message exchange between prospective participants and a change of the Provider's state in case lock is applied successfully.

Although the distinction between a prepare and a locking phase is made in several e-business transaction models, the exact meaning is not always clear [1]. We suggest the following characterization:

Prepare phase – at this phase we propose *functionality verification of prospective participants*, preceding both locking and execution. We assume this phase's activities verify functionality and capacity of prospective participants and don't impose any definite reservation, neither do they impose an obligation on any party (therefore, the terms lock and hold may be misleading). In case of completion or execution failure the preparation doesn't require compensation. Because the participant's profile, initially published in a registry might reflect actual functionality incorrectly (being outdated) or incompletely (containing insufficient information to invoke provider's functionality), the requesting party might want to check this information. Checking (or verification) is performed either by the provider itself (if it is trusted party) or by a third party.

While functionality restrictions are quite rigid, *capacity restrictions* vary due to resources' utilization by other parties. Exact capacity value (or, rather, available capacity) is correct only at lock application, however, any estimate performed before invocation also contributes to efficiency of execution scenario because it allows excluding potentially unavailable participants from the scenario without the need of a lock.

Locking Phase – some protocols employ explicit locking to ensure prospective participants' enlistment. A lock creates *commitments*. This phase follows the prepare phase and typically, it assumes the existence of a contract or similar agreement specifying the type of locks to be applied and their characteristics, but this contract can also be more general and implicit. The lock type should be supported both by requesting party and by functionality-proving participants.

As we said, checking applies either to a participant or the participant capacity (its available resources). The same is true for the locking (Table 1).

Table 1. Phase/object orthogonal representation

	PARTICIPANT	CAPACITY
PREPARE	**Check Participant**	**Check Capacity**
LOCK	**Lock Participant**	**Lock Capacity**

Participant lock serves as a gateway to capacity (or in case of web-services, operation) locks. It allows the use of functionality of the locked participant limited by arrangements specified at the time of locking. This type of lock might be the only one needed if the participant provides only one operation or if it can receive all necessary information to apply operations locks transitively. This kind of lock is not exclusive.
Capacity (operations) lock is an actual locking, it creates mutual obligations between participants. This lock is used as an execution correctness preservation mechanism. With the participants' autonomy, it might be up to the provider to designate, depending on request characteristics, an actual operation to be executed.

This orthogonal architecture provides the following benefits:
it allows transitivity of properties and functionality from participant (Provider) to capacity (resources) it controls, thus optimizing speed of locking (no need to provide additional information for every operation);
it minimizes cost and quantity of compensation. Application of participant/ capacity locks could be relatively extended in time, allowing reservation costs to be minimized. For example, a check can be performed in January, a participant lock in March (for the rest of the year), a capacity lock in June, when the production planning is finalized, while the actual execution is only performed in October. When the check in January fails, another supplier can be looked for. Similarly, when the participant lock fails in March, another pre-selected supplier can be chosen. Etc. In this way, the risks are minimized against minimal costs.

Both prepare and locking phase are preliminary to transaction execution, but their impact is quite different. While check verifies functionality and requests additional information, lock is applied upon known functionality; the check request is based on *advertised* functionality, while lock is based on *confirmed* (verified) functionality; in the case of locking, compensations might follow for unlocking, while check is a request for information with no compensations defined or needed. The provider is considered to be a *prospective* one before lock application and *actual* after.

4 A formal Model of E-Business Locking

In this section, we provide formal semantics for the locking and unlocking operations in terms of Dynamic Deontic Logic [8,9]. This logic allows for the specification of *actions* (locking, unlocking, requesting, compensating etc) and for the specification of *obligations* (which we need to model the commitment aspect of e-business locks).

4.1 Basic Definitions

Let L be a first-order language. DDL [8,9] is L extended with deontic operators (see below) and a dynamic operator. That is, if φ is a wff in DDL and α is an *action*, then $[\alpha]\varphi$ is a wff in DDL, with the intuitive meaning: φ holds after action α has been performed. The action can also be composite: $\alpha_1;\alpha_2$ stands for sequential, and $\alpha_1\&\&\alpha_2$ stands for parallel execution. $\neg\alpha$ stands for not-doing α.

In this case, we have at least the action *lock(r,id,t)* and *unlock(id),* where *r* is a resource, *id* a lock identifier and *t* a lock type (worked out below). The following minimal axioms hold:

Definition 1 (general semantics of lock and unlock)

```
∀r,id,t [lock(r,id,t)] lock-ed(r,id,t)
∀r,id,t lock-ed(r,id,t) ⇒ [¬unlock(id)] lock-ed(r,id,t)
∀r,id,t lock-ed(r,id,t) ⇒ [unlock(id)] ¬lock-ed(r,id,t)
```

These axioms just state that a resource is locked by a lock action, and remains locked until an unlock action is performed. We identify a lock by some unique identifier rather than by the resource as sometimes multiple locks on the same resource are allowed. Note that for simplicity we omit here any variable typing.

The lock operation is performed by the *Provider* (this agent is not included as a parameter, as we deal here only with one Provider at a time). If a *Requestor* wants a lock, he has to send a request message to the Provider. The Provider either accepts or rejects this request (negotiations are not in the scope of this paper). In the case of an accept, the Provider gets an obligation to perform the lock. In fact, this communication logic holds not only for lock but also for unlock and any other operation that P could perform.

Definition 2 (semantics of request)

```
∀R,P,m,α [request(R,P,α,m); accept(P,R,m)] Obl(P,R,α)
∀R,P,m,α ¬Obl(P,R,α) ⇒
    [request(R,P,α,m); reject(P,R,m)] ¬Obl(P,R,α)
```

In this case, the *m* acts as identifier of the request. Obl is the deontic operator for obligation. The most important property of Obl is expressed in the following axiom:

```
    Obl(P,R,α) ⇔ [¬α] Violated
```

which says that if P is obliged to R to perform some action, not doing that action leads to a *violation*. Note that this scheme is a bit naive about time; normally, some deadline will be specified, and the violation only arises when the action is not done before the deadline [5]. For the time being, our abstraction suffices. The semantics of request applied to locking results in:

```
    ∀R,P,m,α [request(R,P,lock,m); accept(P,R,m)] Obl(P,R,lock)
```

that is, the Provider is obliged to lock the resource. If this does not result in a locking action (which can never be excluded, given the autonomy of P), then P certainly has something to explain (within a certain marketplace or agent society, P may get trouble with the market owner).

4.2 Operational Semantics of Locking (Basic Model)

The operational semantics of locking are explored here by considering the intended lock properties one by one. Because of space limitations, we omit our definition of exclusive lock and define reserve lock only. Suppose that an "exclusive lock" has been set, then no one else can use the resource, but what happens when the legitimate requestor appears, with the right *id*. Does it mean that in that case the Provider is *obliged* to accept the request? We do not think this is necessary in all cases. That would mean that *P* has reserved *r* for *R*, and that *R* would be able to charge *P* if for some reason, *P* does not grant the request (perhaps because the resource is no longer available at all). If *R* wants a firm commitment from *P* that the resource is and remains available for him, this is something that is independent from the exclusiveness property. For this purpose, we introduce the notion of *reserve lock* as a conditional obligation.

Definition 4 (reserve lock)

```
    ∀r,id,P,R,m lock-ed(r,id,reserve) ⇒
        [request(R,P,r,m) && pass(R,P,id)] Obl(P,accept(P,R,m))
    ∀r,id,P,R,m lock-ed(r,id,reserve) ⇒
        [request(R,P,r,m); do(r)] Perm(P,unlock(id))
```

The first rule says that *P* becomes obliged to accept the service request. *P* could still for some reason fail to accept the request, but then this leads to a violation, and makes him liable for sanctions or compensations (see below). Note that we take *reserve* and *exclusive* to be orthogonal properties: although they will often go together, other combinations are also possible. For example, reserve but not exclusive: for the requestor in a e-business transaction, usually the most important thing is that he can use the resource, and putting an exclusive lock would be only a means to achieve that. The combination exclusive/no reserve may be useful for example in maintenance situations where some party wants to prevent the resource to be used for some time but without the intention to use it himself. The combination non-exclusive/no reserve is possible theoretically, but then there are no specified effects of the locking at all, and so the operation looses its meaning.

The second rule says that when the Provider has performed the requested operation, he is permitted[1] to unlock and thereby lift the obligation. This is the normal situation: the obligation is fulfilled when the service has been performed. However, the abnormal situations are also relevant and need to be specified. What about the Provider unlocking the resource on his own initiative? In a business context, this may very well occur, for various reasons. For example, because the production capacity of the Provider went down, or because a better-paying service request comes in from another Requestor. Because this may very well occur, it is customary to specify some compensation when it happens, either in the contract or together with the locking request.

If the Provider removes the reserve lock, this leads automatically to the lifting of the obligation (assuming a closed interpretation of the rule in definition 4), so we do not specify that explicitly. However, we do specify that self-unlock is forbidden, that is, leads to a violation. The consequences of that violation differ from one contract to another and can't be specified here.

Definition 5 (self unlock is forbidden, but not impossible)

```
   ∀P,id ¬Perm(P,unlock(id)) ⇔ For(P,unlock(id))
```

So unlocking is forbidden unless permitted (which is at least the case after successful performance, def. 4). We might want to specify in addition that self-unlock, if it happens, leads to an obligation to inform the requestor about this unhappy failure. This would be a good rule for a *locking message protocol*, but such a protocol is beyond the scope of this paper.

Intention and Commitment

Usually, a lock also expresses an intention of the requestor to use the resource. The Provider can see the lock operation as a sign that the resource is going to be used, and could anticipate on that. However, from the agent literature we know that the semantics of "intentions" is rather weak, so we refrain from formalizing it. What can be important in e-business transactions is that the Requestor commits himself to use the resource. A commitment is much stronger than an expression of intent. It means that the requestor is liable to charges (compensation) if he doesn't use the resource. This can be very relevant in business transactions: the Provider may loose interesting business operations because of the locking, and so the lock has an economic value. One could think of various ways to formalize this commitment. The following rule is again naive about timing, but suffices for the idea. Applying a reserve lock creates a commitment to use the service (unless unlocked later).

[1] Permitted is a strong version of the deontic P operator. It implies that doing the action does not lead to a violation (in other words, is not forbidden), but is stronger because it must be set explicitly, cf. the concept of authorization [8].

Definition 6a (reserve locking creates a commitment)

```
∀R,P,r,id,m
    [request(R,P,lock(r,id,reserve),m);accept(P,R,m)]
         locked(r,id,reserve) ⇒ Obl(request(R,P,r,_))
```

We must also account for the situation that the Requestor requests an unlock. In that case, he authorizes the Provider to unlock. Usually, when there is a penalty on not using the service, there is also a penalty for the Requestor on unlocking (otherwise, the first penalty could be easily circumvented), but the conditions and details often differ. For example, the conditions may say that unlocking is possible without penalty till a certain point in time. The following definition deals with unlocking by the Requestor. It says that the Provider should always grant an unlock request (he can not ignore it if that would be more convenient), but at the same time, it is regarded as something the Requestor should not do (that is, it is forbidden and liable to penalty).

Definition 6a (unlocking is forbidden, but not impossible)

```
∀R,P,id,m [request(R,P,unlock(id),m] Obl(P,accept(P,R,m))
∀R,P,r,id locked(r,id,reserve) ⇒
    For(R,request(R,P,unlock(id),_))
```

Compensation

In traditional transaction models (database and advanced) rollback brings participants to the state preceding the locking attempt, while in the case of failure of an e-business transactions return to the previous state might be unnecessary or impossible. Rather, compensation is used to bring the participants into some state specified by a contract.

In the formal model developed in this paper, compensation can be defined as an action specified in the contract that removes a violation of one of the obligations or prohibitions around locking. Note that in the above, we already have identified a number of possible violations, such as failure to deliver the locked service, failure to use the locked service and pre-emptive unlocking. Each violation requires a different compensation. The details of these are specified in the contract, or agreed upon at the time of the lock request, or may even follow from general law.

Definition 7 (violations can be compensated)

```
∀X,Y,γ,r,n (violated(X,r,n) ∧ (violated(x,r) ⇒
    Obl(X,γ)) ∧
    (¬compensated(X,Y,r,n)) ⇔ [γ]compensated(X,Y,r,n))
```

Here *X* and *Y* are participants (subjects), *r* is some rule identifier, *n* is used to identify a particular violation of that rule (the rule might be violated several times), and γ is the compensatory action specified in the contract. We assume here that the DDL is powerful enough to distinguish different violation predicates and is able to count their instances. Note that, strictly speaking, compensation does not remove the violation (which remains as an historical fact, so to say), but only specifies when it is compensated, which is sufficient. Note also that the compensatory action is itself an obligation, and when it fails it may necessitate additional compensations.

Participant lock
Up till now we have considered locking of resources (capacity) only. What is a reasonable semantics for participant locking? We propose to define participant locking as an authorization by means of which P commits himself to accept locking requests (perhaps conditionally, only when capacity is available at that time):

$\forall$`R,P,m,`α `[request(R,P,lock,m)] Obl(P,R,accept(P,R,m))`

This helps to increase the chance that the business transaction will be successful. It is not needed in all circumstances, but in some uncertain situations, it can be helpful.

5 Conclusions and Future Work

While technical aspects of transactions are currently addressed by several industrial standards and protocols, many contract business-related issues are left at the discretion and implementation of business partners, in many cases leaving interoperability issues open. Therefore, a general contract-based interoperability definition should be provided and mechanisms for its contractual specifications should be defined. This paper addresses transactions from contract-based prospective introducing five levels of interoperability and providing descriptions for their characteristics. The main focus is on contract enabling, which is implemented as a set of e-business transaction steps: check and lock.

Further development of contract-based interoperability involves adaptation of one of the available conversation languages to contract negotiation. We also want to express a need for more empirical research on the business requirements for web service technology.

Acknowledgements

This work has been partially supported by NWO (Netherlands Organisation for Scientific Research) and the European NoE INTEROP.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V. (2004) Web services – Concepts, Architectures and Applications. Springer, Berlin Heidelberg New York.
2. Artyshchev, S, Weigand, H. (2005) Interoperable transactions for e-business. Proc. IFAC’05, Prague.
3. Elmagarmid, A. (Ed.) (1995) Database transaction models for advanced applications Morgan Kaufmann, San Mateo.

4. Little, M. (2003) Transactions and Web Services *CACM*, 46(10): pp.9-54.
5. Marjanovic, O. Milosevic, Z.. (2001) "Figaro Should Be in Sydney by the 2nd of July" - Contracting in Many-To-Many e-Services. B. Schmid et al (Eds.): Towards The E-Society: E-Commerce, E-Business, and E-Government, (Proc. I3E 2001). Kluwer, pp.431-444.
6. Milosevic, Z., Dromey (2002) On expressing and monitoring behaviour in contracts. In: Proc. 6th Int. Conf. on Enterprise Object Computing (EDOC'02), Lausanne, Switzerland.
7. Tan, Y.H., Thoen, W. (2003) Electronic Contract Drafting Based on Risk and Trust Assessment". Int. Journal of Electronic Commerce 7(4), pp.55-72.
8. Weigand. H., Dignum, F.Verharen, E. (1997). Integrated Semantics for Information and Communication Systems. In: R. Meersman, L. Mark (eds), Database Application Semantics. Chapman & Hall
9. Wieringa, R.J., Meyer, J.-J.Ch., Weigand, H. (1989) Specifying dynamic and deontic integrity constraints, Data & Knowledge Engineering (4) 2, pp.157-191.
10. Weigand, H., Heuvel, W.J.A.M. van den. (2002) Cross-organizational workflow integration using contracts. Decision Support Systems, 33(3), pp.247-265.
11. Weigand, H., Dignum, V., Meyer, J.J.C. & Dignum, F. Specification by refinement and agreement: Designing agent interactions using landmarks and contracts. In Petta, P., Tolksdorf, R., Zambonelli, F. (Eds.), Engineering Societies in the Agents World III (ESAW 2002*)*. (LNCS, 2577, pp. 257-269), Springer, Berlin Heidelberg New York.
12. Xu, L.,Jeusfeld, M.A. (2003) Pro-active Monitoring of Electronic Contracts, In. Proc.15th Conference On Advanced Information Systems Engineering (CAiSE 2003), LNCS, Springer, Berlin Heidelberg New York.
13. Yang, J., Papazoglou, M.P. (2000) Interoperation Support for Electronic Business, *CACM*, Vol. 43, No. 6, pp. 39-47.
14. Business Transaction Protocol (BTP). An OASIS Committee specification V.1.0 June 2002; See http://www.oasis-open.org
15. Web Services Transactions(WS-T). Joint specification by IBM, Microsoft, and BEA, August 2002; http://www-106.ibm.com/developerworks/library/ws-transpec/
16. Web service Transaction Management (WS-TXM) Joint Specification by Arjuna, Fujitsu, IONA, Oracle, and Sun V1.0 July 2003, http://developers.sun.com/techtopics/webservices/wscaf/wstxm.pdf

Interoperability Middleware for Federated Enterprise Applications in web-Pilarcos

Lea Kutvonen, Toni Ruokolainen, Janne Metso, and Juha-Pekka Haataja

Department of Computer Science, University of Helsinki, Finland
{Lea.Kutvonen|Toni.Ruokolainen}@cs.helsinki.fi
{Janne.Metso|Juha.Haataja}@cs.helsinki.fi

Summary. Participation in electronic business networks has become necessary for the success of enterprises. The web-Pilarcos B2B middleware is designed for lowering the cost of collaboration establishment and to facilitate management and maintenance of networks. The web-Pilarcos architecture and middleware addresses the interoperability of autonomous enterprise applications in inter-organisational context. The approach is a federated one: All business applications are developed independently, and the B2B middleware services are used to ensure that technical, semantic, and pragmatic interoperability is maintained in the business network. In the design, attention has been given to the dynamic aspects and evolution of the network. This paper discusses the concepts provided for application and business network creators, and the middleware knowledge for interoperability support.

1 Introduction

The globalization of business and commerce makes enterprises increasingly dependent on their cooperation partners; competition takes place between supply chains and networks of enterprises. In this competition, the flexibility of enterprise information systems becomes critical. The IT systems and development teams should be able to respond in a timely way to the requirements rising from the changing cooperation networks and their communications needs.

From the computing infrastructure side, the enterprise needs can be addressed by an architecture where business level services, B2B middleware, and abstract communication services are clearly separated from each other, and the relationships between collaboration life cycle, B2B middleware, and software engineering tools are different from those found in the traditional approaches. By B2B middleware we mean general infrastructure services that provides concepts and operations for forming electronic business networks, eCommunities, and managing their life cycle.

The B2B middleware concepts and operations should be such, that strategical, process-related and technological needs of electronic business network management is filled. Such needs we believe include the following

- form new business networks that provide added value services for clients;

- join to multiple networks at the same time without unnecessary restrictions on technologies or operational policies;
- take up new business processes and services cheaply;
- move existing business networks to new phases of life cycle so that new collaboration forms can be used;
- monitor the progress and correctness of the collaborative processes;
- automate some collaboration establishment and correction events; and
- protect local services and computing solutions from the changes and failures of the collaboration partner services and solutions.

Traditionally, inter-enterprise collaboration has required integration of enterprise computing systems or applications. The topical integration techniques vary from new generation ERP systems, process-orientation to distributed workflow management systems. At present, a significant amount of research is focusing on virtual enterprise approaches. Virtual enterprises are joint ventures of independent enterprises joining a shared collaboration process. In many projects, like PRODNET [1], MASSYVE [2], FETISH-ETF [3] and WISE [4], the support environment consists of a breeding environment and operational environment. The breeding environment provides facilities for negotiating and modeling the collaboration processes; the operational environment controls the enactment of the processes. Many of the virtual enterprises' support environments use a unified architecture approach: there is a shared abstract model to which all enterprises have to adapt their local services.

In contrast to this, the approach in the web-Pilarcos project is federated: enterprises seek out partners that have services with which they are able to interoperate (within the strategically acceptable limits). A collaboration model (business network model, BNM) is used for explicitly expressing what kind of collaboration is wanted and comparison of BNMs is used as a semantic interoperability verification tool. Enactment of services and local business processes, either by applications or local workflow management system are required features of the service management facilities of each local computing system. This design choice has been made in order to make the evolution of BNMs and business networks themselves more flexible. Changes in the model to follow require that the model is explicitly available at the operational time, and that there is a synchronization and negotiation mechanism for partners to reach a safe point where new rules can be adopted.

The contents of this paper are a follows: Section 2 discusses interoperability challenges in the context of eCommunity management, and Section 3 briefly describes the web-Pilarcos B2B middleware services and repositories. Section 4 addresses the information repositories presented by the web-Pilarcos middleware. Section 5 discusses methods of finding interoperability problems and potential reactions on them.

2 eCommunity Management and Interoperability

The web-Pilarcos architecture proposes a model of inter-enterprise collaborations as eCommunities comprising independently developed business applications. The applications represent local business services and processes, and are able to collaborate with other enterprises within those limits.

The strategical requirements of a business network member towards the collaboration are expressed as a meta-level model that defines a set of external business processes. The structure is defined in terms of roles and interactions between the roles. For each role, assignment rules define additional requirements for the service offer that can be accepted to fulfill it, and conformance rules determine limits for acceptable behaviour during the eCommunity operation. The explicit use of such model allows comparison and matching of strategic pragmatical goals of members in the network.

Interoperability is a functionality provided by the middleware services, a transparent aspect for application services. Interoperability checking takes place when establishing a community, or entering a new service into an existing community. The applications themselves need only to concentrate on the local business logic, implemented on their local computing platform. Collaboration and eCommunity membership aspects, together with pragmatic process-awareness, however, require application level concepts and services. The inter-enterprise collaboration management concepts supported by the web-Pilarcos architecture include those of

- an eCommunity that represents a specific collaboration, its operation, agreements and state; the eCommunities carry identities and are managed according to their eCommunity contract information;
- services that are provided by enterprises, used as members in eCommunities, and are made publicly available by exporting service offers;
- a set of B2B middleware services for establishing, modifying, monitoring, and terminating eCommunities, or looking from the application service point of view, operations for joining and leaving an eCommunity either voluntarily or by community decision; and
- a set of repositories for storage of meta-models for communities, ontologies of service types, and services.

The eCommunity life cycle is mainly controlled in a eCommunity contract. The contract comprises of the BNM (to define the network structure), information about the member services at each role, some overview state information about the progress of the external business processes, and methods for changing the contract itself.

The eCommunity contract captures shared meta-information about the collaboration; reflective methods are used to keep the real system at each involved computing site correspondent with the meta-information. At each administrative computing domain, there is a local agent for management of knowledge about locally deployed. The local management interfaces are homogenized by a protocol for requesting the system to prepare for running a service (resourcing), querying about communication points, releasing the service, etc. Likewise, all relevant changes in the real system are notified and thus change the meta-information accordingly. The eCommunity contract is an active object itself, and includes logic that may react to changes in the meta-information and request local sites for further negotiations or changes in the system state.

Monitoring interoperability during eCommunity lifetime requires sensors and guards at each communication channel end. We assume an abstract communication infrastructure with selectable transparencies and support for non-functional aspects. From the service specifications it is known what traffic should be seen and in which order; in principle the rules can be extended to view the acceptability of contents structures and making trust related decisions.

3 The web-Pilarcos B2B Middleware Architecture

The B2B-middleware platform provides a) advanced service discovery based on improved services typing and constraint based selection, b) contract based management of collaboration between autonomous services, and c) proactive local monitoring of contract conformance.

The service elements of the web-Pilarcos architecture address the need for joining four important processes: a) introduction of BNMs to the model repository, and introduction of supporting service types to the type repository; b) software engineering processes to provide implementations that correspond to the known service types and thus are applicable for the known BNMs; c) deployment of services and export of corresponding service offers to traders, effectively making a commitment to keep the service consistent with the service offer; d) eCommunity establishment process using the provided information.

These processes are only loosely interleaved. Business network models and the actual application services can be developed independently from each other; indeed their development form a quite separate profession. In the platform, these concepts have to meet at the service description level.

The B2B middleware elements are illustrated in Fig. 1. The BNM design process involves introduction and verification of new models to be stored into the repositories. Implementation of new services or introduction of legacy applications involves interaction with the type repository. Deployment processes are naturally augmented with service offer exports. These processes feed in meta-level knowledge of potential participants in communities to be formed. The feeding processes are independent from each other, even withdrawing or deprecating information may take place.

The functional elements presented in Fig. 1 address the eCommunity life cycle management operations. The *Populator* uses a given BNM for ensuring the pragmatic interoperability of partners to a eCommunity; it also uses a set of compulsory aspects in service offers to determine service types, communication channel requirements, and non-functional aspects to be agreed on for the eCommunity. The populator represents a breeding process where services are selected for eCommunity roles. The population process is a constraint satisfaction challenge between candidates' attribute value spaces and constraints given for roles in the business network model. The service type definitions dictate the attributes and attribute value sets necessary to describe the service, and the actual values for each published service are found in service offer repository. As there are dependencies between selected offers in interacting roles (on channels and NFA), the process is complex. The populator provides its clients with a set of interoperable communities from which to choose during negotiations. Replacement of partners in an existing community, or one partner changing to a significantly different service implementation are also situations where interoperability preconditions need to be checked.

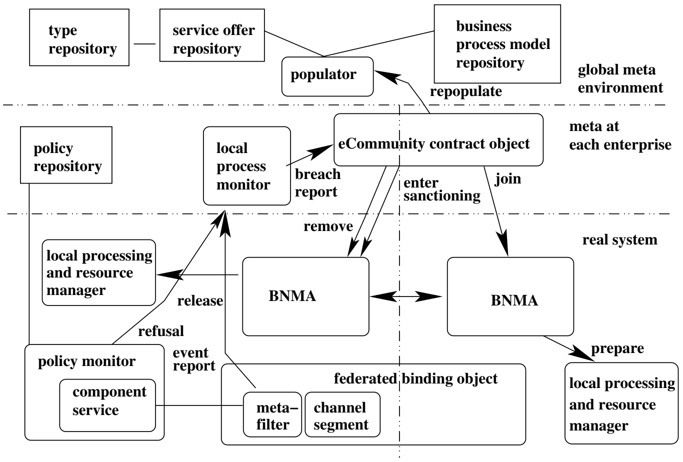


Fig. 1. The web-Pilarcos B2B middleware architecture, and flow of messages when a participant refuses a service due to policy conflict. Component service is the actual implementation for the functionality of a certain eCommunity role.

The eCommunity management is done in cooperation with *Business Network Management Agent (BNMA)* and the *Contract* object. The agents are responsible for managing the inter-organizational coordination and management protocols. The contract object is responsible for making decisions regarding the eCommunity it represents. At each administrative domain, there is a BNMA agent acting as a representative between the eCommunity and the local service-providing system. For local administrators the agents provide a management interface for communities. Between themselves the agents have a protocol for notifications of task completions and contract breaches, and negotiation and commitment protocols for joint contract changes. Each local agent receives notifications of contract breaches and task completions from local monitors and propagates this information forward to other agents as needed. Local agent also feeds monitoring instructions to the monitor.

The *eCommunity contract* itself is a key element in the architecture, because it makes available at operational time aspects from different levels/viewpoints of the business network. The community contract describes technical, semantical, (external business) process-related, and pragmatical aspects. Technical information includes service types and related behaviour descriptions, binding types between services, implementation specific messages or function parameters, and policies used in the eCommunity. The structuring element of the contract is the BNM agreed for the eCommunity: each role is supplemented with information from participants service offer, each binding with connector parametrization information. Semantical aspects cover information representation formats in messages exchanged. The pragmatic aspects covered include functional description of business processes, policies constraining roles, and non-functional aspects. The non-functional aspects govern features like trust, security, QoS that are traditionally considered as additional platform level service solutions required. In addition, non-functional aspects related to business process models capture more business oriented features, like business rules (captured as policies and monitoring rules here).

Monitors are part of the communication channel between participating services. A monitor has a generic sensor element that can be configured to filter traffic by classifying it to expected and unexpected event sequences (task started / completed, unacceptable traffic or lack of expected traffic). The BNMA agents provide each monitor with a behaviour automaton to follow, based on the service choreographies described for the corresponding role. Monitoring reports can be acted on in various ways, scaling from post-operational auditing to proactive prevention of unwanted events. In web-Pilarcos, the intent is to allow major breaches on agreed behaviour or policies to be acted on during the eCommunity operation, and allowing automatic recovery processes to be started. In this respect, the web-Pilarcos approach differs from related projects (like [5]) that otherwise use similar techniques. Because the definition of "severe breach" and the appropriate methods of potentially replacing misbehaving partners are specific to application domain, those rules and process definitions are compulsory parts of BNMs.

4 Interoperability Knowledge in the Global Middleware

The three meta-information repositories in the B2B middleware have a central role in establishing a knowledge base that allows interoperability tests to be made. Essential target concepts are service types, service offers, and business network models. Each repository is distributed for scalability and improved accessibility. Due to different type of load, the best distribution styles may differ [6].

Service types and BNMs have separate life cycles as this provides isolation layers that keep local changes from involving the whole eCommunity and minimizes the effects of BNM enhancements to local services. Furthermore, each model requires only a reasonably narrow expertise to create. In addition to direct relationships between models, the repositories store transformation rules and components for improved transformer/interceptor re-usability [6].

4.1 Type and Service Offer Repositories

The *type repository* provides a structured storage for type information related to services and their access interfaces. The web-Pilarcos type repository design was initially born during the evolution of ODP type repository and OMG MOF specifications [7, 8, 6]. Operations are provided for publishing new types, comparing types, and creating relationships between types.

Service types are abstract descriptions of business service functionality. Services are considered as self describing independent components. Service descriptions are used to ensure technical connectivity, semantic interoperation and behavioural compatibility in possibly heterogeneous environments. Service descriptions do not expose internal properties of applications as this decreases the possibilities of reuse and evolution of services. Implementation specific information, such as binding of a service into specific communication protocol or address, is not covered by service type. A service type is like a contract, which an actual service must implement.

Service types are XML-based descriptions which define interface signatures, service attributes and an interface protocol. An interface signature in web-Pilarcos is described using a WSDL description without technical binding information (see [9]). Each service supports only one kind of behaviour; different behaviour implies different service type. We refer to the definition of service behaviour as *interface protocol* which is a behavioural description defining externally visible behaviour at one endpoint of a bilateral communication. Interface protocols in web-Pilarcos are based on session types (see [10, 11]). For behavioural descriptions we have a simple XML-based process description language. Semantic interoperability of services is supported by binding ontological concepts to the exchanged documents. XML-based ontology description languages, such as general purpose description languages RDF(S) and OWL [12, 13] or more specialized XML-based ontologies such as RosettaNet, can be used [14]. The rules of the type system are based on behavioural session types, structural matching of syntactic information and semantic relations based on description logic [10, 15, 16]. Subtyping-like relationships that support service evolution are also important [17, 11, 16].

The type discipline in the web-Pilarcos platform is strictly managed. Every type definition must be contained by a type repository. Each type name, i.e. URI, must also identify the type repository responsible for managing the corresponding namespace and its type definitions. Without strict management of typing information it would be impossible to ensure that types are unambiguously named, persistently stored, verified to be correct, and relationships between types verified and intact [6]. Type repositories can also be organised into a hierarchy for partitioning of namespaces.

Service types are published by institutions responsible for a business domain or by enterprises willing to promote use of new kinds of services. Standardization of a new service type is however not necessary because the applicability and adoptance of the service type is determined by peer acceptance.

The *service offer repository* refers to services (like UDDI [18] and ODP trading service [19]) for locating services that are published using structured meta-information description of the service. We consider these descriptions as binding offers for the service. When a new service offer is published, type repository functionality is used to validate the conformance between the offer and the corresponding service type. If the validation is successful, the service offer is published into a service offer repository with the claimed service type. The service offer publishing process requires predefined service types.

4.2 Business Network Model Repository

The BNM repository provides interfaces for publishing models, verifying their properties, comparing and querying models for population or software engineering processes.

The structure (topology) and properties of a business network are defined by its BNM that explicates the roles of partners and the interactions between roles that are needed for reaching the objective of the eCommunity. A BNM comprises a collection of roles, a set of connectors and a set of architecture specific non-functional properties. The approach combines ideas from the ODP enterprise viewpoint language [20] and those of separating functional units and their interconnection into distinct concepts of components and connectors [21].

A role represents a logical business service or entity in an administrative domain. The role definition expresses the functional and non-functional properties required. Role functionality is described as a composition of service types and role specific synchronization patterns. Synchronization patterns express causal relationships between actions in distinct services of a role (by setting preconditions for interactions using terms before, after etc).

Interaction relationships between roles of are described by bilateral connectors between service interfaces. Connectors may define other communication related properties, such as control or data adaption, eCommunity coordination and non-functional properties of communication.

Non-functional properties are managed as named values that are used for selecting the right technical configurations from the underlying platform. Some properties are used for dynamic branching of behaviour at operational time. These decisions stem from the business level, but the negotiation and commitment protocols needed are preferably transparent to business services.

5 Verification and Observation of Interoperability

The web-Pilarcos middleware aims to maintain correct collaborative behaviour in eCommunities, involving several aspects of interoperability requirements. The requirements cover technical, semantic, and pragmatic aspects, i.e., awareness of collaborative behaviour and policies. Traditional verification and static analysis methods are complemented by dynamic observation of behaviour conformance against the contracted BNM and policies.

The research and prototype building in the web-Pilarcos project focuses on interoperability and eCommunity management problems at the business service level, i.e. at the level of eCommunity, its participants, behaviour and life cycle. As we presume that services are implemented or wrapped using Web Services technology, technical interoperability at the lower protocol levels is well provided by a service oriented technical middleware layer.

Interoperability problems in software systems stem mainly from components' implicit and incorrect assumptions about behaviour of their surrounding environment [22]. Every aspect of service and eCommunity functionality must be made explicit using unambiguous notations. Concepts of compatibility and substitutability are key issues in integration of autonomous services into communities; descriptions of services and communities must be founded on formal basis.

When an eCommunity is established, we ensure *sufficient* conditions for interoperability of services during service discovery and population. During runtime, however, participants of an eCommunity may behave incorrectly due to outdated service descriptions, changed business policies or technical problems. To overcome, or at least identify, interoperability problems during operation of communities we have adopted an approach based on runtime monitoring of eCommunity contracts.

Conditions for an interoperable eCommunity are fulfilled by three solutions. First, the use of a verified BNM as a basic structuring rule for the eCommunity; the various business process models intertwined into the network model can be verified to be for example deadlock free and complete by traditional protocol verification tools. Second, the use of constraint matching for accepting service offers to fulfill roles in the BNM. And third, the augmentation of the constraint matching process by the inference of further constraints arising from the selected offers for neighbour roles.

Relevant issues in role related constraints cover interface syntax with behaviour descriptions, syntax of documents to be exchanged, semantical aspects of control and information flows, and nonfunctional aspects like trust and business policies that further restrict the behaviour.

To promote evolution of syntactic structures of services, we will adopt by-structure matching instead of by-name matching for service interface comparisons [23]. Using structural typing constructors for WSDL and XML-Schema definitions we can decide if two WSDL interface descriptions are structurally equal. This interface matching is done using an approach similar to [24, 15]. Service selection and matching based on semantic concepts is not addressed in the present version of the web-Pilarcos platform but it will be implemented in future versions. Matching of semantic concepts shall be implemented using standard theories and tools, similarily to [25, 26].

Behavioural interoperability is considered in the extent of verifying that service offers and role requirements for service behaviour match. We do not seek to completely prove that a eCommunity behaves correctly, as this would need verification of behaviours between every possible participant in an eCommunity during its establishment process. Even in theory, a complete pre-operational verification of a eCommunity behaviour would be impossible, because of dynamic changes in the system, such as evolving business policies. Instead service types are considered as contracts, and the subtyping of session types as proof of conformance. Inevitable behaviour and policy conflicts are observed and acted on during operational time by the monitoring system.

The monitoring system can be given a fairly free set of rules to monitor passing message traffic; different informational and behavioural aspects are fairly straightforward to monitor [27]. The monitoring system reports detected situations (task started, completed, unacceptable traffic or lack of expected traffic). In monitoring, the challenges lie in the performance of the communication system, the design of monitoring rules, and decision engine.

Some breaches that can be detected by monitoring include a) messages from parties not partners in the eCommunity; b) transactions that are not acceptable in the current state of the eCommunity life cycle or not fulfilling precedence requirements; c) information content is not allowed to be exchanged (e.g., private documents, unknown structure); d) expected flow of information is broken; and e) obligatory transactions are not performed.

Each administrative domain can have its own decision method on how critical a breach is considered. The eCommunity contract provides methods for BNMAs to invoke in case of breaches, either for information only, or for the removal of the partner in fault. The eCommunity contract carries these rules for deciding which recovery or sanction processes to use.

6 Conclusion

The web-Pilarcos approach allows autonomous services to form the federated communities. Federated approach means that there is no overarching shared collaboration model from which the services would be derived. Instead, the services stand on their own and interoperability, including the collaboration process, the semantics and the technical aspects, must be maintained explicitly by B2B middleware. From the BNM, it would be possible to use the popular model driven approach to generate applications, but the approach would result in move expensive evolution steps [28, 29].

The federated approach has been criticized for the lack of guidance for service elements to be developed. However, making existing business network models globally available and thus exposing repeating patterns of roles - i.e., expected local business processes - gives the required guidance. Examples include RosettaNet. Our solution is to provide a repository for external process descriptions that can be augmented on demand, and that will provide an element of evolution support. These model definitions can be added to the repositories at will, without interfering already operational communities. Existing models can be frozen so that new communities are no longer formed using them, but are not actually removed automatically. The verification and matching hierarchies within the repositories may depend on them, and of course, operational communities may make references.

Another criticism frequently arising is the performance penalty of the eCommunity interoperability checking. From our earlier prototype on the populator process, we can judge that the cost of the process and its scalability are acceptable [30].

Current work extends the monitoring system and the repositories.

Acknowledgment

This article is based on work performed in the web-Pilarcos project at the Department of Computer Science at the University of Helsinki. Project was funded by the National Technology Agency TEKES in Finland and Tellabs with active partners VTT, Elisa and SysOpen. The work much integrates with RM-ODP standards work, and recently has found an interesting context in INTEROP NoE collaboration.

References

1. Afsamanesh, H., Garita, C., Hertzberger, B., Santos Silva, V.: Management of distributed information in virtual enterprises - the PRODNET approach. In: ICE'97 - International Conference on Concurrent Enterprising. (1997)
2. Rabelo, R., Camarinha-Matos, L.M., Vallejos, R.V.: Agent-based brokerage for virtual enterprise creation in the moulds industry. In: E-business and Virtual Enterprises. (2000)
3. Camarinha-Matos, L.M., Afsarmanesh, H.: Service federation in virtual organisations. In: PROLAMAT'01, Budabest, Hungary (2001)
4. Lazcano, A., Alonso, G., Schuldt, H., Schuler, C.: The WISE approach to Electronic Commerce. Computer Systems Science and Engineering **15** (2000) 345–357
5. Neal, S., Cole, J.B., Linington, P.F., Milosevic, Z., Gibson, S., Kulkarni, S.: Identifying requirements for business contract language: a monitoring perspective. In: EDOC2003. (2003)
6. Kutvonen, L.: Trading services in open distributed environments. PhD thesis, Department of Computer Science, University of Helsinki (1998)
7. ISO/IEC JTC1: IS14769 ODP Type Repository Function. (2001)
8. Object Management Group: Common Facilities RFP-5: Meta-Object Facility. (1998) OMG TC Document cf/96-05-02.
9. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C. 1.1 edn. (2001)
10. Takeuchi, K., Honda, K., Kubo, M.: An interaction-based language and its typing system. In: 6th European Conference on Parallel Architectures and Languages. (1994) 398–413
11. Gay, S., Hole, M.: Types and subtypes for client-server interactions. LNCS **1576** (1999) 74–90
12. W3C: RDF Vocabulary Description Language 1.0: RDF Schema. (2004) W3C Recommendation 10 February 2004.
13. W3C: OWL Web Ontology Language Guide. (2004) W3C Recommendation 10 February 2004.
14. RosettaNet Consortium: Rosettanet implementation framework: Core specification v02.00.00 (2004) http://www.rosettanet.org/.
15. Jha, S., Palsberg, J., Zhao, T.: Efficient Type Matching. LNCS **2303** (2002) 187–206
16. Nardi, D., Brachman, R.: An Introduction to Description Logics. In: Description Logic Handbook. Cambridge University Press (2002) 5–44
17. Di Cosmo, R., Pottier, F., Rémy, D.: Subtyping recursive types modulo associative commutative products. Unpublished draft manuscript (2003)
18. OASIS consortium: UDDI version 3.0 published specification. (2002) http://www.uddi.orgspecification.html.
19. ISO/IEC JTC1: IS13235 ODP Trading function. (1997)
20. ISO/IEC JTC1: IS15414 ODP Enterprise Language. (2003)
21. Allen, R., Garlan, D.: Formalizing architectural connection. In: ICSE 1994, Sorrento, Italy (1994) 71–80
22. Garlan, D., Allen, R., Ockerbloom, J.: Architectural mismatch or why it's hard to build systems out of existing parts. In: ICSE 1995, ACM Press (1995) 179–185
23. Ruokolainen, T.: Component interoperability. Master's thesis, Department of Computer Science, University of Helsinki (2004) In Finnish.
24. Palsberg, J., Zhao, T.: Efficient and flexible matching of recursive types. Inf. Comput. **171** (2001) 364–387
25. Peer, J.: Bringing together semantic web and web services. LNCS **2342** (2002) 279–291

26. Sriharee, N., Senivongse, T.: Discovering Web Services Using Behavioural Constraints and Ontology. In Stefani, J.B., Demeure, I., Hagimont, D., eds.: DAIS 2003. Number 2893 in LNCS, Springer-Verlag (2003)
27. Kutvonen, L., Metso, J., Ruokolainen, T., Haataja, J.: Collaboration management in dynamic business networks. (2005) Manuscript.
28. Kutvonen, L.: Relating MDA and inter-enterprise collaboration management. In Akehurst, D., ed.: Second European Workshop on Model Driven Architecture (MDA). (2004) 84–88
29. Kutvonen, L.: Challenges for ODP-based infrastructure for managing dynamic B2B networks. In Vallecillo, A., Linington, P., Wood, B., eds.: Workshop on ODP for Enterprise Computing (WODPEC 2004). (2004) 57–64
30. Vähäaho, M., Haataja, J.P., Metso, J., Suoranta, T., Kutvonen, L.: Pilarcos prototype II. Technical report, Department of Computer Science, University of Helsinki (2002)

Customizable Isolation in Transactional Workflow

Adnene Guabtni, François Charoy, and Claude Godart

LORIA laboratory, BP 239, 54506 Vandouvre-lès-Nancy Cedex, France
{guabtni,charoy,godart}@loria.fr

Summary. In Workflow Management Systems (WFMSs) safety of execution is a main need of more and more business processes and transactional workflows are real needs inside enterprizes. In previous works, transactional models consider mainly atomicity as the main issue regarding long term transactions. It rarely consider the fact that many processes may run concurrently and thus access and update the same data. Usually, the main isolation item is the data on which we apply locking approaches and this attitude don't worry about process dimension. In this work we study more precisely what are the real isolation needs in workflow environment. To realize these needs, we define "Isolation Spheres" inspired from "spheres of control" proposed by C. T. Davies to make a separation of concerns between workflow design and transactional properties specification.

1 Introduction

Defining the transactional requirements of business processes is still an issue in today workflow models and systems. This is even more critical when the complexity of the process increases. It is the case for instance with cooperative process or with distributed and composed e-services. Today's models consider the relationship between transactional properties and processes as something very monolithic. A process is considered as a long term atomic transaction and an activity is considered as a short term transaction. In the workflow terminology, that means that a process is controlled by some kind of advanced transaction model that ensure either that the process terminates or that it can be compensated. The other assumption is that activities can be implemented as short term database transactions. This has an impact on the way processes and activities are defined and it requires that business process designer have some in-depth knowledge of transactional requirements. Moreover these models consider mainly atomicity as the main issue regarding long term transactions. It rarely considers the fact that many processes may run concurrently and thus access and update the same data. Some work has been done on this topic in a recent past (contracts/coo) but it has never been generalized to process.

In this paper we try to consider processes as the concurrent execution of sets of activities that may have different requirements regarding isolation. Usually, isolation in workflow systems is performed by the database system of WFMS. Databases use

ANSI SQL isolation levels to define isolation requirements of a transaction on some data items. The problem is that workflow isolation requirements cannot always be satisfied by a database system. Contrary to database transactions, workflow transactions are defined and organized throw a process. At design time, we know exactly what are possible concurrent transactions and we want to make it possible to allow a transaction to adopt different isolation levels depending on concurrent transactions. This need appears when an activity requires an isolation level to access some data and many activities become unable to access or modify this data even if some of them don't really affect the transaction's correctness or the consistency of data. The way we choose to tackle this problem consists in separating concern between the process definition and its transactional requirements. We consider that a process must be defined independently of the transactional properties that we need to ensure. The process definition depends on the actual user activities and should reflect the actual company organization. Transactions reflects technical and consistency requirements and should not impact on this definition. To perform that, we inspire ourselves from the sphere of control approach proposed by C. T. Davies in [5] in 1978. This approach has been reused in 2001 to produce atomicity spheres in [4] to perform customizable atomicity specification in transactional workflow. We use the same approach to define isolation spheres to allow customizable isolation specification.

In the following sections of the paper, we study isolation needs in database world already applied to workflow processes. Next we try to specify transactional workflow isolation requirements. Finally we develop our approach based on isolation spheres to allow customizable isolation in transactional workflow.

2 Isolation Needs in Transactional Workflow

Isolation is an important and difficult problem as it requires to consider access to data during process and activities execution. It requires to study data manipulation by process, activities and/or sets of activities. It requires also to take into account the fact that long term process execution cannot require locking of whole set of data for all its duration. The requirements regarding these data can be far more complex. Isolation levels in flat transactions has been recognized in ANSI SQL specification [1] where the user can choose between 4 different isolation levels: (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE) to prevent phenomena like dirty read, fuzzy read or phantom problems as described in the table below. Dirty read problem occurs when a transaction reads an uncommitted data roll-backed later. Non repeatable or fuzzy read problem occurs when a transaction reads a data two time and retrieve two different values. Phantom problem occurs when a transaction reads a set of data satisfying some search condition and then repeats its read with the same search condition, it gets a different set of data items.

WFMS and Databases don't refers to the same requirements. Workflow processes are based on a controlled flow of tasks but this control is not sufficient to ensure correct execution and don't prevent from lack of consistency. This is due to workflow data visibility that is a paramount way to distribute access to data in a workflow

Isolation levels	Dirty read	Fuzzy read	phantom
READ UNCOMMITTED	Possible	Possible	Possible
READ COMMITTED	Not possible	Possible	Possible
REPEATABLE READ	Not possible	Not possible	Possible
SERIALIZABLE	Not possible	Not possible	Not possible

Table 1. SQL isolation levels defined in terms of the three phenomena

process but also a real source of concurrency access. In the next section, we expose isolation needs in workflow processes and what is important to do in the case of activities groups.

2.1 Isolation Needs in WFMS

Data accessed during a process execution are heterogeneous. They consists in documents, folders, cases data, local data, database system data and/or data obtain from external sources. Access control on these data may be very different and may have different kind of impact on the level of isolation that can be obtain. Moreover access to these data can be controlled by automatic activities or by users themselves. The level of control differs also in these two cases. Execution of automatic programs can be anticipated. User action cannot. We need to take all these parameters in account to study isolation requirements in workflow processes.

Based on previous conclusions, we need to introduce new elements in the isolation levels use performed by the transactional workflow designer. These elements are the cohesion and the coherence on a group of activities. In the following, we describe these new workflow isolation behaviors

One of the needs of transactional workflow is the control of the **cohesion** of data used in a group of activities (collaborative work, distributed or composed E-services, etc.). The solution used nowadays to ensure this cohesion of data for groups of activities is to create only one transaction imbricating all the others. Admittedly this approach makes it possible to ensure such a cohesion but has a major impact on the competition of access since it calls upon bolts in writing.

A second need is that of the **coherence** of the data. Indeed, the fact of allowing activities external to a group to read some data written by activities of that group. This can cause some inconsistencies outside the group. That is why a control of the data visibility written by activities of a group must be ensured.

Related works were made in [6] to support partial isolation in flat transactions but it was made without a real separation of concerns. In the reality, relativity and extension of isolation are merged to express customizable requirements. These requirements are usually influenced by the requirements of each activity and the pertinence of the isolation is more and more crucial depending on the type of used data and its visibility in the workflow [7]. In the next section, we introduce a new approach based on ”isolation spheres” to take into account workflow isolation needs expressed in this paper.

3 Our Approach: Isolation Spheres

In the last few years, some works has been inspired from the sphere of control proposed by Davies [5] to enhance expressivity of transactional properties, especially in [4] where the notion of atomicity sphere has been developed to allow more customizable atomicity in transactional workflow. A sphere of atomicity is a group of activities on which we apply the transactional property of atomicity. In our work, we inspire from this sphere of control approach and we define ”spheres of isolation”. A sphere of isolation will allow us to generalize isolation in the context of a workflow system. An isolation sphere allows the inside group of activities to be isolated from concurrent outside activities. The level of isolation is defined by the sphere. two kind of constraints are defined by the sphere: Coherence and Cohesion.

An isolation sphere controls the access to some data giving some privileges to a set of activities and some others to the rest of workflow activities depending on the execution evolution inside the set.

An isolation sphere represents a set of activities in concurrency working on some data. All or a part of this data represents the isolation data (data concerned by isolation on which necessary locks need to be applied). To perform cohesion and coherence of this data, we introduce some cohesion levels and some coherence levels:

Read Uncommitted: if an activity of the sphere reads a data, it can read only the last value written before the starting of the sphere or a value written by an activity of the sphere. Thus, the group of activities constituting the sphere starts from the same value.

Read Committed: if an activity of the sphere reads a data then it can read only the last **validated** value written before the starting of the sphere or a value written by an activity of the sphere.

Repeatable Read: As the Read Committed except that it is also concluded that the value of the data is not modified by an external activity as long as the sphere did not finish its execution yet. The end of the execution of a sphere occurs when all its activities finished their execution.

Srialisable: emulate an execution in series of the activities of the sphere with outside ones. This level makes it possible to ensure a serialisability between the sphere and the rest of the process but does not ensure it between the activities of the sphere.

To ensure Coherence, some coherence levels are defined in the following:

Atomic coherence: All the values of a data written by the activities of the sphere are visible outside of the sphere.

Selective coherence: Only the **validated** values written by the activities of the sphere are visible outside of the sphere.

Total coherence: Only the **last validated** value written by an activity of the sphere is visible outside of the sphere.

Imbrication of isolation spheres is possible throw imbrication of sets of activities. Imbrication is a powerful way to express more possibilities in isolation behavior. While isolation levels can be relative to a part of the process, we can generate isolation behavior dependent on execution progress due to isolation relativity over sphere imbrication.

The power of isolation spheres is the simplicity of interpretation: an isolation sphere is represented as a group of activities that need to be isolated from external activities and don't worry about internal concurrency (concurrency between activities of the group). Internal isolation, if needed, can be performed by imbricated isolation spheres. So the work performed by the workflow designer to specify isolation requirements is simplified.

This isolation sphere based transactional workflow take account of more possibilities to customize isolation and introduce more flexibility in isolation behavior. But isolation levels defined in the ANSI SQL specification have been criticized in [3] due to the lack of clarity in the interpretation of these isolation levels and the lack of response to some phenomena other then dirty read, fuzzy read and phantom. Since that, ANSI SQL specification has changed to be SQL 3 but without changes in isolation levels. Non SQL isolation levels have been proposed as cursor stability isolation or snapshot isolation. We need to study the impact of using these isolation approaches on the isolation sphere definition.

4 Conclusion and Perspectives

In this paper, we have focussed on isolation in transactional workflow. Existing approaches use techniques of isolation adapted to databases and this practise is not really adapted to workflow context. A specific adaptation of isolation techniques to transactional workflow increases expressivity in term of isolation and allow process to get rid of long blocking due to database isolation methods. Our study of the problem revealed two main isolation functionalities to make part of the transactional workflow possibilities: Cohesion to make possible the activities of a group to start working from the same values of data and become unified along the sphere execution, and Coherence to make it possible to control the delivery of data values to external activities. Our approach to make these two functionalities possible is based on "Isolation Spheres" inspired from Sphere of control introduced by C. T. Davies.

This work need to be continued referring to many aspects: the relation between isolation spheres declaration and the control flow of the workflow process, the correctness of imbricating spheres and the flexibility criterion that we need to find to ensure that a transaction will be performed with less blocking then before. Also an implementation of "isolation sphere" functionalities need to be performed in a WFMS to validate the feasibility of this work.

References

1. Ansi x3.135-1992, american national standard for information systems - database language - sql. November 1992.
2. W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.
3. Hal Berenson, Phil Bernstein, Jim Gray, Jim Melton, Elizabeth O'Neil, and Patrick O'Neil. A critique of ansi sql isolation levels. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 1–10. ACM Press, 1995.
4. Wijnand Derks, Juliane Dehnert, Paul Grefen, and Willem Jonker. Customized atomicity specification for transactional workflow. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'01)*, pages 140–147. IEEE Computer Society, 2001.
5. Charles T. Davies Jr. Data processing spheres of control. *IBM Systems Journal 17(2): 179-198*, 1978.
6. Randi Karlsen. Supporting partial isolation in flat transaction. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications*, pages 698–711. Springer-Verlag, 1999.
7. Nick Russell, Arthur H. M. ter Hofstede, David Edmond, and W.M.P. van der Aalst. Workflow data patterns. Technical Report FIT-TR-2004-01, Queensland University of Technology, Brisbane, Australia, April 2004.

Transforming Workflow Graphs

Johann Eder, Wolfgang Gruber, and Horst Pichler

University of Klagenfurt, Department of Informatics-Systems,
eder@isys.uni-klu.ac.at

Summary. Workflow management systems are very useful for integrating separately developed application systems by controlling flows of execution. For various purposes (e.g. distribution of activities, workflow evolution, time calculation, etc.) it is necessary to change the representation of a workflow, the structure of a workflow graph without changing it's semantics. We provide an equivalence definition of workflow graphs and introduce a set of basic transformation operations defined on workflow graphs which keep the semantics. We show how these basic operations can be combined to achieve complex transformations and briefly describe a prototypical transformation tool.

1 Introduction

Workflow management systems (WFMSs) improve business processes by automating tasks, getting the right information to the right place for a specific job function, and integrating information in the enterprise. Workflow management systems were also considered as integration tools from the very beginning, with the idea that they allow the representation of processes comprising of activities which are executed within different application systems [5, 8].

Numerous workflow models have been developed, based on different modelling concepts (e.g. Petri Net variants, precedence graph models, precedence graphs with control nodes, state charts, control structure based models) and on different representation models (text based models, flow models, structured graphs, etc.) [2, 7, 11]. Transformations can be difficult, as expressiveness equality between two representations must be ensured (e.g. from graph to text-based). Another sort of transformation is to change the structure of a workflow within its representation model. Such transformations are needed in several situations. During workflow design, it allows the designer to view a workflow graph from different perspectives and change the representation to enhance readability and understandability. For distributing the activities of a workflow to separate information systems, or among different stakeholders in a virtual organization, it is useful to change the workflow graph to make this distribution explicit and easy to understand and to support the dispatcher at runtime. For workflow evolution [4] two workflow graphs (initial and final workflow) have to be compared and a manifold of hybrid workflows have to be generated. This task is much easier, if both workflow graphs are previously prepared in a way that the comparison is facilitated by structural equivalence to the greatest possible extent, such that the comparison of the workflow only has to deal with the actual changes of the workflow which lead to different executions and not with mere representational alterations. Furthermore, such transformations are needed for time management [3], or for organizational changes [1, 11].

The main contributions of this paper are: We present a notion for equivalence of workflow models based on the idea that two workflows are equal, if they admit the same workflow instances at the level of atomic activities. We introduce a series of basic transformations which preserve this semantics of the workflows. We show how complex transformations can be constructed from the basic operations and briefly describe a workflow model management tool prototypically implementing these transformations.

2 Workflow Model

2.1 Structured Workflow Definition

A workflow is a collection of *activities, agents,* and *dependencies* between activities. Activities correspond to individual steps in a business process, agents (software systems or humans) are responsible for the enactment of activities, and dependencies determine the execution sequence of activities. We assume that workflows are *well structured*: A well-structured workflow consists of m sequential activities, $T_1 \dots T_m$. Each activity T_i is either elementary, i.e. it cannot be decomposed any further, or it is complex. A complex activity consists of n_i parallel (and), sequential (seq), conditional (or) or alternative (alt) sub-activities ${T_i}^1, \dots, {T_i}^{n_i}$, each of which is either elementary or complex (cf. [2]). (Complex) activities between *seq-split* and *seq-join* are always executed in sequence. An *and-split* node refers to a control element with several immediate successors, all of which are executed in parallel. An *and-join* node synchronizes the workflow as it can only be executed if all of its immediate predecessors have finished their execution. An *or-split* node refers to a control element whose immediate successor is determined by evaluating some boolean expression (conditional) and the successor node of an *alt-split*s is selected by (user-)choice. Note that the semantics of or-splits and alt-splits demands that exactly one of many successors may be executed. *Or-join*s and *alt-joins* refer to control elements that join all the branches after or-splits and alt-splits respectively. Additionally, predicates have to be defined for each successor node of an or-split, representing a boolean expression which must yield true as precondition for execution (due to the exclusive semantics of an or-split only one of many successors may be executed, thus the predicates must be defined accordingly). Structured Workflows are often represented by *structured workflow graphs*, where nodes represent activities (rectangles) or control elements (circles) and edges correspond to dependencies between nodes (see Fig. 1). Predicates are displayed in angle brackets, attached on top of a node.

According to the definition presented above a workflow graph must be *well structured*, as each split node is associated with exactly one join node and vice versa and each path in the workflow graph originating in a split node leads to its corresponding join node. For the purpose of allowing more transformations (see section 3) and the

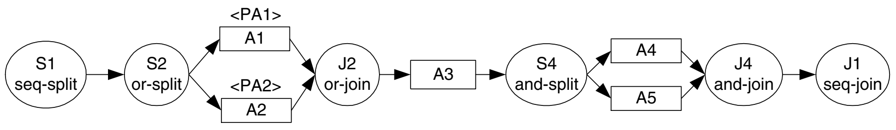


Fig. 1. Workflow graph example

separation of workflow instance types in the workflow model we offer the notion of a *structured* workflow graph. Here a split node may be associated with several join nodes, however, a join node corresponds to exactly one split node. Each path originating in a split node has to lead to an associated join node. Such graphs are results of equivalence transformations.

2.2 Workflow Instance Types

Due to conditionals not all instances of a workflow will process the same activities. We classify workflow instances into workflow instance types according to the actual executed activities. Similar to [9], a *workflow instance type* refers to (a set of) workflow instances that contain exactly the same activities, i.e., for each or-split node in the workflow graph, the same successor node is chosen; resp. for each conditional complex activity the same child-activity is selected. Therefore, a workflow instance type is a submodel of a workflow where each or-split has exactly one successor; resp. each conditional or alternative complex activity has exactly one subactivity [2].

2.3 Equivalence of Workflows

Our goal is to support the transformation of a workflow without changing the semantics. For this purpose we need a clear definition when workflows are equivalent. Our definition is based on the consideration that workflows are equivalent if they provide the same tasks. Therefore, the equivalence of correct workflows bases on equivalent tasks and identical execution order. Workflows are equivalent, if they execute the same activities in exactly the same order. Therefore, the equivalence of structured workflows ($WF1 \equiv WF2$) is based on equivalent sets of workflow instance types [6], i.e. any instance of $WF1$ is equivalent to an instance of $WF2$, and vice versa.

Equivalence of workflows Two workflows are equivalent ($WF1 \equiv WF2$) if their sets of instance types are equivalent, i.e. if and only if for each element of one set there is an equivalent element in the other set.

Equivalence of instance types Two workflow instance types $IT1$ and $IT2$ are equivalent ($IT1 \equiv IT2$) if they consist of the same (elementary) activities with identical execution order, where the position of or-splits and or-joins in instance types is irrelevant, since an or-split has only one successor in an instance type.

3 Workflow Transformations

Workflow transformations are operations on a workflow *SWF* resulting in a different workflow *SWF'* (e.g. moving splits or joins) [6]. It is essential that such changes

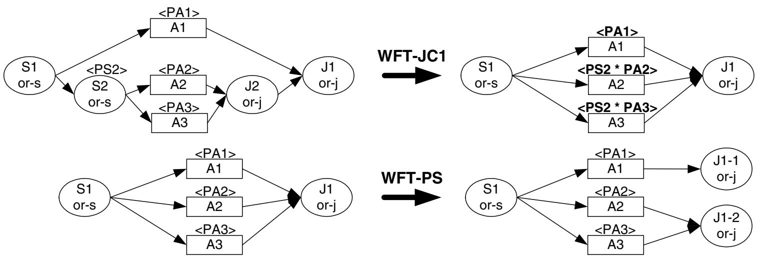


Fig. 2. Coalescing nodes and path separation

are introduced systematically and that their impact is clearly understood. Workflow model transformation is a suitable approach for this purpose [10]. The application of pre-defined transformation operations can ensure that the modified process conforms to constraints specified in the original model. In the following we provide a set of transformations, which do not change the semantics of the workflow according to the definition of equivalence given above. Complex transformations can be established on this basic set of transformations by repeated application. Transformations are feasible in both directions, i.e. from *SWF* to *SWF'* and vice versa from *SWF'* to *SWF*. We distinguish between *basic* and *complex* workflow transformations, where complex transformations are composed of consecutively applied basic transformations. Each transformation of a structured workflow graph must result in another structured workflow graph.

3.1 Basic Workflow Transformations

a) Sequence Encapsulation, Coalescing Nodes and Path Separation

• **Encapsulation in a Sequence (WFT-S)** An activity can always be encapsulated between two sequence control elements (seq-split and seq-join).

• **Or-Join Coalescing (WFT-JC1)** In a workflow *SWF* with a nested or-structure two succeeding or-joins and their according or-splits can be coalesced into a single or-structure. It is necessary to adjust the predicates according to the changed sequence of split-nodes $S1$ and $S2$ by applying the conjunction (*, logical and). An example for this transformation is given in Fig. 2. This transformation is similar to the structurally equivalent transformation presented in [10] as far as the differences in the workflow models are concerned.

• **Alt-Join Coalescing (WFT-JC2)** This transformation is performed analogously to *WFT-JC1*, by replacing each *or*-split with an *alt*-split and each *or*-join with an *alt*-join and vice versa. Note that there are no preconditions to consider in such a scenario.

• **Separating a Conditional or Alternative Path (WFT-PS)** In a workflow *SWF* with an or-structure or alt-structure each path can be separated by means of duplicating join-node $J1$, if (and only if) $J1$ has no successor (see Fig. 2).

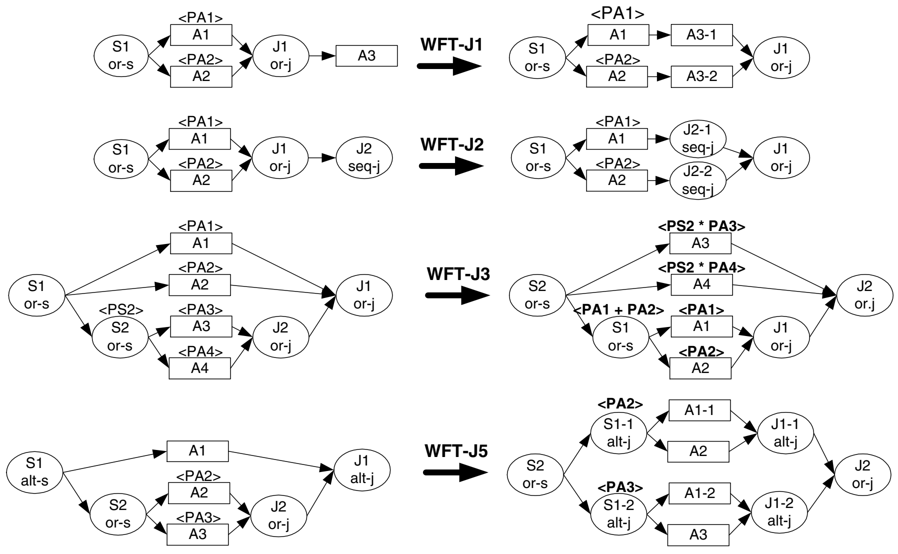


Fig. 3. Moving Joins (1)

b) Moving Joins

Moving Joins means changing the topological position of a join control element (and-join, or-join, alt-join, or seq-join). This transformation separates the intrinsic instance types contained in a workflow model. Some of the following transformations require node duplication. In some cases moving a join element makes it necessary to move the corresponding split element as well.

• **Moving Join over Activity (WFT-J1)** A workflow *SWF* with an or-join or an alt-join $J1$ followed by an activity *A3* can be transformed to a workflow *SWF'* applying node duplication, so that the join $J1$ is shifted behind the duplicates *A3-1* and *A3-2* of activity *A3* as shown in Fig. 3. This transformation, and all of the following ones, can be applied to structures with any number of paths.

• **Moving Join over Seq-Join (WFT-J2)** A workflow *SWF* with an or-join or an alt-join *J1* followed by a sequence join *J2* can be transformed to workflow *SWF'* applying node duplication, so that the join $J1$ is delayed after *J2* as shown in Fig. 3. Here, *J2* will be replaced by its duplicates *J2-1* and *J2-2*, such that *J1* is the successor of *J2-1* and *J2-2*, and *A1* is the predecessor of *J2-1* and *A2* is the predecessor of *J2-2*.

• **Moving Or-Join over Or-Join (WFT-J3)** In a workflow *SWF* with a nested or-structure (i.e. within an or-structure with the split *S1* and the corresponding join *J1* there is another or-structure with the split S2 and the corresponding join *J2*), the inner join *J2* can be moved behind the outer join *J1*, which makes it necessary to move the corresponding split element *S2* and to adjust the predicates according to the changed sequence of $S1$ and *S2* by conjunction (*, logical and) or disjunction (+, logical or). This change causes the inner or-structure to be put over the outer (see Fig. 3).

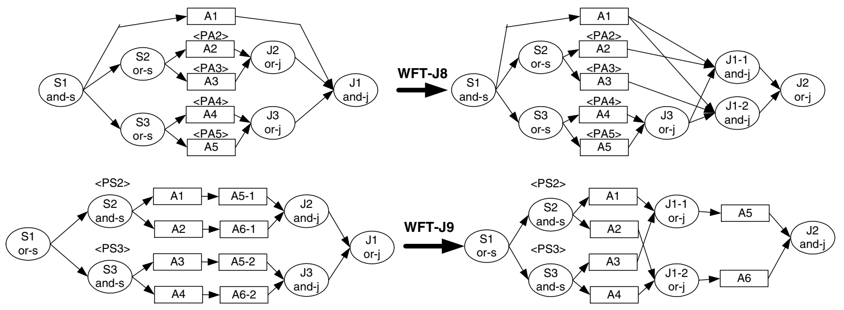


Fig. 4. Moving Joins (2)

• **Moving Alt-Join over Alt-Join (WFT-J4)** This transformation is performed analogously to *WFT-J3*, by replacing each *or*-split with an *alt*-split and each *or*-join with an *alt*-join. Note that there are no preconditions to consider in such a scenario.

• **Moving Or-Join over Alt-Join (WFT-J5)** In a workflow *SWF* with a nested alt/or-structure, i.e. within an alt-structure with the split *S1* and the join *J1* there is an or-structure with the split *S2* and the join *J2*, the inner join *J2* can be moved behind the outer join *J1*. This makes it necessary to move the corresponding split element *S2* and to duplicate control elements and activities and adjust the predicates. In fact, the inner or-structure is put over the outer structure (see Fig. 3).

• **Moving Alt-Join over Or-Join (WFT-J6)** This transformation is performed analogously to *WFT-J5*, by replacing each *or*-split with an *alt*-split and each *or*-join with an *alt*-join and vice versa.

• **Unfold: Moving Join over And-Join (WFT-J8)** The *unfold* transformation produces a *structured* graph-based structure with *multiple sequential successors*, which means that a node, with the exception of splits, can have more than one sequential successor. However, in each instance type of such a graph every node except for and-splits has again exactly one successor. An or-join or alt-join $J2$ can be moved behind its immediately succeeding and-join *J1*, requiring duplication of control elements. An example for this transformation is shown in Fig. 4. To move *J2* behind *J1* we place a copy of *J1* behind every predecessor of *J2*, so that each of these copies of *J1* has additionally the same predecessor as *J1* except for *J2*. A copy of *J2* is inserted, such that it has the copies of *J1* as predecessor and the successor of *J1* as successor. Then *J1* is deleted with all its successor and predecessor dependencies. If *J2* has no longer a successor, it will also be deleted. *Partial unfold*, as it is described in [6], is a combination of already described transformations.

• **Moving And-Join over Or-Join or Alt-Join (WFT-J9)** This transformation is introduced in [7]. Starting with a workflow *SWF* with an or-join $J1$, which has only and-joins $J2_1 \dots J2_m$ as predecessors, each of these and-joins $J2_i \in \{J2_1 \dots J2_m\}$ has the **identical set of predecessors** $M_1 \dots M_n$. Let the sets of the predecessors of $M_1 \dots M_n$ for every and-join be $S_1 \dots S_m$. The or-join $J1$ can be moved before the predecessors of the and-joins, which necessitates the duplicating and coalescing of control elements. The transformation is shown in Figure 4. In order to move $J1$ we place a copy of $J1$ for every predecessor of an and-joins $J2_i \in \{J2_1 \dots J2_m\}$, so that each copy of $J1$ has the same number of predecessors

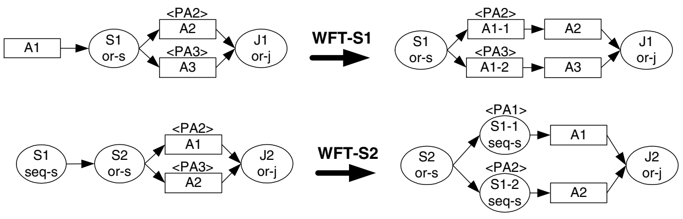


Fig. 5. Split Moving Over Seq-Split

as $J1$ and every copy of $J1$ has as predecessor one element from every set $S_1 \dots S_m$, so that every element from $S_i \in \{S_1 \dots S_m\}$ has only one successor. Furthermore, we place a copy of an and-join $J2_i$ with its predecessors, so that every copied predecessor of the copied and-join $J2_i$ has exactly one copy of $J1$ as its predecessor. The copy of the and-join $J2$ has as successor the successor of $J1$, if existent. Now, the and-joins $J2_1 \dots J2_m$ with their predecessors and with all their successor and predecessor dependencies are deleted.

c) Moving Splits

Moving Splits changes the position of a split control element. This transformation separates (moving splits towards start) or merges (moving splits towards end) the intrinsic instance types contained in a workflow model, in analogy to join moving. Not every split can be moved: Moving alt-splits is always possible and for or-splits the predicates have to be considered. Another aspect of or-split moving to be considered is that the decision which path of an or-split is selected will be transferred towards the workflows start, so that decision uncertainties due to or-splits will be reduced.

• **Moving Split before Activity (WFT-S1)** A workflow *SWF* with an or-split or alt-split *S1* with activity *A1* as predecessor can be transformed in the workflow SWF' through node duplication, so that *S1* is located before *A1* (see Fig. 5). Here, *A1* will be replaced by its duplicates *A-1* and *A-2*, so that *S1* is the predecessor of *A-1* and *A-2*, and *A2* is the successor of *A1-1* and *A3* is the successor of *M1-2*. Predicates are adjusted.

• **Split Moving Over Seq-Join (WFT-S2)** A workflow *SWF* with an or-split or alt-split $S2$ proceeded by a sequence split *S1* can be transformed to a workflow *SWF'* through node duplication, so that the split $S1$ is delayed after *S2* as shown in Fig. 5. Here, *S1* will be replaced by its duplicates *S1-1* and *S1-2*, so that *S2* is the predecessor of *S1-1* and *S1-2*, and *A1* is the successor of *S1-1* and *A2* is the successor of *S1-2*. This transformation results in a *structured* workflow (see Fig. 5 for an example).

3.2 Complex Transformations

The basic workflow transformations are building blocks for more complex transformations as compositions with (repeated) application of these basic transformations. These complex transformations do not change the semantics of the workflow either. In this paper, three complex transformations are constructed: (1) the *backward unfolding procedure*, (2) the *partial backward unfolding procedure*, and (3) the *forward* *unfolding procedure*. Unfolding means that or-joins or alt-joins are moved topologically to the rear and or-splits or alt-splits are moved as near to the start as possible, for a simple separation of different instance types.

a) Backward Unfolding

A procedure for generating an equivalent backward unfolded workflow UW for a workflow W is described in [6]. The transformation specifies how a workflow has to be modified to become fully unfolded. An alternative approach to unfold a workflow is to apply the above listed basic transformations in a way that no or-join or alt-join element has an activity as successor. Every structured workflow can be fully unfolded, because for every constellation there is a basic transformation that can be applied in order to move the corresponding join topologically backwards. The following constellations have been identified:

- or-/alt-join before activity → use WFT-J1
- or-/alt-join before and-join → use WFT-J8
- or-/alt-join before seq-join → use WFT-J2
- or-/alt-join before or/alt-join → use WFT-J3, WFT-J4, WFT-J5, or WFT-J6
- separate path → use WFT-PS

The above procedure suffers from the potential explosion of the number of "duplicate" nodes in the unfolded workflow, since it considers each instance type separately. This is not always desirable when discriminating between instance types. To avoid this problem, we developed the *partial unfolding* technique.

b) Partial Backward Unfolding

We can unfold the workflow only where it is desired. The procedure of partially unfolding a workflow W to a partially unfolded workflow PUW begins by selecting a *hot-node*, with the side effect that all instance types going through the hot-node are factored out, or intuitively, the workflow graph reachable from the hot-node is duplicated. In principle, every node can be a hot-node. For practical reasons, we assume that a hot-node is an immediate predecessor of an or-join. Once a hot-node is identified, partial unfolding takes place as follows:

1. Mark all (transitive) successors of the hot-node;
2. Apply the transformations *WFT-J1, WFT-J2, WFT-J3, WFT-J4, WFT-J5, WFT-J6, WFT-J8, WFT-PS* on the marked workflow elements so that no or-join or alt-join element has an activity element as successor in the context of the marked workflow elements;

Note that the transformation step order is of great importance to avoid unnecessary cancellation of operations (for details see [6]).

c) Forward Unfold Procedure

A procedure for generating an equivalent forward unfolded workflow UW for a workflow W is described below. In order to unfold a workflow forward, the transformations listed above must be applied, such that no or-split and no alt-split element has an activity element as predecessor. Because of data dependencies not every structured workflow can be fully forward unfolded. For the following constellations there is a basic transformation that can be applied in order to move the corresponding split topologically forward. The following constellations have been identified:

- or-/alt-split after activity without data dependencies → use WFT-S1
- or-/alt-split after seq-split → use WFT-S2
- or-/alt-split after or/alt-join → use WFT-J3, WFT-J4, WFT-J5, or WFT-J6

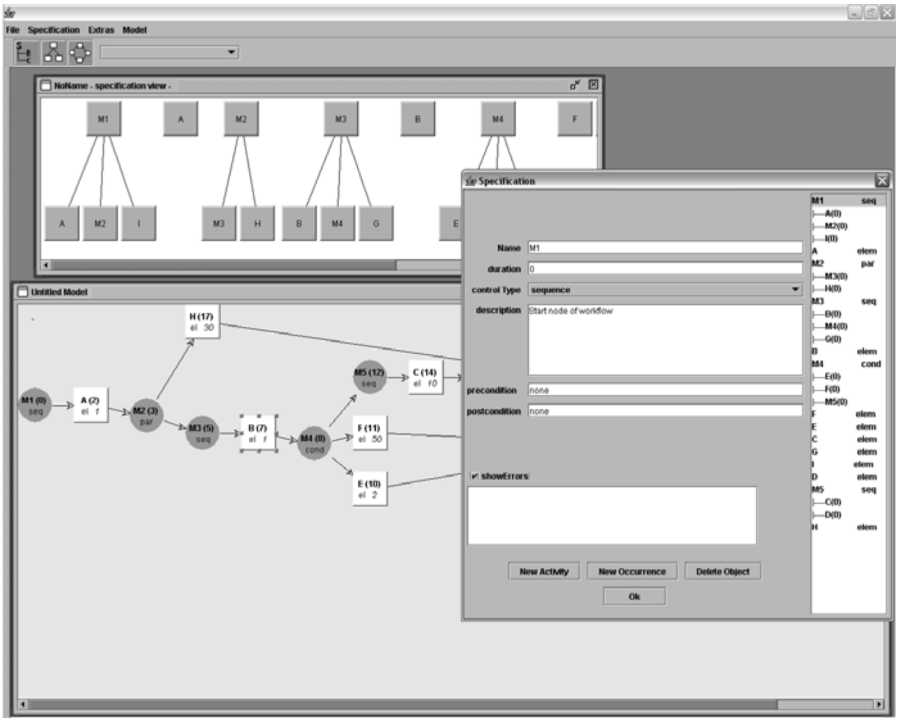
For the constellations *or-/alt-split after and-join* no transformation exists.

4 Graphical Workflow Designer

Our Graphical Workflow Designer (GWfD) has been developed as proof-of-concept prototype, which currently supports workflow modeling, workflow transformations, modeling of time and verification of time constraints. The requirements for the GWfD architecture were primarily platform independency, persistent data management, modularity, extensibility and simplicity. To assure platform independency the GWfD has been implemented in Java.

To achieve a modular and extensible structure we designed a three layered architecture, where each layer provides services used by the layer below. The **Data Source Connectivity Layer** represents the API to the GWfD data source. Either relational databases (accssed via JDBC) or XML files (accessed via JAXP) serve as data source, where workflows are durably stored, using the relational model from our workflow metamodel [2]. The **Application Layer** implements the application logic and functionalities, which are: create, modify, and delete workflow specifications, deduce workflow models from the specification, and perform transformations and time calculations on the workflow model. The **Presentation Layer** is responsible for the intuitive graphical visualization of workflow specifications and models. For the implementation we applied the recommended architectural design pattern *Model-View-Controller* (MVC), using the freely available Swing Component *JGraph* for the representation and modification of graph-based workflow models.

Figure 6 shows the three main views of the editor: In the **Edit View** (on the right hand side) a workflow can be created or modified. Here the basic building blocks of a workflow are specified, which are elementary activities (marked with the key word 'elem') and complex activities (marked as 'seq', 'cond', 'par' and 'alt'). The workflow structure is defined using a bottom-up approach by assigning (complex) activities to superior complex activities. If the workflow definition is complete a correctness-verification of the workflow specifications can be launched, which detects and displays modelling errors (in order to achieve a well structured workflow graph). The graphical visualization of the specification is presented in the **Specification View**, where the hierarchy of elements, as defined in the Edit View can be examined. Finally, the **Model View** reflects an initially generated graph-based *master workflow model*, which allows no transformations. From the master model any number of *Child-Models* can be derived, each displayed in its own *Child-Model*



```
Name            M1
duration        0
control Type    sequence
description     Start node of workflow
precondition    none
postcondition   none
[✓] showErrors
```

```
M1          seq
|----A(0)
|----M2(0)
|----I(0)
A           elem
M2          par
|----M3(0)
|----H(0)
M3          seq
|----B(0)
|----M4(0)
|----G(0)
B           elem
M4          cond
|----E(0)
|----F(0)
|----M5(0)
F           elem
E           elem
C           elem
G           elem
I           elem
D           elem
M5          seq
|----C(0)
|----D(0)
H           elem
```

Fig. 6. Specification Editor, Specification View and Master Model View

View, where workflow transformations can be applied. Figure 7 shows an example workflow graph before and after the unfold operation, which is applied on the conditional-join M4.

In order to separate the intrinsic instance types in a workflow model, we apply the partial unfold transformation operation. Therefore, we have to specify the instance type to be separated. As illustrated in Figure 7, we select an adjacent predecessor of an *or*-join and invoke the context menu by right-clicking the mouse. When selecting the item **transform** in the context menu, a list of all possible transformation operations in this context appears, which is in this case only (partial) **unfold**. When selecting this operation the workflow model is accordingly modified and the graphical representation is updated, as we can see in Figure 7. The other transformation-operations are implemented in a similar manner, where depending on which workflow elements are selected, the GWfD proposes all possible transformation operations.

5 Conclusions

It is generally a good engineering principle to consider manipulations of design artifacts, study their properties and make applications of such manipulations instrumental in design processes. We introduce an equivalence definition on workflow models based on the notion of instance types and thus capture the semantics of workflow

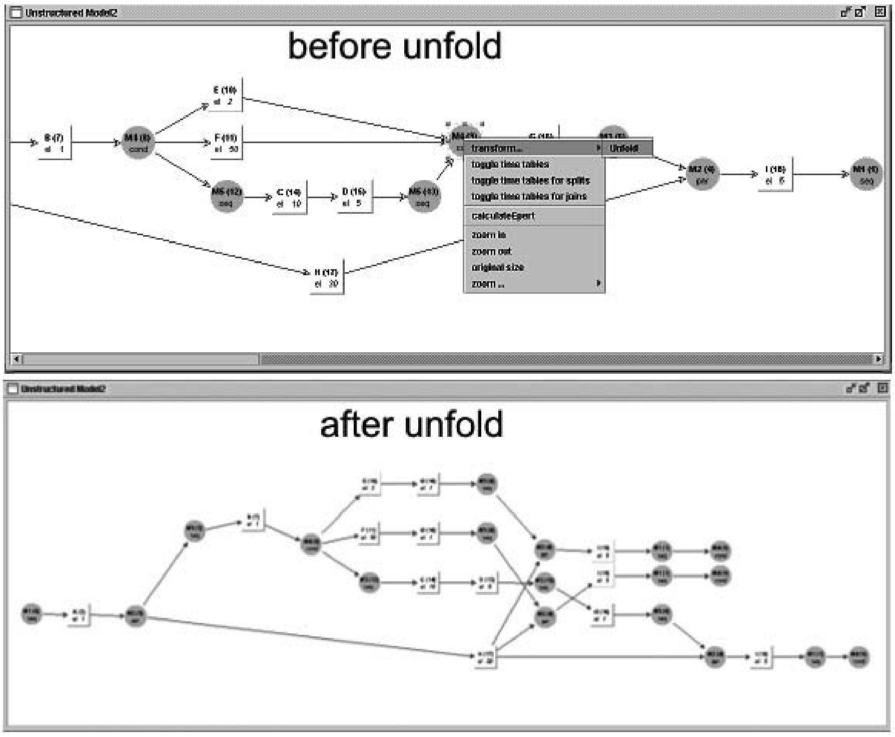


Fig. 7. Graph before and after unfold

models as the set of admitted partial orders of their basic activities. The main contribution of this work is the development of a set of basic schema transformation that maintain this semantics. There are several applications for the presented methodology. It serves as sound basis for design tools. It enables analysts and designers to start from an initial model and improve the quality of the model step by step. We can provide automatic support to achieve certain presentation characteristics of a workflow model. A model can be transformed to inspect it from different points of view. In particular, a model suitable for conceptual comprehension can be transformed to a model better suited for implementation. For workflow evolution it is possible to isolate those parts of a workflow which are actually changed. For cooperating workflows it is possible to arrange a workflow model in a way, that those parts of a workflow which are visible from outside are raised to the highest level while only internal parts of a workflow are hidden in complex activities. Workflow definitions can also be prepared to be applied in different organizational settings, where activities are distributed among other application systems.

References

1. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Conceptual modeling of workflows. *LNCS 1021*. Springer, 1995.

2. J. Eder and W. Gruber. A Meta Model for Structured Workflows Supporting Workflow Transformations. *LNCS 2435*. Springer, 2002.
3. J. Eder, W. Gruber, and E. Panagos. Temporal modeling of workflows with conditional execution paths. *LNCS 1873*. Springer, 2000.
4. J. Eder and M. Saringer. Workflow Evolution: Generation of Hybrid Flows. in: OOIS,*LNCS 2817*. Springer, 2000.
5. D. Georgakopoulos, M. F. Hornick, and A. P. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.
6. W. Gruber. Modeling and Transformation of Workflows with Temporal Constraints. *ISBN 3-89838-484-5*. Akademische Verlagsgesellschaft, Berlin, 2004.
7. B. Kiepuszewski, A.H.M. ter Hofstede, and C. Bussler. On structured workflow modelling. *LNCS 1789*. Springer, 1999.
8. P. Lawrence. *Workflow Handbook*. John Wiley and Sons, New York, 1997.
9. O. Marjanovic and M. E. Orlowska. On modeling and verification of temporal constraints in production workflows. *Knowledge and Information Systems, KAIS*, vol 1. Springer, 1999.
10. W. Sadiq and M. E. Orlowska. On business process model transformations. *LNCS 1920*. Springer, 2000.
11. W. v. d. Aalst, et.al. Advanced workflow patterns. *LNCS 1901*. Springer, 2000.

Interoperability Issues in Metamodelling Platforms

Harald Kühn[1,#] and Marion Murzek[2 *]

1 BOC Information Systems GmbH, Platform Development, Rabensteig 2, A-1010 Vienna, Austria, harald.kuehn@boc-eu.com
2 Women's Postgraduate College for Internet Technologies (WIT), Vienna University of Technology, Austria, murzek@wit.tuwien.ac.at

Summary. Metamodelling platforms are getting more and more base technology, therefore interoperability of metamod-elling platforms becomes a crucial aspect in managing corpo-rations' knowledge assets. This paper describes a generic metamodelling platform architecture and presents an overview of interoperability issues according to conceptual domains in metamodelling platform architectures. Some of these issues are illustrated by a case study from the insurance sector. The collection of interoperability issues can serve as a starting point to stimulate further research on interoperability prob-lems in the metamodelling platform domain.

1 Introduction

Metamodelling platforms are software environments allowing the definition, usage and maintenance of a method's elements: (a) metamodels describing problem-specific modelling languages, (b) mechanisms & algorithms working on models and their underlying metamodels, and (c) procedure models representing process descriptions how to apply the metamodels and the corresponding mechanisms. Some of their functional and non-functional requirements are multi-product ability, web-enablement, multi-client ability, adaptability, and scalability [6].

Metamodelling approaches are an active research field since the past 15 years and since then have found serious application areas in the software and information technology industries. Some of them are Enterprise Model Integration (EMI) [9] in the context of Enterprise Application Integration (EAI) [12], Model Integrated Computing (MIC) [11], domain specific modelling languages such as the Unified Modelling Language (UML) [22] based on Meta Object Facility (MOF) [18], the

[#] This work is partially supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract N° 508011, <http://www.interop-noe.org>).
[*] This research has been partly funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

Unified Enterprise Modelling Language [27], and model-driven development approaches such as Model Driven Architecture (MDA) [19]. Additionally, metamodelling approaches serve as valuable base technology to merge different modelling approaches into a domain specific modelling language, e.g. integrating UML with simulation-oriented modeling languages.

Since widespread industrial and research usage of metamodelling technology such as ADONIS [1], MetaEdit+ [13], and METIS [14], the integration and interoperability of metamodelling platforms is moving into focus of applied research and product-quality implementations [16]. The goal of this paper is to provide an overview of interoperability issues in the domain of metamodelling platforms. This overview can serve as a starting point to stimulate further research on interoperability problems in this domain.

The remainder of the paper is organized as follows: chapter 2 presents a generic metamodelling platform architecture. Chapter 3 gives an overview of issues in metamodelling platform interoperability. These issues provide input for further research areas in metamodelling platform interoperability. Chapter 4 presents a case study in metamodelling platform interoperability. Related work is discussed in chapter 5. Finally, chapter 6 summarizes the paper and gives an outlook to future work.

2 Metamodelling Platform Architecture

Figure 1 presents a generic architecture of metamodelling platforms [6, 8]. An important element is the *meta-metamodel* (meta^2 model). The meta^2 model defines general concepts available for method definition and method usage such as “metamodel”, “model type”, “class”, “relation”, “attribute” etc. *Semantic schemas* are tightly coupled with the meta^2 model. They describe the semantics of each method element defined by using the meta^2 model. Semantic schemas can be described by using approaches such as ontology [5], semantic engines ("mechanisms") [6] etc.

The *metamodel base* contains metamodels of concrete modelling languages. Metamodel editors are used for the definition and maintenance of metamodels. The metamodel base is based on the meta^2 model. The metamodel base forms the foundation of the *model base*, in which all models are stored. Models can be created, changed and visualized by using appropriate editors.

All mechanisms and algorithms used for evaluating and using models are stored in the *mechanism base*. Mechanism editors are used for definition and maintenance of mechanisms. The mechanism base is based on the meta^2 model.

Procedure models describe the application of metamodels and mechanisms. They are stored in the *procedure model base*. Procedure model editors are used for definition and maintenance of procedure models. The procedure model base is based on the meta^2 model.

Persistency services support the durable storage of the various bases. These services abstract from concrete storage techniques and permit storing of modelling information in heterogeneous databases, file systems, web services etc.

Access services serve two main tasks. On the one hand they enable the open, bi-directional exchange of all metamodelling information with other systems using a message-oriented approach, i.e. APIs, or a data-oriented approach, i.e. files. On the other hand they cover all aspects concerning security such as access rights, authorization, en-/decryption etc.

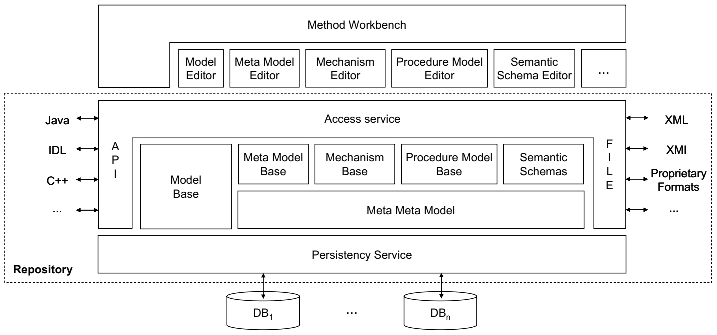


Figure 1. Generic Architecture of Metamodelling Platform

3 Interoperability Issues

OUSKEL AND SHETH identified two major categories of interoperability problems: information heterogeneity and system heterogeneity [24]. In the context of metamodelling platforms, information heterogeneity maps to the modelling hierarchy of meta^2 models, metamodels and models of each platform ("model heterogeneity"). System heterogeneity maps to the diversity of available access services, mechanisms, persistency services, and implementation technologies of each platform (see Figure 2). The further description of interoperability issues in metamodelling platforms will be structured according to the conceptual domains of the generic metamodelling platform architecture as described in Figure 1.

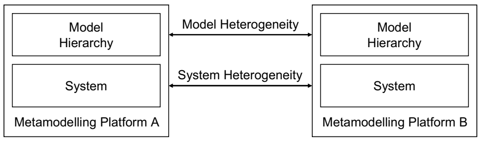


Figure 2. Model Heterogeneity and System Heterogeneity

3.1 Meta^2 Model Domain

The meta^2 model provides the basis for the other conceptual domains (see figure 1.). Interoperability problems in this domain may arise in the syntax, semantics and expressiveness of underlying metamodelling languages to define, integrate and

represent a method's elements [8], and appropriate transformation mechanisms for metamodel transformation.
Some important aspects to be considered by metamodelling languages are:

- inheritance and meta-class features for metamodel definition.
- the expressive power and cardinality features of meta-relationships such as aggregation (part-of), generalisation (is-a), pointer (link), and binary or n-ary relationships.
- the amount of available meta-attributes to define concrete attributes of a certain type.
- Some important aspects to be considered by metamodel transformation mechanisms are:
- handling of different relationship concepts such as n-ary relationships, and circular and recursive dependencies.
- nesting of elements and their handling in flattened structures during metamodel transformation.
- uniqueness of element identification and the possibility of using model annotations to store information in the target metamodel to avoid information loss during transformation.

3.2 Metamodel Domain

Interoperability issues in the metamodel domain may occur in the definition, integration and representation of the syntax, semantics and notation of modelling languages. Additionally, model transformation mechanisms for the horizontal and vertical model transformation are aspects to be considered in interoperable metamodelling platforms.
Some important aspects to be considered on the metamodel level are:

- identification and consideration of syntactic and semantic mismatches among modelling languages (same name – different concept, different name – same concept etc.).
- identification and usage of domain-specific ontology to consider domain-specific aspects and knowledge to establish metamodel interoperability.
- measurements for analysis and evaluation of modelling languages and their underlying metamodels to identify interoperable and non-interoperable parts.
- definition of "hot spots" in participating metamodels to provide linking points for metamodel integration.

3.3 Model Domain

Models correspond to their underlying metamodel. Therefore, the interoperability problems on this level are influenced by the problems concerning metamodels (see 3.2). In addition interoperability issues have already been investigated thoroughly in the realm of distributed database systems [28]. Based on these considerations some additional aspects of Model Interoperability are:

- existence of non-corresponding model fragments, i.e., their metamodels are partly not corresponding. This can result in information loss or in hidden information to avoid losing information in bidirectional model exchange.
- diversity of graphical representations and diversity of the underlying coordinate system to place and arrange modelling objects. In worst case, models cannot be understood after model exchange because of complete loss of graphical information.
- models are input or provide parameters for mechanisms such as simulation, analysis, reporting, and code generation. Even if models correspond to its metamodel, it may occur that mechanisms cannot be used because of incomplete models.
- existence of appropriate domain ontology to support a proper model interpretation in each platform.
- history logs to record model changes which can be necessary in model synchronisation.

3.4 Mechanism Domain

Mechanisms provide possibilities to generate value added out of the different model bases. Typical examples for mechanisms are version management, multi language support, model analysis, and simulation.
Some important aspects to be considered in mechanism interoperability are:

- mechanisms can be implemented either on meta model level or meta^2 model level. Before exchanging mechanisms between metamodelling platforms, the interdependencies of a mechanism to these both levels have to be analyzed.
- the technology used to implement a mechanism (scripting, programming language, query language etc.) has strong influence on its interoperability. A possible way to implement interoperable mechanisms is using standardized interfaces, e.g. applying interface definition language (IDL) or wrapper technology.

3.5 Procedure Model Domain

Procedure models describe the processes how to apply modelling languages and mechanisms to solve certain problem scenarios. This includes concepts such as phases, milestones, responsibilities, work steps, results etc.
Important aspects to be considered in interoperability of procedure models are:

- the availability and mismatch of special procedure model fragments such as contradictory work step descriptions.
- merging of procedure models into consolidated procedure descriptions.

3.6 Semantic Schema Domain

Semantic schemas describe the semantics of each method element. They are connected either to elements of the model level, metamodel level or meta^2 model

level. A semantic schema can be defined, e.g., by semantic engines ("script libraries") or by using ontology.
Important aspects to be considered in interoperability of semantic schemas are:

- semantic similarity among semantic schemas and the measurement of the similarity.
- mismatches of ontological constructs used in the semantic schemas.
- merging and integrating semantic schemas into a consolidated and shared semantic schema.

3.7 Persistency Services Domain

Persistency services provide support for durable storage of the various bases. Some of the relevant interoperability issues in this domain are:

- heterogeneous structures of underlying data sources such as relational DBMS, object-oriented storage systems, XML-based databases, or file systems.
- different transaction systems which encumber an interoperable commit strategy.
- heterogeneous user, user profiles and connect definitions to make consistent data access difficult or even impossible (single sign-on).

3.8 Access Services Domain

Interoperability issues in this domain are mainly caused by system heterogeneity. It can be separated into problems of direct (via API) or indirect (via files) exchange:

- Direct exchange can be supported by metamodelling platform API. Some important interoperability issues are:
- the involved providers must agree on necessary interfaces regarding their *programming languages, method signatures* and in general about the *security handling* and the *access rights*.
- Agreement on standardized "protocols" as suggested in [2] by using a general "model bus" where each vendor could get attached by implementing one of the provided protocols.

In indirect exchange the supported file formats play an important role such as XMI [17], HUTN [20], XML, and proprietary formats:

- In case of *proprietary formats* a parser must be implemented to be able to interpret the file syntax. Then rules must be defined to convert the semantic content and the target file (format) must be generated.
- *Standard formats* and *languages* have the advantage that their syntax and partly their semantics are given. Also standardized script languages, e. g., XSLT [30] or XQuery [29] for XML are provided.

4 Case Study

On the basis of an example from the financial services sector, namely the insurance sector, interoperability issues of the metamodel and the access service domain are demonstrated.

The example consists of three metamodels which are instances of the ADONIS [1] meta^2 model (quality management, business process management and ERP introduction metamodel). These and an additional metamodel of a fixed metamodelling platform should be integrated into a new metamodel (Figure 3). In the following, the four metamodels and their integration on the metamodel level and model level is described.

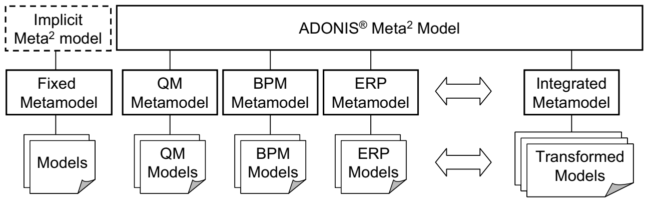


Figure 3. Integration and Interoperability of four Different Metamodels

The main part of the *ERP metamodel* (Figure 4 top right) consists of a process flow and additional process objects. The process flow generalizes the components event, function and the logical operators. There are two types of functions: basic functions and decomposition functions. Specializations of the additional objects are organizational unit, information system, and information object.

The *quality management metamodel* (Figure 4 top left) contains four different model types: process overview, business process model, organizational model and system model. The differences of the type business process model and the process model in the ERP metamodel are the missing class event and the missing operator XOR. Therefore it contains additionally classes such as start and end. The process overview contains processes which could refer to other processes or to business model processes. Additionally, a process could reference a document. The organizational model consists of organisational units and actors which have a role and which could be referenced by an activity from the business process model. The system model contains system components which could be connected via data flows. There are two types of system components, systems and subsystems which could refer to another system model. The systems are referenced by input/output information classes.

The *business process metamodel* is mainly contained in the quality management model, which is described above. The ERP metamodel and the *fixed metamodel* are very similar. Therefore and due to a lack of space only two of the four different source metamodels are illustrated in Figure 4.

To ensure syntactical interoperability, in the target metamodel all of these concepts must be integrated. To enable the transformation of the existing models from the old platform to the new one the mapping of the classes between each source metamodel and the new integrated metamodel must be defined.

Figure 4 graphically illustrates the syntactical mapping between the metamodels. The new integrated metamodel is shown at the bottom of Figure 4. It contains seven model types, the same four as the quality management metamodel and additional three new pool model types. The pool models summarize all documents, all roles and all process owners. Due to the fact that the new system model does not provide the class subsystem, the structure of the ERP models has to be flattened implying a loss of information. Also for each ERP model a process start and an end must be newly created to match the syntax of the integrated metamodel. The events in the ERP models have to be eliminated and the logical operator XOR must be converted into a decision. Moreover, many classes have to be renamed, for example function in activity and decomposition function in sub process as shown in Figure 4.

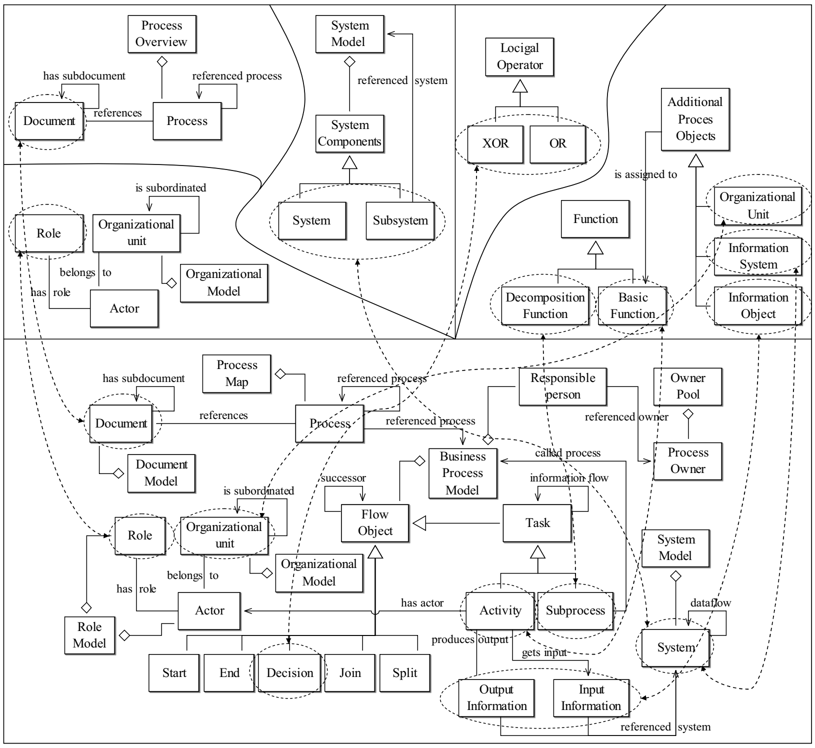


Figure 4. Integration of Metamodels

To physically transform the models to fit to the new integrated metamodel the interoperability problems concerning the access service domain must be solved. In our case the models of three source metamodels are described with the same format, the ADONIS XML format. The additional models of the fixed metamodel environment are also described in XML but in a different structure. Thus, first the models which are described differently must be converted into the structure of ADONIS XML format. This transformation could be done by the means of an XSLT script which does this conversion. After that all model files exist in the same structure. Now a transformation tool is needed to automatically transform the models to fit to the new integrated metamodel. For this purpose the BOC Model Transformer (BMT) [10, 15] has been used. It is a tool that supports the transformation of models between different modelling languages within ADONIS. Different kinds of navigations, rules, functions, conditions and definitions make it possible to specify a rule file which contains the semantical mappings for the transformation of the models between each source metamodel and the new integrated metamodel. Furthermore the BMT supports the creation of graphical information for new objects, for example the start and end classes. Also the interdependencies within and between the models are preserved and newly derived dependencies may be created automatically. After the transformation of all models they comply to the integrated metamodel and could be imported into the new metamodel which has been configured in ADONIS.

5 Related Work

Due to the fact that this work provides an overview of existing problems regarding interoperability issues in metamodelling platforms, the following related work concentrates on technology and approaches in the metamodelling domain.

ADONIS is a meta business process management tool [1]. It offers a three-step modelling hierarchy with a rich meta^2 model. Meta models can be customized as instances of the meta^2 model. Mechanisms such as "simulation" or "analysis" are defined on the meta^2 model level and can be redefined on the metamodel level. The scripting language AdoScript provides mechanisms to define specific behaviour and functionalities.

MetaEdit+ offers also a three-step modelling hierarchy [13]. The meta^2 model forms the "GOPRR" model, offering the basic concepts "Graph", "Object", "Property", "Relationship" and "Role". A diagram editor, object and graph browsers and property dialogs support the definition of a new modelling language without hand coding. Furthermore MetaEdit+ includes XML import and export, an API for data and control access and a generic code generator.

The OMG′s Meta Object Facility (MOF) [18], the open source Eclipse Modelling Framework (EMF) [26] and the Graphical Editor Framework (GEF) [25] are no metamodelling platforms themselves. With the MOF the OMG created a meta^2 model standard, which provides a basis for defining modelling frameworks. UML [22] and the Common Warehouse Metamodel (CWM) [21] are examples of instantiated meta models of the MOF. Interoperability issues concerning the meta model and the model domain are addressed by the ongoing standardisation of MOF Query/Views/Transformations (QVT) [23] which should provide mechanisms for mappings between models and meta models. The EMF which was influenced by the MOF is a shared code base for public use. Together with the GEF it provides a possibility to create a new modelling tool.

The main difference between metamodelling platforms such as MetaEdit+ and ADONIS, and MOF or EMF is that metamodelling platforms provide the user a graphical environment to create new meta models, whereas with MOF and EMF everything must be coded.

It is easier and faster to build new meta models within metamodelling platforms, but due to their implemented meta^2 models the degree of freedom to create a specific meta model is lower than in modelling frameworks like EMF [7].

In [4] E-MORF - a XSLT-based transformation tool - is introduced. E-MORF supports the conversion between MOF and EMF. The transformation is executed by applying the XSLT to XMI which is supported by MOF and EMF. Beside the mapping concept where all fragments of both meta^2 models are related also the mapping problems for example “non-corresponding fragments” and “ame mangling” are described.

The XMF (eXecutable Metamodelling Facility) [3] created by Xactium is a metamodelling facility that fully supports language definition. At the heart of XMF is XCore, the metamodel of XMF, which is comparable to the MOF model. To support mappings between models two further languages are defined, namely XMap, which is a unidirectional pattern based mapping language, and XSync, which is a bidirectional synchronisation language.

6 Conclusion

Metamodelling platforms are getting more and more a kind of base technology [6]. Additionally, domain specific languages, model transformation approaches, and lifecycle management within large model bases are active research issues. The interoperability of metamodelling platforms becomes a crucial aspect in managing corporations' knowledge assets. This paper presented an overview of interoperability issues according to conceptual domains in metamodelling platform architectures. Some of these aspects were illustrated by a case study from the insurance sector. The issues overview can serve as a starting point to stimulate further research on interoperability problems in the metamodelling platform domain.

Additionally to interoperability, we see three important trends in the area of metamodelling platforms in the near future:

- *Metamodelling gets commodity*: metamodelling provides suitable concepts for flexible and interoperable solutions for modelling platforms. Furthermore, metamodelling concepts spread more and more into other domains, such as MOF, UML 2.0, and product-line software development. We expect that metamodelling will also get more attention in domains such as Workflow Management, IT Architecture Management, and Knowledge Management. With this evolution in mind, challenging interoperability issues will have to be solved.
- *Integration of business-oriented and IT-oriented methodologies*: we see strong demands integrating approaches such as Strategy Management,

Process Management and IT Management into single, integrated methods. A promising approach is MOF and MDA. Nevertheless, their focus currently concentrates on system development. Upper-level models such as business specifications and computation independent models (CIM) are not well represented until now. More research dealing with semantic transformations is needed.

- *Method Integration and Knowledge Management*: our society is regarded as "knowledge society". Methods represent experts knowledge, how to do and process things in a certain way. As a future research domain we see the investigation of interdependencies of knowledge management and method engineering and corresponding issues in integrating both.

References

1. ADONIS Homepage. www.boc-eu.com, access 30 November 2004.
2. X. Blanc, M. Gervais, P. Sriplakich: Model Bus: Towards the interoperability of modelling tools, Proceedings of Model-Driven Architecture: Foundations and Applications 2004, Linköping, Sweden.
3. T. Clark; A. Evans; P. Sammut; J. Willans: Applied Metamodelling – A Foundation for
   Language Driven Development, Version 0.1, Xactium, August 2004.
4. K. Duddy; A. Gerber; K. Raymond: Eclipse Modeling Framework (EMF) import/export from MOF / JMI, DSTC Technical Report, May 2003.
5. Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing. In: Guarino, N.; Poli, R. (Eds.): Proceedings of the International Workshop of Formal Ontology, Padova, Italien, August 1993.
6. D. Karagiannis; H. Kühn: Metamodelling Platforms. Invited paper in: Bauknecht, K.; Tjoa, A Min.; Quirchmayer, G. (eds.): Proceedings of the Third International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence, France, September 2-6, 2002, LNCS 2455, Springer-Verlag, Berlin, Heidelberg.
7. S. Kelly: Comparison of Eclipse EMF/GEF and MetaEdit+ for DSM, http://www.softmetaware.com/oopsla2004/kelly.pdf, access 20 November 2004
8. H. Kühn: Methodenintegration im Business Engineering. PhD Thesis, University of Vienna, April 2004.
9. H. Kühn; F. Bayer; S. Junginger; D. Karagiannis: Enterprise Model Integration. In: Bauknecht, K.; Tjoa, A M.; Quirchmayr, G. (Hrsg.): Proceedings of the 4th International Conference EC-Web 2003 - Dexa 2003, Prague, Czech Republic, September 2003, LNCS 2738, Springer-Verlag, pp. 379-392.
10. H. Kühn; M. Murzek; F. Bayer: Horizontal Business Process Model Interoperability using Model Transformation. In: INTEREST'2004 Workshop at ECOOP 2004, Oslo, Norway, June 2004.
11. Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason IV C., Nordstrom G., Sprinkle J., Volgyesi P. The Generic Modeling Environment, Workshop on Intelligent
    Signal Processing at WISP'2001, Budapest, Hungary, May 17, 2001.
12. Linthicum, D. S.: Enterprise Application Integration. Addison-Wesley, 2000.
13. MetaEdit+ Homepage. www.metacase.com, access 30 November 2004.
14. METIS Homepage. www.computas.com, access 30 November 2004.

15. M. Murzek: Methodenübergreifende Modelltransformationen am Beispiel von ADONIS. Diploma Thesis, University of Vienna, April 2004.
16. NoE INTEROP, Interoperability Research for Networked Enterprises Applications and Software, IST Network of Excellence, www.interop-noe.org, 2004.
17. Object Management Group: OMG XML Metadata Interchange (XMI) Specification, Version 1.2, January 2002. http://www.omg.org/cgi-bin/doc?formal/02-01-01.pdf, access 10 November 2004.
18. Object Management Group: Meta Object Facility (MOF) Specification, Version 1.4, April 2002. http://www.omg.org/cgi-bin/doc?formal/02-04-03.pdf, access 30 November 2004.
19. Object Management Group: MDA Guide, Version 1.0.1, 12. June 2003. http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf, access 30 November 2004.
20. Object Management Group: Human-Usable Textual Notation (HUTN) Specification, Final Adopted Specification December 2002, http://www.omg.org/docs/ptc/02-12-01.pdf, access 30 November 2004.
21. Object Management Group: Common Warehouse Metamodel (CWM), OMG Specification/203-03-02, March 2003, http://www.omg.org/docs/formal/03-03-02.pdf, access 30 November 2004.
22. Object Management Group: Unified Modeling Language Specification, OMG Specification/2003-09-01, September 2003, http://www.omg.org/docs/formal/03-03-01.pdf, access 30 November 2004.
23. Object Management Group: Revised submission for MOF 2.0 Query/Views/ Transformations RFP, OMG Specification/2003-08-18, August 2003, http://www.omg.org/docs/ad/03-08-08.pdf, access 30 November 2004.
24. A. M. Ouskel; A. Sheth: Semantic Interoperability in Global Information Systems. A brief Introduction to the Research Area and the Special Section, SIGMOD Record Vol. 28, No. 1, March 1999.
25. The Graphical Editing Framework (GMF), http://www.eclipse.org/gef/.
26. The Eclipse Modeling Framework (EMF), http://www.eclipse.org/emf/.
27. UEML – Unified Enterprise Modelling Language, http://www.ueml.org, access 30 November 2004.
28. P. Valduriez: Parallel Database Systems: Open Problems and New Issues, Distributed and Parallel Databases, April 1993
29. World Wide Web Consortium (W3C): XML Query (XQuery) Version 1.0, February 2005, http://www.w3c.org/XML/query/.
30. World Wide Web Consortium (W3C): XSL Transformations (XSLT) Version 2.0, February 2005, http://www.w3.org/TR/xslt.

Towards Using UML 2 for Modelling Web Service Collaboration Protocols

Gerhard Kramler[1], Elisabeth Kapsammer[2], Werner Retschitzegger[2], and Gerti Kappel[1]

[1] Business Informatics Group, Vienna University of Technology, Austria {kramler, gerti}@big.tuwien.ac.at
[2] Department of Information Systems, Johannes Kepler University of Linz, Austria {ek, werner}@ifs.uni-linz.ac.at

Summary. In a web environment, graphical specifications of service collaborations which focus on the protocols of collaborating services are especially important, in order to attain the desired properties of interoperability and loose coupling. Different to modelling of generic software component collaborations, additional requirements must be considered, including security and transaction aspects, and the characteristics of specific target technologies such as ebXML and BPEL. This paper describes a UML-based approach for platform independent modelling web service collaboration protocols, which takes into account the specific requirements, and supports mappings to relevant target technologies.

1 Introduction

Web services are being used for the coordination of communicating processes, e.g., in the automation of business collaborations such as the standard airline ticketing example. Specification of such collaborations in a manner that facilitates interoperability and loose coupling is provided by a so-called *collaboration protocol*, which provides a global public view on multiple cooperating web services. Collaboration protocols, also called choreographies [15] or conversation policies [8], can be specified using languages like ebXML Business Process Specification Schema (BPSS) [13] and Web Services Choreography Description Language (WS-CDL) [15].

Related to collaboration protocols are interface specification and implementation of a web service. An *interface* describes the public aspects of a web service, including both its provided and required operations as well as its observable behavior. The behavioral aspect of an interface is also called choreography [5, 14], orchestration [5], and abstract process [1]. Web service interfaces are specified using languages like WSDL, BPEL, and WSCI. An *implementation* (also called executable process [1]) is the private aspect of a web service, specified by a language such as BPEL, Java, etc.

Using current Web Services languages, i.e., BPEL and WSDL for the specification of web service collaborations brings up three problems: (1) the languages are XML-based, lacking a standardized graphical representation, which would ease modelling and understanding of collaboration protocols, (2) there is no support for collaboration protocols but only for individual interfaces, leading to potential consistency problems [4], and (3) the level of abstraction is too low for conveniently expressing transactions, as specifically useful in business collaborations [3].

Although there are approaches addressing these problems, no complete solution has been found yet. There is a UML profile for BPEL [7] addressing problem (1). WS-CDL complements BPEL by addressing problem (2) but does not provide a graphical representation. BPSS addresses (3) and there is also a UML representation for BPSS [11] addressing (2), however, these approaches does not support Web Service specification languages and lack flexible specification of intra-transactional interactions. One of the origins of BPSS, the UN/CEFACT Modeling Methodology (UMM) addresses (1-3) but it is limited to the domain of business collaborations and not supporting the specifics of Web Service technology.

Our approach to cope with the identified problems is a UML-based modelling technique that supports platform independent modelling of web service collaboration protocols and that is closely aligned with BPEL and BPSS concepts. In particular, the semantics of UML 2 are refined for applying it to collaboration protocol modelling. This way, no new language and notation need to be invented and problems (1-2) are addressed. Furthermore, different levels of abstraction are supported, thereby supporting both top-down and bottom-up development and addressing problem (3).

In the next section, an overview of our proposed modelling technique is presented. The used UML diagrams and specific semantics are elaborated in sections 3–5. A brief comparison of our approach to related work is given in Section 6. The paper concludes with the open issues that need to be resolved to make the whole approach operational.

2 The Big Picture

The main idea of our approach is to explore UML's existing modelling concepts for collaboration protocol modelling. Therefore, we attempt to identify modelling concepts in UML that are similar to those of the target technologies. Since in many cases no direct equivalence can be found, we first define a platform independent modelling technique, and in a second step define a mapping to specific platforms.

We identify the main specification concepts of the target technologies based on a previously conducted comparison [3]. In that comparison, the layered architecture of the eCo framework [6] was used as basis for comparison, including the layers information items, documents, interactions, and services.

- The concepts dealt with in the *information items* and *documents layer* are the data structures of documents being exchanged among the participants of a collaboration. Re-usable data structures are of particular interest here.
- In the *interaction layer*, the main concept is the message sent from one participant to another, conveying documents. A meaningful interaction is formed by one or more message exchanges.

- The *service layer* considers transactions among participants, and the composition of transactions to build up complex services.
- Furthermore, the *business and market layers*, which were not used in [3] but will be in this paper, consider different types of participants based on the set of services they provide to and require from related participants.

The realization of these layers and concepts in the target technologies, i.e., ebXML and WSDL/BPEL, has been discussed in [3]. For the creation of an UML-based PIM, we need to find modelling concepts that are suitable to both target technologies and that are supported by UML.

UML provides a range of diagrams that cover the concepts in the above named layers. We propose the use of five kinds of UML models as shown in Fig. 1. The models are arranged in a layered architecture which - due to the idiosyncracies of UML - is different from the eCo one. The different levels are interrelated by refinement and usage relationships, meaning that elements specified in one level are refined at lower levels, and conversely, specification elements can be re-used at upper levels. Thus both top-down and bottom-up development is supported.

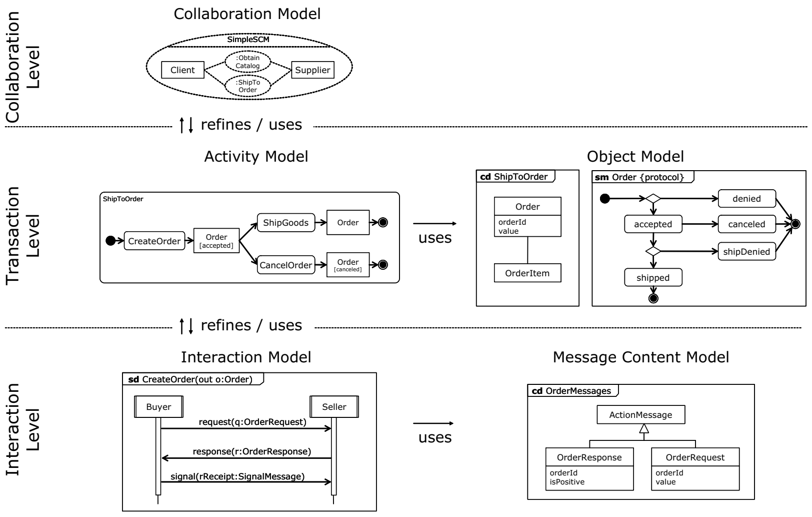


Fig. 1. Abstraction levels and kinds of models

The *collaboration level* roughly corresponds to eCo's business and market layers. It is concerned with participants and their collaboration and communication relationships, thus providing an overview of a collaboration. This is expressed in the collaboration model, a UML collaboration diagram.

The *transaction level* corresponds to the services layer of eCo and considers transactions and transactional processes, each transaction being performed by a set of participants in collaboration. This level abstracts from the distribution of state and control in a collaboration to provide a convenient high-level model. Two kinds of models are proposed in this level. The *activity model*, a UML activity diagram, defines transactional processes, which refine collaborations as defined in the collaboration level. The *object model* uses class diagrams and protocol state machines to define the attributes and states of objects that are used as pre- and post-conditions of transactions.

The *interaction level* covers the eCo interactions, documents, and information items layers. It specifies the messages actually exchanged among participants, thus being at a low level of abstraction. Again two kinds of models are proposed. The *interaction model*, a UML interaction diagram, defines the details of a transaction in terms of message exchanges among the participants, thus refining the individual transactions of the activity model. The *message content model* is a class diagram defining the content of messages.

The following sections discuss how UML is employed to realize the models introduced above, i.e., how a subset of UML 2 is used and what the specific interpretation of the employed UML concepts is.

3 Collaboration Level

UML collaboration diagrams have been chosen for modelling of the business and market layers, because they can express the roles of participants interacting within an overall collaboration. Furthermore, behavior models that refine a collaboration model are always a specification of behavior emerging from the behavior of the individual participants [12], as opposed to specification of executable behavior which is based on a centralized control. Naturally, collaboration protocols are a specification of emergent behavior.

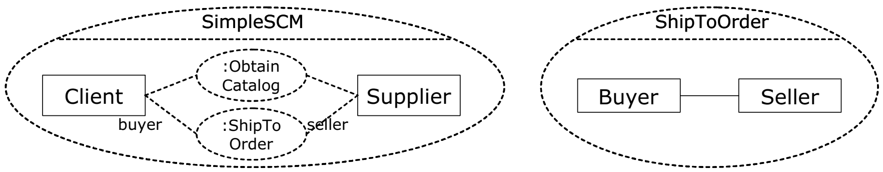


Fig. 2. A simple supply chain collaboration (left) and the sub-collaboration "ShipToOrder" in detail (right)

All of the modelling concepts of UML's collaboration diagrams are used in the collaboration model. Of particular interest is the nesting of collaborations, supporting the definition of composite services or of business areas.

Example 1. Fig. 2 (left) shows the "SimpleSCM" (Simple Supply Chain Management) collaboration between two roles, client and supplier. The collaboration uses two sub-collaborations, the specification of one of them is included in Fig. 2 (right).

There is no behavior attached to the “SimpleSCM” collaboration, meaning that the two sub-collaborations can be performed independent of each other. The sub-collaboration “ShipToOrder” specifies that the two roles, “Buyer” and “Seller”, communicate with each other. The behavior of that collaboration will be further discussed in Example 2.

4 Transaction Level

4.1 Activity Model

The activity model specifies the behavior of a collaboration in terms of transactional processes, using UML activity models. A UML activity is used to define a transactional process, each action within that process representing a transaction performed collaboratively by two or more participants of the overall collaboration. Each transaction may in turn be refined by another activity model, or by an interaction.

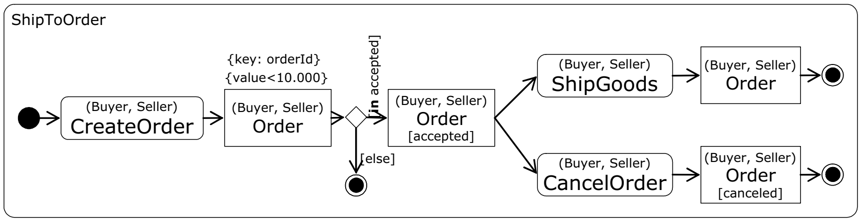

ShipToOrder
(Buyer, Seller) CreateOrder
{key: orderId}
{value<10.000}
(Buyer, Seller) Order
[in accepted]
[else]
(Buyer, Seller) Order [accepted]
(Buyer, Seller) ShipGoods
(Buyer, Seller) Order
(Buyer, Seller) CancelOrder
(Buyer, Seller) Order [canceled]

Fig. 3. Activity model of the “ShipToOrder” collaboration

The input and output parameters of an action define its pre- and post-conditions in terms of collaborative objects (cf. Section 4.2) consumed and produced by the action, respectively. In UML, object nodes are used to define the input and output parameters. Like actions, also object nodes must be assigned to at least two participants, meaning that the pre- and post-conditions apply to those participants.

We emphasize that - opposite to the usual interpretation of activity diagrams - a collaboration activity does not prescribe centralized control. Rather, it represents emergent behavior of all the participants, i.e., each of the participants must behave such that the resulting behavior of the overall collaboration corresponds to the defined collaboration activity. To ensure that a collaboration protocol can be realized without central control, certain restrictions must be met. In particular, a control flow can only be modelled between actions which share at least one participant, otherwise no one of the participants of the succeeding action would have knowledge about when to start. Similarly for object flows. An object flow is only allowed if the target object node is assigned to a subset of the participants that the source object node is assigned to, ensuring that the pre-condition represented by the target object node is known to the participant who is obliged to it.

As an extension to standard UML, we introduce the *key constraint* which specifies for an object node that some of the attributes of the objects contained in the node must be unique among all concurrent instances of the activity. In other words, the key attributes must unambiguously identify the activity instance. This is corresponding to BPELs concept of correlation set. Since key constraints are not natively supported by UML, the proprietary notation "{key: field_1, field_2, ...}" is introduced.

Example 2. The activity shown in Fig. 3 defines the behavior of the "ShipToOrder" collaboration. It comprises the "CreateOrder" action which, in case of success, is followed either by "ShipGoods" or "CancelOrder", based on a non-deterministic choice. Object nodes are used to define pre- and post-conditions on the actions, in particular, state constraints and a key constraint.

An activity may be used to specify the behavior of a collaboration. If the collaboration is used as a top-level collaboration, its activity must not have input and output parameters (e.g., as in Fig. 3). Furthermore, if the collaboration is composed of subcollaborations, the composite collaboration's activity must include (i.e., invoke) its constituent collaborations' activities.

4.2 Object Model

The object model specifies the objects (i.e., both data structure and their behavior) that the collaborative transactions operate on. These objects represent the knowledge common to some or all participants of the collaboration. Note that objects are different from messages. Messages specify the data exchanged among participants, whereas objects specify requirements on the participants' data resources. In many cases, messages represent updates to the objects. Note that this kind of model has not been considered in eCo nor in ebXML or BPEL, but naturally arises by our proposed way of using UML.

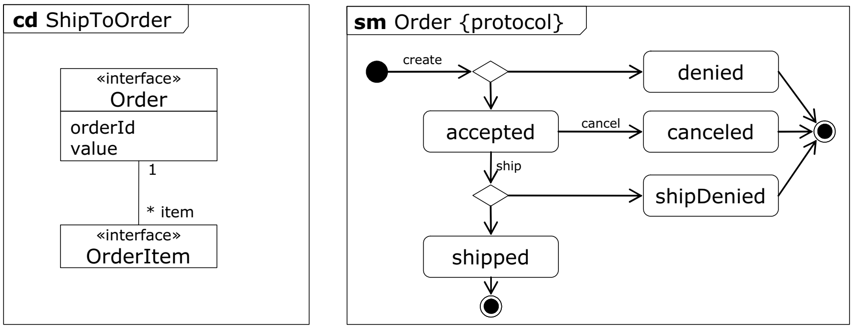


Fig. 4. A data structure used in the collaboration (left) and its permissible behavior (right)

For the purpose of collaboration protocol modelling, only those attributes and states are necessary which are needed for defining the coordination logic, including control flow and data flow constraints. If no decision conditions and no constraints on data are needed, the collaboration object model can be omitted.

Since the collaboration object model must not prescribe the objects used by the participants internally, only interfaces and protocol state machines are used, depicted in a class diagram and state machine diagrams respectively. An *interface* specifies the structure of collaboration-relevant data, whereas a *protocol state machine* specifies the states and permissible state transition of such an interface. The states are used as pre- and post-conditions of transaction specifications, and the transitions are used for consistency checks with the use of objects in activity models. Therefore neither triggering events nor transition conditions need to be specified formally.

Example 3. Fig. 4 (left) shows a class diagram for the data relevant to the "ShipToOrder" collaboration. The specification of the permissible states of an "Order" is shown in the protocol state machine in Fig. 4 (right). Note that transition events are modelled only informally, transition conditions are modelled not at all.

5 Interaction Level

5.1 Interaction Model

The interaction model specifies the interactions among the participants of a transaction in terms of asynchronous message exchanges. The behavior of individual participants is considered, i.e., the notion of a shared state is no longer maintained, but rather different states of the participants and the means of synchronization and coordination need to be defined. Interaction models are intended to be used at a low level of granularity and complexity with request/response as minimal interaction patterns. They refine actions or collaborations. If an interaction model is used to refine an individual action, its parameters must be compatible.

Interaction models must specify how participants achieve a common outcome of the transaction, i.e., at the end of an interaction the participants must know the common outcome in terms of the interaction's output data and the transaction's state. For reasons of loose coupling, the tasks performed by the participants are out of scope of a collaboration model. It is only the overall transaction which represents a common/synchronous task.

The contents of individual messages are interrelated by means of *data flow constraints* rather that operation input/output. In particular, a deterministic data flow constraint defines equality of attributes in different messages or parameters. They are required for key constraints and for specification of post-conditions (see below). To ensure realizability of the collaboration protocol, data flow constraints must be defined based solely on interaction parameters and message arguments, such that both sender and receiver of the involved messages are able to observe the constraint variables.

Similar to the activity model, also the interaction model is extended with the *key constraint* which specifies, for a message, that some of the message's arguments

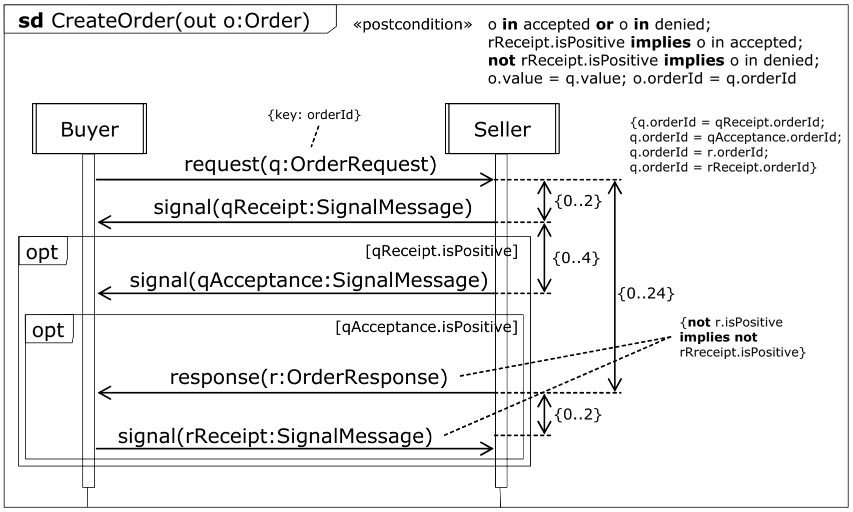

sd CreateOrder(out o:Order)
«postcondition» o in accepted or o in denied;
rReceipt.isPositive implies o in accepted;
not rReceipt.isPositive implies o in denied;
o.value = q.value; o.orderId = q.orderId
Buyer
Seller
{key: orderId}
request(q:OrderRequest)
signal(qReceipt:SignalMessage)
opt
[qReceipt.isPositive]
signal(qAcceptance:SignalMessage)
opt
[qAcceptance.isPositive]
response(r:OrderResponse)
signal(rReceipt:SignalMessage)
{0..2}
{0..4}
{0..24}
{0..2}
{q.orderId = qReceipt.orderId;
q.orderId = qAcceptance.orderId;
q.orderId = r.orderId;
q.orderId = rReceipt.orderId}
{not r.isPositive
implies not
rRreceipt.isPositive}

Fig. 5. Interaction refining the "CreateOrder" action

must be unique among all concurrent instances of the interaction. In other words, the message must unambiguously identify the interaction instance based on the specified key. Sender and recipient (lifelines) may also be used as key components.

The relationship between messages in the interaction model and objects in the activity model is established by means of *post-conditions*. A post-condition specifies the outcome of the interaction in terms of data flow constraints on its output parameters. The post-condition constraints must be observable by all participants who are assigned to the respective output parameter, and the resulting value of all output parameters must be defined.

Example 4. The interaction depicted in Fig. 5 corresponds to a BPSS interaction pattern. It comprises a request and a response message, and accompanying acknowledgement signal messages. Positive acknowledgement messages are a pre-requisite for the interaction to proceed successfully. The data flow constraints shown on the right correlate the "orderId" attributes of the "OrderRequest" and "OrderResponse" messages, and, conditionally, the "isPositive" attributes of the "OrderResponse" and "rReceipt" messages. The post-condition defines the state and value of the output parameter. In particular, the "order" is in state "accepted" only if "rReceipt" was positive, otherwise it is in state "denied".

Within specific domains transactions will often be defined based on generic interaction patterns such as the one used in Example 4. Such generic interaction patterns can be supported by means of interaction templates having template parameters for specification of participants, message types, and timing constraints.

5.2 Message Content Model

The message content model specifies the requirements on message contents. The message model can be specified either completely or in an abstract way. A complete message model defines the message contents unambiguously, i.e., all exchanged documents are specified. Conversely, an abstract message model specifies only the minimal requirements, as needed for the collaboration specification. An abstract message model facilitates re-use of variants of complete message models, e.g., different business document standards could be supported.

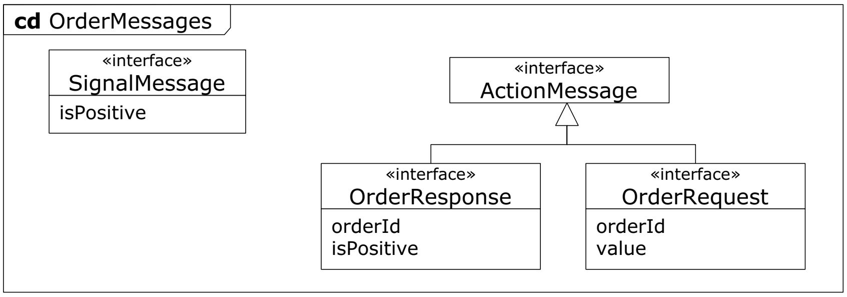


Fig. 6. Abstract messages used by the "CreateOrder" interaction

The concepts used in the message model are interfaces and classes. *Interfaces* are used to specify an abstract message model, whereas *classes* are used to specify complete message models. For specifying XML-related characteristics of message contents, an appropriate UML profile has to be used [2].

Example 5. The example in Fig. 6 shows an abstract message model for the messages in the "ShipToOrder" collaboration. The messages are based on generic ebXML messages "SignalMessage" and "ActionMessage".

6 Related Work

Several approaches to graphically modelling collaboration protocols exist, related to BPSS and based on UML. Considering the web services area, most research deals with modelling of interfaces and implementation of individual web services rather than with collaboration protocols.

Kim [11] takes an approach very similar to ours in that he investigates how UML (version 1.x) diagrams can be used to graphically specify collaboration protocols with an automatic mapping to BPSS. His solution covers the transaction level and the interaction level. At the transaction level, activity diagrams are used. Furthermore, sequence diagrams are used also at the transaction level, with synchronous messages used to represent transactions. This way, behavior of multi-party collaboration protocols is modelled. At the interaction level, both interaction diagrams and class diagrams are used. The major difference of our approach is that we support data flow constraints, both at the interaction level and at the transaction level. Furthermore, our approach is not bound to the limitations imposed by BPSS and is therefore more expressive regarding possible interaction patterns and collaboration processes.

UMM [10] provides not only a rich set of UML-based modelling concepts for B2B collaboration protocols, but also methodological guidance ranging from requirements elicitation to implementation design. UMM has provided the conceptual foundation of BPSS, and since then it was further improved. In particular, it now supports so-called business entities, i.e., business domain objects which are modelled in terms of class diagrams and state diagrams. Furthermore, a business collaboration protocol (the equivalent to a BPSS binary collaboration protocol) can use business entities to define pre- and post-conditions of business transactions. Business entities and their use in business collaboration protocols are very similar to our object model and its use in the activity model. The difference is that business entities capture business semantics, whereas our object model is defined in technical terms. In particular, our object model is formally connected to the messages exchanged in the interaction level, thereby creating an aggregated form of data flow constraint, which is not the case with business entities. As a result, our approach can be directly mapped to the implementation level, which is not the case with UMM.

There exists also a mapping from a subset of UMM to BPEL [9]. It supports the UMM/BPSS interaction patterns, as well as the control flow of business collaboration protocols. In comparison to our approach, that mapping is elaborated in full detail. It considers, however, only a subset of the concepts defined in our models. Difficult mapping problems, e.g., non-deterministic choice, or mapping of failure handling, are still open issues.

The approach described in [8, 4] is not restricted to the business domain but supports web service collaboration protocols in general. Collaboration protocols are specified in terms of a state machine, with states representing the global state of the collaboration, and state transitions representing messages exchanged among its participants. A strong point of this work is that it supports formal verification of consistency between global behavior, i.e., the collaboration protocol, and local behavior of participants, i.e., the interface. In contrast to our approach, only the interaction level is considered, no notion of transaction is provided. Furthermore, it is a purely conceptual model without graphical notation.

7 Summary and Outlook

We have presented a technique for modelling collaboration protocols, which seeks to support the main concepts of both BPSS and BPEL by exploring the features of UML 2. The modelling technique generalizes some of the key concepts of BPSS and UMM, resulting in a language which is no longer specific to the B2B domain but rather supports generic transactional collaboration protocols.

However, several important issues remain open for further research:

- Specification of failures and failure handling has not yet been addressed. At the transaction level, extensions to the activity model are required that cope with failure handling in order to realize transactional properties of long running transactions. At the interaction level, failures of the messaging system have to be considered. Without these extensions, the corresponding aspects need to be defined at the implementation level.
- Non-functional properties such as security and transactional characteristics are still missing. In particular, the respective requirements of BPSS are of interest.
- The mapping to the target technologies has to be elaborated in full detail, to enable automatic code generation. For BPSS this will be straight forward, a mapping to BPEL however amounts to the specification of a protocol implementation. In addition to BPEL and BPSS, support for WS-CDL would be logical but has not yet been considered.
- Finally, to support better integration in a software development process, it would be interesting to consider the relationship of collaboration protocol models to models of web service interfaces and deployments.

References

1. BEA, IBM, Microsoft, SAP, Siebel (2003) Business Process Execution Language for Web Services, Version 1.1. `http://ifr.sap.com/bpel4ws/BPELV1-1May52003Final.pdf`
2. Bernauer M, Kappel G, Kramler G (2004) Representing XML Schema in UML – A Comparison of Approaches. In: Proceedings of the 4th International Conference on Web Engineering (ICWE2004)
3. Bernauer M, Kappel G, Kramler G, Retschitzegger W (2003) Comparing WSDL-based and ebXML-based Approaches for B2B Protocol Specification. In: Proceedings of the 1st International Conference on Service Oriented Computing (ICSOC 2003)
4. Bultan T, Fu X, Hull R, Su J (2003) Conversation specification: a new approach to design and analysis of e-service composition. In: WWW 2003
5. DERI (2004) Web Service Modeling Ontology - Standard, WSMO Working Draft. `http://www.wsmo.org/2004/d2/v02/`
6. eCo Working Group (1999) eCo Architecture for Electronic Commerce Interoperability. `http://eco.commerce.net/rsrc/eCoSpec.pdf`
7. Gardner T (2004) UML Modelling of Automated Business Processes with a Mapping to BPEL4WS. In: Object-Oriented Technology: ECOOP 2003 Workshop Reader, ECOOP 2003 Workshops, Darmstadt, Germany, July 21-25, 2003, Final Reports. Springer LNCS 3013
8. Hanson J E, Nandi P, Kumaran S (2002) Conversation support for Business Process Integration. In: Proceedings 6th IEEE International Enterprise Distributed Object Computing Conference (EDOC-2002)
9. Hofreiter B, Huemer C (2004) Transforming umm business collaboration models to bpel. In: Proc. of the OTM Workshop on Modeling Inter-Organizational Systems (MIOS 2004)

10. Hofreiter B, Huemer C, Naujok K D (2004) Un/cefact's business collaboration framework - motivation and basic concepts. In: Proc. of the Multi-Konferenz Wirtschaftsinformatik (MKWI 2004)
11. Kim H (2002) Conceptual Modeling and Specification Generation for B2B Business Processes based on ebXML. In: SIGMOD Record Volume 31
12. OMG (2003) UML 2.0 Superstructure Specification. OMG Adopted Specification ptc/03-08-02
13. UN/CEFACT and OASIS (2001) ebXML Business Process Specification Schema, Version 1.01. `http://www.ebxml.org/specs/ebBPSS.pdf`
14. W3C (2002) Web Service Choreography Interface (WSCI) 1.0, W3C Note. `http://www.w3.org/TR/wsci`
15. W3C (2004) Web Services Choreography Description Language Version 1.0, W3C Working Draft. `http://www.w3.org/TR/ws-cdl-10/`

Quantitative Analysis of Enterprise Architectures

Maria-Eugenia Iacob and Henk Jonkers

Telematica Instituut, P.O. Box 589, 7500 AN Enschede, the Netherlands
{MariaEugenia.Iacob, Henk.Jonkers}@telin.nl

Summary. Enterprise architecture is concerned with a description of all the relevant elements that make up an enterprise and how these elements inter-relate. It covers aspects ranging from the technical infrastructure, through software applications, to business processes and products. The relations between these layers play a central role. Also from a quantitative analysis perspective, the layers are interrelated: the higher layers impose a workload on the lower layers, while the performance characteristics of the lower layers directly influence the performance of the higher layers. This paper presents an approach for quantitative analysis of layered, service-based enterprise architecture models, which consists of two phases: a 'top-down' propagation of workload parameters, and a 'bottom-up' propagation of performance or cost measures. By means of an example we demonstrate the application of the approach, and show that a seamless integration with other performance analysis methods (e.g., queueing analysis) can be achieved.

1 Introduction

An enterprise can be viewed as a complex 'system' consisting of multiple domains that influence each other. Architectures are used to describe components, their relations and underlying design principles of a system [9]. Constructing architectures for an enterprise may help to, among others, increase the insight and overview required to successfully align the business and ICT. Although the value of architecture has been recognised by many organisations, mostly architectures for various organisational domains, such as business processes, applications, information and technical infrastructure, are developed in isolation. The relations between these architectures often remain unspecified.

Enterprise architecture is a discipline that focuses on making these relations explicit. Models play an important role in all approaches to enterprise architecture. However, currently they strongly focus on functional aspects.

In contrast to detailed design models within domains, the quantitative aspects of such enterprise architecture models have hardly received any attention in literature. Nevertheless, quantitative properties are also important at the enterprise architecture level. For example, "the business" imposes performance requirements on the applications and technical infrastructure, while the performance characteristics of systems influence the quantitative behaviour of business processes. The availability of global performance and cost estimates in the early architectural design stages can provide invaluable support for system design decisions, and prevent the need for expensive redesigns at later stages.

In this paper we present an approach for quantification and performance analysis of enterprise architectures. This approach is based on the propagation of quantitative input parameters and of calculated performance measures through a service-oriented architectural model. It complements existing detailed performance analysis techniques (e.g., queueing analysis), which can be plugged in to provide the performance results for the model elements.

2 State of the Art in Architecture Performance Analysis

As mentioned, enterprise architecture covers a wide range of aspects, from the technical infrastructure layer (e.g., computer hardware and networks), through software applications running on top of the infrastructure, to business processes supported by these applications. In each layer, quantitative analysis techniques can be applied, often requiring detailed models as input. In this section, we will only be able to give a global impression of analysis approaches.

We also noted earlier that enterprise architecture is specifically concerned with how the different layers *interoperate*. Also from a quantitative perspective there is need for interoperability across layers: higher layers impose a workload on lower layers, while the performance characteristics of the lower layers directly influence the performance of the higher layers. However, techniques that cover quantitative analysis throughout this whole 'stack' hardly exist.

2.1 Infrastructure Layer

Traditionally, approaches to performance evaluation of computer systems and communication systems [6] have a strong focus on the infrastructure domain. Queueing models, for example, describe the characteristics of the (hardware) resources in a system, while the workload imposed by the applications is captured by an abstract stochastic arrival process. Also, a lot of literature exists about performance studies of specific hardware configurations, sometimes extended to the system software and middleware level. Most of these approaches have in common that they are based on detailed models and require detailed input data.

2.2 Application Layer

Performance engineering of software applications [17] is a much newer discipline compared to the traditional techniques described above. A number of papers consider performance of software *architectures* at a global level. Bosch and Grahn [3] present some observations about the performance characteristics of a number of often-occurring architectural styles. Performance issues in the context of the SAAM method [13] for scenario-based analysis are considered in [14].

Another direction of research address the approaches that have been proposed to derive queuing models from a software architecture described in an architecture description language (ADL). The method described by Spitznagel and Garlan [18] is restricted to a number of popular architectural styles (e.g., the distributed message passing style but not the pipe and filter style). In [5] queueing models are derived from UML 2.0 specifications which, however, in most cases do not have an analytical solution.

2.3 Business Layer

Several business process modelling tools provide support for quantitative analysis through discrete-event simulation. Also, general-purpose simulation tool, e.g., Arena or ExSpect (based on high-level Petri nets) are often used for this purpose. A drawback of simulation is that it requires detailed input data, and for inexperienced users it may be difficult to use and to correctly interpret the results. Testbed Studio [2] offers, in addition to simulation, a number of analytical methods. They include completion time and critical path analysis of business processes [10] and queueing model analysis [12]. Petri nets (and several of its variations) are fairly popular in business process modelling, either to directly model processes or as a semantic foundation. They offer possibilities for performance analysis based on simulation, but they also allow for analytical solutions (which are, however, fairly computation-intensive). Business process analysis with stochastic Petri nets is the subject of, among others, [15].

3 The ArchiMate Enterprise Modelling Language

Enterprise architecture refers to a consistent whole of principles, methods and models that are used in the design and realisation of organisational structure, business processes, information systems, and infrastructure. However, these domains are not approached in an integrated way, which makes it difficult to judge the effects of proposed changes. Every domain speaks its own language, draws its own models, and uses its own techniques and tools. Communication and decision making across domains is therefore seriously impaired.

In contrast to languages for models within a domain (e.g., the Unified Modelling Language, UML [16] for modelling applications and the technical infrastructure, or the Business Process Modelling Notation BPMN [4] for modelling business processes), a language for enterprise architecture should describe the elements of an enterprise at a relatively *high level of abstraction*, and should pay particular attention to the *relations* between these elements.

Following the principle of cross-domain integration, the ArchiMate project [1] concentrates on modelling, visualisation and analysis of architectures, providing architects with concepts, techniques and tools for architectural design. Central in this is a language for various architectural domains and their relations. As the analysis technique we present was designed for ArchiMate models, we will first briefly introduce this modelling language.

As the basis for our analysis approach, we use a simplified version of the ArchiMate enterprise modelling language. Figure 1 shows an informal representation of the metamodel of this language with the main concepts and the relations that they may have. They cover the business layer of an enterprise (e.g., the organisational structure and business processes), the application layer (e.g., application components) and the technical infrastructure layer (e.g., devices and networks), as well as relation betweens the layers. The language considers the structural, behavioural and informational aspects within each layer. For a description of the full language, we refer to [11].

In order to allow for the quantitative analysis of models expressed in this language, *attributes* are added to quantify some of the concepts and relations. There can be attributes for both input parameters and analysis results, although the distinction may not always be sharp: the result of one analysis phase may be the input of a later analysis phase. In our approach we identify the specific quantitative attributes that we use for ArchiMate models.

4 Viewpoints on Architecture Performance

Architectures can be described from different viewpoints, which result in different views on architectural models [9]. These views are aimed at different *stakeholders* that have an interest in the modelled system. Also for the performance aspects of a system, a number of viewpoints can be discerned, resulting in different (but related) performance measures:

- **User/customer view** (stakeholders: customer; user of an application/system): *response time*, the time between issuing a request and receiving the result; the response time is the sum of the processing time and waiting times (synchronisation losses).
- **Process view** (stakeholders: process owner; operational manager): *completion time*, the time required to complete one instance of a process (possibly involving multiple customers, orders, products etc., as opposed to the response time, which is defined as the time to complete one request).
- **Product view** (stakeholders: product manager; operational manager): *processing time*, the amount of time that actual work is performed on the realisation of a certain product or result: the response time without waiting times. This can be orders of magnitude lower than the response time.
- **System view** (stakeholders: system owner; system manager): *throughput*, the number of transactions/requests that are completed per time unit.
- **Resource view** (stakeholder: resource manager; capacity planner): *utilisation*, the percentage of the operational time that a resource is busy. On the one hand, the utilisation is a measure for the effectiveness with which a resource is used. On

the other hand, a high utilisation can be an indication of the fact that the resource is a potential bottleneck.

This is a refinement of the views mentioned in, e.g.,, [7], which only discerns a user view and a system view. Figure 2 summarises the views on performance.

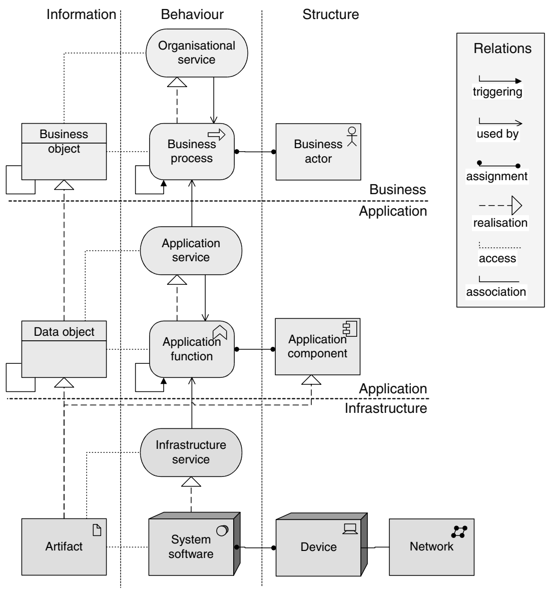


Fig. 1. The ArchiMate metamodel

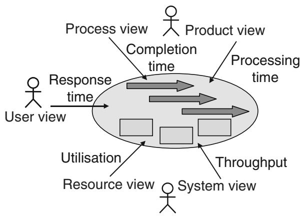


Fig. 2. Performance viewpoints

4.1 Integration of Models and Analysis Results

As indicated in [11], one of the aims of the ArchiMate language is to provide a way to integrate detailed design models expressed in languages such as BPMN [4] for the business layer or UML [16] for the application and infrastructure layers. The ArchiMate model shows the global structure within each domain and the relations between the domains, and contains references to the design models for the details. Language and model integration also forms the basis of tool integration, as described in [20]. Since some modelling tools aimed at a specific domain also offer quantitative analysis capabilities, we ultimately also aim for the integration of quantitative results: in that case, the detailed analysis results are the input for quantitative analysis at the enterprise architecture level. A prerequisite for this is the compositionality of analysis results.

5 Analysis of Enterprise Architecture Models

In this section we present our approach for the quantitative analysis of service-oriented models expressed in the ArchiMate language.

5.1 Model Structure

The metamodel of Figure 1 shows that an architecture model displays a regular structure, which may be viewed as a hierarchy of "layers". We can distinguish layers of two types: *service layers* and *realisation layers*. A service layer exposes external functionality that can be used by other layers, while a realisation layer models the implementation of services. Thus, we separate the externally observable behaviour (expressed as services) from the complex internal organisation (contained within the realisation layers). Figure 6 shows an example of a layered view of an ArchiMate model. Looking at the horizontal structure of the metamodel, we notice that realisation layers basically contain three types of elements. They might model some pieces of *internal behaviour* (expressed as processes, functions or system software). Further, each behaviour element can access *objects*, and is assigned to exactly one *resource* (see Figure 3).

Analysis is possible by propagating quantities through the layers. A natural option for this is to first consider workload measures that are imposed as a "demand" to the model elements from the layers that contain the users of the system (e.g., customers). These quantities propagate to the deeper layers of the architecture, yielding the *demands* of each of the model elements. Once the workloads have been determined, we determine the effort these workloads require from the resources and the behaviour elements. This can be expressed in terms of, e.g., performance measures (e.g., utilisations for resources, processing and response times for behaviour elements) or costs. From the 'deepest' layers of the models, these measures propagate to the higher layers. In summary, our analysis approach consists of the following two phases (see Figure 4): a *top-down* calculation and propagation of the workloads imposed by the top layer; this provides input for a *bottom-up* calculation and propagation of performance measures. In the rest of this section we will show how these phases of analysis can be realised in a systematic manner.

5.2 Quantitative Input

One of the most difficult tasks related to quantitative analysis is to obtain reliable input data. There are several possible sources for this data. For existing systems, measurement is one of the most reliable methods, although it is not easy to do this in a correct way: e.g., it should be clearly defined what exactly is to be measured, the number of measurements must be sufficient and the measurements must be taken under various circumstances that may occur in practice. If the system or organisation is still to be developed, measurement is no option. Possible alternatives are then the use of documentation of components to be used, or to use estimates (e.g. based on comparable architectures). However, it often is very difficult to correctly interpret

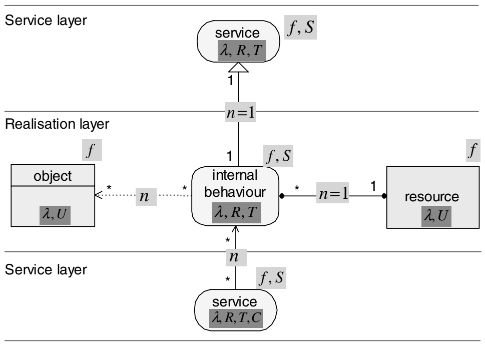


Fig. 3. Structural properties of ArchiMate models

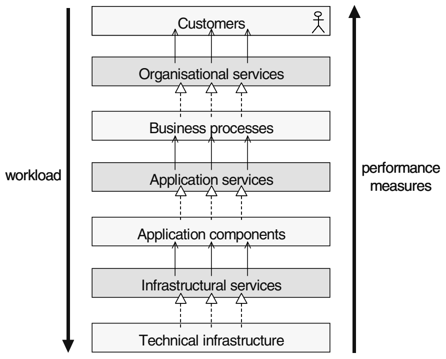


Fig. 4. Layers of ArchiMate models

available numerical data, and to evaluate the reliability of the available data. We assume that the following input is provided for analysis (see Figure 3):

- For any 'used by' and 'access' relation e a weight n_e, representing the average number of uses/accesses. For any 'realisation' and 'assignment' relation, we set $n_e = 1$: they represent a 1-to-1 mappings of a behaviour element to a service and to a resource, respectively.
- For any behaviour element a, a service time S_a representing the time spent internally for the realisation of a service (excluding the time spent waiting for supporting services). Since a service represents the externally observable behaviour of a behaviour element a, we may assume that it inherits the service time from the behaviour element that realises it. Therefore, we leave the choice of specifying this input value for either one of these nodes.
- For any resource r a capacity C_r. By default $C_r = 1$.

- For any node a, an arrival frequency f_a. Typically, arrival frequencies are specified in the top layer of a model, although we do allow for the specification of arrival frequencies for any node in the model.

5.3 Quantitative Results

The goal of our approach is to determine the following performance measures (see Figure 3):

- the workload (arrival rate) λ_a for each node a. (Provided that no resources are overloaded, the throughput for each node is equal to its arrival rate.)
- the processing time T_a and the response time R_a, for each behaviour element or service a
- the utilisation U_r, for each resource r.

5.4 Analysis

To derive performance measures, given the inputs, we proceed in three steps.

Step 1: Model normalisation.

Typical ArchiMate models do often not display the regular structure ArchiMate metamodel. This is due to the fact abstraction rules may be used to create simplified views on the architecture. These abstractions have a formal basis in an operator that has been derived for the composition of relations (see [19] for the details). For instance, a 'realisation' relation with a consecutive used by' relation may be replaced by a new 'used by' relation that short-circuits a service.

The first step in our approach is a model transformation, deriving a normalised version of the input model which conforms to the structure described in Figure 3. Since some concepts and relations are irrelevant for our approach, normalisation starts with eliminating them from the model. Then the model will be subjected to a series of transformations steps, an example of which is given in Figure 5. There is a limited set of transformation rules, eventually resulting in the normalised model. Because model normalisation is not the primary focus of this paper, we omit a formal description of the normalisation algorithm.

The following two steps will be applied to the normalised model.

Step 2: Top-down workload calculation.

For a normalised model, we can calculate the arrival rate for any node a with the following recursive expression:

$$\lambda_a = f_a + \sum_{i=1}^{d_a^+} n_{a,k_i} \lambda_{k_i}, \qquad (1)$$

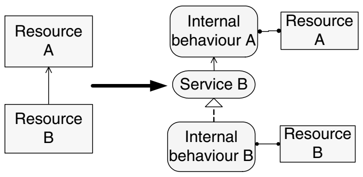


Fig. 5. Example of a normalisation step

where d_a^+ denotes the out-degree of node a and k_i is a child of a. In other words, the arrival rate for a node is determined by adding the requests from higher layers to the local arrival frequency f_a.

Step 3: Bottom-up performance calculation.

Once the workloads on the various model components have been calculated, we can proceed with the bottom-up calculation of the performance measures. The approach is similar to the top-down approach. We focus here on the bottom-up propagation of performance measures. The following recursive expressions apply:

- The utilisation of any resource r is $U_r = \frac{1}{C_r}\sum_{i=1}^{d_r} \lambda_{k_i} T_{k_i}$, where d_r is the number of internal behaviour elements k_i to which the resource is assigned.
- The processing time and response time of a service a coincide with these measures for the internal behaviour element realising it, i.e.: $T_a = T_k$ and $R_a = R_k$, where (k, a) is the *realisation* relation with a as end point. (The service merely exposes the functionality of the internal behaviour element to the environment: there is no additional time consumption).
- The processing time and response time of an internal behaviour element a is computed using the following recursive formulas:

$$T_a = S_a + \sum_{i=1}^{d_a^-} n_{k_i,a} R_{k_i} \qquad\qquad R_a = F(a, r_a) \tag{2}$$

  where d_a^- denotes the in-degree of node a, k_i is a parent of a and r_a is the resource assigned to a and F is the response time expressed as a function of attributes of a and r_a.

For example, if we assume that the node can be modelled as an M/M/1 queue [6], this function is $F(a, r_a) = T_a/(1 - U_{r_a})$. We can replace this by another equation in case other assumptions apply: e.g., the Pollaczek-Khinchine formula for an M/G/1 if T_a has a non-exponential distribution.

In most cases, this will lead to approximate results because the queueing networks are not separable [6]. At the enterprise architecture level, where we are generally interested in global performance estimates, we expect such approximations to be good enough. In case more precise results would be required, instead of simple queueing formulas, more detailed techniques such as simulation can be applied in combination with our approach.

6 Example

In this section we give an example to illustrate the analysis approach. Consider an insurance company using a document management system for the storage of damage reports. We assume that the document management system is a centralised system, used by multiple offices throughout the country, which means that it is quite heavily used. We show how performance measures of this system can be derived using a model of the architecture of the system (see Figure 6). This model covers the whole stack from business processes and actors, through applications, to the technical infrastructure.

There are three applications offering services that are used directly by the business actors. The Administrator can *search* in the metadata database, resulting in a short descriptions of the reports that meet the query and *view* reports that are returned by a search. The *report scanning application* is used to scan, digitise and store damage reports (in PDF-format).

In addition to the two applications that are used directly by the end-user, there are two supporting application components: a *database access* component, providing access to the metadata database, and a *document management* component, providing access to the document base. Finally, the model shows the physical devices of which the database access and document management components make use. They use file access services provided by these devices.

In the model we also specify the analysis inputs. On the 'used by' relations, we specify workloads, in terms of the average number of uses n of the corresponding service. For the business processes, an arrival frequency f is specified. Finally, for services we may specify a service time S. We now proceed to analyse this model, using the three steps described in the previous section.

Step 1: Model normalisation.

We first derive a normalised model that is compliant with the modelling rules described in Figure 3. Figure 7 shows the normalised version of the model in Figure 6. The input parameters for the workload on the 'used by' relations are the same as in the original model. The service times are now transferred also to the inserted internal behaviour elements.

Step 2: Top-down workload analysis.

Figure 8 shows the workload for the services s in the model, in terms of the arrival rates λ_s. The arrival rates depend on the frequencies of the customer input requests

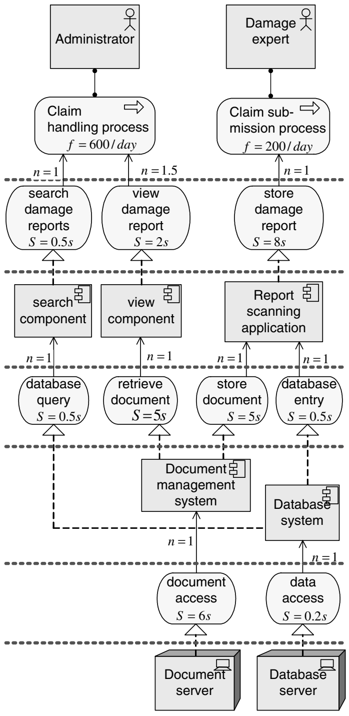

Administrator
Damage expert
Claim handling process $f = 600/day$
Claim submission process $f = 200/day$
$n=1$
$n=1.5$
$n=1$
search damage reports $S = 0.5s$
view damage report $S = 2s$
store damage report $S = 8s$
search component
view component
Report scanning application
database query $S = 0.5s$
retrieve document $S = 5s$
store document $S = 5s$
database entry $S = 0.5s$
Document management system
Database system
document access $S = 6s$
data access $S = 0.2s$
Document server
Database server

Fig. 6. Example model

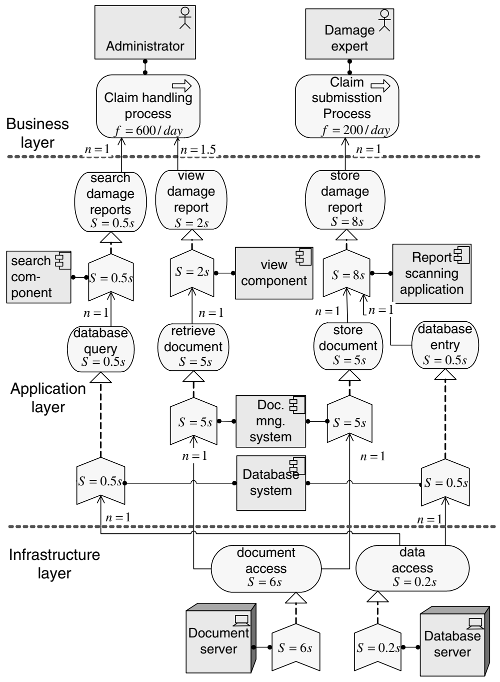

Business layer
Application layer
Infrastructure layer
Administrator
Damage expert
Claim handling process $f = 600/day$
Claim submisstion Process $f = 200/day$
search damage reports $S = 0.5s$
view damage report $S = 2s$
store damage report $S = 8s$
search com-ponent
view component
Report scanning application
database query $S = 0.5s$
retrieve document $S = 5s$
store document $S = 5s$
database entry $S = 0.5s$
Doc. mng. system
Database system
document access $S = 6s$
data access $S = 0.2s$
Document server
Database server

Fig. 7. Normalised model

and the cardinalities n of the 'used by' relations. The table also shows the scaled arrival rates expressed in arrivals/second (assuming that systems are operational eight hours per day).

Step 3: Bottom-up performance analysis.

Figure 8 shows the performance results for the example, i.e., the processing and response times for the services and the utilisations for the resources at the application and infrastructure layer.

The results show that queueing times from the lower layers accumulate in the higher layers, resulting in response times that are orders of magnitude greater than the local service times. E.g., the 'view' component of the 'claim handling support' application has a utilisation of over 84%, which results in a response time of the 'view damage report' application service of almost 3 minutes.

Using our approach, it is easy to study the effect of input parameter changes on the performance. For example, the graph in Figure 8 shows how the response time of the View component depends on the arrival frequency associated with the Administrator (assuming a fixed arrival frequency for the Damage expert). The maximum arrival frequency, which results in a utilisation of the View component of 100%, is 651 arrivals per day. In the design stage these results may help us to decide, e.g., if an extra View component is needed.

Resource (r)	Service (s)	λ_s (sec^{-1})	T_s (sec)	R_s (sec)	U_a (%)
Doc. srv.	doc. acc.	0.0382	6.0	7.8	22.9
DB srv.	data acc.	0.0278	0.2	0.2	0.6
Doc.mgt. sys.	retr. doc.	0.0313	12.8	25.0	48.8
Doc.mgt. sys.	store doc.	0.0069	12.8	25.0	48.8
DB syst.	DB query	0.0278	0.7	0.7	1.9
DB syst.	DB entry	0.0069	0.7	0.7	1.9
Search comp.	search rep.	0.0278	1.2	1.2	2.5
View comp.	view rep.	0.0313	27.0	174.0	84.3
Rep. scanning	store rep.	0.0069	33.7	44.0	23.4

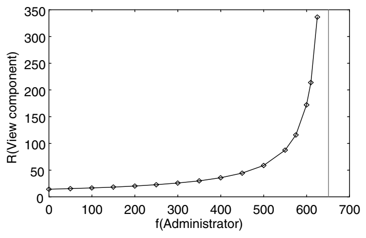


Fig. 8. Workloads and performance results

7 Conclusions and Future Work

Although the importance of enterprise architecture modelling has been recognised, hardly any attention has been paid to the analysis of their quantitative properties. Most existing approaches to performance evaluation focus on detailed models within a specific domain. In this paper we demonstrated the applicability of quantitative modelling and analysis techniques for the effective evaluation of design choices at the enterprise architectures level. We discerned a number of architecture viewpoints with corresponding performance measures, which can be used as criteria for the optimisation or comparison of such designs.

We introduced a new approach for the propagation of workload and performance measures through an enterprise architecture model. This can be used as an analysis framework where existing methods for detailed performance analysis, based on, e.g., queueing models, Petri nets or simulation, can be plugged in. The presented example illustrates the use of our top-down and bottom-up technique to evaluate the performance of a document management system for the storage and retrieval of damage reports. Using a simple queueing formula for the response times, we showed that queueing times from the lower layers of the architecture accumulate in the higher layers, which results in response times that are orders of magnitude greater than the local service times. A prototype has been developed for further illustration and validation of the approach.

Several improvements and extensions to the approach are conceivable. E.g., in [8] we show that our "vertical" approach to propagate workloads and performance measures could also be combined with "horizontal" analysis techniques to evaluate completion times in business processes [10]. Finally, for a further integration of the architectural design process, combining quantitative analysis with functional analysis or visualisation techniques could be fruitful.

Acknowledgement

This paper results from the ArchiMate project [1]. The ArchiMate consortium consists of ABN AMRO, Stichting Pensioenfonds ABP, the Dutch Tax and Customs Administration, Ordina, Telematica Instituut, Centrum voor Wiskunde en Informatica, Radboud Universiteit Nijmegen, and the Leiden Institute of Advanced Computer Science.We thank René van Buuren, Timber Haaker and Hugo ter Doest for their useful remarks that helped to improve our approach.

References

1. ArchiMate Project. http://archimate.telin.nl.
2. BiZZdesign B.V. http://www.bizzdesign.nl.
3. J. Bosch and H. Grahn. Characterising the performance of three architectural styles. In *Proceedings First International Workshop on Software and Performance*, Santa Fe, NM, Oct. 1998.
4. Business Process Management Initiative. Business process modeling notation. working draft (1.0), Aug. 2003.
5. A. Di Marco and P. Inverardi. Compositional generation of software architecture performance QN models. In J. Magee, C. Szyperski, and J. Bosch, editors, *Proceedings Fourth Working IEEE/IFIP Conference on Software Architecture*, pages 37–46, Oslo, Norway, June 2004.
6. P. Harrison and N. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1992.
7. U. Herzog. Formal methods for performance evaluation. In *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science (LNCS 2090)*, pages 1–37. Springe Verlag, 2001.
8. M.-E. Iacob and H. Jonkers. Quantitative analysis of enterprise architectures. Technical Report ArchiMate D3.5b, Telematica Instituut, Enschede, the Netherlands, Mar. 2004.
9. IEEE Computer Society. IEEE standard 1471-2000: Recommended practice for architectural description of software-intensive systems, 2000.
10. H. Jonkers, P. Boekhoudt, M. Rougoor, and E. Wierstra. Completion time and critical path analysis for the optimisation of business process models. In M. Obaidat, A. Nisanci, and B. Sadoun, editors, *Proceedings of the 1999 Summer Computer Simulation Conference*, pages 222–229, Chicago, IL, July 1999.
11. H. Jonkers, M. Lankhorst, R. van Buuren, S. Hoppenbrouwers, M. Bonsangue, and L. van der Torre. Concepts for modelling enterprise architectures. *International Journal of Cooperative Information Systems*, 13(3), Sept. 2004.
12. H. Jonkers and M. van Swelm. Queueing analysis to support distributed system design. In *Proceedings of the 1999 Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 300–307, Chicago, IL, July 1999. M.S. Obaidat and M. Ajmone Marsan.
13. R. Kazman, L. Bass, G. Abowd, and M. Webb. SAAM: A method for analyzing the properties of software architectures. In *Proceedings 16th International Conference on Software Engineering*, pages 81–90, Sorento, Italy, 1994.
14. C.-H. Lung, A. Jalnapurkar, and A. El-Rayess. Performance-oriented software architecture analysis: An experience report. In *Proceedings First International Workshop on Software and Performance*, Santa Fe, NM, Oct. 1998.

15. A. Schömig and H. Rau. A petri net approach for the performance analysis of business processes. Technical Report 116, Lehrstuhl für Informatik III, Universität Würzburg, 1995.
16. K. Scott. *Fast Track UML 2.0.* Apress, 2004.
17. C. Smith. *Performance Engineering of Software Systems.* Addison-Wesley, 1990.
18. B. Spitznagel and D. Garlan. Architecture-based performance analysis. In *Proceedings 1998 Conference on Software Engineering and Knowledge Engineering*, San Francisco Bay, June 1998.
19. R. van Buuren, H. Jonkers, M.-E. Iacob, and P. Strating. Composition of relations in enterprise architcture models. In H. Ehrig, G. Engels, F. Parisi-Presicce, and G. Rozenberg, editors, *Graph Transformations – Proceedings of the Second International Conference (LNCS 3256)*, pages 39–53, Rome, Italy, Sept. 2004.
20. D. van Leeuwen, H. ter Doest, and M. Lankhorst. A tool integration workbench for enterprise architecture. In *Proceedings 6th International Conference on Enterprise Information Systems*, Porto, Portugal, Apr. 2004.

Mapping Specification in MDA: From Theory to Practice [*]

Denivaldo Lopes[1,2], Slimane Hammoudi[1], Jean Bézivin[2], and Frédéric Jouault[2,3]

[1] ESEO, France
[2] Atlas Group, INRIA and LINA, University of Nantes, France
[3] TNI-Valiosys, France
{dlopes, shammoudi}@eseo.fr
{jean.bezivin, frederic.jouault}@lina.univ-nantes.fr

Summary. In this paper, we present a metamodel for supporting the mapping specification between two metamodels. A mapping model based on this proposed metamodel defines correspondences between elements from two metamodels. It can then be used to generate a transformation definition, e.g. using Atlas Transformation Language (ATL). This metamodel is based on the Eclipse Modeling Framework (EMF). A plug-in for Eclipse implements this metamodel for mapping with the aim of providing a tool for supporting mapping specification and for generating the transformation definition.

1 Introduction

Recently, the Object Management Group (OMG) has stimulated and promoted the adoption of the Model Driven Architecture (MDA^{TM})[4] [1] approach for developing large and complex software systems. In this approach, models become the hub of development, separating platform independent characteristics (i.e. Platform-Independent Model - PIM) from platform dependent characteristics (i.e. Platform-Specific Model - PSM).

The MDA approach aims to provide an architecture with which complex software systems (such as B2B applications on the Internet) can evolve for meeting new requirements or new technologies, business logic is protected against the changes in technologies, and legacy systems are integrated and harmonized with new technologies. However, before that becomes a mainstream reality, several issues in MDA approach need solutions, such as mapping, transformation [2], handling of *semantic distance* between metamodels [3], bidirectional mapping [4], and so on.

In this paper, we limit our objectives to providing some insights into *mapping specification* and *transformation definition*. We propose a metamodel for mapping and a tool for editing mapping models based on this metamodel. Afterwards, we

[*] This work has been done in the context of the INTEROP European Network of Excellence.
[4] MDA^{TM} is a trademark of the Object Management Group (OMG).

can use this tool to generate *transformation definition*. Our approach is implemented through a tool using a plug-in for Eclipse [5].

This paper is organized in the following way. Section 2 is an overview of MDA. Section 3 presents our approach for mapping two metamodels within the context of MDA. Section 4 shows the implementation of our proposed metamodel for mapping through a plug-in for Eclipse. Section 5 contains conclusions and presents the future directions of our research.

2 Overview

Some areas, such as civil and electrical engineering, have assimilated the importance of models. In these areas, the model predates the building. Moreover, in some countries, it is impossible to start a building without its models (i.e. an electrical and civil engineering project). Models are largely used during the building, and then for maintenance, for modification (i.e. evolution) and for criminal investigation (such as police investigation in the case of a building that crumbles). In fact, we have much to learn from good practice in other areas. At times, we need to develop more the discipline of model engineering in computer science and its use in software enterprises.

For a long time, modeling languages, such as UML, were only used for documenting software systems, but nowadays models become the impetus for software development thanks to the MDA approach.

MDA is based on an architecture with four meta-layers: metametamodel, metamodel, model and information (i.e. an implementation of its model). In this approach, everything is a model or a model element, and a high level of abstraction and comprehension about a problem and its solution are provided.

The MDA approach has been accepted by companies, universities, governments and organizations as a potential solution for the problem of complexity in the development and maintenance of software systems. The period of skepticism about MDA seems to be over. Several case studies have demonstrated some benefits of the MDA approach, such as gain in productivity and protection against error-prone factors linked with manual programming [6]. Other possible benefits of MDA include:

- an increase in the return on investments in technology.
- a uniform approach for business models and for technologies to evolve together.
- an architecture ready for handling the problems of yesterday, today and tomorrow; and for integrating old, current and new technologies used in software development [7].

Several potential benefits need time to be demonstrated. However, the MDA approach has been used with success in some projects such as the case study presented at [6].

Before reaching a stage comparable to civil and electrical engineering, the MDA approach needs to be more developed and experimented. Recently, some initiatives have contributed to enhancing MDA, such as some response [8] [9] to the RFP-QVT [2] and the creation of tools for implementing the concepts around MDA [10] [11]

[12] [13]. Among these initiatives, we call attention to the Eclipse projects based on the concepts of MDA, such as Eclipse Modeling Framework (EMF) [10] developed under an open source philosophy. This project provided the basis for researchers to develop and implement their ideas and contributions on software engineering. Our aim is to contribute to this project providing an approach for *mapping specification* and *transformation definition*.

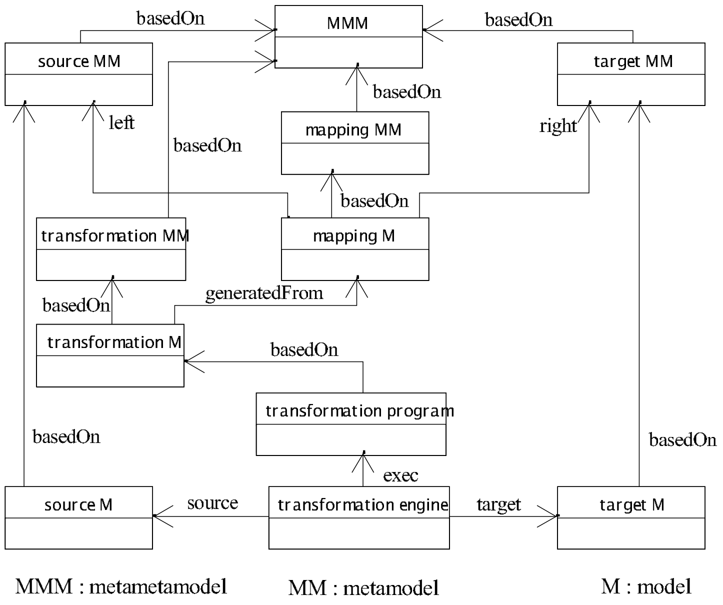


Fig. 1. Transformation Process in MDA

In fact, the transformation of a PIM into a PSM is a process [7]. Here, we extend this process, separating *mapping specification* and *transformation definition*. Figure 1 depicts this transformation process. The new proposed process involves several entities: a metametamodel (such as MOF and Ecore [10]), a source and target metamodel (such as UML), a source and target model, a mapping model, a transformation language model (such as a model based on the ATL [14] and YATL [15] metamodel), and a transformation engine. Mapping model specifies correspondences between the source and target metamodel. A transformation model is generated from a mapping model. A transformation program is based on its transformation model. The transformation is achieved by a transformation engine that executes a transformation program. The transformation engine takes a source model, executes the transformation program, and provides a target model as output.

In this paper, we use the term *mapping* as a synonym for correspondence between the elements of two metamodels, while *transformation* is the activity of transforming a source element into a target element in conformity with the *transformation definition*. Following these two concepts, the *mapping specification* precedes the

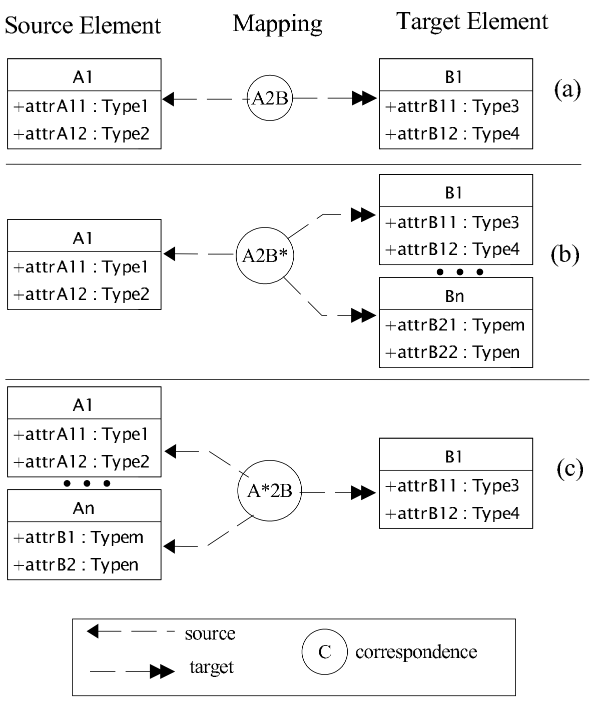


Fig. 2. Categories of mappings: (a)one-to-one, (b)one-to-many and (c) many-to-one

transformation definition. Figure 2 presents three categories of mapping: one-to-one, one-to-many and many-to-one. This classification is based on the concept of similar structure and semantics between the elements of metamodels. Two or more elements from two metamodels present similar structure and semantics, if the target element(s) can represent the same information contained in the source element(s).

A one-to-one mapping is characterized by one element from a target metamodel that may represent the similar structure and semantics of one element from a source metamodel. A one-to-many mapping is characterized by a non-empty and non-unitary set of elements from a target metamodel that presents similar semantics to one element from a source metamodel. A many-to-one mapping is characterized by an element from a target metamodel that presents similar semantics to a non-empty and non-unitary set of elements from a source metamodel. This last mapping is not directly implemented in some languages such as ATL, but it can be simulated using a transformation rule that takes only one element from the source metamodel and determines the other elements based on the relationships between the first element with the others.

Several research projects have studied the specification of mapping between metamodels [16] [17] [18]. However, the ideas around mapping are not sufficiently developed to create efficient tools to enable automatic mappings.

3 Mapping

The creation of *mapping specification* and *transformation definition* is not an easy task. In addition, the manual creation of *mapping specification* and *transformation definition* is a labor-intensive and error-prone process [19]. Thus the creation of an automatic process and tools for enabling them is an important issue. Some propositions enabling the *mapping specification* have been based on heuristics [18] (for identifying structural and naming similarities between models) and on machine learning (for learning mappings) [20]. Other propositions enabling *transformation definition* have been based on graph theory [21]. *Mapping specification* is not a new issue in computer science. For a long time, the database domain has applied mappings between models (i.e. schema) and transformation from different conceptual models, e.g. entity-relationship (ER), into logical or physical models (relational-tables and SQL schema). However, these same issues have taken a new dimension with the sprouting of MDA, because models become the basis to generate software artifacts (including code) and in order to transform one model into another model, *mapping specification* is required. So, both *mapping specification* and *transformation definition* have been recognized as important issues in MDA [7].

In this section, we present our proposition for specifying mappings (i.e. correspondences between metamodels). This approach for mapping is based on a metamodel and implemented as a tool on Eclipse. This tool provides support for mapping, which is a preliminary step before the creation of a *transformation definition*, e.g. using Atlas Transformation Language (ATL) [14].

3.1 Foundation for Mapping

Given $M_1(s)/M_a$, $M_2(s)/M_b$, and $C_{M_a \to M_b}/M_c$, where M_1 is a model of a system s created using the metamodel M_a, M_2 is a model of the same system s created using the metamodel M_b, and $C_{M_a \to M_b}$ is the mapping between M_a and M_b created using the metamodel M_c, then a transformation can be defined as the function $Transf(M_1(s)/M_a, C_{M_a \to M_b}/M_c) \to M_2(s)/M_b$. In this section, we aim to detail $C_{M_a \to M_b}/M_c$. In general, M_a, M_b and M_c are based on the same meta-metamodel which simplifies the *mapping specification*. For now, we can define $C_{M_a \to M_b} \supseteq \{M_a \cap M_b\}$, where $\cap$ is a binary operator that returns the elements of M_a and M_b which have equivalent structure and semantics.

The process of identifying and characterizing inter-relationships between metamodels is denominated *schema matching* [18]. In fact, *mapping* describes how two metamodels [5] are related to each other. So, *schema matching* results in a *mapping*. According to *model management algebra* [22], a *mapping* is generated using an operator called *match* which takes two metamodels as input and returns a *mapping* between them.

The identification of inter-relationships between metamodels is generally based on the structure of the metamodels. The structure of a metamodel is a consequence

[5] In our approach, we prefer employ the term metamodel in the definition of the term mapping.

of relationships between its elements. These relationships relate two metamodel elements and have some characteristics such as kind. Generally, five relationship kinds can relate a model element to another model element [17]: *Association*, *Contains*, *Has-a*, *Is-a*, *Type-of.*

These relationship kinds can be formalized as follow:

- **Association**: $A(a, b)$ means a is associated with b.
- **Contains**: $C(c, d)$ means container c contains d.
- **Has-a**: $H(e, f)$ means e has an f.
- **Is-a**: $I(g, h)$ means g is an h.
- **Type-of**: $T(i, j)$ means i is a type of j.

In [17], the authors propose the five cross-kind-implications:

- if $T(q, r)$ and $I(r, s)$ then $T(q, s)$.
- if $I(p, q)$ and $H(q, r)$ then $H(p, r)$.
- if $I(p, q)$ and $C(q, r)$ then $C(p, r)$.
- if $C(p, q)$ and $I(q, r)$ then $C(p, r)$.
- if $H(p, q)$ and $I(q, r)$ then $H(p, r)$.

The authors of [17] propose that two models are defined equivalents *"if they are identical after all implied relationships are added to each of them until a fix point is reached"*. Applying these relationship kinds and cross-kind-implications to metamodels, they can also be simplified and compared between them to find equivalences and similarities.

3.2 Eclipse Modeling Framework (EMF)

Eclipse Modeling Framework (EMF) is a modeling framework and code generation facility for supporting the creation of tools and applications driven by models [10]. EMF represents the efforts of Eclipse Tools Project to take into account the driven model approach. In fact, MDA and EMF are similar approaches for developing software systems, but each one has different technologies. MDA was first designed by OMG using MOF and UML, while EMF is based on Ecore and stimulates the creation of specific metamodels.

EMF is a framework which meets the requirements of Domain-Specific Languages (DSL) [3]. DSLs are defined by Steve Cook as *"languages that instead of being focused on a particular technological problem such as programming, data interchange or configuration, are designed so that they can more directly represent the problem domain which is being addressed"* [3]. DSL presumes the existence of many metamodels, despite UML which is a general-purpose metamodel. MOF can also be used for creating DSLs, but OMG has focused its efforts on the adoption of UML and UML profiles, to the detriment[6] of specific metamodels.

[6] The OMG has provided a limited number of metamodels, such as UML and Enterprise Distributed Object Computing (EDOC).

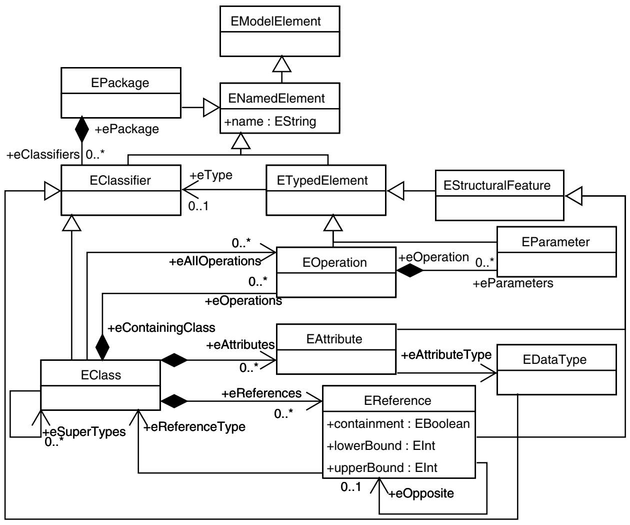


Fig. 3. Ecore metamodel (fragment)

Within the Eclipse's philosophy, EMF is one more plug-in that can be used such as it is or extended by other plug-ins. In the context of Eclipse, plug-in is the mechanism for extending the basic architecture of Eclipse with things that have common or different purposes, where they share a common environment [23].

Figure 3 presents a fragment of Ecore metametamodel [10]. Ecore has some points in common with MOF, but the former is more simple than the latter[7]. In addition, the serialization of a metamodel based on Ecore is more clear and simple than the similar metamodel based on MOF.

3.3 A Metamodel for Mappings

In order to define a mapping, we need a metamodel which enables:

- identification of what elements have similar structures and semantics to be mapped.
- explanation of the evolution in time of the choices taken for mapping one element into another element.
- bidirectional mapping. It is desirable, but is often complex [4].
- independence of model transformation language.
- navigation between the mapped elements.

Figure 4 presents a metamodel for *mapping specification*.

In this metamodel, we consider that a mapping can be unidirectional or bidirectional. In unidirectional mapping, a metamodel is mapped into another metamodel.

[7] This comparison is made using MOF 1.4. However, MOF 2.0 (i.e. Essential MOF - EMOF) is closer to Ecore.

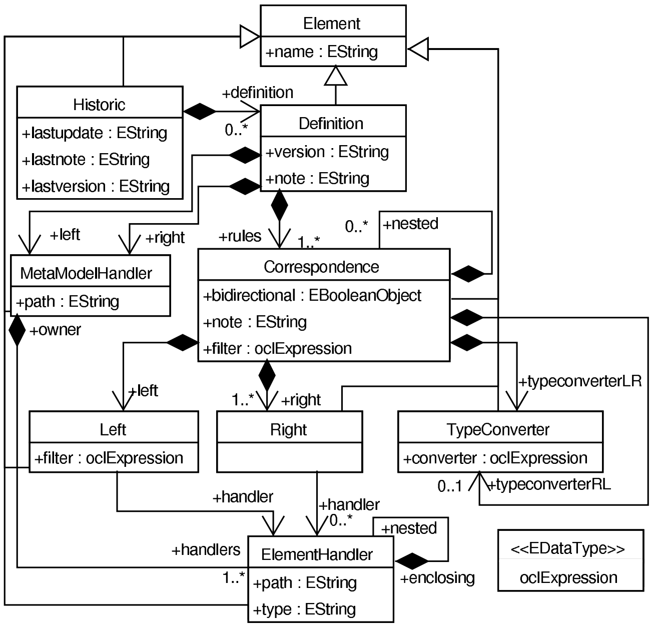


Fig. 4. Metamodel for Mapping Specification

In bidirectional mapping, the mapping is specified in both directions. Thus, we prefer to denominate two metamodels in a mapping as left or right metamodels.

This metamodel presents the following elements:

- `Element` is a generalization for the other elements.
- `Historic` enables the explanation of the different choices taken for making the mapping. It has the date of the last update, a note, and the number of the last version, and a collection of `Definitions`.
- `Definition` is the main element and it contains all correspondences (i.e. `Correspondence`) between two metamodels (i.e. each correspondence has one `left` element and one or more `right` elements).
- `Correspondence` is used to specify the correspondence between two or more elements, i.e. `left` and `right` element. The correspondence has a filter that is an OCL expression. When `bidirectional` is `false`, a mapping is unidirectional (i.e. left to right), and when it is `false` it is bidirectional (i.e. in both directions). It has two `TypeConverters` identified by `typeconverterRL` and `typeconverterLR`. `typeconverterRL` enables the conversion of the elements from a right metamodel into the elements from a left metamodel.
  `typeconverterLR` enables the conversion of the elements from a left metamodel into the elements from a right metamodel. We need often specify only the `typeconverterLR`.
- `Left` is used to identify the left element of a mapping.
- `Right` is used to identify the right elements of a mapping.

- `MetaModelHandler` is used to navigate into a metamodel. It has the information necessary for accessing a metamodel participating in a mapping. A mapping is itself a model, and it must not interfere with the two metamodels being mapped.
- `ElementHandler` enables access to the elements being mapped without changing them.
- `TypeConverter` enables the type casting between a left and a right element. If one element of a left metamodel and another element of a right metamodel are equals, then the mapping is simple and direct. However, if one element of a left metamodel and another element of a right metamodel are similar, then the mapping is complex and it is achieved using type converter, i.e. a complex expression to adapt a left element to a right element.

4 A Plug-in for Eclipse

A tool supporting our proposed metamodel for mapping should provide:

- simplification for visualizing mappings. In order to specify a mapping, two metamodels are necessary. From experience, metamodels have generally a considerable number of elements and associations. So the visualization becomes complex, putting two metamodels so large side by side and the mapping in the center. A tool should allow the creation of views, navigation and encapsulation of details unnecessary for each mapping in order to facilitate the visualization and comprehension of the mapping without modifying the involved metamodels.
- creation of *transformation definition* from *mapping specification*. A *mapping specification* is a model itself, then it can also be transformed into another model. For example, a mapping model can be transformed into a transformation model.

4.1 An Illustrative Example

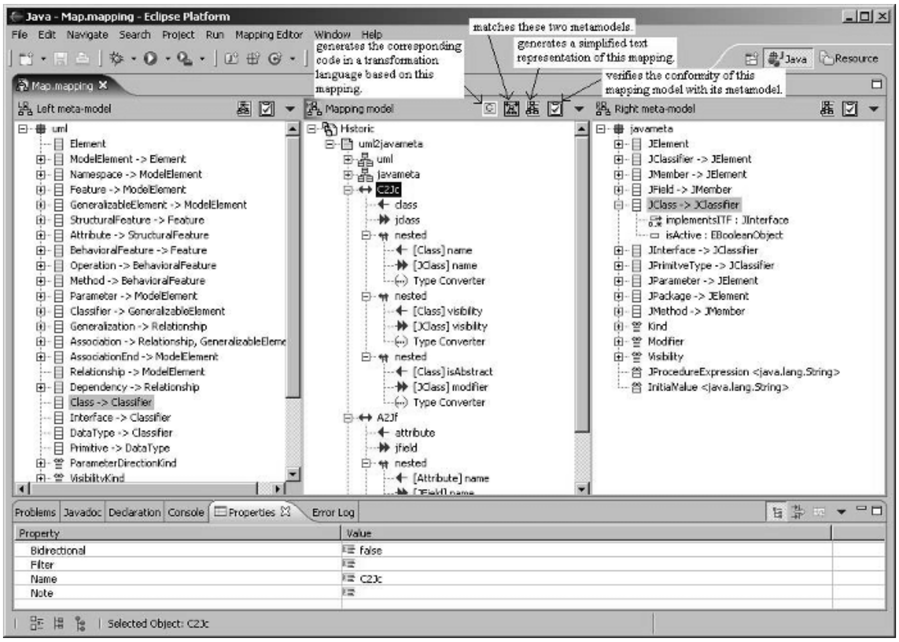
Figure 5 presents our first prototype to support *mapping specification*. This tool is denominated Mapping Modeling Tool (MMT). It was implemented as a plug-in for Eclipse that uses our proposed metamodel for mapping (see sections 3.1 and 3.3).

This tool presents a first metamodel on the left side, a mapping model in the center, and a second metamodel on the right. In this case, a fragment of the UML metamodel is mapped into a Java metamodel. In the bottom, the property editor of mapping model is shown. A developer can use this property editor to set the properties of a mapping model.

According to figure 5, `C2Jc` maps `UML Class` into `Java JClass`, mapping also the attributes such as `name` and `visibility` from `Class` into the corresponding attributes of `JClass`.

This tool simplifies the creation of *mapping specification* and provide some other features such as:

- verification of the conformity between a mapping model and our proposed metamodel for mapping.


Java - Map.mapping - Eclipse Platform
File Edit Navigate Search Project Run Mapping Editor Window Help
Java
Resource
Map.mapping
Left meta-model
Mapping model
Right meta-model
uml
Element
ModelElement -> Element
Namespace -> ModelElement
Feature -> ModelElement
GeneralizableElement -> ModelElement
StructuralFeature -> Feature
Attribute -> StructuralFeature
BehavioralFeature -> Feature
Operation -> BehavioralFeature
Method -> BehavioralFeature
Parameter -> ModelElement
Classifier -> GeneralizableElement
Generalization -> Relationship
Association -> Relationship, GeneralizableEleme
AssociationEnd -> ModelElement
Relationship -> ModelElement
Dependency -> Relationship
Class -> Classifier
Interface -> Classifier
DataType -> Classifier
Primitive -> DataType
ParameterDirectionKind
VisibilityKind
Historic
uml2javameta
uml
javameta
C2Jc
class
jclass
nested
[Class] name
[JClass] name
Type Converter
nested
[Class] visibility
[JClass] visibility
Type Converter
nested
[Class] isAbstract
[JClass] modifier
Type Converter
A2Jf
attribute
jfield
nested
[Attribute] name
[JField] name
javameta
JElement
JClassifier -> JElement
JMember -> JElement
JField -> JMember
JClass -> JClassifier
implementsITF : JInterface
isActive : EBooleanObject
JInterface -> JClassifier
JPrimitiveType -> JClassifier
JParameter -> JElement
JPackage -> JElement
JMethod -> JMember
Kind
Modifier
Visibility
JProcedureExpression <java.lang.String>
InitialValue <java.lang.String>
Problems Javadoc Declaration Console Properties Error Log

generates the corresponding code in a transformation language based on this mapping.

matches these two metamodels.

generates a simplified text representation of this mapping.

verifies the conformity of this mapping model with its metamodel.

Property	Value
Bidirectional	false
Filter	
Name	C2Jc
Note	

Selected Object: C2Jc

Fig. 5. Mapping Modeling Tool (MMT)

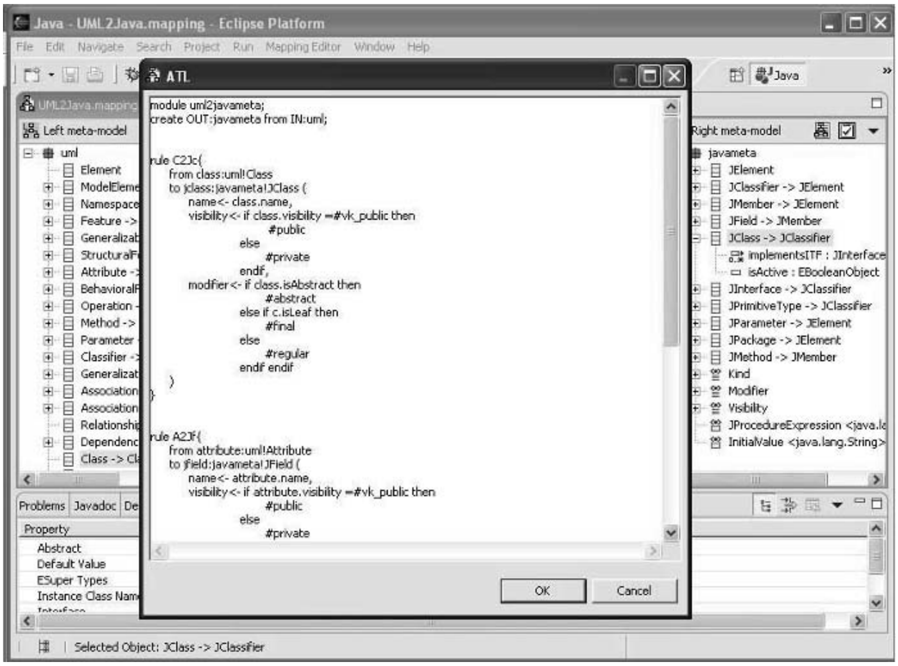
- transformation of mappings and metamodels into a simplified text representation which makes them more understandable.
- transformation of mappings specification into *transformation definition*.

This tool can export a mapping model as *transformation definition*. For the moment, we have implemented a generator for ATL[14], but we envisage creating generators to other model transformation languages such as YATL [15], in order to evaluate the power of our proposed metamodel for mapping. The resulting code in ATL of the mapping between UML (fragment) and this Java metamodel is presented in figure 6. This figure shows the ATL code fragment as `module uml2java;...`, the rules `C2Jc` and `A2Jf`. The last rule transforms an `UML Attribute` into `Java JField`.

The code in ATL is generated based on the mapping model. This tool leave the developer free to think only in the mapping between two metamodels, helping him to specify how the metamodels can be inter-related. Afterwards, it can generate the transform definition.

5 Conclusion

In this paper, we have presented our approach to determine mappings (i.e. correspondences) driven by models. A tool for mapping was also provided. However, the *schema matching* [18], i.e. finding correspondences between models, was not sufficiently covered here, because in the current stage of our research, we concentrated


Java - UML2Java.mapping - Eclipse Platform
File Edit Navigate Search Project Run Mapping Editor Window Help
ATL
Left meta-model
Right meta-model
OK
Cancel
Selected Object: JClass -> JClassifier

```
module uml2javameta;
create OUT:javameta from IN:uml;


rule C2Jc{
    from class:uml!Class
    to jclass:javameta!JClass (
        name<- class.name,
        visibility<- if class.visibility =#vk_public then
                        #public
                    else
                        #private
                    endif,
        modifier<- if class.isAbstract then
                        #abstract
                    else if c.isLeaf then
                        #final
                    else
                        #regular
                    endif endif
    )
}


rule A2Jf{
    from attribute:uml!Attribute
    to jfield:javameta!JField (
        name<- attribute.name,
        visibility<- if attribute.visibility =#vk_public then
                        #public
                    else
                        #private
```

Fig. 6. The generated ATL code using the plug-in for mapping

our efforts on taking into account mapping driven by models, defining a metamodel and tool for mapping.

If transformation is the heart of MDA, and to transform a PIM into a PSM, it is necessary to find correspondences between metamodels, then *mapping specification* is also another important issue within MDA context.

MDA simplifies the development of software, but this is not obtained without effort. In fact, MDA allows developers to develop, maintain and evolve their systems encapsulating the complexity in models, model transformation, mapping, and so on. Thus, what a developer must know in order to develop software artifacts with MDA is only the tip of the iceberg, i.e. models.

In future research, we will apply this plug-in for Eclipse (i.e. MMT) to build mappings between the UML metamodel and Web Service metamodel. We also aim to develop more fully the *schema matching* in order to integrate it also into our plug-in.

Acknowledgments

The work described in this paper was partly financed by “Conseil Général de Maine-et-Loire”, through a fellowship provided to Denivaldo Lopes.

References

1. OMG (2001) Model Driven Architecture (MDA) - document number ormsc/2001-07-01.
2. OMG (2002) Request for Proposal: MOF 2.0 Query/Views/Transformations RFP. Available at http://www.omg.org/docs/ad/02-04-10.pdf.
3. Cook, S. (2004) Domain-Specific Modeling and Model Driven Architecture. MDA Journal 1-10
4. Kent, S., Smith, R. (2003) The Bidirectional Mapping Problem. Electronic Notes in Theoretical Computer Sciences 82
5. Eclipse Project (2004) Eclipse. Available at http://www.eclipse.org.
6. Middleware Company (2003) Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach. Technical report, The Middleware Company. Available at www.middleware-company.com/casestudy.
7. Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F. (2004) Applying MDA Approach for Web Service Platform. 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004)
8. Interactive Objects Software GmbH and Project Technology, Inc. (2004) 2nd Revised Submission to MOF Query / View / Transformation RFP.
9. QVT-Merge Group (2004) Revised submission for MOF 2.0 Query/Views/Transformations RFP (ad/2002-04-10).
10. Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T.J. (2003) Eclipse Modeling Framework: A Developer's Guide. 1st edn. Addison-Wesley Pub Co
11. GMT Generative Model Transformer Project (2004). Available at http://dev.eclipse.org/-viewcvs/indextech.cgi/checkout/gmthome/description.html.
12. Java Community Process (2002) Java^{TM}Metadata Interface(JMI) Specification, Version 1.0.
13. netBeans.org (2004) MetaData Repository (MDR) Available at http://mdr.netbeans.org.
14. Bézivin, J., Dupé, G., Jouault, F., Pitette, G., Rougui, J.E. (2003) First Experiments with the ATL Model Transformation Language: Transforming XSLT into XQuery. 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture
15. Patrascoiu, O. (2004) Mapping EDOC to Web Services using YATL. 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004)
16. Hausmann, J. H. and Kent, S. (2003) Visualizing Model Mappings in UML. Proceedings ACM 2003 Symposium on Software Visualization (SOFTVIS 2003) 169-178
17. Pottinger, R.A., Bernstein, P.A. (2003) Merging Models Based on Given Correspondences. Proceedings of the 29th VLDB Conference
18. Rahm, E., Bernstein, P.A.(2001) A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10 334-350
19. Madhavan, J., Bernstein, P.A., Domingos, P., Halevy, A.Y. (2002) Representing and Reasoning about Mappings between Domain Models. Eighteenth National Conference on Artificial intelligence (AAAI'02) 80-86
20. Lacher, M.S., Groh, G. (2001) Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. 14th International FLAIRS conference 305-309
21. Agrawal, A., Levendovszky, T., Sprinkle, J., Shi, F., Karsai, G. (2002) Generative Programming via Graph Transformation in the Model-Driven Architecture. OOPSLA 2002 Workshop on Generative Techniques in the Context of Model Driven Architecture
22. Bernstein, P.A. (2003) Applying Model Management to Classical Meta Data Problems. Proceedings of the 2003 CIDR
23. Gamma, E., Beck, K. (2003) Contributing to Eclipse: Principles, Patterns, and Plug-Ins. 1st edn. Addison-Wesley Pub Co

Experiment in Model Driven Validation of BPEL Specifications

David H. Akehurst

University of Kent at Canterbury, Canterbury, Kent, CT2 7NF, United Kingdom, D.H.Akehurst@kent.ac.uk

Summary. The Business Process Execution Language (BPEL) is an XML based language; the BPEL standard defines the structure, tags and attributes of an XML document that corresponds to a valid BPEL specification. In addition, the standard defines a number of natural language constraints of which some can be ambiguous and are complex. This paper uses the Unified Modelling Language (UML) and Object Constraint Language to provide a model of the XML based BPEL language. Based on this model the paper shows how OCL can be used to give a precise version of the natural language constraints defined in the BPEL standard. We then use this precise specification to generate a Validation tool automatically that can check that a BPEL document is well-formed.

1 Introduction

The problems of interoperability can be viewed as having two aspects: one being the matching of concepts and ontologies; the other being the problem of matching different technologies. In fact, most interoperability problems straddle this division, requiring both conceptual and technological compatibility to be addressed at the same time. If we can simplify one of these two aspects, we free up the engineering process to focus more fully on the other.

BPEL or BPEL4WS (Business Process Execution Language for Web Services) [7] is a language for specifying the Business Process logic that defines a choreography of interactions between a number of Web Services [15]. It is a technology that can be used to address aspects of interoperability between components that offer their services as Web Services. Although the language provides technical means to specify choreography patterns, it does not directly provide support for understanding the conceptual information associated with the related services. Making the use of this language as simple as possible via provision of good support tools will enable an engineer to focus on the conceptual issues rather than focusing on the difficulties of the language itself.

The BPEL4WS language is an XML based language and the BPEL standard defines the structure, tags and attributes of an XML document that corresponds to a valid BPEL specification. In addition to the precise specification of tag names and attributes, the standard defines a number of constraints on the way in which the

XML elements should be put together. These constraints are given using natural language, which although being very descriptive is not always precise and some of the constraints are ambiguous. In addition some of the constraints are so complex that it is difficult to understand from the text what the constraint is actually saying; this leaves the possibility of creating what appears to be a valid XML BPEL document, which in actual fact violates one or more usage constraints.

This paper uses the Unified Modelling Language (UML) [13] and Object Constraint Language (OCL) [8] to provide a model of the XML based BPEL language. Based on this model the paper shows how OCL can be used to give a precise version of the natural language constraints defined in the BPEL standard.

The model and constraints are subsequently used to automatically generate a BPEL validation tool as a plug-in to IBM's eclipse IDE. The plug-in uses the Eclipse Modelling Framework (EMF) [4] code generation facilities along with the OCL code generation tools developed at the University of Kent [2].

Section 2 of this paper gives an explanation of the Model Driven approach to constructing the validator. Section 3 specifies (some of) the OCL constraints that correspond to natural language from the BPEL4WS standard. Section 4 discusses some of the more difficult issues involved in mapping the model and constraints to an implementation. Section 5 looks at some alternative approaches and the paper concludes in section 6.

2 Explaining the Approach

Model Driven Development (MDD) or Model Driven Architecture (MDA) [11] is an approach to software development in which the focus is on Models as the primary artefacts. Model Transformations are considered the primary operation on models, used to map information from one model to another. One of the simplest instantiations of MDD is the idea of code generation from UML models, which can be viewed as a transformation from the UML model to a Code model.

Code generation is not new and has been around for far longer than the recent activity surrounding MDD. What we consider to be of interest with regards to the work in this paper is the combination of code generation from models and code

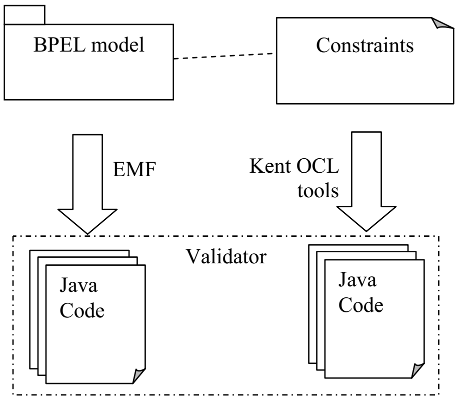


Figure 1. Code Generation

generation from constraints; with the resulting code facilitating the validation of the constraints against instances of the model. In particular, the model code and the constraint code are generated separately, see Figure 1.

The work carried out in this paper is (as implied by the title) an experiment in using a modelling and constraint based approach to specifying the BPEL4WS language and automatically generating a tool that aids a user in writing valid BPEL specifications. This is different from the work described in [6] where UML is used as a language for writing BPEL specifications.

A set of UML Class Diagrams are created to model of the structural aspects of the BPEL language. We then use OCL as a mechanism for formally specifying constraints on instances of the model; these constraints on the model correspond to constraints on how elements of the BPEL language can be put together. A model constraint that fails would indicate an invalid combination of BPEL constructs.

OCL is a text based constraint language that has been officially part of the UML standard in recent versions. The language is based primarily on Set theory concepts and can be written using the ASCII character set. Although, defined initially as a “constraint” language the core expression part of OCL can also be used as an object-based Query Language.

The BPEL language is defined as an XML document, however, in order to write constraints using OCL we form a UML model that represents the structure of the BPEL XML document. There is not room in this paper to give the specification of this model but it can be found in the technical report [1]. The model of the BPEL language defined is in accordance with the version 1.1 of the BPEL standard.

3 The Constraints

The BPEL4WS standard contains approximately 20 natural language constraints that we have mapped into OCL constraints on the BPEL metamodel. The full details of this work can be found in the technical report [1], in this paper we have room to show only a few. We start by showing a simple constraint, then move on to a couple of more complex (and interesting) ones.

Each of the following subsections gives an extract from the BPEL standard, defining a natural language constraint on the use of the language, which is then expressed using OCL to give a more precise specification as a constraint on the UML model of BPEL. The later constraints of greater complexity require the definition of additional model properties; OCL, through the use of the ‘def’ context, enables us to give these definitions, which can subsequently be used to specify the required OCL expressions.

3.1 Partner definitions must not overlap

The first constraint we look at is quite a simple constraint to define, however it does highlight an interesting issue regarding the relationship between the structural model of the language, OCL constraints on that model, and the implementation policy.

"7.3 Business Partners

... Partner definitions MUST NOT overlap, that is, a partner link MUST NOT appear in more than one partner definition."

This constraint is a restriction on the connections between a 'partner' construct and a 'partnerLink' construct. These two constructs are represented in BPEL as sub elements of the top level 'BusinessProcess' element. E.g. the following two XML segments show a valid and an invalid process specification:

```
<process                        <process
  name="Invalid" ...              name="Valid" ...
 <partner                        <partner
   name="SellerShipper"            name="SellerShipper"
   xmlns="http:...">               xmlns="http:...">
   <partnerLink                    <partnerLink
     name="Seller"/>                 name="Seller"/>
   <partnerLink                    <partnerLink
     name="Shipper"/>                name="Shipper"/>
 </partner>                      </partner>
 <partner                        <partner
   name="Shipper">                 name="Shipper">
   <partnerLink                    <partnerLink
     name="Shipper"                  name="Shipper2"
 ...                             ...
</process>                      </process>
```

This constraint requires that the union of partnerLink objects from all partners in a process is a Set; i.e. each partnerLink in that union is unique.

We could attempt to enforce this constraint by the structure of the BPEL model, however; i.e. if we model the relationship between Partner and PartnerLink as an [0..1]-to-[0..*] association, this states that any one partnerLink can only be associated to a single partner and thus the above constraint would not be violated. However, if we consider the generation of a BPEL validator, this is where the implementation policy is important; the structure of the XML language happily allows the constraint to be broken, so we must look at the process of mapping the XML document into an implementation of the BPEL model.

A typical implementation of a [0..1]-to-[0..*] association would simply update the [0..1] end if an object were added to the [0..*] end (this is certainly what happens within the EMF implementation). So a naïve mapping from XML to model implementation of the partner and partnerLink constructs would not provide any notification that the constraint was broken, it would simply update the links between instances.

For a BPEL validator, it is essential that notification is given that the constraint is violated. So either, the mapping from XML to model instance should check that the link between partner and partnerLink has not already been set; or we can model the association as a [0..*]-to-[0..*] association and add an explicit OCL constraint to check that the required uniqueness properties are met.

As can be seen by the UML diagrams above, we have adopted the second approach and the necessary OCL constraint is given below.

```
context bpel::Process
  inv partnerDefinitionsMustNotOverlap :
    let x=self.partner.partnerLink in
      x->asSet()->asBag() = x->asBag()
```

As can be seen by this example, there is a balance to be made between constraining the BPEL language by the structural model and by constraining it using OCL constraints.

3.2 Link Control Cycles and Boundary Crossing

Some of the most complex constraints to check manually are those that define the use of Links within the Flow construct; i.e. those that check for boundary crossing conditions and control cycles. By using OCL to defining some additional (temporary) properties on the model we can construct OCL invariants that check the constraints.

Consider the following natural language constraints:

"... Every link declared within a flow activity MUST have exactly one activity within the flow as its source and exactly one activity within the flow as its target. The source and target of a link MAY be nested arbitrarily deeply within the (structured) activities that are directly nested within the flow, except for the boundary-crossing restrictions.

... In general, a link is said to cross the boundary of a syntactic construct if the source activity for the link is nested within the construct but the target activity is not, or vice versa, if the target activity for the link is nested within the construct but the source activity is not.

... A link MUST NOT cross the boundary of a while activity, a serializable scope, an event handler or a compensation handler (see 13. Scopes for the specification of event, fault and compensation handlers)."

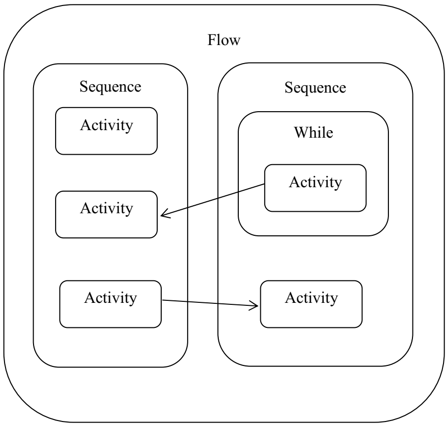


Figure 2. Links in a Flow Activity

We can illustrate a specification with Links that do and don't adhere to these constraints in Figure 2. The specifications are shown using a graphical notation rather than the XML syntax of BPEL as it is easier to see the links between activities in this manner. The arrows in the diagram represent links between activities. The top link is invalid as it crosses the boundary of a while activity; the lower link is valid.

To express this constraint in OCL, we require a property subActivities to be defined for each subtype of Activity. The property returns a set containing all activities directly nested within that Activity. Also required is a property allSubActivities which returns all nested and sub-nested activities. For basic Activities this set will typically be empty. These properties are defined for each Activity subtype in the technical report, overriding the property defined on their common supertype Activity as shown here:

```
context bpel::Activity
  def: subActivities : Set(bpel::Activity) = Set{}

  def: allSubActivities : Set(bpel::Activity) =
    self.subActivities
    ->union( self.subActivities.allSubActivities->asSet() )
```

The constraint requiring the source and target activity for each link of a flow to be contained within the flow can be expressed as follows:

```
context bpel_11::Flow
  inv sourceAndTargetActivitiesAreContainedWithinTheFlow :
    self.link->forAll( lnk |
      self.allSubActivities->includes(lnk.source.activity)
        and
      self.allSubActivities->includes(lnk.target.activity) )
```

We can subsequently test for boundary crossing violations on While, Scope and EventHandler constructs as follows:

```
context bpel::While
  inv boundryCrossing :
    self.allSubActivities->includesAll(
             self.allSubActivities.sourceOf.activity )
     and
    allSubActivities->includesAll(
             self.allSubActivities.targetOf.activity )

context bpel::Scope
  inv boundryCrossing :
    not self.variableAccessSerializable.oclIsUndefined()
      implies
   (self.variableAccessSerializable
      implies
    self.allSubActivities->includesAll(
                self.allSubActivities.sourceOf.activity )
      and
    self.allSubActivities->includesAll(
                self.allSubActivities.targetOf.activity ) )

context bpel::EventHandler
  inv boundryCrossing :
```

```
  self.activity.allSubActivities->includesAll(
        self.activity.allSubActivities().sourceOf.activity )
    and
  self.activity.allSubActivities->includesAll(
        self.activity.allSubActivities().targetOf.activity )
```

Lastly, we check that links do not create control cycles in accordance with this natural language constraint:

> *"... Finally, a link MUST NOT create a control cycle, that is, the source activity must not have the target activity as a logically preceding activity, where an activity A logically precedes an activity B if the initiation of B semantically requires the completion of A. Therefore, directed graphs created by links are always acyclic."*

This constraint requires us to construct expressions that enable navigation around a causality graph of the BPEL specification. To enable this we define properties that return the set of possible 'next' or 'previous' activities for any activity. Given the possibility to mix structured flow constructs (Switch, While, etc) with free form (Flow) constructs, it is not trivial to construct expressions that define 'next' and 'previous' properties.

If we define the property 'next', along with a transitive closure version 'allNext' that returns the set of all following activities, we can define a constraint that ensure links do not create control cycles as follows:

```
context bpel::Link
  inv noControlCycles :
    self.target.allNext->flatten()->excludes(self.source)
```

The complex part of this constraint is hidden in the expressions that form the 'next' properties, part of the definition for these properties is given below, the overriding versions for the other Activity subtypes can be found in the technical report:

```
context bpel::StructuredActivity
  def: next(prev:bpel::Activity) : Set(Activity) =
    Set{}

context bpel::Activity
  def: next : Set(bpel::Activity) =
   let
    x =  self.parent.next(self)
   in
    if x->isEmpty()
    then self.parent.next
    else x endif

  def: allNext : Sequence(Set(bpel::Activity)) =
    Sequence{Set{self}}->transitiveClosure(x|x.next->asSet())

context bpel::While
  def: next(prev:bpel::Activity) : Set(Activity) =
    self.activity.initialActivities
    ->union( self.parent.next )
```

Note here that we require a transitive closure operation in order to define the 'allNext' property. It would potentially be possible to manage without it; however the definition of ‘allNext’ would require the use of additional recursive operations and be more complex to define.

3.3 Correlations between Receive and Reply Activities

By far the most complex constraints described using natural Language are those that discuss the relationship between receive and reply activities that use the same correlation sets, operations and ports. A correlation set is a mechanism for identifying a particular instance of a remote statefull service. These constraints address the situation of receiving a message from just such a remote service and the subsequent reply to that service. They attempt to reduce the possibility of causing deadlocks and inconsistencies due to miss matching of receive-reply pairs.

The following is one of the constraints addressing this issue:

“... The correlation between a request and the corresponding reply is based on the constraint that more than one outstanding synchronous request from a specific partner link for a particular portType, operation and correlation set(s) MUST NOT be outstanding simultaneously. The semantics of a process in which this constraint is violated is undefined. ...”

To construct an OCL constraint to check that this situation does not occur, it is necessary to form an expression that first collects the sequence of activities that may possibly follow a receive activity; we can potentially use the previously define ‘next’ and ‘allNext’ properties to do this. This sequence of following activities must then be searched for the first matching reply activity, and finally an expression needs to formed to check that no other matching receive activity occurs in-between the first receive activity and first subsequent matching reply activity. The following OCL invariant forms the required constraint:

```
context bpel_11::Receive
  inv noSimultaneousOutstandingSynchronousRequests  :
    let
      request = self,
      actionSeq = self.allNext->flatten(),
      replySeq = actionSeq->select( a |
                     a.oclIsTypeOf(bpel_11::Reply)
                  ).oclAsType(Sequence(bpel_11::Reply)),
      reply = replySeq->select( r |
                  r.portType= request.portType and
                  r.operation = request.operation and
                  r.correlation = request.correlation
              )->first(),
      replyIndex = actionSeq->indexOf(reply)
    in
      not actionSeq->subsequence(0,replyIndex)
          ->select(a|a.oclIsTypeOf(bpel_11::Receive))
          .oclAsType(Sequence(bpel_11::Receive))
          ->exists( rec |
           rec.portType= request.portType and
           rec.operation = request.operation and
           rec.correlation = request.correlation )
```

Unfortunately this constraint does not fully perform an accurate check. The semantics of the Link construct cause execution to block at the target of a link until the action at the source of a link has completed. The defined property ‘next’ on the Flow activity does not take into consideration the full link-semantics of the language; thus causing the constraint to falsely fail under conditions that are in actual fact not a problem. To provide a fully accurate check the definition of the ‘next’ property would need to take into consideration the blocking semantics of links between activities that are nested within a Flow activity. This would amount to providing a true flow analysis of the activity structure and although it may be possible to do this using OCL, the complexity of the expressions comes close to constructing a model checking algorithm, and we consider this to be outside the scope of appropriate use of OCL.

4 Generating a BPEL Validator

It is all very well to draw a few diagrams containing boxes, lines and snippets of text and then write constraints in a funny kind of ASCII compatible set theory – but what use is it?

Firstly, it does allow us to define unambiguously and precisely the usage constraints of the language. Secondly, and even more usefully, we can automatically generate a validation tool that will check that the constraints have been adhered to, for any particular language expression (in this case, any particular BPEL document).

Using one of the UML (or MOF or ECORE) model to code automatic generation tools, such as the Eclipse Modelling Framework (EMF), we can quickly and easily generate a tool that supports a model based representation of a BPEL specification. At the University of Kent we have developed an OCL library that operates in conjunction with EMF repositories to facilitate both the evaluation of OCL constraints and the generation of code that corresponds to an OCL constraint.

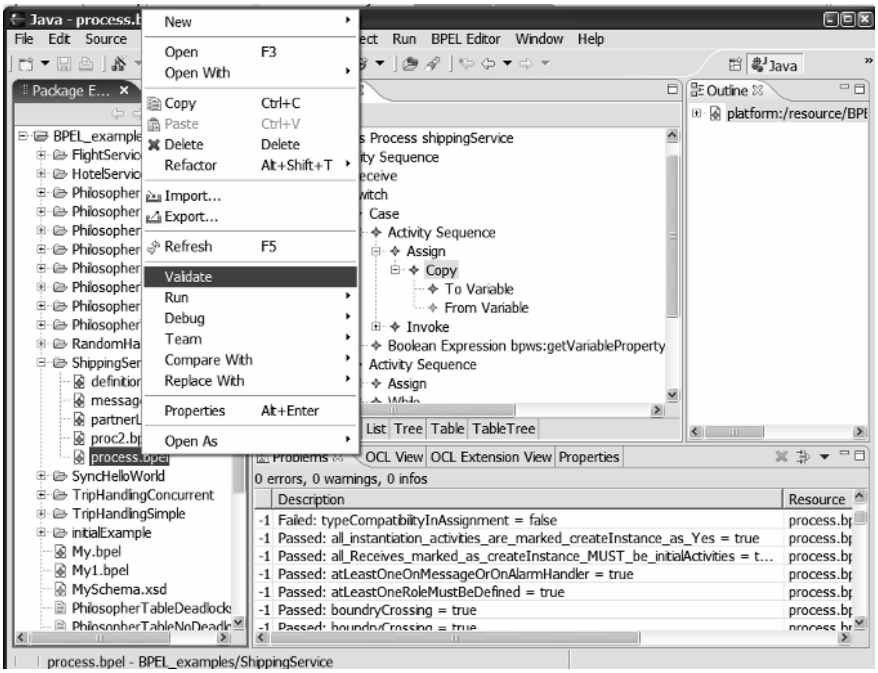
Thus, using these two tools (EMF and Kent OCL) the core code for a BPEL validator can be generated. The code can then be wrapped up to give the required user interface; for example we have wrapped up the validator as an eclipse plug-in, see Figure 3.

There are two mechanisms provided by the Kent OCL library for evaluating OCL expressions. There is an interpreter, that directly evaluates an expression and there is a code generation interface that converts the OCL expression into equivalent Java code. For the BPEL validator we use the Java code generation interface, as this allows the validator to be used independently from the OCL library. Constructing the BPEL validator has helped to develop the code generation facility of the Kent OCL library.

There are several issues worth drawing out regarding our experiences of mapping OCL expressions to Java code, in addition to the highlighting of inconsistencies in the specification of the OCL standard; below we discuss a couple of the more interesting OCL-to-Java issues.

The first issue we consider worth mentioning is the mapping of Boolean expressions between OCL and Java. The Java language uses true two-value

Boolean logic, whereas the Boolean logic of OCL is actually based on three-value logic; i.e. true, false, and undefined. Providing a mapping from the three-value OCL to the two-value Java is thus not a simple process; in addition, Java expressions that would typically cause an exception (e.g. a NullPointerException), in OCL simply evaluate to the undefined value. Consequently, the Java code for an OCL Boolean expression must first catch exceptions for each sub-part of the expression, and then use a structure of conditional statements to convert the two-value semantics of Java into the correct interpretation of the OCL three-value semantics.



0 errors, 0 warnings, 0 infos

	Description	Resource
-1	Failed: typeCompatibilityInAssignment = false	process.bp
-1	Passed: all_instantiation_activities_are_marked_createInstance_as_Yes = true	process.bp
-1	Passed: all_Receives_marked_as_createInstance_MUST_be_initialActivities = t...	process.bp
-1	Passed: atLeastOneOnMessageOrOnAlarmHandler = true	process.bp
-1	Passed: atLeastOneRoleMustBeDefined = true	process.bp
-1	Passed: boundryCrossing = true	process.bp
-1	Passed: boundryCrossing = true	process.bp

Figure 3. BPEL Validator as an eclipse plug-in

The second issue we illustrate here is that of providing support for the 'def' context of OCL. This part of the OCL language is used to define additional properties and operations on Classes and Types defined by the structural part of a UML model. We have seen the use of this in the constraints specified in the previous section.

The difficulty here is that the Java code for the model types is generated by the EMF toolkit; however we wish to add additional properties and operations as a separate process. One option is to alter the code generated by EMF, however this is not ideal as it would require parsing and understanding the generated code; additionally in a more general scenario we might not have access to the source code of the model types. Ideally we require a mechanism that can be used to 'semantically' add operations and properties (from the perspective of the OCL code) without touching the implementation of the model types.

Our solution to this has been to generate static methods that represent the ‘defined’ properties and operations. When generating code for an OCL expression that calls one of these ‘defined’ features rather than calling a method on the source object, the source object is passed as a parameter to the static method. To facilitate the overriding semantics, the static method itself is not called directly, but called via a special method that looks for a parameter match on the closest runtime type of the source object.

5 Alternative Approaches

The technique for validating BPEL specifications presented here is one possible approach; two other potential candidates are to program the constraints directly or to use an XML constraint language. The first of these options can be easily dismissed; it is necessary to more formally specify the constraints in the BPEL standard in order to ensure that they are not misinterpreted; if we use a more formal language to define the constraints it is clearly more efficent and less error prone to map the formal specifications to code automatically, rather than perform the task by hand. In addition, due to the complexity of some of the constraints, it has been hard to specify them using OCL; seeing as the specifications in OCL are at a higher level of abstraction than program code, it can be assumed that it would be even harder to try and write the constraints directly in code; in fact the initial ideas for this work arose from conversations with engineers who had attempted (and struggled) to implement the constraints directly in code.

The second option, to use an XML constraint language is a quite feasible alternative. There are a number of XML constraint languages under development [3, 5, 6, 9, 10] although none have yet been standardised. The draw back of this approach would also be the level of abstraction; the XML based view of BPEL is at a lower level of abstraction than its equivalent UML model and hence it is likely than the specification of the complex constraints would be harder in XML context (consider the relationship between XSLT [14] and the requirement for a QVT language [12]). A useful item of future work would be to carry out some comparisons between OCL and XML versions of the same constraints.

6 Conclusion

In this paper we have shown the use of OCL and UML to provide a precise version of natural language constraints on the structuring of BPEL XML documents. From these precise specifications of the constraints we can automatically build a validator to check that the constraints have been met for any example BPEL document. This is particularly useful in the case of BPEL as some of the natural language constraints are ambiguous or complex to understand.

We have discussed in this paper a number of consecutively more complex forms of constraint: those that can be formed directly from OCL expressions; those that require the addition of extra properties; and those that require complex algorithms.

In particular, the constraint discussed in section 3.1 highlights an interesting requirement to balance constraints impose indirectly by the structure of a model, explicit constraints added using OCL and the implementation strategy involved. The later constraint of section 3.3 illustrates a limit to what we consider appropriate use of OCL and the constraints in section 3.2 illustrate the requirement for a transitive closure operation to be added to OCL.

Finally the paper has discussed some of the issues regarding the automatic generation of code, in this case a BPEL validator, using OCL constraints as the source.

References

1. Akehurst D. H., "Validating BPEL Specifications Using OCL," University of Kent at Canterbury, technical report: 15-04, August 2004.
2. Akehurst D. H. and Patrascoiu O., "OCL 2.0 – Implementing the Standard for Multiple Metamodels," in proceedings UML 2003 Workshop, OCL 2.0 - Industry standard or scientific playground?, San Francisco, USA, October 2003.
3. Buneman P., Fan W., Simeon J., and Weinstein S., "Constraints for Semi-structured Data and XML," SIGMOD Record (ACM Special Interest Group on Mangement of Data), vol. 30, pp. 47-54, 2001.
4. Eclipse.org, "Eclipse Modelling Framework (EMF)," www.eclipse.org/emf
5. Fan W. and Simeon J., "Integrity Constraints for XML," Journal of Computer and System Sciences (JCSS), vol. 66, pp. 254-291, February 2003.
6. Hu J., "Visual Modeling of XML Constraints Based on A New Extensible Constraint Markup Language," thesis, Department of School of Computer Science and Information Systems, Pace University, 2003
7. IBM, "Business Process Execution Language for Web Services," 2003, www.ibm.com/developerworks/library/ws-bpel/
8. Kleppe A. and Warmer J., "The Object Constraint Language and its application in the UML metamodel," in proceedings <<UML>>'98 Beyond the Notation, Mullhouse, France, June 1998.
9. Meißner E., "XML-Constraints with Scheme," in proceedings XML Europe 99, Granada, May 1999.
10. Nentwich C., Capra L., Emmerich W., and Finkelstein A., "xlinkit: A Consistency Checking and Smart Link Generation Service," ACM Transactions on Internet Technology, vol. 2, pp. 151-185, May 2002.
11. OMG, "Model Driven Architecture (MDA)," Object Management Group, ormsc/2001-07-01, July 2001.
12. OMG, "Request for Proposal: MOF 2.0 Query / Views / Transformations RFP," Object Management Group, ad/2002-04-10, April 2002.
13. OMG, "The Unified Modeling Language Version 1.5," Object Management Group, formal/03-03-01, March 2003.
14. W3C, "XSL Transformations (XSLT) Version 1.0," J. Clark (eds), W3C Remomendation, REC-xslt-19991116, November 1999.
15. W3C, "Web Services Definition Language (WSDL) 1.1," 2001, http://www.w3.org/TR/wsdl

Integrating Business Processes with Peer-to-Peer Technology

Florian Kupsch and Dirk Werth

German Research Center for Artificial Intelligence (DFKI), Stuhlsatzen-hausweg 3, D-66123 Saarbrücken, Germany, {kupsch|werth}@iwi.uni-sb.de

Summary. Since file sharing tools as eDonkey or Grokster are one of the favourite applications for a large number of internet users, the peer-to-peer paradigm experiences a new boom, as it enables very robust, scalable and fault-tolerant architectures. Indeed, the functionality of current applications is limited to quite simple operations such as searching and downloading music files or movies. Hence, to benefit economically from the characteristics of Peer-to-Peer technology, there is need of additional, innovative applications for business usage. A predestined scope of application is the context of Business Integration, where various distributed IT-systems act jointly by exchanging data records and the corresponding control flow. In this article, we present the concept of a Peer-to-Peer based integration architecture that solves the structural problems of traditional integration approaches. It allows a holistic integration of data, applications and business processes without the common insufficiencies of existing EAI solutions.

1 Introduction

Emerging the internet was an unpredictable development. All the same, important operating criteria such as reliability, local structures and robustness always came to the fore. As a result, users had the opportunity to get access to a highly available international data network that stays operative even in case of a breakdown of some single nodes.

One important aspect of globalization is concentration. As a logical consequence, international companies have to merge or to collaborate with each other to meet the requirements for a global distribution of their goods and services. In a networked economy, these enterprises are bound to unseal some parts of their IT-infrastructure to allow the engaged parties an exchange of product and accounting data as well as current status information in order to sustain the supply chain.

Furthermore, employees need improved business applications with enhanced functionalities to manage their all-day work. These applications replenish existing legacy systems by degrees. Long-ranging, these enhancements result in a heterogeneous network of computers and applications, as all those components require each other and have to be consolidated.

In the context of integrating distributed business application systems, the vision of redundancy and robustness was not resumed consequently: A changing business environment results in new technical developments and increasing demand of IT supported execution of business processes.

A first naive solution to provide an interaction of different systems is shown in Figure 1. Apparently, an appropriate description for this point-to-point integration of single applications and platforms is described with the term *Spaghetti Integration.* This (n:n)-connection of resources is not able to be maintained properly, as it contains too many non-standardized interfaces. In the last resort, the number of interfaces that have to be implemented will arise quadratic.

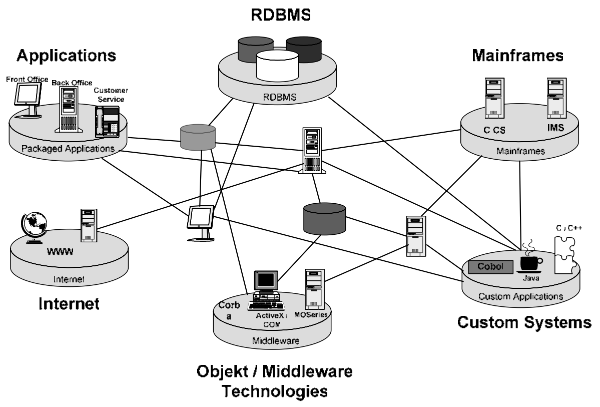


Figure 1. Heterogeneous IT-Infrastructure

After having widely experienced those scenarios, most enterprises recognized that there has to be found another solution for a reliable integration of business applications.

Some years later, the efforts of bringing together single systems to one logical unit are paraphrased with the term *Enterprise Application Integration (EAI)* [1]. EAI means a bold venture, as the interaction of business applications is restricted due to several constraints:

- *different communication protocols* and interfaces,
- *syntactical differences* between data of the single applications
- *proprietary semantic* of particular system messages, and finally
- Business modelling methodology differs from the technical implementation. There is an essential difference between an integration at technical level and business process integration

However, the market for specific integration software is estimated to grow up to 30 billion US$ in 2004 [2].

In chapter 2, we mention why the existing state-of-the-art solutions for business integration have tremendous disadvantages. We will also propose an advanced Peer-to-Peer (P2P) integration architecture.

Certain applications based on P2P technology are already in use for non-commercial purposes. A prominent example is the exchange of music, movies and software via file-sharing networks where single peers communicate directly without detouring via any central servers. In the specific field of business integration, most development activities focus on 'old-school' solutions that rely on client/server architectures. All the same, some crucial advantages of P2P technology (performance, resilience, load-balancing, etc.) are well-known to many software architects. Chapter 3 gives an overview about existing P2P applications.

A widely used standard for exchanging data between enterprises is Electronic Data Interchange (EDI) and its sectoral implementations such as EDIFACT. Hence, these standards determine a relationship where only two parties are involved. As soon as an inter-organizational process stretches across more than two companies, conventional EDI operations will fail. With the appearance of *Web-Services*, a new paradigm was born. Now, it is technically possible to encapsulate functionality within services that are made available via internet. The interaction of these services is syntactically supported by the XML standard. Unfortunately, common Web-Services are stateless and therefore do not include any functionality that would be capable to control complex business processes. Furthermore, one needs a central Directory for the Universal Description, Discovery and Integration of Web Services (UDDI) where a catalogue of all offered services is stored. However, there are ambitions to enable Event-Driven architectures using Web Services [3].

At this time, P2P technology comes into play: Why should it not be possible to accomplish a P2P-based integration architecture that is not dependent on any central server and that allows a standardized exchange of both, business process data and control information?

The advantages would be evident: The abandonment of EAI-Servers that are hard to maintain is as fascinating as the easy customization of the single peers and the dynamic aggregation of new peers during run-time. Chapter 4 will introduce our idea of a P2P integration environment.

2 Business Integration and EAI

As already mentioned in the first chapter, an integration of the core IT-systems is essential to guarantee a frictionless handling of business processes. In an optimal scenario, the result would be a seamless IT infrastructure that is completely transparent for all participants: It appears as one single system, supporting a service-oriented interaction of all relevant business processes.

So far, the efforts to manage this integration are pooled with the keyword Enterprise Application Integration (EAI). They contain a set of technologies and concepts such as Middleware, ETL-Tools and EAI-Software that focus on a central planning and control of business application data in real-time. In Figure 2, an example for embedding an EAI Server into the enterprise architecture is shown:

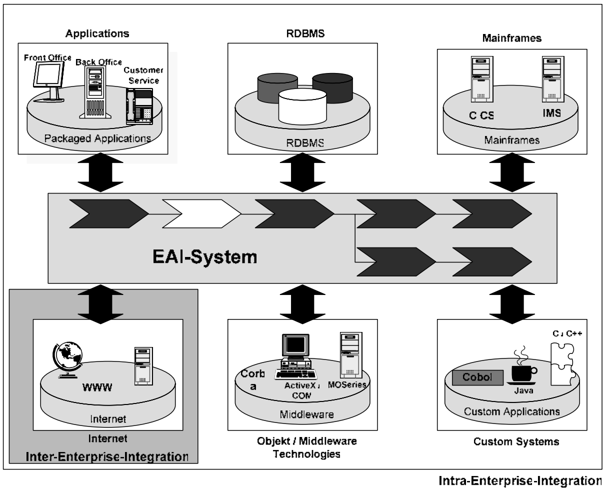


Figure 2. EAI Scenario

In this scenario, a central EAI-Server manages the coordination of both, control and data flow, between the attached systems. In this manner, the number of required interfaces can be reduced, as there exists only one bidirectional connection from each system to the server.

Because of the central component (which is ordinarily called *Information Hub*), these approaches cause several problems that complicate a reliable integration of numerous systems. Major well-reported problems are the following:

- **Single point of failure**: The Information Hub is the central node between the different applications. In case of a breakdown, all business processes are affected, possibly even inoperable.
- **Bottleneck**: All network traffic has to be forwarded through the Information Hub. This results in an extremely high load of data that has to be handled. In a situation of peak load, the system performance will crash down dramatically.
- **Configuration Icebergs**: The EAI application must contain all relevant business transactions: The distribution of information has to be represented by formal rules for transformation and routing. Because of its complexity, interdependencies and lots of exceptions, the number of configuration rules increases exponentially. This may cause insufficient and fault-prone integration solutions.

We consider EAI systems as mission critical applications whose blackout is associated with a substantial business risk. In recent years, awareness of cost-intensive administration and insufficient management of complex business application systems by centralistic approaches has arisen. New fields of research such as *Autonomic Computing* gave thought-provoking impulses to find better alternatives for an efficient management of business integration [4].

3 Recent P2P-Applications

As denoted at the end of the last chapter, an important field of research consists in *Autonomic Computing*. However, the underlying principles are not new, but became popular by the introduction of file sharing systems, and thus have found their breakthrough.

P2P means a networked structure where the participants (herein called *peers*) interact and share resources directly and equitable, i.e. "sharing of computer resources and services by direct exchange between systems". In contrast to client/server architectures, P2P networks do not include any kind of hierarchical structure. In the following, the term peer describes a participant within a P2P network.

In principle, single peers are independent from certain hardware platforms. This comprises a wide range of supposable systems, ranging from PDAs via desktop computers up to mainframes [5]. They are all characterized by the following properties [6]:

- **Client and server functionality:** Every peer is able to receive data from other peers as well as to provide data for others.
- **Direct exchange:** There is no central coordinating instance controlling the communication between the peers.
- **Autonomy:** It is in charge of each single peer at which time it provides which service, data or output to the network.

Especially the property of autonomy is of special importance for the integrated use of mobile computers such as notebooks, PDAs or mobile smart phones, as these per se can not be available to the network permanently.

Nowadays, P2P technology is used for non-commercial, private or academic purposes. Well-known examples are file-sharing applications (e.g. Napster, eDonkey, Kazaa) or grid computing projects (e.g. seti@home).

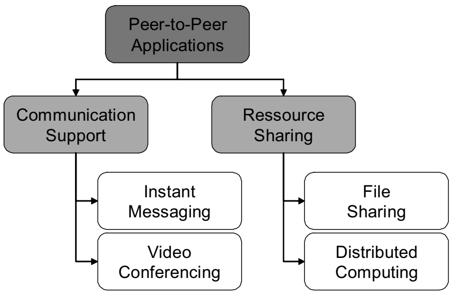


Figure 3. Typical P2P applications

Current applications that are usable within business integration scenarios are seldom [7]. In particular, existing approaches can be categorized in two main clusters as shown in Figure 3.

Communication support is the most common kind of applications for P2P networks within enterprises. The best known representatives are Microsoft's Netmeeting (for video conferencing) as well as ICQ, AOL Instant Messenger or the Microsoft Instant Messanger for sending and receiving instant messages. These applications are characterized by an open and highly dynamic number of participants.

Instant messaging applications are used for the direct, real-time communication between peers. Thus, they allow the determination who of the known participants is online at a particular time. This accelerates business communication processes, as it ensures a synchronous, text-based communication that enables a direct reaction on incoming messages and accordingly business events. For example, the retailer *Land's End* uses instant messages for after-sales service and support. The probability of selling goods was enhanced by 67%, compared to customers not being supported by this instrument [8].

Video conferencing means the real-time transmission of video and audio data between two or more participants. It mainly serves as an appliance to support cooperation and collaboration processes. Thus, natural communication trough wide distances is possible. Currently, real meetings can be substituted by the use of videoconferencing. Simultaneously, time and traveling costs can be reduced significantly. Moreover, video conferencing systems are deployed for personal education purposes (classroom function) [9].

Resource sharing can benefit from the exponential development of computer performance in the last decades. In parallel, the price for computation power has decreased in at least the same dimension. Subsequently, the available capacity of today's clients is only rarely used, as most of the work load is assigned to server(s). P2P can profit from these idle resources and hereby achieve a drastic cost reduction, combined with other competitive advantages. The sharing of resources is implemented in two concepts:

File sharing provides shared access to any files that are stored locally. It consists of special, efficient mechanisms for searching as well as algorithms for a non-central storage [10].

Compared to central data storage applications, this kind of data access eliminates the single point of failure, transfers data to low-cost mass storages and levels off the peaks in network traffic. The main problem is to ensure data consistency within the network as well as a 24/7 availability. After all, only those applications that do not affect mission-critical cases found their way in the business context.

Distributed computing is used for complex business computing tasks such as product development, simulation, financial forecasting or data mining tasks. The main resource for these time-consuming processes is computation power. Distributed computing can substitute powerful single computer systems with a network of peers by decomposing (dividing) the problem in small sub-problems and spreading them within the network. Hereby in the ideal case, the computation power of host systems can be reached with a fractional amount of the costs being calculated originally. The costs even can be minimized by using idle time of the peers to calculate single computation tasks. Thus such computations can be even processed by small or medium-sized enterprises (SME) that usually do not have the capability to invest in powerful host systems [11]. Even world-class enterprises use distributed computing for this purpose: Intel introduced distributed computing in 1990 for the development of new microprocessors [12].

4 Our P2P Integration approach

Assigning the peer-to-peer paradigm to the context of business integration means that all systems and components of an enterprise work in a self-organizing manner. Thereby, administration and integration costs can be reduced. [13] A basis for this approach is the non-centralized architecture of a peer-to-peer system which can offer the following advantages:

- **Non-central topology**: The complete IT-infrastructure is defined by a variable number of flat (non-hierarchical) peers. Every peer offers (and receives) at least one service (e.g. generate an order, create an invoice, check consistency of data, dispose payment, etc.), where several peers may contain the same services.
- **Reliability**: There is no central component that may cause problems. If a peer breaks down, another peer with similar functionality can replace the broken peer. If a peer with unique services crashes, only those business process instances are affected that require that service.
- **Scalable performance**: The performance of the network can be enhanced nearly unlimited by appending additional peers. Already existing components do not have to be replaced.
- **Easy configuration**: It is no longer necessary to customize the whole EAI system by central transformation rules. Every peer only contains the business knowledge it requires to accomplish its functionality. The configuration of the complete architecture results from the sum of the configuration of the single peers.
- **Adaptive self-configuration**: By implementing intelligent search mechanisms, a peer can find the next service in the process chain by a broadcast into the network. If another peer is able to offer the desired service, it responds. From now on, this peer is part of the process chain and can accept tasks from other peers as well as delegating services to any peers.

While the advantages mentioned above are mainly of a technical nature, these features will not be sufficient to manage the complete field of business integration. There is also a high demand of adequate logical representations of business processes to provide an essential process-oriented view that focuses also economical aspects. The vision of both, distributed business processes that are associated with distributed IT-systems, allows an optimization of business processes as well as improving IT applications without interacting each other.

In our research project *Peer-to-Peer Enterprise Environment (P2E2)*, we develop an integration architecture that focuses typical business integration scenarios. As a proof of concept, we will also implement and evaluate a prototypic system. For achieving this ambitious goal, we collaborate with the working group for Databases and Information Systems of the Max-Planck-Institute in

Saarbruecken (Germany) and several partners of the software industry. In Figure 4, a P2P scenario is shown that gives a first impression of our intention:

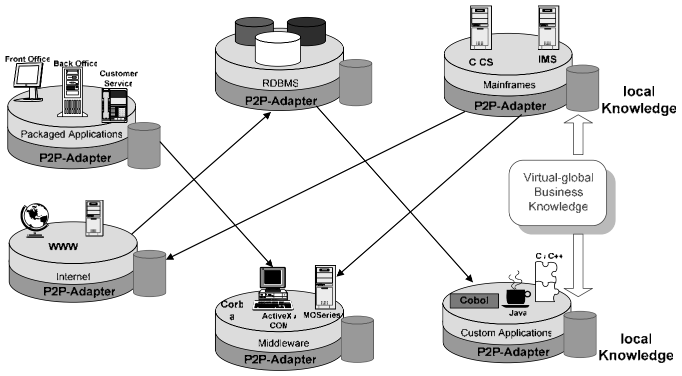


Figure 4. P2E2 integration scenario

Every application system (AS) that participates in the whole business process is encapsulated by a P2P-Adapter. Every Adapter enhances the functionality offered by the single components with additional Web Services (WS) that allow a composition of very complex services by a dynamic interaction of different adapters. A peer can initiate business processes, embed local processes in the application flow and even get embedded by other peers. It only has its own *local business knowledge*, but can also acquire *global business knowledge* by interacting with other peers. In this way, a comprehensive management of meta-data can be achieved without requiring centralistic client/server architectures.

To implement the P2E2 Adapter, a technical architecture was specified. It consists of four detailed interface specifications, whose interrelation is illustrated in Figure 5.

The *Configuration Interface* reads the configuration data provided by the P2E2 Configurator. The Business API (*BAPI*) is an automated interface for executable Business Process models that can be created with standard BPM tools such as the ARIS toolset by IDS Scheer AG. To find the following Service that provided the next step in a process chain, a *Search API* is implemented that offers efficient P2P-Search algorithms. The completed API consists in the monitoring and controlling component; it covers the whole BPM-Lifecycle and allows to specify Controlling queries to measure the performance of the integration environment.

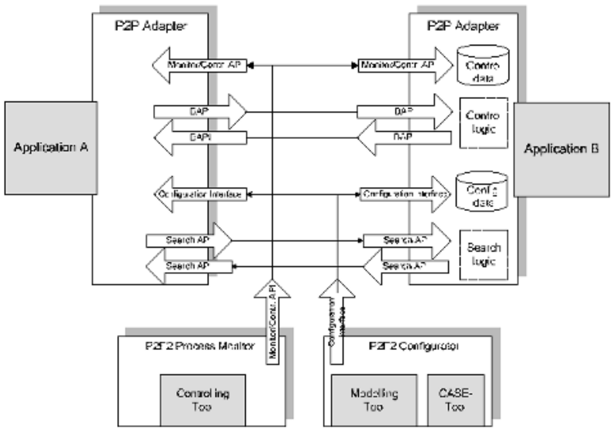


Figure 5. Architecture of a P2E2 Adapter

To ensure the efficient use of the procedure models and methods, we will conceptually and technologically develop an integrated support tool that includes an *Adapter Development Kit* as well as some important pre-customized standard adapters and an *Adapter Management Tool.*

The problem faced above is complex and versatile. Therefore, a highly structured and planned proceeding will be necessary. In contrast to other approaches, we follow a 'meet in the middle' strategy, analysing the problem space and creating solving concepts both from the business oriented as well as from the system/technology oriented direction. This ensures that the conceptual solutions are suitable for the business problems targeted and that they are realizable with today's state-of-the-art technologies.

The business oriented approach will evaluate the requirements of enterprises within internal or collaborative business integration scenarios. The main reference object here is the (abstract) business process that has to be supported. Thus, the business oriented conceptual solution has to provide mechanisms and techniques how to interconnect independent business processes using the P2P paradigm. The main challenge is the lack of a central coordination instance. As a logical consequence, appropriate business process negotiation techniques have to be developed.

Looking at the system oriented problems, the main question is how to find a mapping between heterogeneous application systems in conformity with the business processes and rules to be supported. This does not only mean to connect interfaces, but also requires to find *reasonable matches* within concrete contexts, as well as to handle a reliable control of the interaction.

Peer-to-peer technologies have been proven to be very flexible and robust. Hence, new methods and algorithms for a distributed interface and interaction management will be created using the P2P paradigm. Our idea is to enable an auto-configuration of the interaction between two independent application systems that succeeds to predefined business processes and that is also compliant to existing, constraining business rules.

Finally, only developing the two solving concepts described above is not sufficient, as there are strong interdependencies between business- and system-layer. The formation of business process chains within business integration use cases is always limited by the capabilities of the existing applications supporting the business processes. On the other hand, applications themselves can only be interconnected in a way that the combination of systems is realizing a predefined business process.

Hence, a relationship between the two partial solutions has to be found and specified. The combination of these three concepts will be a conceptual methodology for a dynamic binding of business processes to the behaviour of the distributed environment via a (semi)automatic reconfiguration in case of need.

The technological *P2E2* results will consist of a family of congeneric components (adapters) that can be linked to application systems. These components will provide a non-central IT-support for interactive business processes within distributed business integration environments. Interactive business processes are business processes that are structurally influenced by humans (user interaction). Therefore supporting interactive business processes have to provide operative user interfaces (front-end) whereas non-interactive ones only need a system-to-system connection (EDI). The adapters themselves will not have to rely on a central control unit. It mainly addresses those business cases where a central integration approach is not reasonable, not desired or not realizable due to organizational or technical restrictions. Moreover, *P2E2* extends the focus of classic EAI solutions by the aspect of user's interaction and influence. This also reflects the business level view where such a distinction is not reasonable. Expected properties of the *P2E2* implementation are robustness, fault tolerance, adaptability, dynamic behaviour and scalability.

For business usage, a coherent computing of business process is necessary, but not sufficient. For the purpose of analyzing and optimizing processes continuously, a detailed history of key performance indicators (KPI) is required. This data is in particular used on tactical, dispositive level to manage the performance of critical core processes. Within business integration scenarios with several systems involved, the gathering of these figures by measuring the processes is difficult and complex. *P2E2* will cope with this problem domain by integrating special measurement and control instruments within each peer. Thus, an integrated and consistent business process controlling and measurement within distributed environments will be enabled. Hence, *P2E2* will develop the measuring methods, implement them in the adapters as well as create a reporting and analyzing tool for an efficient process performance measurement.

5 Conclusions

The sustainable success of peer-to-peer technology within the non-commercial sector (file sharing and instant messaging) advises it also for the business sector. Within enterprises, there were several potential application cases for the productive use of P2P technology.

Especially for business integration scenarios, P2P seems to be an interesting alternative, in particular those with are characterized by non-hierarchical, flat organization structures. Thus, the integration of application systems using peer-to-peer technology is obvious.

Indeed, only integrating applications is not sufficient for efficient usage in business environments. In order to effectively and efficiently run a business, the respective business process is the central object. Therefore, an efficient support and integration of operative business processes is the primary objective.

P2E2 addresses this aim by creating a family of generic components that provide a general connectivity between business application systems and that act with client and server functionality. Hence, the classical separation between client and server will be resolved in favor of functional ad-hoc decisions. The coordination and control between the single adapters will not be managed by a central unit, but case-based negotiated between the adapters and afterwards transferred as needed. Thus, there is no functional dependency of the whole functionality network from only one edge.

Within a consortium of 6 partners, covering basic academic research, applied research, software industry and IT-consultancy, *P2E2* deals with that problem domain for a total of 24 months. During the project lifetime, several showcases realizing real business scenarios are built up and deployed in order to prove the practical use of the project results.

Finally, the use of P2P technologies within business integration scenarios marks a young area of research and development within the field of business information systems. It will remain an interesting object for actual research. *P2E2* has made a first step into this domain trying to solve some of the common problems, leaving some others open and maybe also rising some new ones. The way to consistent, operative environments that integrate enterprises and enterprise application trough peer-to-peer technology will be long, but with P2E2 we have entered the road.

References

1. Linthicum, D.: Enterprise Application Integration. Boston (2000)
2. Meta Group: E-Business und Enterprise Application Integration. (2001)
3. Smith, D.: Web Services Enable Service-Oriented and Event-Driven Architectures. In: Business Integration Journal, (May 2004) 12-14
4. Kephart, J.; Chess, D.: The vision of Autonomic Computing. IEEE Computer, (2003)
5. Intel Corporation: P2P Filesharing at work in the Enterprise. http://www.intel.com/deutsch/ebusiness/pdf/ wp011301_sum.pdf, online: 2004-01-21
6. Schoder; D., Fischbach, K.: Peer-to-Peer: Anwendungsbereiche und Herausforderungen. In: Schoder, D.; Fischbach, K.; Teichmann, R. (Ed.): Peer-to-peer: ökonomische, technische und juristische Perspektiven. (Springer) Berlin et al. (2002) 3-21
7. Saini, A.: Peer to Peer Distributed Business Processes. http://www.dmreview.com/whitepaper/WID463.pdf, online: 2004-06-10
8. Tischelle, G.; Swanson, S.: Not just Kid Stuff. http://www.informationweek.com/story/showArticle.jhtml?articleID=6506155, online: 2004-06-14

9. ViDe Video Conferencing Cookbook, www.videnet.gatech.edu/cookbook/ , online: 2004-04-02
10. Schoder, D.; Fischbach, K.: Peer-to-Peer : Anwendungsbereiche und Herausforderungen. In: Schoder, D.; Fischbach, K.; Teichmann, R. (Ed.).: Peer-to-peer: ökonomische, technische und juristische Perspektiven. (Springer) Berlin et al. (2002) 3-21, 6
11. Intel Corporation: Peer-to-Peer Computing in the Enterprise: Implications of IT and Business Decision Makers, http://www.intel.com/deutsch/ebusiness/pdf/wp020702_sum.pdf, online: 2003-02-12, 3
12. Intel Corporation: Peer-to-Peer: Rechenleistung auf breitester Basis. http://www.intel.com/deutsch/eBusiness/products/peertopeer/ar011102_p.htm, online: 2004-03-01
13. Leymann, F.: Web Services: Distributed Applications Without Limits. In: Weikum, G.; Schöning, H.; Rahm, E. (Ed.), Proceedings of the 10. Conference of Databases for Business, Technology und Web (BTW 2003). LNI 26, Gesellschaft für Informatik, (2003)
14. Zieger, A.: What Can P2P Apps Do for Enterprise Users. http://e-serv.ebizq.net /p2p/zieger_1a.html, online: 2003-04-17

Design and Implementation of a Peer-to-Peer Data Quality Broker

Diego Milano, Monica Scannapieco and Tiziana Catarci

Dipartimento di Informatica e Sistemistica, Universitá degli Studi di Roma "La Sapienza"
Via Salaria 113, Rome, Italy {milano,monscan,catarci}@dis.uniroma1.it

Summary. Data quality is becoming an increasingly important issue in environments characterized by extensive data replication. Among such environments, this paper focuses on Cooperative Information Systems (CISs), for which it is very important to declare and access quality of data. Specifically, we describe the detailed design and implementation of a peer-to-peer service for exchanging and improving data quality in CISs. Such a service allows to access data and related quality distributed in the CIS and improves quality of data by comparing different copies of the same data. Some experiments on real data will show the effectiveness of the service and the performance behavior.

1 Introduction

Data quality is a complex concept defined by various *dimensions* such as accuracy, currency, completeness, consistency [18]. Recent research has highlighted the importance of data quality issues in various contexts. In particular, in some specific environments characterized by extensive data replication high quality of data is a strict requirement. Among such environments, this paper focuses on Cooperative Information Systems.

Cooperative Information Systems (CISs) are all distributed and heterogeneous information systems that cooperate by sharing information, constraints, and goals [14]. Quality of data is a necessary requirement for a CIS. Indeed, a system in the CIS will not easily exchange data with another system without a knowledge on their quality, and cooperation becomes difficult without data exchanges. Also, when poor quality data are exchanged, there is a progressive deterioration of the quality of data stored in the whole CIS. Moreover, when a CIS is a data integration system, data integration itself cannot be performed if data quality problems are not fixed. As an example, results of queries executed over local sources must be reconciled and merged, and quality problems resulting from a comparison of results need to be solved in order to provide the data integration system with the required information [5]. On the other hand, the high degree of data replication that characterizes a CIS can be exploited for improving data quality, as different copies of the same data may be compared in order to detect quality problems and possibly solve them.

In [16, 11], the DaQuinCIS architecture [1] is described as an architecture managing data quality in cooperative contexts, in order to avoid the spread of low-quality data and to exploit data replication for the improvement of the overall quality of cooperative data .

The core component is the *Data Quality Broker*. The Data Quality Broker has two main functionalities: *(i) quality brokering*, that allows users to select data in the CIS according to their quality; *(ii) quality improvement*, that diffuses best quality copies of data in the CIS.

With reference to the quality brokering functionality, the Data Quality Broker is in essence a data integration system that allows the access to the best available quality data without having to know where such data are stored. The Data Quality Broker adopts a *wrapper-mediator* architecture, in which wrappers hide technological and model-related details of organizations, while a mediator interacts with the user, presenting a unified view of the databases on which queries can be posed.

With reference to the quality improvement functionality, when retrieving data, they can be compared and a best quality copy can be constructed. Organizations having provided low quality data are notified about higher quality data that are available in the CIS.

This paper will focus on the design and implementation features of the Data Quality Broker as a Peer-to-Peer (P2P) system. More specifically, the Data Quality Broker is implemented as a peer-to-peer distributed service: each organization hosts a copy of the Data Quality Broker that interacts with other copies and has both the functions of wrapper and mediator. While the functional specification of the Data Quality Broker is not a contribution of this paper, and has been presented in [11], its detailed design and implementation features as a P2P system are a novel contribution of this paper. Moreover, we will present some results from tests made to prove the effectiveness and efficiency of our system. The Data Quality Broker is implemented by a peer-to-peer architecture in order to be as less invasive as possible in introducing quality controls in a cooperative system. Indeed, cooperating organizations need to save their independency and autonomy requirements. Such requirements are well-guaranteed by the P2P paradigm which is able to support the cooperation without necessarily involving consistent re-engineering actions; in Section 5, we will better detail this point, comparing our choice with a system that instead does not adopt a P2P architecture.

The rest of this paper is organized as follows. In Section 2, an overview of both the component architecture and the system architecture of the Data Quality Broker is provided. In Section 3, the modules Query Processor and Transport Engine of the Data Quality Broker are respectively described in detail. The set of performed experiments is described in Section 4. Finally, related work and conclusions are presented in Sections 5 and 6.

[1] The DaQuinCIS approach has been developed in the context of the project "DaQuinCIS - Methodologies and Tools for Data Quality inside Cooperative Information Systems" (`http://www.dis.uniroma1.it/~dq/`).

2 The Data Quality Broker Architecture

In this Section, we provide an overview of the Data Quality Broker, first summarizing the Data Quality Broker functionality in Section 2.1 and then presenting the design and implementation details in Section 2.2.

2.1 The Data Quality Broker Functionality

In the DaQuinCIS architecture, all cooperating organizations export their application data and quality data (i.e., data quality dimension values evaluated for the application data) according to a specific data model. The model for exporting data and quality data is referred to as *Data and Data Quality (D^2Q) model* [11]. The Data Quality Broker allows users to access data in the CIS according to their quality. Specifically, the Data Quality Broker performs two tasks, namely *query processing* and *quality improvement.*

The Data Quality Broker performs *query processing* according to a global-as-view (GAV) approach by unfolding queries posed over a global schema, i.e., replacing each atom of the original query with the corresponding view on local data sources [17, 9]. Both the global schema and local schemas exported by cooperating organizations are expressed according to the D^2Q model. The specific way in which the mapping is defined stems from the idea of performing a quality improvement function during the query processing step. Concept from the global schema may be defined in terms of extensionally overlapping concepts at sources. When retrieving results, data coming from different sources can be compared and a best quality copy can be constructed. Specifically, in our setting, data sources have distinct copies of the same data with different quality levels, i.e., there are *instance-level conflicts*. We resolve these conflicts at query execution time by relying on quality values associated to data: when a set of different copies of the same data are returned, we look at the associated quality values, and we select the copy to return as a result on the basis of such values. More details on the algorithm implemented for processing queries can be found in [16]. The best quality copy is also diffused to other organizations in the CIS as a *quality improvement* feedback.

2.2 The Data Quality Broker Component Architecture

The Data Quality Broker is implemented as a peer-to-peer distributed service: each organization hosts a copy of the Data Quality Broker that interacts with other copies (see Figure 1, left side). Each copy of the Data Quality Broker is internally composed by four interacting modules (see Figure 1, right side). The modules Query Processor and Transport Engine are general and can be installed without modifications in each organization. We have implemented both the Query Processor and the Transport Engine; details on their implementation will be provided in the next sections. The module Wrapper has to be customized for the specific data storage system. The module Propose Manager must be implemented by each cooperating organization.

The *Query Processor* performs query processing, as detailed in Section 3. It unfolds queries posed over the global schema. On receiving query results, a query refolding phase is performed, in order to make the execution of the global query possible. A record matching activity is then performed in order to identify all copies of same data returned as results. Matched results are then ranked on the basis of associated quality and the best quality copies are selected to be returned as a result. The *Wrapper* translates the query from the language used by the broker to that of the specific data source. In this work, the wrapper allows read-only access to data and associated quality stored inside organizations. The *Transport Engine* is a communication facility that transfers queries and their results between the Query Processor module and data source wrappers. The *Propose Manager* receives feedbacks sent to organizations in order to improve their data. This module is customizable by each organization according to its internal policy for quality improvement. As an example, an organization can choose to trust the quality improvement feedbacks and automatically update its data based on the improvement notifications.

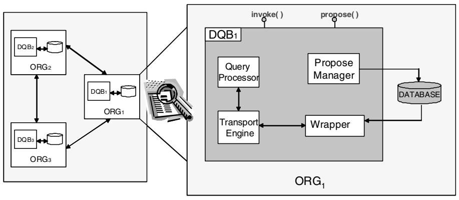


Fig. 1. The Data Quality Broker as a P2P system and its internal architecture

The Query Processor is responsible for query execution. Each copy of the Query Processor unfolds user queries into subqueries, on the basis of the defined mapping. Subqueries are then sent to other copies of the Query Processor for execution. In order to send subqueries and receive the answers, each Query Processor interacts with the local Transport Engine.

The Transport Engine provides general connectivity among all Data Quality Broker instances in the CIS. Copies of the Transport Engine interact with each other in two different scenarios: *(i)* during *query execution*, the Transport Engine of the requesting peer sends the subqueries to the Transport Engine of the target data sources by executing the `invoke()` operation (see Figure 2, right side), and asynchronously collects the answers; *(ii)* when a Query Processor has selected the best quality result of a query, it contacts through its local Transport Engine the Transport Engines of the interested sources, to enact *quality feedback* propagation. The `propose()` operation (see Figure 2, right side) is executed as a callback on each organization, with the best quality selected data as a parameter.

Another function performed by the Transport Engine is the evaluation of the *availability* of data sources that are going to be queried for data. This feature is encapsulated into the Transport Engine as it can be easily implemented exploiting Transport Engine's communication capabilities.

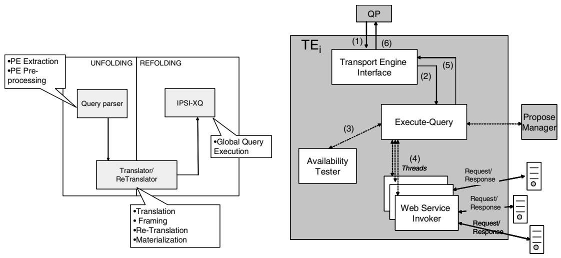
The Data Quality Broker component architecture has been implemented by web services technologies. To implement web services, we have chosen the J2EE 1.4 Java

Platform, specifically the Java API for XML-based Remote Procedure Call (JAX-RPC) [8]. In JAX-RPC, request/response of remote methods is performed through the exchange of SOAP messages over an HTTP connection.

3 Query Processor and Transport Engine: Design and Implementation Issues

The Query Processor module of the Data Quality Broker implements the *mediation* function of a data integration architecture [19]. It performs query processing according to a GAV approach, by unfolding queries posed over a global schema. Both the global schema and local schemas exported by cooperating organizations are expressed according to the D^2Q model. The D^2Q model is a semistructured model that enhances the semantics of the XML data model [7] in order to represent quality data. The schemas and instances of the D^2Q model are almost directly translated respectively into XML Schemas and XML documents. Such XML-based representations are then easily and intuitively queried with the XQuery language [4]. The unfolding of an XQuery query issued on the global schema can be performed on the basis of well-defined mappings with local sources. The exact definition of the mapping is beyond the scope of this paper, more details can be found in [13].

The Query Processor has been implemented as a Java application. Figure 2 shows the main components; the phases of query processing that are executed by each component module are also represented. The *Query Parser* performs the first query



(a) Query Engine (b) Transport Engine

Fig. 2. Implementation Modules of the Query Processor (left side) and Internal Modules of the TE of organization i (right side)

processing steps. To implement it, a parser for the XQuery language has been generated with the help of the JavaCC tools. The *Translation/Retranslator* module manages everything related to the translation and retranslation of queries and their results. For query execution a third-party query engine may be used. The engine used in our implementation is *IPSI-XQ* [1]. Let us note that we made IPSI-XQ quality-aware by adding some *quality functions* to it. These functions are written in XQuery, and allow to access quality data; they are simply added to the query prolog of each query submitted to the engine.

The Transport Engine component of the Data Quality Broker provides the connectivity and communication infrastructure of the DaQuinCIS system. In Figure 2 the internal components of the TE are shown; the sequence of interactions among such modules is also depicted. The *Availability Tester* module works in background continuously executing connectivity tests with servers from other organizations. It executes a ping function on the servers in the cooperative system opening HTTP connections on them. The *Transport Engine Interface* is the module that interfaces the Query Processor and the Transport Engine. Specifically, it uses a data structure to store queries and query results, once the latter have been gathered from each organization. The data structure is organized as an array: each element is representative of a single query execution plan and is composed by a list of queries that are specific of such a plan. Such queries are passed by the Query Processor (step 1). Then, the *Transport Engine Interface* activates the *Execute-Query* module with plans as input parameters (step 2). The *Execute-Query* interacts with the *Availability Tester* module that performs an availability check of the sources involved in the query execution (step 3). Then, the *Execute-Query* activates the *Web Service Invoker* module that carries out the calls to the involved organizations (step 4). The call is performed in an asynchronous way by means of suitable proxy SOAP client. Before invoking data management web services, an availability check is performed by the *Availability Tester* module. When the result of the different plans are sent back, the *Execute-Query* module stores them in a specific data structure and gives it to the *Transport Engine Interface* (step 5) that, in turn, gives it back to the Query Processor (step 6). The data structure is very similar to the input one; the main difference is the substitution of the query field with a special record containing data and associated quality provided as query answers.

Notice that the same interaction among modules shown in Figure 2 occurs when quality feedbacks need to be propagated. The Query Processor selects the best quality copies among the ones provided as query answers and then sends back the result to the *Transport Engine Interface* that activates the *Execute-Query* module with the best quality copies and the organizations to be notified about them as input parameters. The best quality copies are then sent by the *Web Service Invoker*. On the receiver organization side, the *Execute-Query* module notifies the *Propose Manager* modules of involved organizations about the better quality data available in the system, thus implementing the quality feedback functionality that the Data Quality Broker provides at query processing time. Notice also that the *Execute-Query* module, on the sender organization side, also interacts with the *Availability Tester* modules: this makes quality notification not to be performed in a one-step process. Instead, a transaction starts that commits only when the set of sources that has to be notified is exhausted.

4 Experiments

In this Section, we first show the experimental methodology that we adopted in Section 4.1; then, in Sections 4.2 and 4.3, we show respectively quality improvement experiments and performance experiments.

4.1 Experimental Methodology

We performed two types of experiments. The first set shows the effectiveness of the Data Quality Broker to improve data quality. The second set shows some performance features of the Data Quality Broker.

We used two real data sets, each owned by an Italian public administration agency, namely: *(i)* the first data set is owned by the Italian Social Security Agency, referred to as INPS (in Italian, Istituto Nazionale Previdenza Sociale). The size of the database is approximately 1.5 millions of records; *(ii)* the second data set is owned by the Chambers of Commerce, referred to as CoC (in Italian, Camere di Commercio). The size of the database is approximately 8 millions of records. Some data are agency-specific information about businesses (e.g., employees social insurance taxes, tax reports, balance sheets), whereas others are common to both agencies. Common items include one or more identifiers, headquarter and branches addresses, legal form, main economic activity, number of employees and contractors, information about the owners or partners.

As far as quality improvement experiments, we have associated quality values to the INPS and CoC databases. Specifically, we have associated completeness and currency quality values to each field value. Completeness refers to the presence of a value for a mandatory field. As far as currency values, timestamps were already associated to data values in the two databases; such timestamps refer to the last date when data were reported as current. We have calculated the degree of overlapping of the two databases that is equal to about 970000 records.

As far as performance experiments, a P2P environment has been simulated. Each data source has been wrapped by a web service; such web services are deployed on different computers connected by a LAN at 100 Mbps and interacting each other using the SOAP protocol.

4.2 Quality Improvement Experiments

The experimental setting consists of the two described real data bases that are queried by a third data source that cooperates with them. We consider how this CIS behaves with regards to the quality of its data, in two specific cases. In the first case, a "standard" system is considered; this system does not perform any quality based check or improvement action. In the second case, the CIS uses the Data Quality Broker functionality of query answering and quality improving.

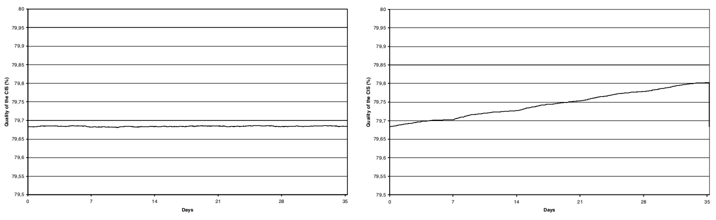
Values for the frequency of queries and updates on the data bases and average query result size are derived from real use cases. We have estimated the frequency of changes in tuples stored in the two databases to be around 5000 tuples per week.

Average query frequency and query result size are, respectively, of 3000 queries per week and 5000 tuples per query. In a real setting, updates are distributed over a week. Anyway, to simplify our experimental setting, we have chosen to limit updates to the beginning of each week.

We consider how the quality of the entire CIS changes throughout a period of five weeks. Note that such variations are due to both updates on the databases and exchanges of data between them. In the standard system, these exchanges are only due to queries. With the Data Quality Broker, each time a query is performed, an improvement feedback may be propagated. For both the Data Quality Broker and the standard system, we calculate the overall *Quality* of the system, as the percentage of the high quality tuples in the system. We adopt simplified quality metrics by considering that a tuple has high quality if it is complete and current on all its fields. Conversely, a tuple has low quality if it is not complete and/or current on some fields.

To clarify how the two systems reacts to updates, we use an update set composed by both high quality and bad quality tuples equally distributed.

Figure 3 shows the behavior of the Data Quality Broker and the standard system with respect to quality improvement. In the standard system (Figure 3.a), the overall quality is roughly constant, due to the same number of high quality and low quality tuples spread in the system. Instead, with the Data Quality Broker (Figure 3.b), the improvement of quality in each period is enhanced by data quality feedbacks performed by the system and low quality data are prevented to spread. This causes a growing trend of the Data Quality Broker curve, in spite of low quality inserted tuples. The actual improvement is about 0.12%; given that the size of the two databases is about 9.500.000 tuples, the improvement consists of about 11.500 tuples.



(a) Data quality improvement in the standard system (b) Data quality improvement with the Data Quality Broker

Fig. 3. Data quality improvement in the standard system and with the Data Quality Broker.

4.3 Performance Experiments

For the performance set of experiments, we have considered the Data Quality Broker and the standard system behavior with fictitious sources, in order to vary some parameters influencing performance experiments.

The first performance experiment shows the time overhead of the Data Quality Broker system with respect to the standard system. In such experiment we draw a *normalized transaction time* defined by the fraction:

$$\frac{\texttt{DataQualityBrokerElaborationTime}-\texttt{StandardElaborationtime}}{\texttt{StandardElaborationtime}}$$

The elaboration time is the time required by the system for processing a query. The normalized transaction time is drawn when varying the degree of overlapping of data sources. The overlapping degree significantly influences the Data Quality Broker. Indeed, the Data Quality Broker accomplishes its functionalities in contexts where data sources overlap and such an overlapping can be exploited to improve the quality of data. The Figure 4 left side shows how the normalized transaction time varies in dependance on the percentage of data sources overlapping with two fixed query result sizes, namely q1=1000 tuples, q2= 5000 tuples. The number of overlapping sources is fixed to 3. This means that once a query is posed over the system, three sources have data that can be provided as answer to the query, though the system can have a larger number of sources. Figure 4 shows the actual time overhead of the Data Quality Broker systems with respect to a standard system. The Data Quality Broker system has an acceptable time overhead. The worst depicted case is for the query result size q2=5000 and a percentage of overlapping equal to 40%; in such a case, there is a 50% time overhead with respect to the standard system.

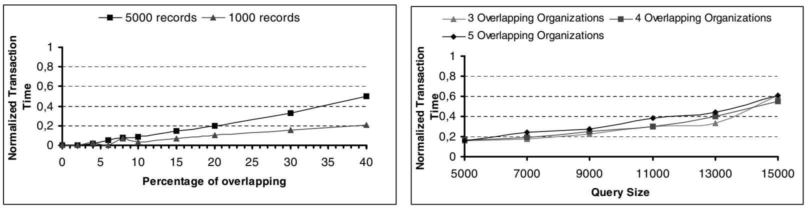


Fig. 4. Normalized transaction time wrt percentage of overlapping data sources (left side) and normalized transaction time wrt query sizes (right side)

The second performance experiment shows the normalized transaction time with query size varying (see Figure 4 right side). For a fixed degree of overlapping equals to 15%, we draw the normalized transaction time for three different numbers of overlapping organizations, namely n1=3, n2=4 and n3=5. This experiment shows the behavior of the Data Quality Broker when increasing the number of organizations and the size of queries. Specifically, the normalized transaction time increases slowly with an almost linear trend. The positive result shown in Figure 4 is that when the number of overlapping data sources increases, the trend does not substantially change.

5 Related Work

Data quality has been explicitly addressed in a few works. In [15], an algorithm for querying for best quality data in a LAV integration system is proposed. In the Data

Quality Broker framework, we share with [15] the idea of querying for best quality data. The main difference of our work with respect to [15] is the semantics of our system. Our aim is not only querying, but also improving quality of data. To such a scope, the query processing step has a specific semantics that allows to perform quality improvement on query results.

The MIT Context Interchange project (COIN) [6] is based on the idea of modeling a “context" for integrating heterogeneous sources. Such a context consists of metadata that allows for solving problems, such as instance level conflicts that may occur in the data integration phase. However, the COIN approach focuses only on one aspect of data quality, namely *data interpretability*, while our approach offers a much more general and explicit way of representing the quality of data.

In [12], the basic idea is to select web data sources based on quality values on provided data. Specifically, the authors suggest to publish metadata characterizing the quality of data at the sources. Such metadata are used for ranking sources, and a language to select sources is also proposed. In the Data Quality Broker system, we associate quality to data (at different granularity levels) rather than to a source as a whole. This makes things more difficult, but allows to pose more specific queries.

In 1999, the Italian Public Administration started a project, called “Services to Businesses”, which involved extensive data reconciliation and cleaning [3]. The approach therein followed consisted of three main steps: *(i)* linking *once* the databases of three major Italian public administrations, through a record matching process; *(ii)* correcting matching pairs and *(iii)* maintaining such status of aligned records in the three databases by centralizing record updates and insertions only on one of the three databases. This required a substantial re-engineering of administrative processes, with high costs and many internal changes for each single administration. In out framework, conversely, the choice of implementing the Data Quality Broker functionality in a completely distributed way through a P2P architecture avoids bottlenecks on a single cooperating rganization. Even more important, no kind of re-engineering actions need to be engaged when choosing to use the Data Quality Broker, as query answering and quality improvement can be performed with a very low impact in terms of changes on cooperating organizations.

6 Concluding Remarks

In this paper, we have described the detailed design and implementation of the Data Quality Broker service, and in particular two modules, namely the Query Processor and the Transport Engine. We have also described some experiments that validate our approach with respect to quality improvement effectiveness. Such experiments show that the Data Quality Broker succeeds in controlling and improving quality of data in a CIS. Moreover, when compared to a standard system, i.e. a system with no quality management features, the Data Quality Broker exhibits a limited performance degradation. Such a performance degradation is not a serious problem in specific scenarios, such as e-Government, in which the quality of data is the main enabling issue for service provisioning. Indeed, we remark that such scenarios are the reference ones for the DaQuinCIS system. Future works will include the deep validation based on the adoption of the proposed P2P system in some Italian e-Government pilot initiatives.

References

1. *IPSI-XQ*, Available from `http://ipsi.fhg.de/oasys/projects/ipsi-xq/index_e.html`.
2. K. Aberer and Z. Despotovic, *Managing Trust in a Peer-2-Peer Information System*, Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM'01), Atlanta, Georgia, USA, 2001.
3. M. Bertoletti, P. Missier, M. Scannapieco, P. Aimetti, and C. Batini, *Improving Government-to-Business Relationships through Data Reconciliation and Process Reengineering*, Advances in Management Information Systems-Information Quality Monograph (AMIS-IQ) Monograph (Richard Wang, ed.), Sharpe, M.E., to appear, 2004. Shorter version also in ICIQ 2002.
4. S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simèon, *XQuery 1.0: An XML Query Language*, W3C Working Draft. Available from `http:///www.w3.org/TR/xquery`, November 2003.
5. M. Bouzeghoub and M. Lenzerini (editors), *Special Issue on Data Extraction, Cleaning, and Reconciliation*, Information Systems **26** (2001), no. 8.
6. S. Bressan, C.H. Goh, K. Fynn, M.J. Jakobisiak, K. Hussein, K.B. Kon, T. Lee, S.E. Madnick, T. Pena, J. Qu, A.W. Shum, and M. Siegel, *The COntext INterchange Mediator Prototype*, Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD 1997), Tucson, Arizona, USA, 1997.
7. M.F. Fernandez, A. Malhotra, J. Marsh, M. Nagy, and N. Walshand, *XQuery 1.0 and XPath 2.0 Data Model*, W3C Working Draft. Available from `http:///www.w3.org/TR/query-datamodel`, November 2002.
8. JSR-101 Expert Group, *Java(tm) api for xml-based remote procedure call (jax-rpc) specification version 1.1*, Sun Microsystems, Inc., October 2003.
9. M. Lenzerini, *Data Integration: A Theoretical Perspective*, Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS 2002), Madison, Wisconsin, USA, 2002.
10. M. Mecella and C. Batini, *A Review of the First Cooperative Projects in the Italian* e-*Government Initiative*, Proceedings of the 1st IFIP Conference on *e*-Business, *e*-Commerce, *e*-Government, Zurich, Switzerland, 2001.
11. M. Mecella, M. Scannapieco, A. Virgillito, R. Baldoni, T. Catarci, and C. Batini, *The DaQuinCIS Broker: Querying Data and their Quality in Cooperative Information Systems*, Journal of Data Semantics **1** (2003), no. 1. Shorter version also appeared in CoopIS 2002.
12. G. Mihaila, L. Raschid, and M. Vidal, *Using Quality of Data Metadata for Source Selection and Ranking*, Proceedings of the 3rd International Workshop on the Web and Databases (WebDB'00), Dallas, Texas, 2000.
13. D. Milano, M. Scannapieco, and T. Catarci, *Quality-driven Query Processing of XQuery Queries*, Submitted to International Conference, 2004.
14. J. Mylopoulos and M.P. Papazoglou (eds.), *Cooperative Information Systems (Special Issue)*, IEEE Expert Intelligent Systems & Their Applications **12** (1997), no. 5.

15. F. Naumann, U. Leser, and J.C. Freytag, *Quality-driven Integration of Heterogenous Information Systems*, Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99), Edinburgh, Scotland, UK, 1999.
16. M. Scannapieco, A. Virgillito, M. Marchetti, M. Mecella, and R. Baldoni, *The DaQuinCIS architecture: a Platform for Exchanging and Improving Data Quality in Cooperative Information Systems*, Information Systems (to appear, 2004).
17. J.D. Ullman, *Information Integration Using Logical Views*, Proceedings of the 6th International Conference on Database Theory (ICDT '97), Delphi, Greece, 1997.
18. R.Y. Wang and D.M. Strong, *Beyond Accuracy: What Data Quality Means to Data Consumers*, Journal of Management Information Systems **12** (1996), no. 4.
19. G. Wiederhold, *Mediators in the Architecture of Future Information Systems*, IEEE Computer **25** (1992), no. 3.

Moving from Internal to External Services Using Aspects

Martin Henkel, Gustav Boström and Jaana Wäyrynen

Department of Computer and Systems Science, Stockholm University and Royal Institute of Technology, Forum 100, SE-164 40 Kista, Sweden, `martinh@dsv.su.se`, `gusbo@kth.se`, `jaana@dsv.su.se`

Summary. Service oriented computing and web service technologies provide the means to structure an organisation's internal IT resources into a highly integrated network of services. In e-business and business process integration the internal services are interconnected with other, external organisations' resources to form virtual organisations. This move from using services internally to external use puts new non-functional requirements on the service implementation. Without any supporting technologies, meeting these new requirements can result in re-writing or changing a large part of the service implementation. In this paper we argue that aspect oriented programming is an important technique that can be used to facilitate the implementation of the new requirements that arises when moving from internal to external services. The suggested solution is illustrated by an example where quality of service metrics is implemented by using aspect oriented programming.

1 Introduction

Service oriented computing, in particular the web service technologies, has drawn a lot of attention in recent years. The reason for this attention is multi-faceted. One reason is based on the view that service oriented computing is a natural step of evolution from object-oriented and component based computing. Another aspect that makes services interesting is that they can be used to structure and interconnect an organisation's internal IT systems. Even more importantly, services can be used externally to enable interconnection of enterprises [1], thus enabling the forming of networked or "virtual" enterprises [2].

The interconnecting of enterprises via service technology requires that part of an enterprise's internal systems must be made available to external organisations. This shift from internal to external use puts new requirements on existing IT systems. First of all, the systems must be able to communicate, regardless of differences in platforms and languages. This interoperability can be achieved by conforming to technical standards. The existing and upcoming web service standards such as SOAP [3] and WSDL [4] are important steps towards interoperable services. A second, somewhat overlooked part is that providing external services also puts new non-functional requirements on existing systems, such as scalability, security and quality. These requirements are sometimes also called “illities” [5].

Within the border of a company these requirements might be implicitly met by assumptions about the company’s secure intranet, the well-defined number of users, the support department’s ability to monitor the quality etc. However, exposing parts of the system to external partners will require that these implicit assumptions need to be converted into explicit, measurable, and monitorable implementations. Furthermore, the new non-functional requirements posed need to be implemented as part of the existing systems.

Implementations of new non-functional requirements often cut across the entire system, i.e. a large part of the existing system code is affected by new non-functional requirements. Thus, when moving from internal to external service use, a potentially large part of the system code needs to be changed. The need to change large portions of code to implement new non-functional requirements is a problem, since it increases the risk of major redesign when moving from internal to external use of services. The question is how to overcome this problem.

Aspect Oriented Programming (AOP) is proposed as a technique to implement functionality that cuts across an entire system [6]. Thus, aspect orientation may be a solution that can facilitate the implementation of the non-functional requirements that arises when internal systems are to be exposed as external services.

In this paper we argue that aspect oriented programming can be a useful technique for moving from internal to external services. The paper begins with a short introduction to service oriented computing and aspect oriented programming. The introduction is followed by a description of the problem and the proposed solution. Then, we provide an example of how aspect-orientation can be applied in the context of a service that needs to be changed due to changed non-functional requirements. The article ends with a discussion of the proposed solution’s applicability and a discussion of further work.

1.1 Service Oriented Computing

A *service* is an “act or performance offered by one party to another” [7]. Services offered by IT systems have been dubbed “e-Services” [8] and “software services”. In this paper the term service denotes services offered by one system to another system, i.e. where both parties are software systems. This excludes services that are offered to (human) users via graphical user interfaces.

Service Oriented Computing (SOC) builds on the component based development (CBD) principles of building systems by combining software parts. In contrast to components, a service is a run-time programming interface rather than a physical/binary entity that needs to be installed before use. This distinction between runtime provisioning and implementation is made by component based development methods [9]. The distinction might seem finicky, but it has a profound impact on how services are used. A provider of a service is responsible for the run-time availability of the service, whereas a component provider is only responsible for the construction and delivering of the binary component. Thus, building a component based system is about assembling software parts, while building a service-oriented system is about communicating with services offered by different providers.

The focus on run-time availability and provider responsibility makes services an ideal metaphor for interconnecting a organisation's IT-systems (internal use), where each system is a separate run-time entity. For the same reason, service oriented computing can play a major role in the interconnection of systems belonging to separate organisations (external use).

1.2 Aspect Oriented Programming

Aspect oriented programming is a paradigm that attempts to help in implementing concerns that are present in many modules and therefore crosscuts a system, that is *cross-cutting concerns*. Cross-cutting concerns are difficult to modularise using existing object-oriented techniques since there is no logical place in which to implement them. An illustrative example is logging of method-calls. Using existing object-oriented techniques the code for implementing this would be spread out in all methods that require logging. Changing the way logging is done, and especially, where it is performed is therefore difficult to accomplish without changing all methods that need to be logged. This poor modularisation leads to code that is difficult to maintain. The fact that you need to deal with several concerns, logging and business logic, in the same method also adds to the complexity of the code.

Aspect oriented programming provides a way to modularise these cross-cutting concerns in an efficient manner by factoring out logic belonging to a specific concern into an *Aspect* [10]. In this article we use AspectJ as a tool to implement aspect oriented programming [11]. An aspect in AspectJ consists of *Pointcuts*, and *Advice* (in AspectJ an aspect can also contain *Inter Type declarations*, but this concept is not used in this article). *Pointcuts* describe *where* the aspect should apply in terms of the object-oriented systems structure, e.g. the pointcuts of the logging aspect would describe *where* in the system logging should be performed in terms of the classes and methods of the system. *Advice* describes *what* should happen at these pointcuts, e.g. how the logging should be carried out. The AspectJ keywords *aspect, pointcut* and *advice* are added to the Java-syntax in order to support these concepts. The process of combining the aspects and the classes into an executable system is called *aspect weaving*.

1.3 Functional, Non-functional and Cross-cutting Requirements

Functional requirements are statements of services that the system should provide. This can also be said as describing *what* the system should do. Non-functional requirements, on the other hand, are focused on *how* the system should perform the services [12]. An example of a functional requirement on an ATM-machine could for example be that an ATM-machine should be able to dispense money to the bank's customers. A non-functional requirement could be that this service has to be performed *securely* and with an acceptable *response time*. Other examples of non-functional requirements are performance, traceability, scalability and error handling. A problem with non-functional requirements is that they are often *crosscutting,* i.e. they affect many modules of the system. For example, security needs to be addressed in many parts of an ATM-system. It is therefore difficult to modularise crosscutting requirements. This can make systems difficult to maintain and evolve [13].

2 Moving from Internal to External Services

As stated in the introduction, service oriented computing promises to interconnect organisations. This is done by integrating and automating business processes that span across several organisations. Service oriented architectures (SOA) and service technologies, such as web services form the fundament for such integration. Integrating business processes and automating them relies on the integration of the organisation's IT systems. This in turn, requires that contact patches between the systems need to be established. Systems, which previously only where used within a company, need to exchange information with external systems through these contact patches. Interconnecting the processes of two organisations commonly does not require that all the IT systems need to be integrated. Rather than making an entire system available externally, a selection of functionality is made. This functionality is then exposed as services that can be used by external organisations [14]. As mentioned earlier, the exposed services commonly need to adhere to a new set of non-functional requirements. Examples of new non-functional requirements are security, quality of service measurements, and performance monitoring. Implementing support for these new requirements is instrumental in making the services available externally.

There exist several solutions to this problem. The first solution that comes to mind is to rework the entire code to support the new requirements. This can be achieved by following common refactoring principles, such as those proposed by Fowler [15]. However, this solution requires a lot of work, since each method that should adhere to the new requirements has to be reviewed. More generic approaches have also been proposed, such as the addition of an extra layer to existing component-environments by using generated proxies controlled by proprietary description languages [16]. These generic approaches have a much better chance of reducing the amount of work required. They are, however, based on proprietary languages and technologies. The ideal would be a technique which is generic (to avoid too much rework), and at the same time does not rely on proprietary technologies, servers, and languages. We propose that aspect oriented programming can be such a technique.

Aspect oriented programming separates the code required to fulfil the new requirements from the existing code. The new requirements can thus be separately implemented as aspects, without changing the existing code. These aspects can then selectively be applied (by aspect weaving) to the parts of the code that need to adhere to the new requirements. Applying the aspects does not require changing the original code. For a large system this can save a lot of time.

In the next section we will give an example of how aspect oriented programming can be used to implement non-functional requirements without a major rework of the original system.

3 An Example: Adding QoS Metrics to Web services

The example in this section describes the steps necessary to extend a web service with quality of service metrics monitoring (QoS monitoring). The example elucidates the main point of this paper, that by using aspects, the move from internal services to external services does not require a major rework of the code. The need to extend web services with QoS metrics is selected as an example both because it is a likely scenario, and because it clearly demonstrates how non-functional requirements can be implemented using aspects. What makes the scenario likely is that enterprises starting to use the web service technology internally will need to further define, and monitor their quality of service when starting to use web service technologies as an external communication mean between enterprises, thus the need to add QoS metrics to web services.

3.1 Scenario

To illustrate how AOP will help in implementing non-functional requirements such as QoS metrics let's imagine a company that provides financial services such as mortgages and loans for cars. In this business it is essential to know your customers' credit worthiness. Credit worthiness is determined using the customers' credit history, income and other variables. Different financial services require different definitions and levels of credit worthiness. This information is used in determining whether to grant applications for both loans and mortgages. Since credit checking is an important part of this organisation's business, it is implemented as a web service that can be reused from all systems within the organisation. Figure 1, below, shows how the interface to this credit checking service might look like implemented in Java.

```
public interface CreditCheckingServiceInterface
{
      public boolean hasPaymentRemarks(String name);
      public boolean hasCreditHistory(String name);
      public boolean checkCreditForAmount
                        (String name, int amount);
}
```

Figure 1. The interface of the CreditCheckingService

The next step in the organisation's business plans could be to provide the credit checking service to external businesses, such as mobile phone operators and car leasing companies that also need efficient credit check processing. However, before using the credit checking from their systems, external businesses will require some form of quality guarantee. For example, a potential customer of the service would probably ask the following questions:

- How can it be ensured that the service paid for is reliable and running when needed?
- How can the organisation monitor that the performance is acceptable?

In short, the customers will require some form of agreement that states the intended quality of service. The agreement can include measurable limits for performance, cost, up time and other dimensions that affect the overall quality of the provided web service. For this example we use three QoS dimensions for web service processes as defined by Sheth et al. [17]: time, cost and reliability.

- Time is a measure of response time of the web service that is to be monitored. The response time is measured from request arrival to the completion of the request.
- Cost can be measured by either estimating an average cost for each service invocation, or by measuring the resources that are consumed to complete a request (such as processor time, cost of information storage etc).
- Reliability is a measure of technical failure rate, that is, monitoring the reliability will discover how many times the service failed to deliver a response. Sheth et al. [17] suggest that reliability is to be measured as a ratio of successful executions/scheduled executions.

The credit checking web service mentioned above is not built with QoS metrics in mind, since it was designed for internal use only. Adding QoS metrics to the existing service can be a major undertaking, since code that monitors the metrics need to be inserted in all parts of the service. Without a technique that helps implement cross-cutting, non-functional requirements such as QoS metrics, developers are running the risk of having to redesign a major part of the code. However, applying aspect oriented programming can reduce this risk. An example of how aspects can be applied in this case is described in the next section.

3.2 Applying Aspects

Let's look at how aspects could be applied in the described scenario. The three QoS dimensions time, cost and reliability define what is to be measured. Before implementing the actual metrics, it has to be decided where in the application code the dimensions should be measured. A basic approach would be to add code to register each metric in the beginning and end of each request, i.e. before and after each call to the web service. Without using aspects, this approach would require additional code that has to be inserted in *all* web service methods. However, using an aspect-oriented approach, adding QoS metrics to web services would only require the metrics *aspects* and their *join points* to be defined once, without any change to the original web service implementation.

To implement QoS metrics for the credit checking service, one aspect for each of the QoS dimension can be implemented. Thus, as an example we have implemented the aspects PerformanceQoSAspect, CostQoSAspect and ReliabilityQoSAspect .

3.3 Performance Aspect

The performance aspect is intended to measure the Time QoS dimension. Time can be measured by recording the request/method name, when the request arrived and when the response was sent. The implementation of this metric requires that two aspect join points are defined; one at the beginning of each method call and one at the end. These join points are defined within the AspectJ pointcut “timedMethods”, see Figure 2.

```
public aspect PerformanceQoSAspect
{
      Timer timer=new Timer();
      pointcut timedMethods() : (
            execution(public *
                  CreditCheckingService.* (..)));

      before() : timedMethods()
      {           // Start timing
      }
      after() : timedMethods()
      {           // End timing
      }
}
```

Figure 2. Performance QoS aspect

3.4 Cost Aspect

Cost can be measured by recording the request/method name for each request. Using predefined cost for each type of request, the total cost can be calculated. The measurement of cost can be done by using a join point at the end of each web service method. The AspectJ example in Figure 3 shows how an aspect that logs each method call can be implemented.

```
public aspect CostQoSAspect
{
      pointcut costMethods() : (
            execution(public *
                  CreditCheckingService.* (..)));

      after() : costMethods()
      {           // Log the cost of
                  // the executed method
      }
```

Figure 3. Cost QoS aspect

3.5 Reliability Aspect

Reliability can be measured by recording if the response of a request is a valid response or an error. In this case, a join point can be defined at the end of each method. The AspectJ implementation shown in Figure 4 defines an aspect that logs every method call that ends with a non-application Exception.

```
public aspect ReliabilityQoSAspect
{
      pointcut reliabilityMethods() : (
            execution(public * CreditCheckingService.*
(..)));

      after() throwing(Exception e):
reliabilityMethods()
      {
            if(!(e instanceof ApplicationException))
            {                 // Log error
            }
      }
}
```

Figure 4. Reliability QoS aspect

The example given above can be extended with more QoS metrics. This example illustrates the main points in using aspects for the implementation of non-functional requirements.

4 Conclusion

In this article we proposed that AOP could help the transition from internal to external services. By using AOP, non-functional requirements can be implemented without doing a major redesign of the existing system. The feasibility of the proposed solution has been demonstrated with a simple example written in AspectJ.

The example demonstrated that AOP could be a useful tool when an application needs to accommodate QoS metrics that have not been previously designed into the system. It also shows that this can be easily achieved using just a few lines of code. In fact, the bigger the application, the more amount of time will be saved by using aspects.

AspectJ was used in the example. However, there are other ways to implement non-functional requirements in an "aspect oriented" way. It could be argued that by using a component technology such as Enterprise Java Beans (EJB), QoS metrics could be provided by the application server (e.g. through the use of method interceptors in the JBoss EJB server [18]). These QoS metrics, however, are not currently standardised, they would therefore be different for each component server. It would also require the application to be built as a component-based application from the start, which is often a lot more time-consuming and skill-intensive than using plain Java objects. Using design patterns such as the "proxy" pattern [19] could also alleviate the need for using specific AOP technologies such as AspectJ. An example of this is provided by Filman et al. [5]. This approach, however, is considerably more time-consuming and therefore also more error-prone.

The proposed solution is applicable when the move from internal to external services poses new non-functional requirements. Clearly, if no additional non-functional requirements need to be fulfilled, the need to introduce aspect-oriented concepts is not as obvious. Furthermore, the proposed solution presumes that the non-functional requirements can be implemented in a generic, separated way using aspects. In the case that not all new requirements can be implemented in this way, aspects can still contribute to the implementation of some of the requirements. Thus, we believe that the use of aspect-oriented programming can be a valuable technique when moving from internal to external services.

5 Further work

In this paper we examined how aspect oriented programming can be used to tackle the non-functional requirements when moving from internal to external services. However, when integrating processes it is likely that other changes need to be implemented in parallel with the new non-functional requirements. For example, when integrating processes there might be a need for further process automation, i.e. new functional requirements. A possible future extension of our work might include principles guiding the combination of aspect-oriented programming with traditional refactoring techniques to implement both non-functional and functional requirements.

Another interesting question is whether AOP could prove useful for solving other "architecture breaking" problems. There are several indications that this could be the case. De Win et al. [20] have shown how AspectJ can be used to help implement security features in an application. Filman et al. [5] have described how AOP can be used for inserting "ilities", such as stability and reliability. These examples, however, do not prove that AOP can handle every possible new requirement.

References

1. Fremantle, P., Weerawarana, S., Khalaf, R., Enterprise Services. Communications of the ACM, October 2002, Vol. 45, No 10 (2002)
2. Yang , J., van den Heuvel, W. J., Papazoglou, M. P., Service Deployment for Virtual Enterprises. Australian Computer Science Communications, Vol. 23, Iss. 6 (2001)
3. Gudin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H. F., SOAP Version 1.2. W3C Candidate Recommendation. (2002)
4. Chinnici, R., Gudin, M., Moreau, J., Weerawarana, S., Web Services Description Language (WSDL) Version 1.2. W3C Working Draft (2003)
5. Filman R., et al., Inserting Ilities by Controlling Communications. Communications of the ACM, Vol. 45 (2002)
6. Duclos F., Estublier J., Morat P., Describing and Using Non Functional Aspects in Component Based Applications. 1st International Conference on Aspect-Oriented Software Development (2002)

7. Lovelock, C., Vandermerwe, S., Lewis, B., Services Marketing. Prentice Hall Europe (1996)
8. Piccinelli, G., and Stammers, E., From E-Process to E-Networks: an E-Service-oriented approach. OOPSLA Workshop on Object-Oriented Web Services (2001)
9. Allen, P., Frost, S., Component-Based Development for Enterprise Systems: Applying the Select Perspective. Cambridge University Press (1998)
10. Elrad, T., Filman, R., Bader, A., Aspect-oriented Programming an Introduction, Communications of the ACM Vol. 44 (2001)
11. AspectJ, www.aspectj.org. Accessed in April 2003
12. Cysneiros, L., do Prado Leite, J., Non-Functional Requirements: From Elicitaion to Modelling Languages. International Conference on Software Engineering (2002)
13. Moriera, A., Araújo, J., Brito, I., Crosscutting Quality Attributes for Requirements Engineering**.** 1st International Conference on Aspect-Oriented Software Development (2002)
14. Georgakopolos, D., Schuster, H., Cichocki, A., Baker, A., Managing Process and Service Fusion in Virtual Enterprises. Information System Vol. 24, No6 (1999) 429-456
15. Fowler, M., Refactoring – Improving the Design of Existing code. Addison-Wesley (1999)
16. Becker, C., Geihs, K., Quality of service and object-oriented middleware - multiple concerns and their separation. International Conference on Distributed Computing Systems (2001) 117 -122
17. Sheth A., Cardoso J., Miller J., Kochut K., Kang M., QoS for Service-Oriented Middleware.. Proceedings of the Conference on Systemics, Cybernetics and Informatics, (2002)
18. JBoss, www.jboss.org, Accessed in April 2003
19. Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
20. De Win, B., Vanhaute, B., De Decker, B., How Aspect-Oriented Programming Can Help to Build Secure Software. Informatica Journal, Vol. 26, (2002) 141-149

Methodology for the Definition of a Glossary in a Collaborative Research Project and its Application to a European Network of Excellence

Paola Velardi[1], Raúl Poler[2] and José V. Tomás[2]

[1] University of Rome "La Sapienza", Via Salaria 113, 00198 Rome, Italy
velardi@di.uniroma1.it
[2] Polytechnic University of Valencia, Camino de Vera, s/n, 46002 Valencia, Spain
rpoler, jotomi@cigip.upv.es

Summary. The aim of this paper is to describe the methodology of creation of a Glossary in a collaborative research project and its application to the Network of Excellence IST-508011 INTEROP "Interoperability Research for Networked Enterprise Applications and Software" for the definition of a glossary in the area of interoperability of enterprise applications and software. The proposed methodology is based on an adaptation of a method of the University of Rome for the semiautomatic acquisition of terms and definitions starting from a source of documents related to the research areas of a collaborative project.

Introduction

Knowledge Management has been gaining significant importance within organisations and is considered an important success factor in enterprise operation. For some time, there have been many techniques to model processes and other elements of the enterprise in order to capture the explicit knowledge. Modelling in this context means creating an explicit representation, usually computable, for the purposes of understanding the basic mechanics involved.

But knowledge can mean different things to different people and companies must spend some time looking for an appropriate mechanism to avoid misunderstanding in knowledge transmission. One mechanism to avoid this problem is to build a Glossary. The goal is to make accessible the organizational knowledge by unifying the language used in representing explicit knowledge. The semantic unification is a key factor for the success of the knowledge dissemination and diffusion through an organization.

2 Methodology to Obtain a Glossary

A general method for constructing a glossary is: collect a vocabulary, collect definitions, establish format rules, establish rules for writing definitions, examine definitions for multiple meanings, write basic definitions, review and compare for consistency of meaning, classify, select preferred words, group words, review and finalize the glossary. In practice, some of these steps are omitted, while other steps are developed to considerable depth, depending on the final objective of the glossary. The complexity of detail for any phase of the glossary depends upon the scope of the glossary, the size of the vocabulary, and the number of persons participating in the project. This is understandable because a large working group reflecting a wide range of subjects, introduces more words for consideration and supplies multiple meanings relative to different backgrounds. Starting from a wide list of terms with the objective of building a whole glossary with inputs of several researchers could consume a great amount of time and effort. Therefore, a specific methodology was defined:

- **1st stage**: to define the purpose of the glossary.
- **2nd stage**: to built an initial list of terms and definitions using a semi-automatic glossary acquisition.
- **3rd stage**: to set up the collaborative glossary online module to support the sharing and extension of the glossary.
- **4th stage:** to complete the glossary by means of manual inputs and reviews, that is, the extension of the glossary.

2.1 1st Stage: Purpose of the Glossary

In a collaborative environment, all participants must have in mind what is the main purpose of the glossary and the benefits they are going to obtain as well. Because of this, the project representatives must meet in order to clarify objectives and possible applications of the glossary. This fact becomes critical when the research project has different related research areas. There are several benefits in creating a durable glossary facility:

- *Semantic unification*: the glossary represents an informal, but shared view of relevant concepts. This activity will let semantics *emerge* naturally from applications and collaborative work.
- *Classification/retrieval* of documents: glossary terms may be used as meta-data for indexing documents and databases.
- *Integration of competences* from different research areas.

2.2 2nd Stage: Semi-automatic Glossary Acquisition

The second stage of the methodology will lead to obtain the first version of the Glossary. This stage is based on a semi-automatic tool for ontology building called OntoLearn. This first version of the Glossary must be addressed as a preliminary stage for the generation of the final glossary. The extension and the diffusion between the research community are strictly required to meet the projected requirements.

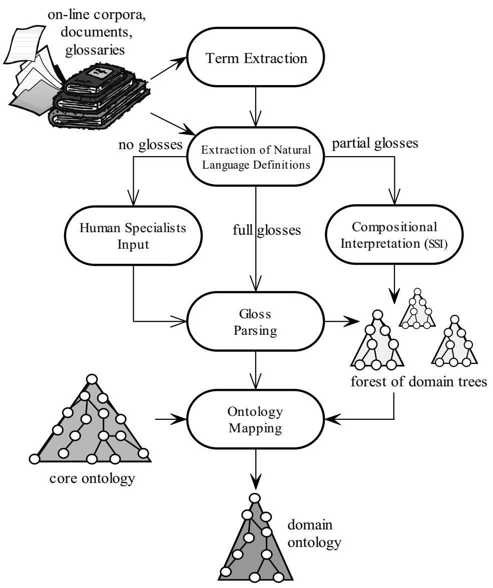


Figure 1. An outline of the ontology learning phases in the OntoLearn system

Main steps of OntoLearn semi-automatic procedure. Figure 1 provides a snapshot of the OntoLearn ontology learning methodology. The following steps are performed by the system:

Step 1. Extract pertinent domain terminology. Simple and multi-word expressions are automatically extracted from domain-related corpora, like enterprise interoperability (e.g. *collaborative work*), hotel descriptions (e.g. *room reservation*), computer network (e.g. *packet switching network*), art techniques (e.g. *chiaroscuro*). Statistical and natural language processing (NLP) tools are used for automatic extraction of terms [3]. Statistical techniques are specifically aimed at simulating *human consensus* in accepting new domain terms. Only terms uniquely and consistently found in domain-related documents, and not found in other domains used for contrast, are selected as candidates for the domain terminology.

Step 2. Search on the web for available natural language definitions from glossaries or documents. Available definitions are searched on the web using on-line glossaries or extracting definitory sentences in available documents. A context free (CF) grammar is used to extract definitory sentences. An excerpt is:

S $\rightarrow$ PP ‘,’ NP SEP
NP $\rightarrow$ N1 KIND1
KIND1 $\rightarrow$ MOD1 NOUN1
MOD1 $\rightarrow$ Verb | Adj | Verb ‘,’ MOD1 | Adj ‘,’ MOD1

NOUN1 → Noun
N1 → Art | Adj
SEP → ‘,’ | ‘.’ | Prep | Verb | Wh
PP → Prep NP

In this example, *S*, *NP* and *PP* stand for sentence, noun phrase and prepositional phrase, respectively. KIND1 captures the portion of the sentence that identifies the kind, or genus, information in the definition. This grammar fragment identifies (and analyses) definitory sentences like e.g.: “[In a programming language]$_{PP}$, [an *aggregate*]$_{NP}$ [that consists of data objects with identical attributes, each of which may be uniquely referenced by subscription]$_{SEP}$”, which is a definition of *array* in a computer network domain. The grammar is tuned for high precision, low recall. In fact, certain expressions (e.g. *X is an Y*) are overly general and produce mostly noise when used for sentence extraction.

Step 3. IF definitions are found:

Step 3.1. Filter out non relevant definitions. Multiple definitions may be found on the internet, some of which may be not pertinent to the selected domain (e.g. in the interoperability domain federation as “the forming of a nation” rather than “a common object model, and supporting Runtime Infrastructure.”). A similarity-based filtering algorithm is used to prune out “noisy” definitions, with reference to a domain. Furthermore, an extension of the CF grammar of step 2 is used to select[1], when possible, “well formed” definitions. For example, definitions with genus(kind-of) and differentia (modifier), like the array example in step 2, are preferred to definitions by example, like: Bon a Tirer ”When the artist is satisfied with the graphic from the finished plate, he works with his printer to pull one perfect graphic and it is marked “Bon a Tirer,” meaning “good to pull”. These definitions can be pruned out since they usually do not match any of the CF grammar rules.

Step 3.2. Parse definitions to extract kind-of information. The CF grammar of step 3.1 is again used to extract kind-of relations from natural language definitions. For example, in the array example reported in step 2, the same grammar rule can be used to extract the information (corresponding to the KIND1 segment in the grammar excerpt): $array \xrightarrow{kind-of} aggregate$

Step 4. ELSE IF definitions are not found:

Step 4.1. IF definitions are available for term components (e.g. no definition is found for the compound integration strategy but integration and strategy have individual definitions).

Step 4.1.1. Solve ambiguity problems. In technical domains, specific unambiguous definitions are available for the component terms, e.g.: strategy: “a series of planned and sequenced tasks to achieve a goal” and integration: “the ability of applications to share information or to process independently by requesting services and satisfying service requests” (interoperability domain). In

[1] The grammar used for analysing definitions is a superset of the grammar used to extract definitions from texts. The analysed sentences are extracted both from texts and glossaries, therefore expressions like X is an Y must now be considered.

other domains, like tourism, definitions of component terms are often extracted from general purpose dictionaries (e.g. for housing list, no definitions for list are found in tourism glossaries, and in generic glossaries the word list is highly ambiguous). In these cases, a word sense disambiguation algorithm, called SSI[2] [4] [5], is used to select the appropriate meaning for the component terms.

Step 4.1.2. Create definition compositionally. Once the appropriate meaning components have been identified for a multi-word expression, a generative grammar is used to create definitions. The grammar is based on the presumption (not always verified, see [5] for a discussion) that the meaning of a multi-word expression can be generated compositionally from its parts. According to this compositional view, the syntactic head of a multi-word expression represents the genus (kind-of), and the other words the differentia (modifier). For example, integration strategy is a strategy for integration. Generating a definition implies, first, to identify the conceptual relations that hold between the complex term components[3], and then, to compose a definition using segments of the components' definitions. For example, given the term integration strategy, the selected underlying conceptual relation is purpose:

$$Strategy \xrightarrow{purpose} Integration$$

and the grammar rule for generating a definition in this case is:

<MWE> = **a kind of** <H>, <HDEF>, **for** <M>, <MDEF> . **(1)**

Where MWE is the complex term, H is the syntactic head, HDEF is the main sentence of the selected definition for H, M is the modifier of the complex term and MDEF is the main sentence of the selected definition for M.

For example, given the previous definitions for *strategy* and *integration,* the following definition is generated by the rule (1): *integration strategy*: a **kind of** *strategy*, a series of planned and sequenced tasks to achieve a goal, **for** *integration*, the ability of applications to share information or to process independently by requesting services and satisfying service requests. As better discussed in [5] this definition is quite verbose, but has the advantage of showing explicitly the sense choices operated by the sense disambiguation algorithm. A human supervisor can easily verify sense choices and reformulate the definition in a more compact way.

Step 4.2. ELSE ask expert. If it is impossible to find even partial definitions for a multi-word expression, the term is submitted to human specialists, who are in charge of producing an appropriate and agreed definition.

Step 5. Arrange terms in hierarchical trees. Terms are arranged in forests of trees, according to the information extracted in steps 3.2 and 4.1.1.

Step 6. Link sub-hierarchies to the concepts of a Core Ontology. The semantic disambiguation algorithm SSI (mentioned in step 4.1.1) is used to append sub-trees under the appropriate node of a Core Ontology. In our work, we use a

[2] The SSI algorithm (Structural Semantic Interconnections) is one of the novel and peculiar aspects of the OntoLearn system. SSI recently participated to an international challenge, Senseval-3, obtaining the 2nd best score in a word sense disambiguation task.

[3] Machine learning techniques are used to assign appropriate conceptual relations.

general purpose wide-coverage ontology, WordNet. This is motivated by the fact that sufficiently rich domain ontologies are currently available only in few domains (e.g. medicine).

Step 7. Provide output to domain specialists for evaluation and refinement. The outcome of the ontology learning process is then submitted to experts for corrections, extensions, and refinement. In the current version of OntoLearn, the output of the system is a taxonomy, not an ontology, since the only information provided is the kind-of relation. However, extensions are in progress, aimed at extracting other types of relations from definitions and on-line lexical resources.

2.3 3rd Stage: Setting up a Glossary Online Collaborative Platform

Once completed the first list of terms and definitions using a semi-automatic glossary acquisition, the procedure selected to extend the glossary is the use of a Glossary Collaborative Online Module (GCOM). At the same time, this tool allows the sharing and utilization of the glossary. A methodology to implement the GCOM is defined: i) GCOM requirements definition: data, safety and interfaces, ii) Existing Glossary based tools analysis and iii) Selection of the solution to be implemented.

2.4 4th Stage: Glossary Extension and Sharing

The last stage of the methodology comprises the extension and validation of the glossary by means of the GCOM. The semi-automatic glossary acquisition procedure generates a set of interrelated terms inside a domain starting from a group of documents of that same domain. Although this procedure generates an important number of definitions, it is common that some terms belonging to the research domain may be excluded, either because they don't appear in enough number in the evaluated documents or because they have appeared in later dates to the development of the stage 2. Based on this, the project researchers must extend the glossary terms to complete the final version of the glossary. Likewise, the project researchers must unify their approaches regarding the generated definitions. Therefore, this stage consists on a combined process of sharing-extension-validation using the newest ICT and developed by all the researchers of the project. The stage may be split up in:

- **Step 1. Glossary sharing:** The glossary must be uploaded in the GCOM in order to share the definitions between the research community.
- **Step 2. Glossary extension:** Then, the project researchers will extend the glossary with new definitions.
- **Step 3. Glossary validation**: The project researchers must check each term and definition in terms of clarity and coherency.

3 Application of the Methodology

3.1 Introduction

INTEROP is a Network of Excellence (NoE) whose primary goal is to sustain European research on interoperability for enterprise applications and software. The originality of the project lies in the multidisciplinary approach merging different research areas which support the development of interoperable systems: architectures and platforms, enterprise modelling, and ontology.

Since the use of WordNet[4] as a reference ontology is not a current choice of the INTEROP project, and since for the glossary acquisition task some additional feature was foreseen, we conceived a partly new experiment, using some of the tools and algorithms provided by the OntoLearn system [2] [3], and some new feature that we developed for the purpose of the task at hand. In this chapter the methodology and the results of this experiment are described, that led to the acquisition of a hierarchically structured glossary of about 380 interoperability terms, subsequently evaluated by a team of 6 domain experts.

3.2 1st Stage: Purpose of the Glossary for the INTEROP NoE

Semantic unification is needed in order to facilitate the existing knowledge exchange within the NoE. The creation of a glossary is then a critical task for the project. Several INTEROP working groups have ascertained the need of identifying a glossary of interoperability terms for a variety of tasks, e.g.:

- Building a knowledge map and classifying knowledge domains.
- Classifying partner's competences for the INTEROP mobility matrix.
- Providing a list of relevant domain concepts for educational objectives.
- More in general, indexing with a set of common meta-data the various deliverables, state of art, scientific papers and databases.

Finally, the glossary will be used as main information source to build an Ontology on Interoperability. This ontology will allow structuring the knowledge all over the NoE, facilitating the information retrieval and clustering on the collaborative platform.

3.3 2nd Stage: Application of the Semi-automatic Glossary Acquisition Procedure on the Interoperability Domain. The INTEROP Experiment

For the purpose of the INTEROP glossary acquisition task, step 6 of the proposed methodology has been omitted, since an interoperability Core Ontology was not available, and the adoption of an available reference ontology (like WordNet) is not agreed in the project. The preliminary objective in this phase of INTEROP was to obtain a sort of partially structured glossary, rather than an ontology, i.e. a forest

[4] http://www.cogsci.princeton.edu/~wn/

of term trees, where, for each term, the following information has to be provided: *definition* of the term, *source* of the definition, *kind-of* relation.

Step 1. Term extraction: The first step of the INTEROP glossary procedure was to derive an initial list of terms using the evidence provided by interoperability-related documents. The INTEROP collaborative workspace was used to collect from all partners the relevant documents, among which, the proceedings of INTEROP workshops and deliverables, highly referenced scientific papers, partners' papers, tutorials, etc. The OntoLearn TermExtractor module [3] extracted from these documents 517 terms. A generic computer science glossary was used to remove overly general technical terms and the list was then quickly reviewed manually to delete clearly identifiable spurious terms. The final list included 376 terms.

Step 2. Generation of definitions: Given the list of terms, we activated step 2 of the automatic glossary acquisition procedure. During this step, 28 definitions were not found, 22 were generated compositionally, and the remaining terms were extracted either from glossaries or from available documents. For each definition, we kept track of the source (URL of the web page). For some term, more than one definition survived the well-formedness and domain similarity criteria (step 3.1 of the OntoLearn semi-automatic procedure), therefore the total number of definitions submitted to the experts for revision was 358.

Step 3. Evaluation by experts: Six domain experts[5] in INTEROP were asked to review and refine the glossary. Each expert could review (rev), reject (rej), accept (ok) or ignore (blank) a definition, acting on a shared database. The experts added new definitions for brand-new terms, but they also added new definitions for terms that may have more than one sense in the domain. There have been a total of 67 added definitions, 33 substantial reviews, and 26 small reviews (only few words changed or added). Some term (especially the more generic ones, e.g. business domain, agent, data model) was reviewed by more than one expert who proposed different judgements (e.g. ok and rev) or different revised definitions. In order to harmonise the results, a first pass was conducted automatically, according to the following strategy: If a judgement is shared by the majority of voters, then select that judgement and ignore the others (e.g. if a definition receives two ok and one rev, then, ignore rev and accept the definition as it is). If the only judgement is rej(ect), then delete the definition. If a definition has a rej and one (or more) reviewed versions, then, ignore the reject and keep the reviews. This step led to a final glossary including 425 definitions, 23 of which with a surviving ambiguity that could not be automatically conciliated. Therefore a second, short manual pass was necessary, involving this time only three reviewers. After resolving ambiguity, one definition (the most representative) per term was selected. Final glossary has 283 terms and definitions.

Step 4. Speed-up factors: The objective of the methodology is to speed-up the task of building a glossary by a team of experts. Evaluating whether this objective has been met is difficult, since no studies are available for a comparison. We consulted several sources, finally obtaining the opinion of a very experienced

[5] The experts have been chosen according to their expertise in the three INTEROP domains.

professional lexicographer[6] who has worked for many important publishers. He outlined a three-steps procedure for glossary acquisition including: i) internet search of terms, ii) production of definitions, and iii) harmonization of definitions style. He evaluated the average time spent in each step in terms of 6 minutes, 10 min. and 6 min. per definition, respectively. He also pointed out that conducting this process with a team of experts could be rather risky in terms of time[7], however he admits that in very new fields the support of experts could be necessary. Though the comparison is not fully possible, the procedure described in this paper has three phases in which man-power is requested:

After term extraction (step 1), to prune non-terminological and non-domain relevant strings. This requires 0.5 minutes per term. After the extraction of definitions (step 2), to evaluate and refine definitions. We asked each expert to declare the time spent on this task, and we came out with an average of 4 minutes per definition. Since some definition was examined by more than one expert, this amount must be increased to 6 min. approximately. In a second-pass review, to agree on the conflicting judgements. This depends on the number of conflicts, that in our case was less than 10%, mostly solved automatically (section 3.3). Overestimating, we may still add 1 minute per definition. The total time is then 7.5 minutes per definition, against the 16 declared by the lexicographer for steps 1 and 2 of his procedure. In this comparison we exclude the stylistic harmonisation (step 3 of the lexicographer), which is indeed necessary to obtain a good quality glossary, but has not been conducted in the case of the INTEROP experiments. However, since this phase would be necessarily manual in both cases, it does not influence the computation of the speed-up factor. The above evaluation is admittedly very questionable, because on one side we have an experienced lexicographer, on the other side we have a team of people that are certainly experts of a very specific domain, but have no lexicographic skills. Our intention here was only to provide a very rough estimate of the manpower involved, given that no better data are available in literature. Apparently, a significant speed-up is indeed obtained by our procedure.

Generation of domain sub-trees. As remarked in the introduction, the glossary terms must have some kind of hierarchical ordering, leading eventually to a formal ontology. A hierarchical structure simplifies the task of document annotation, and is a basis for further developments such as automatic clustering of data (e.g. for document classification), identification of similarities (e.g. for researchers mobility), etc. In other words, it is a first step towards semantic annotation. To arrange terms in term trees, we used the procedure described in steps 3.2 and 4.1.1 of the OntoLearn semi-automatic procedure. The definitions have been parsed and the word, or complex term, representing the hyperonym (genus) has been identified. Given the limited number of definitions, we verified this task manually, obtaining a figure of 91,76 % precision, in line with previous evaluations that we did on other domains (computer networks, tourism, economy). Contrary to the

[6] We thank Orin Hargraves for his very valuable comments.

[7] To cite his words: "no commercial publisher would subject definitions to a committee for fear of never seeing them in recognizable form again"

standard OntoLearn algorithm, we did not attached sub-trees to WordNet, as motivated in previous sections. Overall, the definitions were grouped in 125 sub-trees, of which 39 including only 2 nodes, 43 with 3 nodes, and the other with >3 nodes. Examples of two term trees are shown:

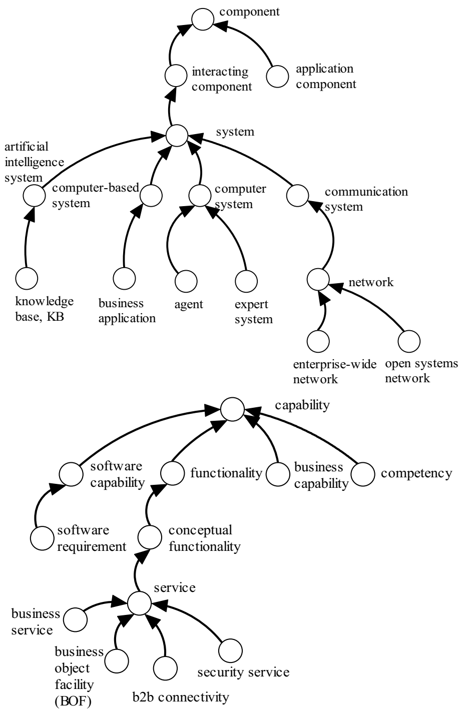


Fig 2. Sub-trees extracted from the Interoperability domain

In Figure 2, the collocation of the term *system* might seem inappropriate, since this term has a very generic meaning. However, the definition of *system* in the interoperability glossary is quite specific: "*a set of interacting components for achieving common objectives*", which justifies its collocation in the tree. A similar consideration applies to *service* in the bottom tree. An interesting paper [1] provides an analysis of typical problems found when attempting to extract (manually or automatically) hyperonymy relations from natural language definitions, e.g. attachments too high in the hierarchy, unclear choices for more general terms, or-conjoined heads, absence of hyperonym, circularity, etc. These problems are more or less evident – especially over-generality – when analysing the term trees forest extracted from the glossary. However, our purpose here is not to overcome problems that are inherent with the task of building a domain concept hierarchy: rather, we wish to automatically *extract, with high precision, hyperonymy relations* embedded in glossary definitions, just as they are: possibly over-general, circular, or-conjoined. The target is, again, to speed up the task of ontology building and population, extracting and formalizing domain knowledge expressed by human specialists in an unstructured way. Discrepancies and inconsistencies can be corrected later by the human specialists, who will verify and rearrange the nodes of the forest.

Conclusion. As already remarked, the glossary provides a first set of shared terms to be used as metadata for annotating documents and data in the INTEROP platform. Several features/improvements are foreseen to improve this initial result, both on the interface/architecture and the methodological side. For example, annotation tools must be defined and integrated in the INTEROP platform. The taxonomic structuring of the glossary must be manually reviewed in the light of a core ontology to be defined, and methods to include new terms must be provided. Finally, the use of terms for document access, clustering and retrieval must be implemented and evaluated.

3.4 3rd Stage: Setting up the INTEROP Glossary Web Module

Extending, accessing and using the glossary are collaborative and sharing activities that need to be supported by specific tools. Furthermore, the spreading of the glossary requires tools that can have a wide access by the research community. Currently, web environments have proved to be a suitable solution to address interaction based applications between several actors in different locations. The main features of web environments are the use of standard interfaces, the ease of implementation, the Worldwide access and the advanced interaction capabilities. These features provide the required functionality in order to allow a controlled and validated development of the future INTEROP Glossary. A methodology has been defined in order to facilitate the definition of the technical and operational specifications of the Glossary Module. Furthermore, this methodology is also aimed to select the software to support the INTEROP Glossary Web Module (GWM in what follows). The stages of this methodology are:

- **Stage 1. INTEROP GWM requirements**: A set of requirements will be defined related to the data and graphical user interface specifications.
- Stage 2. Analysis of existing Glossary Web based modules.
- **Stage 3. Selection of the solution**: Based on the previous analysis, the software to support the INTEROP GWM will be selected.

Stage 1 was done based on the requirements defined by the project researchers. Concerning to stages 2 and 3 some decisions were taken. Based on the general specifications, the Glossary module must be integrated within the INTEROP collaborative platform (PLONE-based system[8]) in order to take profit of the benefits of the mutual existence of a glossary and a set of resources (documents, papers, etc.). A search of glossary tools in the market has been performed. There exist some glossary building tools and some OpenSource e-learning platforms that provide glossary facilities, but none of them are integrated in PLONE. These solutions are discarded. Furthermore, there does not exist any commercial software based on PLONE to support the building of a glossary. This fact leads to consider

[8] http://www.plone.org

the possibility to develop by an external company an *ad hoc* software on PLONE to support the Glossary extension and spreading.

3.5 4th Stage: Extending the INTEROP Glossary

The INTEROP Project has foreseen 5 tasks to be developed in the next year:

- **Glossary structuring:** Currently, the glossary is a "flat" list of terms and textual definitions. This flat structure may be inadequate for many tasks. A first activity is to structure the terms in taxonomic order. Taxonomic structuring of keywords is a first step towards concept-based search.
- **Glossary extension and updating:** In this sub-task the procedures and software tools for updating and extending the glossary will be defined.
- **Glossary Usages:** New usages must be specified in the working groups.
- **Implementation of defined glossary-based applications.**
- **Evaluation and assessment:** Finally, an evaluation will be carried out to check the consistency of the tasks developed.

4 Conclusions

A 4-stage methodology to create a glossary in a collaborative research project has been defined. The new proposed methodology is based on an adaptation of a methodology of the University of Rome for the semiautomatic acquisition of terms and definitions starting from a source of documents related to the research areas of the collaborative project. Based on this methodology, a first glossary of terms related to the interoperability domain has been obtained inside the IST-508011 INTEROP Network of Excellence. The requirements of a web tool to share and to permanently enlarge the INTEROP glossary have been defined. An analysis of the existing glossary tools has been carried out. As conclusion, an *ad hoc* tool will be developed.

References

1. Ide N. and Véronis J. (1993). Refining Taxonomies extracted from machine readable Dictionaries, In Hockey, S., Ide, N. *Research in Humanities Computing*, 2, Oxford Univ. Press.
2. Navigli R., Velardi P and Gangemi A. (2003). Ontology Learning and its Application to Automated Terminology Translation. *IEEE Intelligent Systems*, vol. 18, pp. 22-31
3. Navigli R. and Velardi P. (2004). Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. *Comp. Linguistics*, vol. 50 (2).
4. Navigli R. and Velardi P. (2004b). Structural Semantic Interconnection: a knowledge-based approach to Word Sense Disambiguation,Proc. *3rdWorkshop on Sense Evaluation*
5. Navigli R., Velardi P., Cucchiarelli A. and Neri F. (2004). Quantitative and Qualitative Evaluation of the OntoLearn Ontology Learning System. *Proc. 20th COLING 2004*, Geneva.

Ontology-based Semantic Interoperability Tools for Service Dynamic Discovery *

Devis Bianchini and Valeria De Antonellis

Università di Brescia, Dip. di Elettronica per l'Automazione, Via Branze, 38, 25123 Brescia - Italy, {bianchin|deantone}@ing.unibs.it

Summary. Enterprises and networked enterprises need today effective communication and exchange of distributed data and services under dynamic and context-dependent requirements. Semantic interoperability is considered a key issue to enforce dynamic discovery and composition of distributed data and services. In this paper we specifically address the problem of service discovery and we present an ontology-based approach for dynamic discovery in a highly variable environment where cooperative enterprises interoperate to dynamically combine available services selecting the best possible offers in a given moment. The ontology structure and its deployment are discussed.

1 Introduction

Enterprises and networked enterprises require advanced semantic interoperability methods and tools to enable cooperation and communication at application level. In particular, semantic interoperability techniques are being proposed to support data and service discovery and sharing [13]. Current approaches for service discovery address the treatment of dynamical aspects both with respect to the continuous addition and removal of services in a highly variable environment and with respect to different contexts in which a service could be invoked [4, 6]. The possibility of using Description Logics to implement matching algorithms and reasoning procedures to enhance service discovery has been proposed in [7, 12]. Ontologies are considered as an enabling technology for the Semantic Web and methods and tools for ontology definition are being studied for interoperability purposes. The use of ontology during service search allows for scalability of the systems when a large number of services is considered. In [1] a service retrieval approach based on the use of ontologies is presented. In [14] a service ontology specifies domain concepts with a set of synonyms to allow a flexible search and a set of service classes to define the

* This work has been partially supported by the MAIS (Multichannel Adaptive Information Systems [10]) FIRB Project funded by the Italian Ministry of Education, University and Research, and by NoE INTEROP [9] IST Project n. 508011 - 6th EU Framework Program.

properties of services, its attributes and operations. The service ontology also supports a service quality model that is used to describe non functional aspects. In [3] a new technique for Web service discovery which features a flexible matchmaking by exploiting DAML-S ontologies is proposed.

In this paper we specifically address the problem of service discovery and we present an ontology-based approach for dynamic discovery in a highly variable environment where cooperative enterprises interoperate to dynamically combine available services selecting the best possible offers in a given moment. With respect to existing approaches to ontology-based service discovery, original contribution of our approach regards: (i) the ability of abstracting service characteristics from their operating environment to be able to dynamically select services on the basis of contextual features; in fact, our ontology has a three-layer architecture with different abstraction levels properly exploited to scale service discovery; (ii) the possibility of selecting services through a deductive matching algorithm taking into account semantic relationships among services. In the approach, an ontology-based service description is used as a basis for service retrieval in an extended UDDI Registry. Instead of associating semantic information directly to services, semantic information is extracted from published services on the basis of a domain ontology and a service ontology.

The paper is organized as follows: in Section 2 we propose a three-layer service ontology architecture; in Section 3 we present a logical framework for service ontology representation; in Section 4 service discovery based on the proposed ontological framework is discussed and in Section 5 the architecture underlying our work is presented. Finally, conclusions are discussed in Section 6.

2 The Service Ontology Model

The proposed service ontology architecture organizes services at different layers of abstraction, according to proper semantic relationships [2], that are exploited to improve traditional service discovery mechanisms (mainly keyword-based) with more sophisticated retrieval modalities based on reasoning services, as explained in Section 4.

The service ontology contains *concrete services*, *abstract services* and *subject categories*, organized into three layers of increasing abstraction (called *Concrete*, *Abstract* and *Category* layer, respectively).

Concrete services are directly invocable services and they are featured by their public WSDL interfaces, that define the functional description of the services, that is, the names of provided operations and of input/output parameters for each operation. Concrete services are registered into UDDI Registry, where they are associated to one or more *concrete bindings* (for example, SOAP or HTTP binding) and one or more *endpoints* (that is, the physical localization of the concrete services). UDDI Registry offers searching utilities that are mainly keyword-based; the aim of our work is to maintain backward compatibility with these existing technologies and their searching functionalities and in the meantime to improve these functionalities. Suitable descriptors are defined for concrete services to cluster them on the basis of their functional similarity evaluated by means of properly defined coefficients; each cluster is associated to an abstract service [5].

Abstract services are not directly invocable services, but represent the functionalities of sets of similar concrete services; their description is obtained from the concrete service descriptors by means of an integration process; mapping rules are maintained among the abstract operations and I/O parameters and the original concrete counterparts. Moreover, abstract services are related to each other by two kinds of semantic relationships: (i) *specialization/generalization*, when an abstract service offers at least the same functionalities of another one; (ii) *composition*, when the functionalities offered by a single abstract service can be provided also by a group of other abstract services, considered in their totality; in this case, the first service is often called the *composite service*, while the other ones are called the *component services*. Abstract services are intended to shorten the way towards a variety of alternative concrete services that can be invoked.

Subject categories organize abstract services into standard available taxonomies (such as UNSPSC or NAICS) to provide a mechanism for an easy access to the underlying levels on the basis of standard topics.

The three-layer service ontology is intended to enhance finding of generic services (*abstract services*) describing the required capabilities that can be actually provided by several specific existing services (*concrete services*). The elements used for service functional descriptions (operation names, input/output parameter names) can be related to concepts in a *domain ontology*, where they are semantically defined and organized by means of traditional semantic relationships (*generalization/specialization*, *equivalence*, *disjunction* and *instance-of*)). Figure 1 shows a portion of three-layer service ontology in the tourism domain. Note that the `ReserveAirTravel` abstract service is composed of `ReserveFlight` and `ReserveHotel` ones and that `ReserveFlight` service is specialized by the `ReserveLowCostFlight` abstract service. Moreover, operations inherited from the more general or the composite services are not repeated in the specialized or component services.

3 A Description Logic for Representing Domain and Service Ontologies

In order to enhance semantic interoperability, Description Logics have been adopted for representing services in the ontology and for reasoning in service discovery. We consider the $\mathcal{SHOIN}(\mathcal{D})$ Description Logic, that is the logical foundation of OWL-DL, a sub-language of OWL.

Basic elements in Description Logic formalism are a set of concept names, a set of individual names and a set of role names; a concept C can be defined recursively as follows:

- a concept name A is a concept (called *atomic concept*);
- an enumeration of individuals $\{i_1, i_2, \ldots i_n\}$ is a concept;

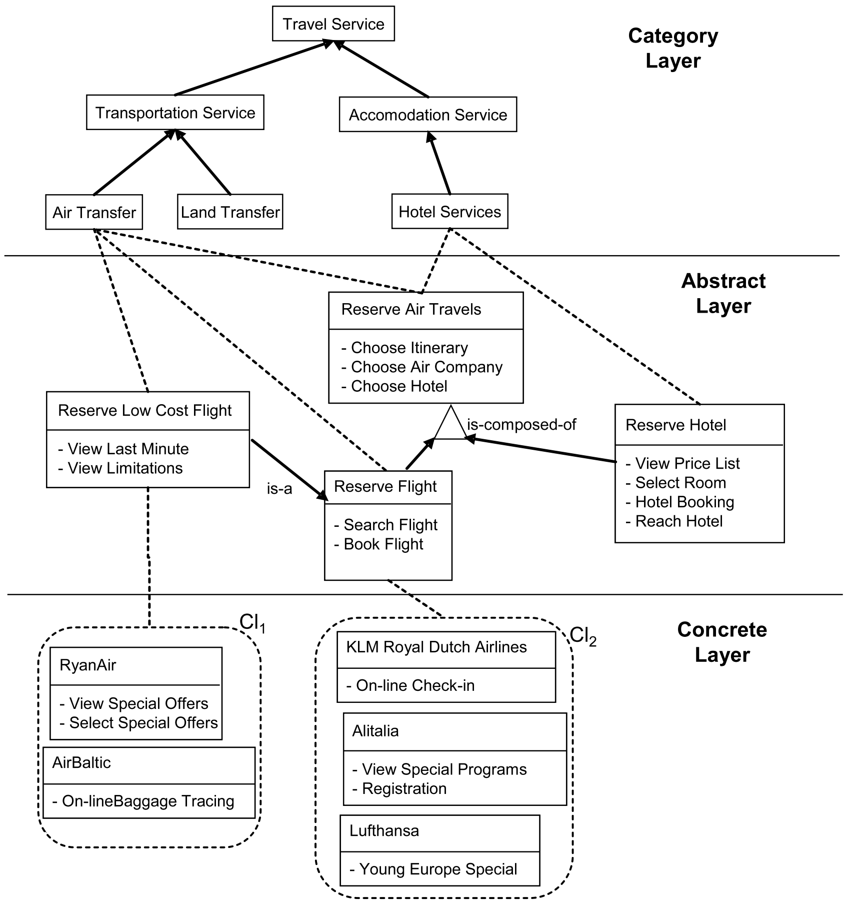


Fig. 1. A portion of three-layer service ontology in tourism domain.

- given two concepts C_1 and C_2, $C_1 \sqcap C_2$ (*intersection*), $C_1 \sqcup C_2$ (*union*), $\neg C$ (*negation*) and (C) are concepts;
- $\exists R.C$ (*existential role restriction*) and $\forall R.C$ (*universal role restriction*) are concepts.

Moreover, the *universal* $(\top)$ and *empty concept* $(\bot)$ are defined. The number and kinds of admitted constructs define the Description Logic family and its expressiveness. In $\mathcal{SHOIN}(\mathcal{D})$ it is possible to define:

- *concept equivalence* $(C_1 \equiv C_2)$;
- *inclusion* $(C_1 \sqsubseteq C_2)$ and *disjointness* $(C_1 \sqsubseteq \neg C_2)$ between concepts;
- the inverse role R^-;
- transitivity of the role R;
- hierarchies of roles $(R_1 \sqsubseteq R_2)$;

- number restrictions, that is, $\leq nR$ (at-most cardinality constraint), $\geq nR$ (at-least cardinality constraint) and $= nR$ (exactly cardinality constraint).

The semantics of Description Logics is defined by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \bullet^{\mathcal{I}})$, consisting of a *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\bullet^{\mathcal{I}}$; given an atomic concept A, a role name R with arity n, a set of individuals $\{i_1, i_2, \ldots i_n\}$, two generic concepts C_1 and C_2, we have:

$$\begin{aligned}
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
R^{\mathcal{I}} &\subseteq (\Delta^{\mathcal{I}})^n \\
i^{\mathcal{I}} &\in \Delta^{\mathcal{I}} \\
(\{i_1, i_2, \ldots i_n\})^{\mathcal{I}} &= \{(i_1)^{\mathcal{I}}, (i_2)^{\mathcal{I}}, \ldots (i_n)^{\mathcal{I}}\} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} &= (C_1)^{\mathcal{I}} \cup (C_2)^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} - C^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{c \in \Delta^{\mathcal{I}} \mid \exists d \in \Delta^{\mathcal{I}} s.t. (c, d) \in R^{\mathcal{I}} \wedge d \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{c \in \Delta^{\mathcal{I}} \mid \forall d \in \Delta^{\mathcal{I}} s.t. (c, d) \in R^{\mathcal{I}} \Rightarrow d \in C^{\mathcal{I}}\} \\
(\leq nR)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \; s.t. \; (x, y) \in R^{\mathcal{I}}\}) \leq n\} \\
(\geq nR)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \; s.t. \; (x, y) \in R^{\mathcal{I}}\}) \geq n\} \\
(= nR)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \#(\{y \; s.t. \; (x, y) \in R^{\mathcal{I}}\}) = n\} \\
(R^{-})^{\mathcal{I}} &= (R^{\mathcal{I}})^{-} \\
(Tr(R))^{\mathcal{I}} &= (R^{\mathcal{I}})^{+}
\end{aligned}$$

where $\#()$ denotes the set cardinality. The empty concept is mapped to the empty set, while the universal concept is mapped to $\Delta^{\mathcal{I}}$. An assertion $C_1 \equiv C_2$ is satisfied by an interpretation $\mathcal{I}$ if $(C_1)^{\mathcal{I}} = (C_2)^{\mathcal{I}}$; an assertion $C_1 \sqsubseteq C_2$ is satisfied by $\mathcal{I}$ if $(C_1)^{\mathcal{I}} \subseteq (C_2)^{\mathcal{I}}$; an assertion $C_1 \sqsubseteq \neg C_2$ is satisfied by $\mathcal{I}$ if $(C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}} = \emptyset$.

An interpretation $\mathcal{I}$ that satisfies all the assertions in a knowledge base $\mathcal{KB}$ is called a *model* for $\mathcal{KB}$; a generic concept C can be satisfied in $\mathcal{KB}$ if $\mathcal{KB}$ allows for a model $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$; $\mathcal{KB}$ *logically implies* an assertion between concepts if, for each model $\mathcal{I}$ of $\mathcal{KB}$, the application of $\mathcal{I}$ to the assertion can be satisfied. In our approach, the knowledge base is constituted by a domain ontology $\mathcal{D}om\mathcal{ONT}$, that is a set of assertions on concepts of the domain to which the services refer and a Service Ontology $\mathcal{S}erv\mathcal{ONT}$, that is a set of assertions on service elements and semantic relationships between them, as explained in the following.

3.1 Service Description

We consider the chosen Description Logic to represent the functional description of abstract and concrete services. To do this, we do not exploit the full expressiveness of $\mathcal{SHOIN}(\mathcal{D})$, but we make a restricted use of its constructs. We consider subsumption, conjunction, disjunction and negation of atomic and complex concepts and existential restriction; in particular, we use two specific roles: the role `hasCategory` to express the association link of a service with a subject category and the role `hasOperation` to express the relationship of a service with its operations or capabilities.

In our approach, a service is semantically described by means of a category, representing the area of interest of the service, and the service capability in terms of provided operations. Formally, the description is given by a conjunction of:

- a concept in the form $\exists\texttt{hasCategory}.CAT$, where CAT is a concept which represents the associated service category;
- one or more concepts in the form $\exists\texttt{hasOperation}.OP$, where OP is a concept representing an operation of the current service; each operation OP is described as a conjunction of:
  - the operation name, expressed by means of an atomic concept;
  - a conjunction of one or more concepts IN, where IN is a concept representing an input parameter;
  - a conjunction of one or more concepts OUT, where OUT is a concept representing an output parameter.

IN and OUT are specified in the form $\exists R.C$, where R represents the name of the parameter and C is a concept representing possible parameter values. C can be defined as an atomic concept, an enumeration $\{i_1, i_2, \ldots i_n\}$ of individuals or a complex concept obtained by applying the *intersection*, *union* and *negation* operator. We represent a functional request $\mathcal{R}$ as well as the functional description of a service interface.

Example - If we consider the abstract service `ReserveFlight` in Figure 1, the following new concept is added to the $\mathcal{S}erv\mathcal{ONT}$:

$$\begin{array}{l}
\texttt{ReserveFlight} \sqsubseteq \exists\texttt{hasCategory.AirTransfer} \sqcap \exists\texttt{hasOperation.(searchFlight} \sqcap \\
\quad \exists\texttt{departureCity.City} \sqcap \exists\texttt{arrivalCity.City} \sqcap \exists\texttt{departureTime.Date} \\
\quad\quad \sqcap \exists\texttt{arrivalTime.Date} \sqcap \exists\texttt{ticket.flightTicket)} \sqcap \\
\quad \exists\texttt{hasOperation.(bookFlight} \sqcap \exists\texttt{ticket.flightTicket} \sqcap \\
\quad\quad \exists\texttt{bookingConfirmation.flightReservation)}
\end{array}$$

□

4 An ontological Infrastructure for Service Discovery

In this section we show how to exploit the proposed ontological infrastructure to enhance service discovery by means of a matching and ranking algorithm that finds desired concrete services according to a set of functionalities required by the user (that is, a request $\mathcal{R}$) and ranks them with respect to their kind of match w.r.t. $\mathcal{R}$. As in [8], we consider five kinds of match, that can be intuitively described as follows:

- *exact match*, when the request and the offer present the same functionalities (this is a strong condition);
- *plug-in match*, when the offer provides at least the required functionalities and possibly adds new ones;
- *subsume match*, when the functionalities provided by the offer are less than ones provided by the offer (it is like *plug-in match*, but with the roles of $\mathcal{R}$ and $\mathcal{S}$ exchanged);

- *intersection match*, when the request and the offer present some common functionalities;
- *mismatch*, when no common functionalities exist between the request and the offer.

Note that, from the request viewpoint, the first two kinds of match can be considered equivalent, since in both cases the offer fulfills the request; in the case of *subsume* and *intersection match*, otherwise, the offer satisfies only partially the request. To verify the five kinds of match listed above, we separate service description components by considering service categories, operation names and input/output parameter names for each operation and we verify the satisfiability w.r.t. $\mathcal{D}om\mathcal{ONT}$. Firstly, we consider if the service categories CAT_R of the request and CAT_S of the offer are related in any generalization hierarchy, that is

$$\mathcal{D}om\mathcal{ONT} \models CAT_R \sqsubseteq CAT_S$$

that is, $CAT_R \sqsubseteq CAT_S$ is true in $\mathcal{D}om\mathcal{ONT}$. If this is not verified, then the match fails (*mismatch*), otherwise the other kinds of match are investigated.

Exact match.

It occurs when, for each operation OP_{iR} there exists a corresponding operation OP_{jS} such that

- $\mathcal{D}om\mathcal{ONT} \models OP_{iR}.name \equiv OP_{jS}.name$;
- for each output $OP_{iR}.OUT_h$ there exists a corresponding output $OP_{jS}.OUT_q$ such that $\mathcal{D}om\mathcal{ONT} \models OP_{iR}.OUT_h \equiv OP_{jS}.OUT_q$;
- for each input $OP_{jS}.IN_p$ there exists a corresponding input $OP_{iR}.IN_k$ such that $\mathcal{D}om\mathcal{ONT} \models OP_{jS}.IN_p \equiv OP_{iR}.IN_k$.

Each operation of $\mathcal{R}$ is compared with each operation of $\mathcal{S}$. Note that in the matching process (for each kind of match) we require that for each comparison between two operations (between corresponding parameters of two operations) when a kind of match is established for a pair of corresponding operations (corresponding parameters) such operations (parameters) do not participate in further comparisons.

Plug-in match.

It occurs when, for each operation OP_{iR}, there exists a corresponding operation OP_{jS} such that

- $\mathcal{D}om\mathcal{ONT} \models OP_{iR}.name \sqsubseteq OP_{jS}.name$;
- for each output $OP_{iR}.OUT_h$ there exists a corresponding output $OP_{jS}.OUT_q$ such that $\mathcal{D}om\mathcal{ONT} \models OP_{iR}.OUT_h \sqsubseteq OP_{jS}.OUT_q$;
- for each input $OP_{jS}.IN_p$ there exists a corresponding input $OP_{iR}.IN_k$ such that $\mathcal{D}om\mathcal{ONT} \models OP_{jS}.IN_p \sqsubseteq OP_{iR}.IN_k$.

The *subsume match* is verified in the same way, with the roles of $\mathcal{R}$ and $\mathcal{S}$ exchanged.

Intersection match.

If neither *exact* or *plug-in* or *subsume match* occurs, but there exist pairs of operations OP_{i_R} and OP_{j_S} with the following conditions verified

- $\mathcal{D}om\mathcal{ONT} \models \neg(OP_{i_R}.name \sqcap OP_{j_S}.name \sqsubseteq \bot)$;
- for at least one output $OP_{i_R}.OUT_h$ there exists a corresponding output $OP_{j_S}.OUT_q$ such that $\mathcal{D}om\mathcal{ONT} \models \neg(OP_{i_R}.OUT_h \sqcap OP_{j_S}.OUT_q \sqsubseteq \bot)$;
- for at least one input $OP_{j_S}.IN_p$ there exists a corresponding input $OP_{i_R}.IN_k$ such that $\mathcal{D}om\mathcal{ONT} \models \neg(OP_{j_S}.IN_p \sqcap OP_{i_R}.IN_k \sqsubseteq \bot)$;

then *intersection match* is recognized between $\mathcal{R}$ and $\mathcal{S}$.

If all the previous comparisons fail, then the match fails (*mismatch*). A qualitative ranking among the considered kinds of match can be defined, that is, `exact` $>$ `plug-in` $>$ `subsume` $>$ `intersection` $>$ `mismatch`. To verify these kinds of match, an automatic reasoner based on Description Logics is used (Racer [11]).

Semantic relationships between concrete services, abstract service and subject categories in the Service Ontology can be exploited to make more efficient the service discovery procedure. The matching algorithm is applied at the category and abstract layers and can effectively exploit the semantic relationships between abstract services, according to the following intuition: if an abstract service $\mathcal{S}_a$ matches with a given service request $\mathcal{R}$, then also abstract services that provide the same capabilities of $\mathcal{S}_a$ (as expressed by means of semantic relationships) match with $\mathcal{R}$. According to this intuition, the following rules are applied:

- if $\mathcal{S}_{ai}$ has an *exact* or a *plug-in match* with $\mathcal{R}$ and $\mathcal{S}_{ai}$ is a generalization of another abstract service $\mathcal{S}_{aj}$, then also $\mathcal{S}_{aj}$ presents a *plug-in match* with $\mathcal{R}$ and the application of matching algorithm to $\mathcal{S}_{aj}$ is not required;
- if $\mathcal{S}_{ai}$ has a *mismatch* with $\mathcal{R}$ and another abstract service $\mathcal{S}_{aj}$ is a generalization of $\mathcal{S}_{ai}$, then we can say that also $\mathcal{S}_{aj}$ presents a *mismatch* with $\mathcal{R}$;
- the same procedure applies when $\mathcal{S}_{ai}$ is a composite service and $\mathcal{S}_{aj}$ is the union of its component ones.

Finally, once abstract services that match with the request are found, then concrete services belonging to the corresponding clusters are included into the searching results, by setting the kind of match of each concrete service w.r.t. $\mathcal{R}$ equal to that of the corresponding abstract service.

Example - We consider a request $\mathcal{R}$ of a flight booking service for a trip from Milan or Venice to Rome, represented as follows:

$$\mathcal{R} \sqsubseteq \exists\texttt{hasCategory.AirTransfer} \sqcap \exists\texttt{hasOperation.(findFlight} \sqcap \exists\texttt{departureCity.}\{\texttt{Milain,Venice}\} \sqcap \exists\texttt{arrivalCity.}\{\texttt{Rome}\} \sqcap \exists\texttt{ticket.flightTicket)}$$

Moreover, we suppose that $\mathcal{D}om\mathcal{ONT}$ contains the following knowledge about the tourism domain: {`City(Milan)`, `City(Venice)`, `City(Rome)`, `findFlight` $\equiv$ `searchFlight`, `bookFlight` $\equiv$ `FlightReservation`}. We can assert that the kind of match between `ReserveFlight` and $\mathcal{R}$ is `'Plug-in'`, since the category is the same, $\mathcal{D}om\mathcal{ONT} \models$ `(findFlight` $\equiv$ `searchFlight)` and for each parameter of $\mathcal{R}$ there exists a corresponding one that is more general in the advertised abstract service, as satisfied by $\mathcal{D}om\mathcal{ONT}$. Finally, since `ReserveLowCostFlight` is a specialization of `ReserveFlight`, then also the kind of match between `ReserveLowCostFlight` and $\mathcal{R}$ is `'Plug-in'` and all the concrete services shown in Figure 1 are returned to the user. □

5 The Ontology-based COMPAT System Architecture

Figure 2 shows the architecture of the COMPATibility ontology-based system to enhance service discovery and publication. Domain and Service ontologies are two of the main elements of the COMPAT (COMPATibility) system and they are described by using OWL-DL, whose logical foundation is constituted by $\mathcal{SHOIN}(\mathcal{D})$ Description Logic.

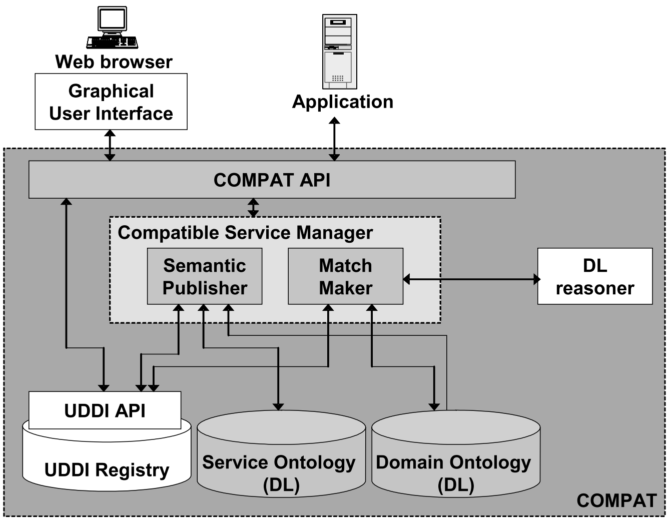


Fig. 2. The COMPAT system architecture.

The discovery and advice tasks are performed by the `Compatible Service Manager` in charge of organizing services in the ontology (`Semantic Publisher`) and of searching services by exploiting underlying ontologies (`Matchmaker`). The `Semantic Publisher` is in charge of extracting from the UDDI Registry a new registered concrete service and of placing it into the Service Ontology; it associates the new service with a suitable cluster and possibly modifies the corresponding abstract service to keep into account the registered service interface; it can also support the ontology designer in finding semantic relationships between abstract services. The `MatchMaker` selects abstract services that match a given service request from the functional perspective, according to the approach in Section 4; then, it selects concrete services associated with the abstract ones, taking into account also non functional aspects like quality requirements, contextual conditions or concrete service availability; finally, it returns to the requestor a list of descriptions of concrete services that match his request, together with links to their implementations (these links are stored in the UDDI Registry). The `Compatible Service Manager` uses the `JENA API` (a Java interface that includes a subsystem for management of ontologies supporting OWL, DAML+OIL and RDF) to access directly the ontologies and an automatic reasoner, RACER [11], that supports reasoning tasks on $\mathcal{SHOIN}(\mathcal{D})$.

External users and applications interact with the system by means of the COMPAT API, that provides the mechanisms to: (i) submit the service request description to the Compatible Service Manager (in particular, to the MatchMaker) in order to discover matching concrete services; (ii) submit a new service description to the Semantic Publisher in order to be registered in the UDDI Registry and placed into the service ontology. Moreover, the COMPAT API allows for using traditional UDDI Registry searching functionalities, ensuring backward compatibility with existing technologies and, in this sense, extending the UDDI API. The COMPAT API can be used by a human user by means of a `Graphical User Interface` implemented with currently existing java-based technologies (such as *Java Servlet* and *Java Server Pages*) or can be programmatically invoked by another application when the registry is used in a wider context, where automated retrieval, composition and invocation of services is required.

The proposed knowledge-based architecture for service discovery and advice system has the capability of providing specific service advice at different levels of granularity. At the highest level, the system can help to determine what kind of abstract service is required against a contextual functional request. Once all the services that can fulfill the required functionalities are discovered, the advice system can recommend an appropriate concrete service, taking into account both problem characteristics and quality considerations.

6 Conclusions

In this paper we have addressed the problem of semantic interoperability in service discovery to support enterprises in dynamically selecting the best possible offers at a given moment. We have proposed a service ontology architecture and a matching algorithm to exploit it in order to find concrete services according to a given service request. We have also proposed a mechanism to rank the resulting set of concrete services in a qualitative way. The proposed approach is being extended by refining the matching and ranking algorithm by means of similarity-based techniques. In this paper we have considered only functional aspects of services, while future work will address also a QoS-based and context-aware selection of concrete services. At the moment, a prototype of the architecture shown in the previous section and the experimentation in the domain of touristic information services are being completed.

References

1. A. Bernstein and M. Klein. Towards high-precision service retrieval. *IEEE Internet Computing*, 8(1):30–36, 2004.
2. D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani. Ontology-based Methodology for *e*-Service discovery. *Accepted for publication on Journal of Information Systems, Special Issue on Semantic Web and Web Services*, 2004.
3. A. Brogi, S. Corfini, and R. Popescu. Flexible Matchmaking of Web Services Using DAML-S Ontologies. In *Proc. Forum of the Second Int. Conference on Service Oriented Computing (ICSOC 2004)*, New York City, NY, USA, November 15-19th 2004.
4. F. Casati, M. Castellanos, U. Dayal, and M. Shan. Probabilistic, Context-Sensitive and Goal-Oriented Service Selection. In *Proc. of the Second Int. Conference on Service Oriented Computing (ICSOC 2004)*, pages 316–321, New York City, NY, USA, November 15-19th 2004.
5. V. De Antonellis, M. Melchiori, B. Pernici, and P. Plebani. A Methodology for *e*-Service Substitutability in a Virtual District Environment. In *Proceedings of the 2003 Conference on Information Systems Engineering (CAiSE 2003)*, Velden, Austria, 2003.
6. I. Elgedawy, Z. Tari, and M. Winikoff. Exact Functional Context Matching for Web Services. In *Proc. of the Second Int. Conference on Service Oriented Computing (ICSOC 2004)*, pages 143–152, New York City, NY, USA, November 15-19th 2004.
7. J. Gonzalez-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. of the KI-2001 Workshop on Applications of Description Logics*, Vienna, Austria, September 2001. http://sunsite.informatik.rwthaachen.de/Publications/CEUR-WS/Vol-44/.
8. I. Horrocks and L. Li. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 331–339, 2003.
9. The INTEROP NoE Portal. http://www.interop-noe.org/.
10. The MAIS Project Home Page. http://www.mais-project.it.
11. The Racer Home Page. http://www.sts.tu-harburg.de/ r.f.moeller/racer/.
12. B. Motik S. Grimm and C. Preist. Variance in e-Business Service Discovery. In *Proc. of the Third International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, November 8th 2004.
13. HP - Web Services concepts. A technical overview. http://www.bluestone.com/downloads/pdf/web_services_tech_overview%w.pdf.
14. L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, 2004.

Identifying Business Components on the Basis of an Enterprise Ontology

Antonia Albani[1], Jan L.G. Dietz[2] and Johannes Maria Zaha[1]

[1] University of Augsburg, Business School, Chair of Business Informatics and Systems Engineering, 86159 Augsburg, Germany
{antonia.albani|johannes.maria.zaha}@wiwi.uni-augsburg.de
[2] Delft University of Technology, Chair of Information Systems
PO Box 5031, 2600 GA Delft, The Netherlands
j.l.g.dietz@ewi.tudelft.nl

Summary. Companies are more and more focusing on their core competencies, outsourcing business tasks to their business partners. In order to support collaboration between business partners, adequate information systems need to be built automating inter-organizational business processes. The bases for such information systems are *business components* combining software artefacts from different vendors to applications which are individual to each customer. The crucial factors in identifying and building reusable, marketable and self-contained business components are the appropriateness and the quality of the underlying business domain models. This paper therefore introduces a process for the identification of business components based on an *enterprise ontology*, being a business domain model satisfying well defined quality criteria.

1 Introduction

The software components of an information system that support directly the activities in an enterprise are usually called *business components* [1, 2]. All other software components are considered either to deliver services to these business components or to offer some general functionality. The identification of business components thus is the first step in the development of an information system according to current standards (component-based, object-oriented etc.). Needless to say that this step is a very crucial one and that it consequently should be performed at the highest possible level of quality. The starting point is the set of requirements that have been elicited from the business domain. Requirements engineering is still a weak link, although considerable progress has been made since it is being based on a business domain model. These models offer a more objective starting point for extracting requirements and a more objective criterion for evaluating them than the traditional 'waiter strategy' [3]. In a very true sense however, this new approach to engineering requirements only shifts the problem to an earlier stage instead of solving it. The crucial factor now is the appropriateness and the quality of the business domain model. In [4] some quality criteria are proposed regarding a business domain model, which we adopt for our current research:

- It should make a clear and well-founded distinction between the *essential* business actions and informational actions. For example, requesting a supplier to deliver articles is essential, but computing the amount of articles is informational (it is no new fact, only the outcome of a computation).
- It should have the right granularity or level of detail. "Right" means in this respect: finding the actions that are *atomic* from the business point of view. They may be composite only in their implementations. For example, the request to a supplier is atomic from the business point of view, but to perform a request by postal mail, a number of non-essential actions have to be taken like mailing the order form, transporting it and delivering it to the supplier.
- It should be *complete*, i.e. it should contain everything that is necessary and it should not contain anything that is irrelevant. As will be shown in the sequel, this requirement is probably the most hard to satisfy since it is common practice in most organizations to perform several kinds of coordination acts tacitly, according to the rule "no news is good news".

We will call a business domain model that satisfies these requirements an *enterprise ontology*. The goal of the research that is reported in this paper is to identify business components on the basis of an enterprise ontology. It builds on previous work regarding enterprise ontology [5] and regarding business components [6, 7]. The outline of the paper is as follows. In section 2, the method is presented that we apply to arrive at a right ontological model of an enterprise and to derive from this model the business components. In section 3, the method is applied to the example of strategic supply network development (SSND), as reported in [8, 9]. On the basis of the ontological model that is the outcome of section 3, we derive the corresponding business components in section 4. Discussions of the findings as well as the conclusions that can be drawn are provided in section 5.

1.2 The Method and the Example Case

A precondition to component based development of application systems by using business components is a stable component model. In order to obtain stable business component models, a well defined identification process is necessary. The basis for the identification of reusable, marketable and self-contained business components is an appropriate and high quality business domain model. Such a model not only serves to satisfy the requirements for a single application system but rather for a family of systems – and therefore for a certain domain. In order to achieve that, we adapted the Business Component Modeling (BCM) [6] process by modeling the business domain using an enterprise ontology as introduced in section 1. An overview of the adapted *Component Based Domain Analysis* phase of the BCM process is given in Figure 1. In this paper we concentrate on the *Domain Scope* and the *Business Components Identification* sub phases and will not describe the *Specification* sub phase.

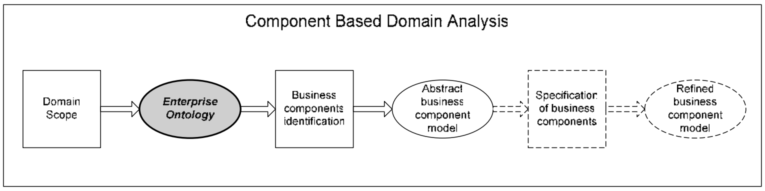

Component Based Domain Analysis
Domain Scope
Enterprise Ontology
Business components identification
Abstract business component model
Specification of business components
Refined business component model

Figure 1. Component Based Analysis Phase of the BCM process

In the *Domain Scope* sub phase, the method that is used for constructing the ontology of an enterprise is taken from DEMO (Design & Engineering Methodology for Organizations) [10-12]. As is explained in [10, 11] a distinction is made between production acts and facts and coordination acts and facts. The transaction axiom puts these acts/facts together in the standard pattern of the (business) transaction. Consequently, two worlds are distinguished: the production world (P-world) and the coordination world (C-world). The complete ontological model of an organization in DEMO consists of four aspect models (Figure 2).

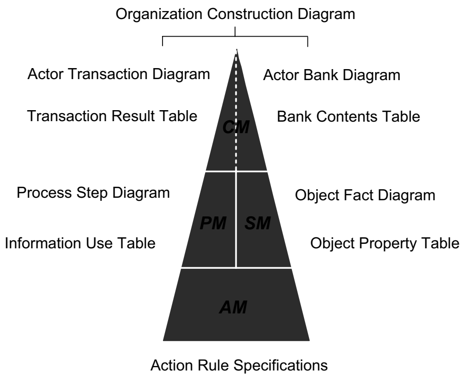


Figure 2. The four aspect models

The Construction Model (CM) specifies the composition, the environment and the structure of the organization: the identified transaction types and the associated actor roles, as well as the information links between the actor roles and production banks or coordination banks. The Process Model (PM) contains for every transaction type in the CM the specific transaction pattern of the transaction type. Next to these patterns, it contains the causal and conditional relationships between transactions. The Action Model (AM) specifies the action rules that serve as guidelines for the actors in dealing with their agenda. It contains one or more action rules for every agendum type. These rules are grouped according to the actor roles that are distinguished. The State Model (SM) specifies the entity types and fact types in the P-world, but only those object classes, fact types and ontological coexistence rules that are contained in the AM.

In Figure 2, the CM triangle is split by a dashed line in a left and a right part. This has got to do with the logical sequence of producing the aspect models. First, the left part of the CM can be made straight away after having applied the elicitation procedure as discussed in [13]. It contains the active influences among actor roles, through their being initiator or executor of a transaction type. The CM is expressed in an Actor Transaction Diagram (ATD) and a Transaction Result Table (TRT). Next, the Process Step Diagram (PSD), which represents a part of the PM, is produced, and after that the AM, which is expressed in Action Rule Specifications (ARS). The action rules are expressed in a pseudo-algorithmic language, by which an optimal balance is achieved between readability and preciseness. Then the SM is produced, expressed in an Object Fact Diagram (OFD) and an Object Property Table (OPT). Next, the right part of the CM is produced. It consists of an Actor Bank Diagram (ABD) and a Bank Contents Table (BCT). Usually the Actor Bank Diagram is drawn as an extension of the Actor Transaction Diagram. Together they constitute the Organization Construction Diagram (OCD). After that we are able to complete the PM with the Information Use Table (IUT).

Having defined the enterprise ontology, the complete information related to the business domain is available in order to identify business components as denoted in the *Business Components Identification* sub phase of the BCM process in Figure 1. In order to optimize the process of identifying high quality, reusable and marketable business components the Business Components Identification (BCI) method is used. BCI is based upon the Business System Planning (BSP) [14] method and has been modified for the field of business components identification. BCI takes as input the object classes and fact types from the SM and the process steps from the PM, obtained from the domain scope phase and summarized in the Create/Use Table (CUT), an extension of the IUT. Using a genetic algorithm a number of possible solutions (component models) are generated in order to select the most suitable solution fitting best to the specified quality factors. One of the most important quality factors concerning component models is the minimal communication between components. The result of the BCI is an abstract business component model with defined dependencies between components.

To illustrate the domain scope and component identification sub phases with their resulting diagrams and models, the BCM process is applied to the domain of strategic supply network development in the next sections. The main tasks in the domain of strategic supply network development derive from the tasks of strategic sourcing. The most evident changes regard the functions with cross-enterprise focus. Purchasing has become a core function in enterprises in the 90ies. Current empiric research shows a significant correlation between the establishment of a strategic purchasing function and the financial success of an enterprise, independent from the industry surveyed [15]. One of the most important factors in this connection is the buyer-supplier-relationship. At many of the surveyed companies, a close cooperation between buyer and supplier in areas such as long-term planning, product development and coordination of production processes led to process improvements and resulting cost reductions that were shared between buyer and suppliers [15]. In practice, supplier development is widely limited to suppliers in tier-1. With respect to the superior importance of supplier development we postulated the extension of the traditional frame of reference in strategic sourcing from a supplier-centric to a supply-network-scope [8] i.e., the further development of the strategic supplier development to a strategic supply network development. This refocuses the object of reference in the field of strategic sourcing by analysing supplier networks instead of single suppliers.

2 Constructing the Ontology of the SSND Case

This section contains the result of applying the method for constructing the ontology of an enterprise, as presented above, to the SSND case. Space limitations prohibit us to provide a more extensive account of how the models in the figures below are arrived at. Also, we will not present and discuss the action rules. Figure 3 exhibits the Organization Construction Diagram (OCD). The Transaction Result Table (TRT) that belongs to it is shown in Table 1.

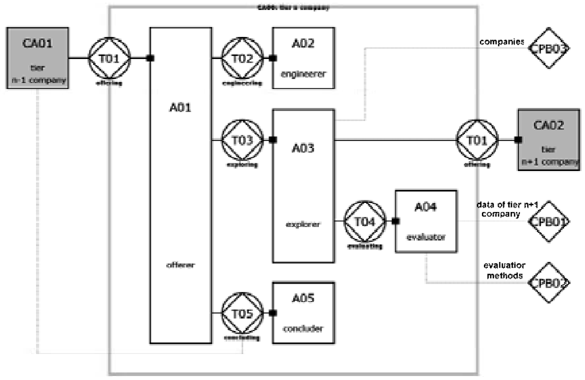


Figure 3. Organization Construction Diagram of the SSND case

Table 1. Transaction Result Table of the SSND case

transaction type	**resulting P-event type**
T01 offering	PE01 supply contract C is offered
T02 engineering	PE02 the BoM of assembly A is determined
T03 exploring	PE03 supply contract C is a potential contract
T04 evaluating	PE04 supply contract C is evaluated
T05 concluding	PE05 supply contract C is concluded

The top or starting transaction type is T01. Instances of T01 are initiated by the environmental actor role CA01, which is a company in tier n-1 and executed by actor role A01. This company asks for an offer regarding the supply of a particular product P. In order to make such an offer, A01 first initiates a T02, in order to get the bill of material of P. This is a list of (first-level) components of P, produced by A02. Next, A01 asks A03 for every such component to get offers from companies that are able to supply the component. So, a number of transactions T03 may be carried through within one T01, namely as many as there are components of P. In order to execute each of these transactions, A03 has to ask companies for an offer regarding the supply of a component of P. Since this is identical to the starting transaction T01, we model this also as initiating a T01. Now however, the executor of the T01 is a company in tier n+1. Consequently, the model that is shown in Figure 3 must be understood as to be repeated recursively for every tier until the products to be supplied are elementary, i.e. non-decomposable. Note that, because of the being recursive, an offer (the result of a T01) comprises the complete bill of material of the concerned component of P. Every offer from the companies in tier n+1 is evaluated in a T04. So, there is a T04 for every 'output' T01, whereby each company can use their own evaluation and decision rules. The result of a T04 is a graded offer for some component of P. So, what A03 delivers back to A01 is a set of graded offers for every component of P. Next, A01 asks A05, for every component of P, to select the best offer. The result is a set of concluded offers, one for every component of P. This set is delivered to A01. Lastly, A01 delivers a contract offer for supplying P, together with the set of concluded offers for delivering the components of P. Because of the recursive character of the whole model, this offer includes the complete bill of material of P, regardless its depth. The OCD in Figure 3 contains three external production banks. Bank CPB01 contains the data about a company that are relevant for the evaluation of offers. Bank CPB02 contains the different evaluation methods that can be applied. In every instance of T04, one of these methods is applied. CPB03 contains identifiers of all companies that may be addressed for an offer. Lastly, in the transaction result table (Table 1), the supply of product by a (supplying) company to a (customer) company is called a contract.

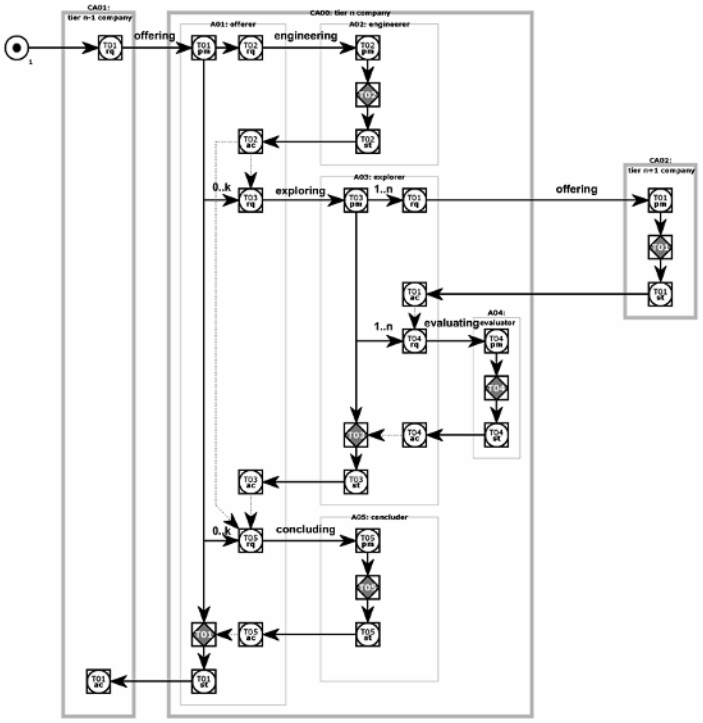


Figure 4. Process Step Diagram of the SSND case

Figure 4 exhibits the process step diagram of the SSDN case. It shows how the distinct transaction types are related. From the state T01/pm (promised) a number of transactions T03 (possibly none) and a number of transactions T05 (possibly none) are initiated, namely for every first-level component of a product. This is expressed by the cardinality range 0..k. Likewise, from the state T03/pm, a number of transactions T01 and a number of transactions T04 are initiated, namely for every offer or contract regarding a first-level component of a product. The dashed arrows, from an accept state (e.g. T02/ac) to some other transaction state, represent waiting conditions. So, for example, the performance of a T03/rq has to wait for the being performed of the T02/ac. Figure 5 exhibits the object fact diagram and Table 2 the object property table. Together they constitute the State Model of the example case.

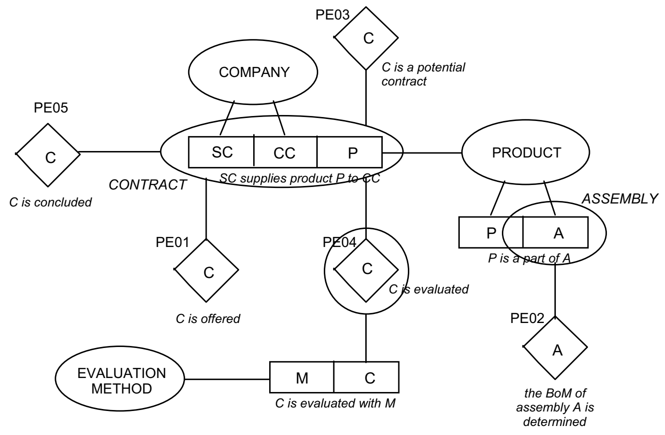


Figure 5. Object Fact Diagram of the SSND case

Table 2. Object Property Table of the SSND case

property type	**object class**	**scale**
< company information >	COMPANY	< aggregated data >
< contract terms >	*CONTRACT*	< aggregated data >
sub_products(*)	PRODUCT	set of PRODUCT
#sub_products(*)	PRODUCT	NUMBER
companies(*)	PRODUCT	set of COMPANY
sub_contracts(*)	*CONTRACT*	set of *CONTRACT*
evaluation_mark	*CONTRACT*	NUMBER
evaluation_marks(*)	*CONTRACT*	set of NUMBER

Properties are nothing more or less than binary fact types that happen to be a pure mathematical function of which the range is set of, usually ordered, values, called a scale. The OFD is a variant of the ORM model [16]. Diamonds represent unary fact types that are the result of transactions, also called a production event type. They correspond with the transaction results in Table 1. An ellipse around a fact type or a role defines a concept in an extensional way, i.e. by specifying the object class that is its extension. For example, the ellipse around the ternary fact type “SC supplies product P to CC” defines the concept of contract. The ellipse around the production event type “C is evaluated” defines the concept of evaluated contract. Lastly, the ellipse around the role “A” of the fact type “P is a part of A” defines all assemblies, i.e. all products that do have parts. The property types marked by "(*)" in the OPT are derived fact types. The derivation rules are:

sub_products (P) = {X | X is a product **and** X is a part of P};
#sub_products(P) = card(sub_products (P));
companies(P) = {X | X is a company **and** X supplies P to the 'this company'};
sub_contracts(C) = {X | X is a contract **and** the product of X is Z **and** the product of C is Y **and** Z is a part of Y};
evaluation_marks (C) = {X | X is an evaluation mark of C};

Table 3. Create/Use Table of the SSND case

object class or fact type	**process steps**
PRODUCT	T01/rq T01/pm T02/rq T02/pm *T02/st* T02/ac T03/rq T03/pm
product P is a part of product A	*T02/st* T03/pm
the BoM of assembly A is determined	*T02/ac*
COMPANY	T01/rq T03/pm T04/pm
< company information >	T04/pm T04/st
CONTRACT	T01/rq T01/pm T02/ac T03/rq *T03/pm* T01/st T01/ac T04/rq T04/pm T04/st T04/ac T03/st T03/ac T05/rq T05/pm T05/st T05/ac
< contract terms >	*T05/st*
supply contract C is offered	*T01/ac*
supply contract C is a potential contract	*T03/ac*
supply contract C is evaluated with method M	*T04/ac*
supply contract C is concluded	*T05/ac*
EVALUATION METHOD	T04/pm T04/st
sub_products(*)	*T02/st* T03/rq T03/pm
#sub_products(*)	*T02/st* T03/rq
companies(*)	T03/pm
sub_contracts(*)	*T03/pm* T03/st T01/st T03/ac T04/rq T04/ac
evaluation_mark(*)	*T04/st* T04/ac T03/st T04/pm T03/ac
evaluation_marks(*)	*T03/st*

Table 3 exhibits the Create/Use Table of the SSND case. It consists of an Information Use Table, extended with the process steps in which an instance of an object type or fact type is created. These steps are printed in italics, as are the fact types that are production event types.

2 Identifying Business Components for the SSND case

Based on the enterprise ontology introduced in the previous section and providing a complete and formal description of the business domain, business components for the domain of strategic supply network development are identified in this section and the resulting component framework is introduced.

The underlying idea of business components combines components from different vendors to an application which is individual to each customer. This principle of modular black-box design demands reusable, marketable, self-contained, reliable and manageable business components. Therefore the business components need to provide services at the right level of granularity and a formal and complete specification of its external view. The description of the specification step of the BCM process is not within the scope of this paper. Instead, for the identification of business components, an enhanced version of the Business Component Identification (BCI) [6] method is applied to the SSND domain and is described next.

The basis for BCI builds the Create/Use table (see Table) of the enterprise ontology. In a first step a matrix is built defining the relationships between the object class respectively fact types and the single process steps, gained from the Create/Use Table. The relationships are visualized inserting "C" and "U" in the matrix. "C" denotes that the object class respectively fact types are *created* in a specific process step and "U" denotes the *usage* of informational data by a given process step. In changing the order of the rows and the columns and placing the "C" as far left and as up as possible in the matrix, groups of relationships can be recognized (see Figure 6). These groups identify potential business components.

Object class and fact types / Process steps	contract	sub_contracts	contract terms	supply contract C is a potential contract	supply contract C is evaluated with method M	supply contract C is concluded	supply contract C is offered	product	product P is part of product A	sub_products	#sub_products	the BoM of assembly A is determined	evaluation_results	evaluation_result	evaluation method	companies	company information	company
T03/pm	c	c						u	u	u						u		u
T05/st	u		c															
T03/ac	u	u		c										u				
T04/ac	u	u			c									u	u			
T05/ac	u					c												
T01/st	u	u																
T04/rq	u	u																
T01/pm	u							u										
T01/ac	u						c											
T02/st								c	c	c	c							
T03/rq	u							u		u	u							
T02/ac	u							u				c						
T01/rq	u							u										
T02/rq								u										
T02/pm								u										
T03/st	u	u											c	u				
T04/pm	u													u	u		u	u
T04/st	u													c	u		u	
T05/rq	u																	
T05/pm	u																	

Manual execution:	▤
External banks:	▥
Business Components:	▨

Figure 6. Business Components Identification Matrix

In order to ensure optimal grouping regarding required metrics – such as minimal communication between components, maximum compactness of components – an optimization problem needs to be solved for which a genetic algorithm has been developed. The genetic algorithm starts with a predefined solution (specific assignment of process steps to components) and while iterating, generates better solutions using mutation and crossing-over of the best generated solutions available. For any iteration, the algorithm assigns each process step to a potential component (1, 2, 3, etc) and evaluates the potential solution with the following quality function:

$$q = \sum_{ps} \sum_{io} e(m(ps, io), p(ps), r(io)) \tag{4.1}$$

ps: process step in the matrix (row)
io: information object (object class or fact type) in the matrix (column)

$$p(ps) \mapsto \mathrm{N} : \text{assignment of a } ps \text{ to a component} \in \mathrm{N} \tag{4.2}$$

$$r(io) \mapsto \mathrm{N} : \text{assignment of an } io \text{ to a component} \in \mathrm{N} \tag{4.3}$$

$$m(ps,io) \in \{'c','u','o'\} : \text{content of matrix entry (create; use; other)} \tag{4.4}$$

$$e(m(ps,io),p(ps),r(io)) = \begin{cases} e_{in}(m(ps,io)) : p(ps) = r(io) \\ e_{out}(m(ps,io)) : p(ps) \neq r(io) \end{cases} \tag{4.5}$$

$$e_{in}(m(ps,io)) = \begin{cases} 0 : m(ps,io) = 'u' \vee 'c' \\ 0.1 : m(ps,io) = 'o' \end{cases} \tag{4.6}$$

$$e_{out}(m(ps,io)) = \begin{cases} 0 : m(ps,io) = 'o' \\ 1 : m(ps,io) = 'u' \\ +\infty : m(ps,io) = 'c' \end{cases} \tag{4.7}$$

The evaluation function (4.5) evaluates for each potential solution any entry in the matrix as follows. Entries which are a *use* or a *create* and located in a component are evaluated with the value 0, anything else is evaluated with a 0.1 (see formula 4.6). Entries outside a component which are a *use* are evaluated with 1 and a *create* with $+\infty$ (see formula 4.7). The quality function (4.1) calculates the entire sum over each evaluation for all entries in the matrix. After all iterations the *min(q)* provides the best component solutions for a given matrix, whereby different solutions with the same *min(q)* may exist. That means that all *creates* are located inside a component, few *uses* are outside of the components and few *other* (empty entries) are inside a component, fulfilling the required metrics of minimal communication and maximum compactness.

The result of the BCI is an abstract business component model with already defined dependencies between components. The dependencies are defined by the *uses* which are located outside the components and which are substitute by arrows as shown in Figure 7.

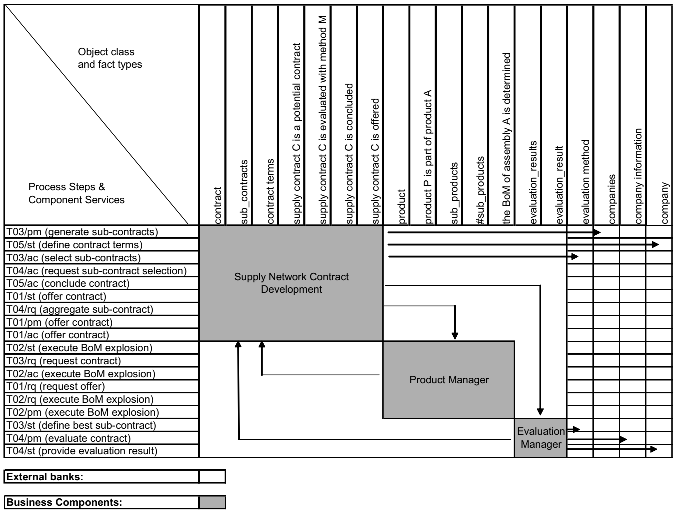

Object class and fact types
Process Steps & Component Services
contract
sub_contracts
contract terms
supply contract C is a potential contract
supply contract C is evaluated with method M
supply contract C is concluded
supply contract C is offered
product
product P is part of product A
sub_products
#sub_products
the BoM of assembly A is determined
evaluation_results
evaluation_result
evaluation method
companies
company information
company
T03/pm (generate sub-contracts)
T05/st (define contract terms)
T03/ac (select sub-contracts)
T04/ac (request sub-contract selection)
T05/ac (conclude contract)
T01/st (offer contract)
T04/rq (aggregate sub-contract)
T01/pm (offer contract)
T01/ac (offer contract)
T02/st (execute BoM explosion)
T03/rq (request contract)
T02/ac (execute BoM explosion)
T01/rq (request offer)
T02/rq (execute BoM explosion)
T02/pm (execute BoM explosion)
T03/st (define best sub-contract)
T04/pm (evaluate contract)
T04/st (provide evaluation result)
Supply Network Contract Development
Product Manager
Evaluation Manager
External banks:
Business Components:

Figure 7. Identified SSND Business Components

The first business component shown in Figure 7 offers services for the development of the contract offers for the whole supply network and is therefore called *supply network contract development.* The second business component is responsible for the specification of products and for the execution of the bill of material explosion and is therefore called *product manager.* The last component identified is the one responsible for the *evaluation* of the offered contracts.

For the business components identified, the services they provide need to be defined. Single process steps need therefore to be assigned to component services. The mapping for the strategic supply network development domain is shown in Figure 7 in the row *process steps and component services.* As can be seen, some process steps are mapped one to one to business component services, e.g. *T03/pm* is mapped to *generate sub-contracts,* or *T05/st* is mapped to *define contract terms*, whereas other process steps are combined to one business component service e.g. *T01/st, T01/pm, T01/ac* are all mapped to one and the same component service *offer contract.* Figure 8 shows the refined business component model with the services defined for each component.

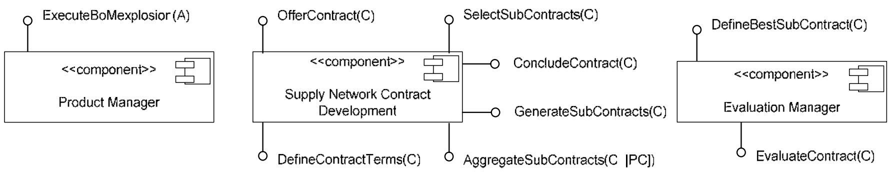


Figure 8. SSND Business Components with provided Services

The information for the mapping of process steps to business component services is gained from the action rules of the enterprise ontology. A detailed description of the mapping would go beyond the scope of this paper. The deployment of the components and their communication can be illustrated in using different types of diagrams, such as deployment and sequence diagrams. These diagrams allow the evaluation of the metrics applied and serve to improve them. The deployment of the components is part of the Component Based Domain Design phase of the BCM process. For a detailed description of this process step please refer to [6].

3 Conclusion

In this paper, we have addressed the problem of identifying business components, defined as the highest level software components, i.e. the software components that directly support business activities. Although component-based software development may be superior to more traditional approaches, it leaves the problem of requirements engineering unsolved. Put differently, the component-based approach provides no criteria for determining that a set of identified components is complete (no component is missing) and minimal (there are no redundant components). Next to that, it provides no means for determining the point of departure, i.e. an appropriate model of the business domain.

The method presented and demonstrated in this paper does solve these problems. First, the enterprise ontology constructed by means of DEMO is an appropriate model of the business domain. It satisfies the quality criteria as proposed in the introduction. As a consequence, the identified business components do really and directly support the business activities in which original new facts are created. Moreover, since the process steps (cf. Figure 4) are atomic from the business point of view, one can be sure to have found the most fine level of granularity that has to be taken into account. Also, one can be sure that this set of process steps is complete and minimal.

Second, the BCI method, based on the resulting models and tables of the enterprise ontology, provides an automated approach for the identification of business components satisfying defined metrics which are relevant for the development of reusable, marketable and self-contained business components. The metrics defined in this paper – being minimal communication between and maximum compactness of business components – are the basic metrics for the component-based development of inter-organizational business applications focusing on the deployment of components which can be on different enterprise systems. Additional metrics can easily be added to the BCI method in extending the genetic algorithm. Further investigation is needed in evaluating reliability and stability of the resulting component models.

References

1. Turowski, K. and J.M. Zaha, Methodological Standard for Service Specification. International Journal of Services and Standards, 2004.
2. Turowski, K., Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. 2003, Aachen: Shaker Verlag.

3. Dietz, J.L.G. and J.A. Barjis. Petri net expressions of DEMO process models as a rigid foundation for requirements engineering. In 2nd International Conference on Enterprise Information Systems. 2000. Escola Superior de Tecnologia do Instituto Politécnico, Setúbal: ISBN: 972-98050-1-6.
4. Dietz, J.L.G. Deriving Use Cases from Business Process Models. In Conceptual Modeling - ER 2003, LNCS 2813. 2003: Springer Verlag.
5. Dietz, J.L.G. and N. Habing. A meta Ontology for Organizations. In Workshop on Modeling Inter-Organizational Systems (MIOS), LNCS 3292. 2004. Larnaca, Cyprus: Springer Verlag.
6. Albani, A., et al. Domain Based Identification and Modelling of Business Component Applications. In 7th East-European Conference on Advances in Databases and Informations Systems (ADBIS-03), LNCS 2798. 2003. Dresden, Deutschland: Springer Verlag.
7. Albani, A., et al. Identification and Modelling of Web Services for Inter-enterprise Collaboration Exemplified for the Domain of Strategic Supply Chain Development. In On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2003, LNCS 2888. 2003. Catania, Sicily, Italy: Springer Verlag.
8. Albani, A., et al. Dynamic Modelling of Strategic Supply Chains. In E-Commerce and Web Technologies: 4th International Conference, EC-Web 2003 Prague, Czech Republic, September 2003, LNCS 2738. 2003, Springer-Verlag.
9. Albani, A., C. Winnewisser, and K. Turowski. Dynamic Modelling of Demand Driven Value Networks. In On The Move to Meaningful Internet Systems and Ubiquitous Computing 2004: CoopIS, DOA and ODBASE, LNCS. 2004. Larnaca, Cyprus: Springer Verlag.
10. Dietz, J.L.G., The Atoms, Molecules and Fibers of Organizations. Data and Knowledge Engineering, 2003. 47: p. 301-325.
11. Dietz, J.L.G. Generic recurrent patterns in business processes. In Business Process Management, LNCS 2687. 2003: Springer Verlag.
12. van Reijswoud, V.E., J.B.F. Mulder, and J.L.G. Dietz, Speech Act Based Business Process and Information Modeling with DEMO. Information Systems Journal, 1999.
13. Dietz, J.L.G. The notion of business process revisted. In OTM confederated International Conferences (CoopIS, DOA, ODBASE), LNCS 3290. 2004. Larnaca, Cyprus.
14. IBM, Business Systems Planning-Information Systems Planning Guide, ed. I.D. (Ed.). 1984, Atlanta: International Business Machines.
15. Carr, A.S. and J.N. Pearson, Strategically managed buyer - supplier relationships and performance outcomes. Journal of Operations Management, 1999. 17: p. 497 - 519.
16. Halpin, T.A., Information Modeling and Relational Databases. 2001, San Francisco: Morgan Kaufmann.

Using Connectors for Deployment of Heterogeneous Applications in the Context of OMG D&C Specification

Lubomír Bulej[1, 2] and Tomáš Bureš[1, 2]

[1] Charles University, Faculty of Mathematics and Physics, Department of Software Engineering, Malostranske namesti 25, 118 00 Prague 1, Czech Republic
{bulej,bures}@nenya.ms.mff.cuni.cz
[2] Academy of Sciences of the Czech Republic, Institute of Computer Science
{bulej,bures}@cs.cas.cz

1 Introduction

Component-based software engineering is a paradigm advancing a view of constructing software from reusable building blocks, components. A component is typically a black box with a well defined interface, performing a known function. The concept builds on the techniques well known from modular programming, which encourage the developers to split a large and complex system into smaller and better manageable functional blocks and attempt to minimize dependencies between those blocks.

Several aspects of component-based programming have been embraced by the software development industry and as a result, there are now several component models, such as Enterprise Java Beans [12] by Sun Microsystems, CORBA Component Model [8] by OMG, and .Net [5] by Microsoft, which are extensively used for production of complex software systems.

There are also a large number of other component models, designed and used mainly by the academic community. While most of the academic component models lack the maturity of their industrial counterparts, they aim higher with respect to fulfilling the vision of the component-based software engineering paradigm. This is mainly reflected in support for advanced modeling features, such as component nesting, or connector support. While we are aware of a number of component models used in academia, we are most familiar with SOFA [11,7] and Fractal [6]. Throughout the paper, we will use these models along with EJB as a test-bed for our experiments.

Typically, component applications are modeled as distributed and platform independent, with a particular execution platform selected during development. However, the current trends in software industry concerning enterprise integration may hint that platform independence on the level of design is not enough.

Our aim is to make possible creating heterogeneous applications which, in addition to the above, can consist of components written using different component models. This brings us two problems that need to be solved – 1) making the different components work together, and 2) deploying the resulting heterogeneous application.

The two problems may seem orthogonal, but in fact they are connected due to the nature of the differences between component models. These differences comprise mainly component packaging format and deployment, component instantiation and lifecycle management, communication middleware, hierarchical composition of components, etc. To make the different components work together and create a truly heterogeneous component application, we need to overcome those differences.

A key problem in making components from different component models work together is communication. Connections in different component models have different semantics and typically use different communication middleware to achieve distribution. Contemporary solutions to this problem usually employ middleware bridges (e.g. BEA WebLogic, Borland Janeva, IONA Orbix and Artix, Intrinsyc J-Integra, ObjectWeb DotNetJ, etc.) to connect components form different component technologies, which only tackles the issue of different middleware and leaves out the issue of different semantics and other (connection related) differences between the component models.

We propose to use software connectors [1] to define the semantics for connections between components from different component models. Based on requirements placed on a specific connection, the implementation of a connector can be automatically generated or, if the semantics allows it, a suitable middleware bridge can be used to mediate the connection.

Deployment of component applications is one of the most burning problems for the majority of component models. The deployment process generally consists of several steps, which have to be performed in order to successfully launch a component application. Without deployment support and tools, a component model is unusable for serious software development.

Most of the component models address the deployment issue in some way, but the differences between various component models have made it difficult to arrive at a common solution. Therefore the deployment process for component applications is specific to a particular component technology and a vendor. Worse, even applications written for a standardized component technology (e.g. EJB) have to be deployed in a vendor specific way using the vendor's proprietary tools.

The above mentioned situation makes the integration of components from different component models and the deployment and maintenance of the resulting application practically impossible.

A promising approach to deployment of heterogeneous component applications is modeling the application in a platform independent manner and mapping the platform independent model into the target environment to ensure interoperability. The first step in this direction has been done by the Object Management Group (the body behind CORBA and the CORBA Component Model [8]), who has published a specification concerning deployment and configuration of distributed component-based applications [9]. Following the MDA [10] approach, the specification presents platform independent models of the application, target environment, and the deployment process, which are then expected to be transformed to platform

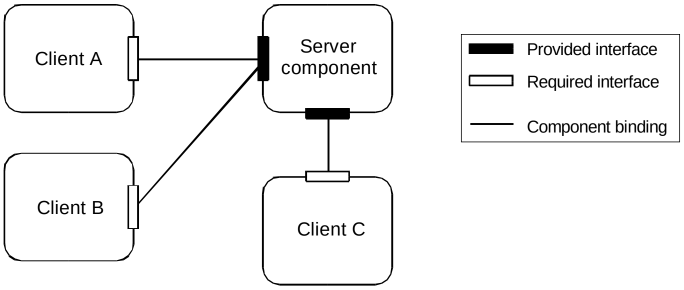


Figure 1. A model of a simple component application. There is a server component providing two different services with two clients connected to one service and another client connected to the other service.

specific models suitable for specific component technologies and mapped to particular programming environment.

Upon careful examination, though, the OMG specification stops short of providing support for deployment of heterogeneous component applications. The failure rests with the fact that the OMG expects the platform independent model to be mapped into a single target environment at a time. In effect, this means that the specification can be used to define a number of deployment mechanisms, but each of the deployment mechanisms will only support a single target environment. Interoperability between heterogeneous target environments is only provided at a conceptual level, which is rather insufficient.

We believe that the specification should support deployment of heterogeneous applications, rather than conceptually compatible but functionally incompatible deployment mechanisms for heterogeneous target environments. We extend the OMG specification to support deployment of heterogeneous component applications by introducing connectors as bridges between the heterogeneous parts of an application, and by extending the model to support construction of connectors during deployment.

Throughout the paper, we will use a model of a simple component application depicted in Figure 1 as a running example. The rest of the paper is organized as follows. Section 2 presents an overview of software connectors and their use in overcoming the differences between component models. Section 3 provides a short overview of the relevant parts of the OMG specification, and Section 4 demonstrates the deployment of a heterogeneous application using connectors and the OMG model of deployment process. We discuss related work in Section 5 and conclude the paper in Section 6.

2 Software Connectors

Software connectors are first class entities capturing communication among components. Although connectors may be very complex, modeled by complicated architectures reflecting different communication styles [4], it is sufficient for the

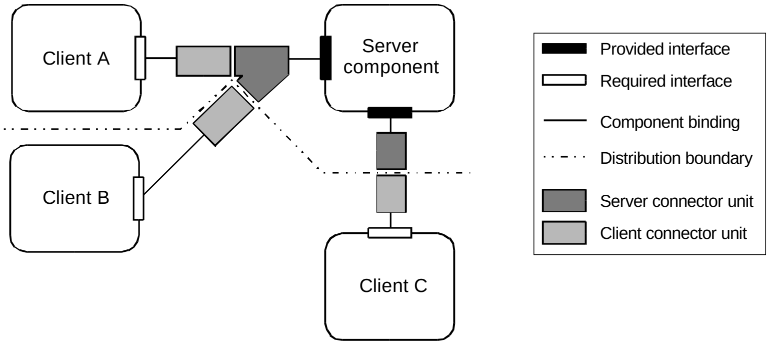


Figure 2. A model of a simple component application using connectors to capture inter-component interactions. Each connector is split into two parts respecting the distribution boundary. In the case of the procedure call-based connectors (i.e. those used in the example) there is typically one server unit and zero to many client units.

purpose of this paper to view a connector as a number of connector units attached to their respective components (see Figure 2).

Apart from modeling and representing inter-component communication, connectors have another important feature – they can be generated automatically based on a high-level description expressed in terms of a communication style and non-functional properties [4]. That allows a developer to concern herself just with components' business logic and not with glue code (often containing middleware dependent parts) used to provide communication among components in distributed environment.

The whole trick of using connectors is to plug an appropriate connector instance between every two components. However, as the concept of connectors is not typically present in current component models, it is often necessary to extend them to support connectors. That is easily done by hooking in the process of component instantiation and binding. Upon instantiation of a component, we create the server connector units and make sure that whenever component interfaces are queried, a connector reference to a corresponding server connector unit is returned (instead of returning a direct reference to a component interface). Similarly, whenever an interface is being connected to another component, a client connector unit is created and bound using the connector reference. In our approach, the connector reference is a container holding a set of server unit identities depending on available transport methods (e.g. Java reference for in-address-space connections, RMI reference for RMI connections, or IOR for CORBA connections).

When instantiating a connector unit at runtime, the information as to what implementation is to be used for that particular connector unit is looked up in a structure called *connector configuration map*, which contains pairs <interface discriminator, connector unit implementation>. The interface discriminator uniquely identifies either a server interface, in which case the discriminator is a pair <component, interface name>, or a client interface, in which case the discriminator is a tuple <client component, client interface name, server dock

name, server component, server interface name>. Such a description reflects the fact that a connector unit attached to a server interface is created in advance and can exist on its own, while a connector unit attached to a client interface is created during component binding when the binding target is already known.

3 Overview of the OMG D&C Specification

As mentioned earlier, the OMG Deployment and Configuration Specification is the first step towards unified deployment of component applications. The specification provides three platform independent models, the component model, the target model, and the execution model. These models represent the three major abstractions used during the deployment process, which uses these models to deploy an application.

For use with specific component models, the platform independent models should be transformed to platform dependent models, capturing the specifics of the concrete platform. A more detailed overview can be found in [3], and yet more details can be found in the specification itself [9].

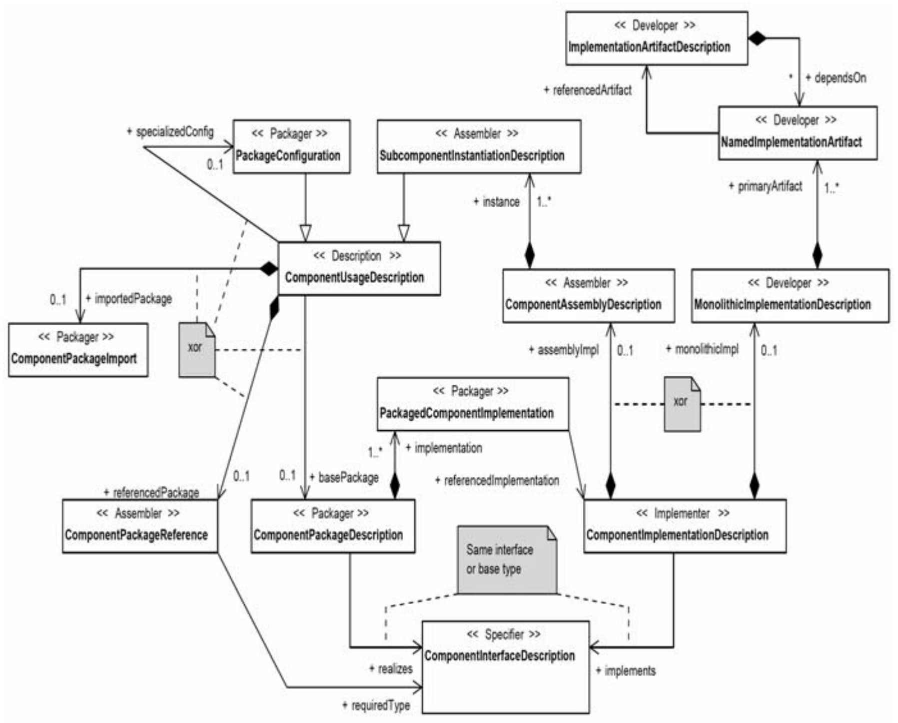
To reduce the complexity, the models are split into the data model and the management (runtime) model, with the management models describing runtime entities dealing with the data models. The management models are not important in the scope of this paper; therefore we will only deal with the data models.

3.1 Component Data Model

The component data model captures the logical view of a component application. A high level overview of the component data model is depicted in Figure 3. The key concept is a component package, which represents a reusable work product. A component package is a realization of a specific component interface, and contains possibly multiple implementations of the realized interface. As a reusable product, the package contains configuration of the encapsulated implementations, and selection criteria for choosing an implementation by matching the criteria to the capabilities of the individual implementations.

The implementation of a component interface can be either monolithic, or an assembly of other components. A monolithic implementation consists of a set of implementation artifacts that make up the implementation. The artifacts can depend on each other and can be associated with a set of deployment requirements and execution parameters. The deployment requirements have to be satisfied before an artifact can be deployed.

An assembly of components, depicted in Figure 4, contains references to other component packages to serve as subcomponents of the assembly. The instances of subcomponents are connected using connections between endpoints defined by subcomponents and external endpoints of the assembly. To allow for configuration of an assembly, which in itself does not carry any implementation code, the configuration properties of an assembly are delegated to its subcomponents through a defined mapping.



<< Developer >> ImplementationArtifactDescription
<< Developer >> NamedImplementationArtifact
+ referencedArtifact
+ dependsOn
+ primaryArtifact
<< Packager >> PackageConfiguration
<< Assembler >> SubcomponentInstantiationDescription
+ specializedConfig
+ instance
<< Description >> ComponentUsageDescription
<< Assembler >> ComponentAssemblyDescription
<< Developer >> MonolithicImplementationDescription
+ importedPackage
<< Packager >> ComponentPackageImport
xor
+ assemblyImpl
+ monolithicImpl
<< Packager >> PackagedComponentImplementation
+ implementation
+ referencedImplementation
+ basePackage
+ referencedPackage
<< Assembler >> ComponentPackageReference
<< Packager >> ComponentPackageDescription
<< Implementer >> ComponentImplementationDescription
Same interface or base type
<< Specifier >> ComponentInterfaceDescription
+ realizes
+ requiredType
+ implements

Figure 3. An overview of the component data model

3.2 Target Data Model

The target data model describes the computational environment into which the application is deployed. The environment, termed domain, consists of computational nodes, interconnects and bridges. Since the target model is not important in the scope of the paper, we will not describe the model in greater detail.

3.3 Execution Data Model

The execution data model depicted in Figure 5 describes the physical structure of a component application. The model represents a flattened view of the original component data model describing the logical structure of an application. The execution data model the application in terms of component instances, connections between endpoints of the instances, and assignment of the instances to computational nodes in the target environment

The component instances carry configuration properties which can be used to influence their behavior, and their implementation is in turn described in terms of implementation artifacts. The artifacts, which are binary files containing implementation code, can carry individual execution parameters, which can be used to tell the computational node how to treat a particular implementation

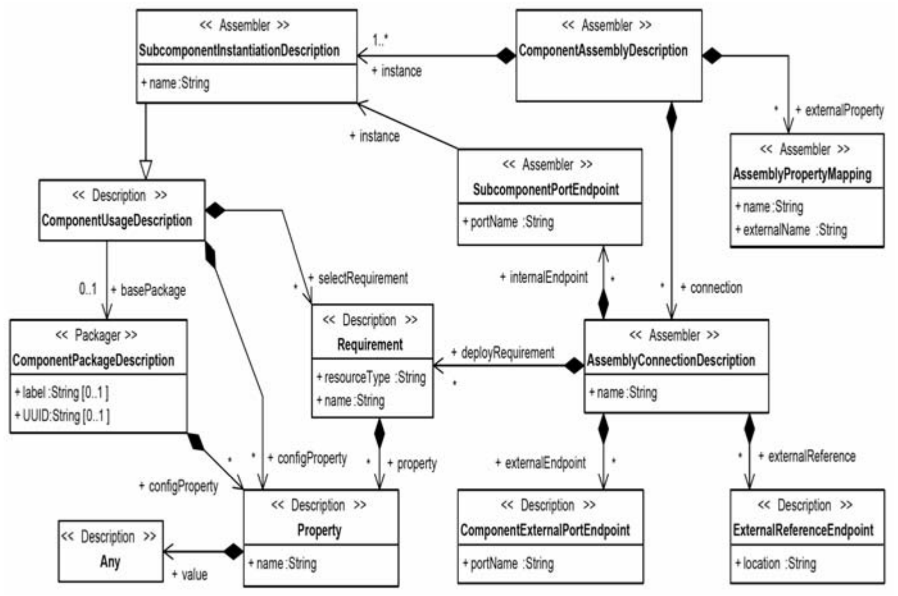


Figure 4. A detailed view of the component assembly description

artifact. Since the artifacts can be shared by multiple implementations, the execution parameters can be defined per implementation.

3.4 Deployment Process

Prior to deployment, the component application must be developed, packaged, and published by the provider and obtained by the user. The deployment process defined in the specification then consists of five stages and is performed by a designated actor called deployer.

Installation. During installation, the software package and its component data model is put into a repository, where it will be accessible from other stages of deployment. The location of the repository is not related to the target execution domain. Also, the installation does not involve transfer of binary files to the computational nodes in a domain.

Configuration. When the software is installed in the repository, its functionality can be configured by the deployer. The software can be configured multiple times for different configurations. The configuration stage is meant solely for functional configuration of the software, therefore the configuration should not concern any deployment related decisions or requirements.

Planning. After a software package has been installed into a repository and configured, the deployer can start planning the deployment of the application. The process of planning involves selection of computational nodes the software will run on, the resources it will require for execution, deciding which implementation will be used for component instances, etc. The planning does not have any immediate effects on the environment, but produces a physical description of the application in the execution data model, termed deployment plan.

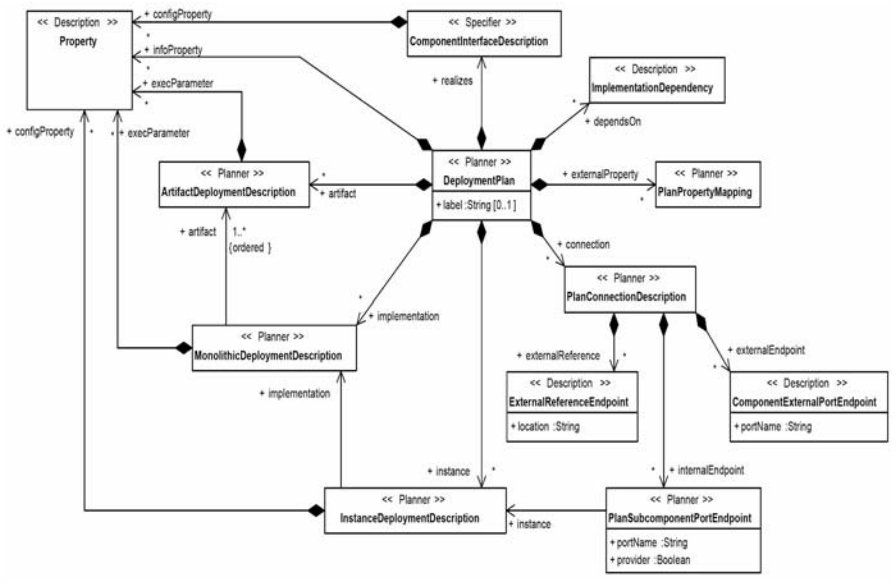


Figure 5. An overview of the execution data model

Preparation. Unlike the planning stage, the preparation stage involves performing work in the target environment in order to prepare the environment for execution of the software. The actual transfer of files to computational nodes in the domain can be postponed until the launch of the application.

Launch. After preparation, the application is brought up to the executing state during the launch stage. As planned, instances of components are created and configured on computational nodes in the target environment and the connections among the instances are established. The application then runs until terminated.

4 Integrating Connectors with Deployment

We have presented the two basic concepts we intend to employ to support deployment of heterogeneous component applications. Software connectors, described in Section 2 will be used to overcome the differences between various component models, while OMG D&C Specification, briefly introduced in Section 3 will be used to model the deployment process of a heterogeneous application.

Since the OMG specification does not support the description of heterogeneous component applications and does not directly support connectors, we have to find a way to combine the two approaches. To use connectors with a component application, there are basically two tasks that need to be done, and which need to be integrated with the OMG deployment process:

1. At some point, the implementation of all connectors needs to be generated, which comprises connectors for component bindings a) present in the initial

architecture of an application, and b) that can emerge at runtime as a result of passing a reference to component interface

2. The connectors need to be instantiated and bound to their respective components when launching an application. Additionally, the *connector configuration map* (described at the end of Section 2) must be prepared for each node to allow for later instantiation of connectors that result from reference passing.

4.1. Preparing connectors

To generate a connector, a connector generator needs to have enough information concerning the requirements for the communication the connector is expected to mediate. The specification of connector features has the form of a communication style and non-functional properties. Each connection among instances of components in an assembly can have different requirements.

The planning stage of the deployment process appears to be the most suitable moment for generating connectors. The planning is performed by the deployer using a planner tool. The tool takes as input the component data model, describing the component application, and the target data model, describing the target environment. Using the tool, the deployer assigns instances of components to nodes in the target environment and verifies that an instance can be placed on a particular node. The planner tool has all the information required for generation of connectors, except for the connection requirements.

The specification of connection requirements is not a part of the OMG specification, which therefore needs to be slightly modified. To make the information available, we have extended the AssemblyConnectionDescription in the component data model class with another association named `connectionRequirement`. The association is used to describe the connection requirements.

The connector-aware part of the planner can then communicate with a connector generator [4] and provide the necessary information. For each assembly connection, the generator synthesizes the implementation artifacts and configuration required to instantiate connector and returns the code fragments to the planner. The connector-aware part of the planner then replaces the assembly connections in the component data model with pairs of components encapsulating the connector units connected to the original components.

This step in fact transforms the enhanced component data model (see Figure 6) back into the original plain data model, which can be then transformed to deployment plan according to the original specification, and for which the planner does not need to be modified.

What is important to note, though, that while we transform connectors to components in the context of the OMG specification, connectors are not really components that would be present in the architecture of an application. Connectors are instantiated at runtime, the instance depends on the type of the server a connector unit connects to, and their encapsulation in components as seen in the component data model is merely an implementation convenience.

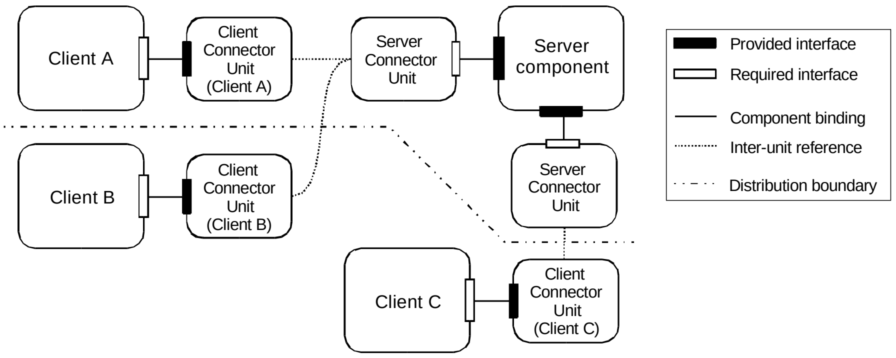


Figure 6. Application architecture after transformation of the enhanced description

4.2. Instantiating connectors

The output of the planning stage is a deployment descriptor, describing the physical structure of the application as assigned onto nodes in the target environment. The plan is then broken into pieces with respect to the distribution boundaries and disseminated to individual nodes. The runtime on each of the nodes uses the fragments of the deployment plan to instantiate components and connector units. Depending on the type of the connector unit one of the following actions is taken:

1. Server connector unit is instantiated and bound to its corresponding component. The unit registers its reference (using a name obtained from execution parameters in the deployment plan) in a naming service so as to be accessible by clients.
2. Bound client connector unit (i.e. representing a binding in the initial architecture) is instantiated and its corresponding component is bound to it. The unit retrieves a reference to a previously registered server connector unit from the naming service (using a name obtained from the deployment plan) and establishes the binding.
3. Future client connector unit (i.e. a unit that does not exist in the initial architecture but which can emerge at runtime as the result of reference passing) is stored in the *connector configuration map* for later use.

Since the implementation of a client connector unit depends also on the server component, there can be multiple implementations of a client unit. This is addressed by providing all the implementations in the deployment plan as different artifacts implementing one component and performing the actions 2 or 3 stepwise for the individual artifacts.

5 Evaluation and Related Work

Our approach to deployment of heterogeneous component applications builds upon two major concepts, the concept of software connectors, which is used to define semantics of connections between components from different component models, and the concept proposed by the OMG D&C specification, which unifies deployment of component based applications on conceptual level, but which in itself cannot provide for deployment of heterogeneous applications.

The original component data model present in the specification assumed direct communication between components. That requires that the artifacts providing component endpoints have to be connected together, which makes it impossible to abstract away the middleware technology used for communication. We have made a slight modification to the original OMG specification to enable expression of connection requirements in form of communication styles and nonfunctional properties, which can be used to generate connectors. This allows postponing the selection of communication middleware until the planning stage of the deployment process, or introducing logging, monitoring, or encryption into communication without changing the original application or its description.

We have also described the integration of the connector generator into the deployment process and pointed out places where the planner tool needs to be modified to support the connector generator. By transforming the enhanced component data model to the original component data model, we have avoided excessive modifications to the planner tool. The transformation of component data models is a generic process that can be used to enhance the expressive power of the component data model as long as the advanced constructs can be transformed back to the original data model.

To our knowledge, there is no other work concerning the use of connectors and the OMG specification to support deployment of heterogeneous component applications. There are, however, a number of mature business solutions for interconnecting the leading business component models such as EJB [12], CCM [8], and .NET [5]. A common denominator of these models is the lack of certain features (e.g. component nesting), which makes the problem of their interconnection a matter of middleware bridging. Each of those component models has a native middleware for communication in distributed environment (RMI in case of EJB, CORBA in case of CCM, and .NET remoting in case of .NET).

Even though the bridges represent mature software products, they limit the heterogeneity of the application by prescribing the use of specific communication mechanisms for the components. The connectors, on the other hand, represent a high level view on the connection between components, and allow for the bridges to be employed in the implementation of a connector if necessary.

6 Conclusion

We have presented an approach which we consider a step forward towards deployment of heterogeneous component applications, which allows us to create component applications composed of components implemented in different component models. We have shown how to employ software connectors to overcome the differences between component models and combined the use of connectors with the OMG D&C specification for unified deployment of component applications.

The presented solution is generic and platform independent, and can be used for different component models. We have verified our approach on a prototype implementation, which supports interconnection of components from SOFA, Fractal, and EJB component models, and are currently developing basic tools for deploying and execution of heterogeneous component applications. Mainly due to space constraints, we had to omit some of the details, which can be found in [3].

In the future, we plan to enhance the connector generator and develop more sophisticated tools, mainly the deployment planner and its integration with the connector generator. The work presented in this paper is part of our efforts within the Deployment Framework task of WP2 of the OSMOSE project. The results related to the OMG D&C specification will be submitted to the OMG.

Acknowledgments

The authors would like to thank Petr Tůma for his valuable comments. This work was partially supported by the Grant Agency of the Czech Republic project number 201/05/H014.

References

1. Balek, D., Plasil, F., Software Connectors and Their Role in Component Deployment, Proceedings of DAIS'01, Krakow, Kluwer, Sep 2001
2. Bulej, L., Bures, T., A Connector Model Suitable for Automatic Generation of Connectors, Tech. Report No. 2003/1, Dep. of SW Engineering, Charles University, Prague, Jan 2003
3. Bulej, L., Bures, T., Addressing Heterogeneity in OMG D&C-based Deployment, Tech. Report No. 2004/7, Dep. of SW Engineering, Charles University, Prague, Nov 2004
4. Bures, T., Plasil, F., Communication Style Driven Connector Configurations, Copyright Springer-Verlag, Berlin, LNCS3026, ISBN: 3-540-21975-7, ISSN: 0302-9743, pp. 102-116, 2004
5. Microsoft Corporation, .NET, http://www.microsoft.com/net, 2004
6. ObjectWeb Consortium, Fractal Component Model, http://fractal.objectweb.org, 2004
7. ObjectWeb Consortium, SOFA Component Model, http://sofa.objectweb.org, 2004
8. Object Management Group, Corba Components, version 3.0, http://www.omg.org/docs/formal/02-06-65.pdf, Jun 2002
9. Object Management Group, Deployment and Configuration of Component-based Distributed Applications Specification, http://www.omg.org/docs/ptc/04-08-02.pdf, Feb 2004
10. Object Management Group, Model Driven Architecture, http://www.omg.org/docs/ormsc/01-07-01.pdf, Jul 2001
11. Plasil, F., Balek, D., Janecek, R., SOFA/DCUP: Architecture for Component Trading and Dynamic Updating, Proceedings of ICCDS'98, Annapolis, Maryland, USA, IEEE CS Press, May 1998
12. Sun Microsystems, Inc., Java 2 Platform Enterprise Edition Specification, version 1.4, http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf, Nov 2003

Development of an Integrated Retrieval System on Distributed KRISTAL-2002 Systems with Metadata Information

Young-Kwang Nam[1], Gui-Ja Choe[1], Byoung-Dai Lee[2] and Jung-Hyun Seo[3]

[1] Department of Computer Science, Yonsei University, Korea,
{yknam, gjchoe}@dragon.yonsei.ac.kr
[2] Digital Home Solution Lab., Samsung Electronics, Co. LTD., Korea,
byoungdai.lee@samsung.com
[3] Group for Information Standardization, Korea Institute of Science and Technology Information, Korea, jerry@kisti.re.kr

Summary. In this paper, we propose an integrated information retrieval system for multiple KRISTAL-2002 systems for different areas or systems for the same area with the different schemas by using the metadata information so that the users can get the answers by once from the whole systems. Our approach utilizes integrated metadata and mapping information between the metadata and the actual database schema information of participating systems. Therefore, we do not require modification of the participants for integration. We have implemented and deployed the proposed system that integrates six different databases distributed across multiple sites.

1 Introduction

Since the early 70's, many organizations in diverse areas have been developing and deploying information retrieval systems. Due to the availability of Internet, user demand for integrated search through the information retrieval systems that have been deployed has increased while the speed with which these systems are integrated does not catch up with the user demand. Integrating the information retrieval systems in each special area requires storing millions of bibliographic information either into different tables through generalization as in relational database systems or into a single table. The former approach may suffer from poor performance caused by JOIN operations. The latter approach may have low space utilization because there is significant duplicate information.

From service provider's point of view, it is not appropriate to modify existing information retrieval systems to build an integrated information retrieval system because they may need to stop servicing. However, from user's point of view, it is tedious to visit, register and search each system manually whenever they need information. In this paper, we propose an integrated information retrieval system (which we call ***Integrated Server***) that does not require modification of participant systems (which we call ***Source Server***). In the proposed system, mapping information between integrated metadata and actual databases or tables of each source server is maintained and is utilized for integrated search. For integrated metadata, we follow ISO/IEC 11179 metadata registry procedure.

In order to make the integration process simple, the source server administrator is able to map the database schema of the source server to the meta-fields that will be used for integrated search. In this approach, users are able to search distributed information sources without considering their locations and the kinds of information each server maintains. The user query, however, must be re-generated for each source server based on the mapping information. This relationship is shown on Figure 1.

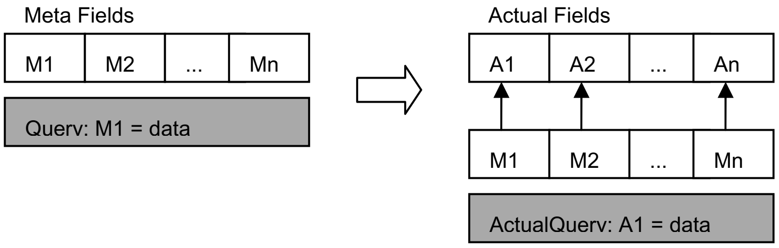


Figure 1. Relationship between meta fields and actual fields

We have implemented and deployed the proposed system using KRISTAL-2002 system, which will be explained in later section. We include six sources servers in different areas that are currently servicing independently: Science Literature DB, Scientific Technical Trend DB, Scientific Technical Report DB, Scientific Technical Analysis DB, Patent DB and Human Resource DB.

The paper is organized as follow: Section 2 gives related work and Section 3 describes KRISTRAL-2002 system. Section 4 presents the proposed integrated information retrieval system in details. Section 5 presents the prototype implementation and finally, we conclude in section 5.

2 Related Work

There have been considerable efforts in developing algorithms and protocols for integrating heterogeneous data distributed across multiple sources. However, schema integration and schema mapping techniques dealing with discrepancies of schema among distributed data sources require more research to address semantic interoperability

Techniques for web search engines and directory-based portal services are the driving forces for advancing the information retrieval area. They must be able to search a large amount of data with short queries given by users. To address the problem, these techniques utilize ranking algorithms that use link information or structural information of the documents. This type of information is not included in the original documents. Crawling 18 is a new technique for web search engines. It collects and indexes documents distributed across multiple sites and the primary focus of crawling technique is how fast a given document can be included in the search target. Clustering 19 focuses on presenting accurate information to users by combining related search results. Meta-searching technique (2021) sends a user query to multiple web search engines and presents the results after combining the partial results obtained from them. It does not require web crawler or indexing a large amount of documents. However, it must be able to combine the results effectively that are found by different ranking algorithms of each source system.

Ontology (891011) and similar researches address semantic interoperability between metadata in consideration of mapping the meaning and the presentation of source data into real-life objects and concepts. Examples of such research include RDF 4, Schema Integration 14, Intelligent Integration of Information 15 and Knowledge Sharing Effort 16, to name a few.

3 KRISTAL-2002

KRISTAL-2002 is an information retrieval and management system developed by Korea Institute of Science & Technology Information. It runs on both Windows and Unix systems. KRISTAL-2002 consists mainly of five modules (Job Scheduler, FIRE, Data Manager, Set Manager and Indexer) and they communicate one another through sockets or pipes (Figure 2). Job Scheduler distributes user requests to FIRE, which conducts actual searching. Set Manager stores and manages documents found by FIRE. Data Manager processes Document Update or Document Store requests from Job scheduler. Once it completes the requests, it sends acknowledgement to the Job Scheduler. If the operation is successful, it contacts Set Manager so that the Set Manager can update the documents it maintains. Indexer is able to analyze input documents and extract indexing words that can best describe the documents. Indexer supports analysis of Korean

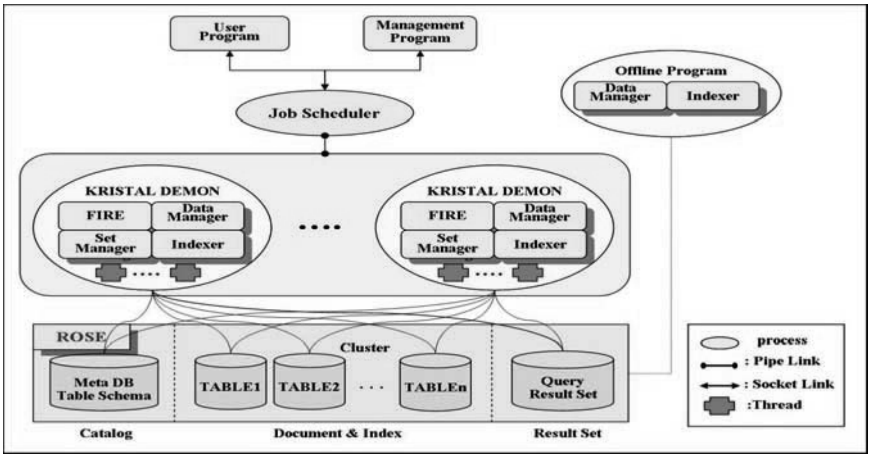


Figure 2. The architecture of KRISTAL-2002

morpheme, English stamping, Chinese character conversion and user-defined dictionary, to name a few. FIRE, Data Manager, Set Manager and Indexer are integrated into a single daemon and it is able to manage multiple databases.

KRISTAL-2002 database supports processing simultaneous user queries and on/off-line data management and fast and safe backup. Searching and processing BLOB (Binary Large OBject) is time-consuming. To address this, KRISTAL-2002 utilizes multiple threads for distributed query processing. The primary components of the databases are: Catalog, Document and Index, and Result Set. Catalog database maintains schema information such as table structures, indexing methods and primary keys and so on. Document and Index database is structured by a single or multiple clusters and each cluster is composed of tables that have the same schemas. Cluster is the basic unit of ranking. Each table in Document and Index database is composed of documents, primary keys, and index database and the structure of the table is defined by table schema stored in Catalog database. Result Set database maintains documents found so that it can respond to user requests quickly.

4 System Architecture of the Integrated Information Retrieval System

4.1 System Architecture

The proposed integrated information retrieval system is based on KRISTAL-2002 and it utilizes metadata information registered by participating source servers. To address the heterogeneity of data and schema information of each source server, metadata maintained in the integrated server is used for structural integration.

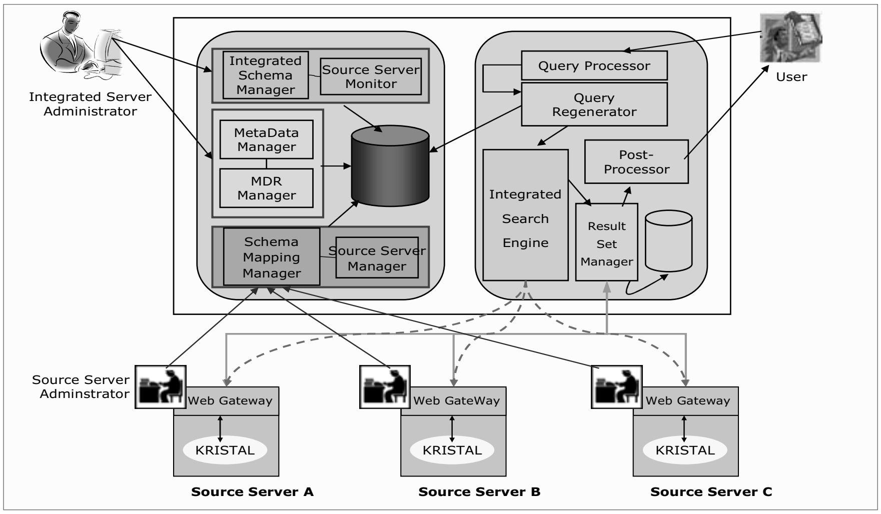


Figure 3. System architecture of the integrated information retrieval system

Furthermore, standardized procedure proposed in ISO/IEC 11179 is employed for metadata registry procedure. Schema information of each source server must be mapped into metadata maintained in the integrated server. Therefore, schema that hasn't been mapped will not be considered for search process. Figure 3. shows the system architecture of the proposed system.

The source server administrator manages the source server and the structural information of the database maintained by the source server while MDR (MetaData Registry) manager standardizes data elements for metadata registry. Integrated server administrator manages and controls all the information of the whole system to support rapid and effective user query processing. Each component of the system will be explained in more details in later sub sections.

The control and information flow among the source servers and the integrated servers is depicted in Figure 4. Each source server administrator registers necessary information to the integrated server and the integrated server administrator authorizes the source server administrator so that he or she can register the source servers. The registered source servers must go through the same authorization process as with the source server administrator to participate in the whole system. Once the registered source servers are authorized by the integrated server, the source server administrators are entitled to do schema mapping. As with previous steps, schema mapping done by the source server must be confirmed by the integrated server. After these steps, the source server is included for integrated search.

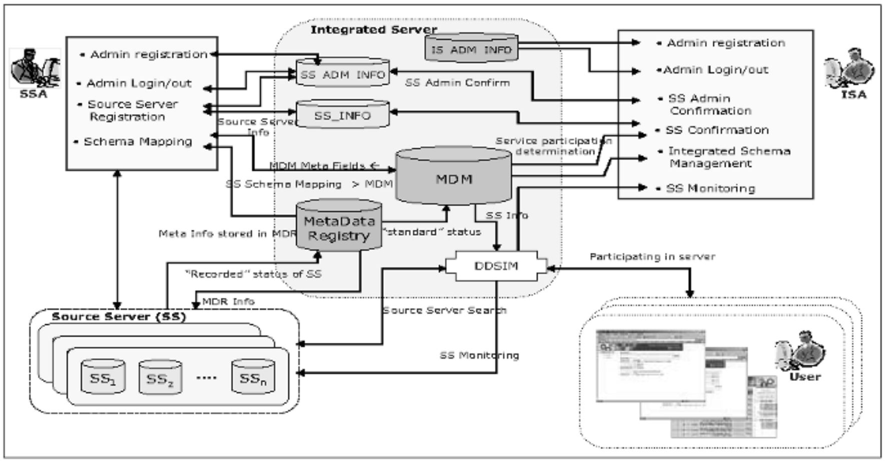


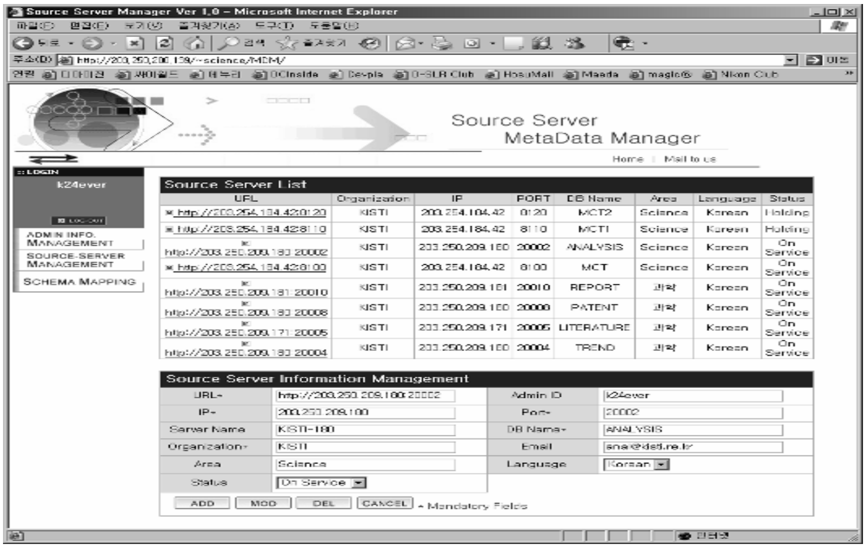
Figure 4. Control and information flow between source server and integrated server

4.2 Source Server Manager

Each source server can run its own database systems and are responsible for searching and providing documents stored in the database when requested by the integrated server. The source server administrator is able to add/remove the source server to/from the set of source servers that will be included for integrated search, and to map the database schema of the source server to the database schema of the integrated server. All these operations are conducted through web interfaces.

4.2.1 Source Server Information Management

Figure 5. shows the web interface for source server registration. When *URL* column on Source Server List is clicked, the information of the selected server will be shown on Source Server Information Management and the source server



Source Server List

URL	Organization	IP	PORT	DB Name	Area	Language	Status
http://203.254.184.42:8120	KISTI	203.254.184.42	8120	MCT2	Science	Korean	Holding
http://203.254.184.42:8110	KISTI	203.254.184.42	8110	MCT1	Science	Korean	Holding
http://203.250.209.180:20002	KISTI	203.250.209.180	20002	ANALYSIS	Science	Korean	On Service
http://203.254.184.42:8100	KISTI	203.254.184.42	8100	MCT	Science	Korean	On Service
http://203.250.209.181:20010	KISTI	203.250.209.181	20010	REPORT	과학	Korean	On Service
http://203.250.209.180:20008	KISTI	203.250.209.180	20008	PATENT	과학	Korean	On Service
http://203.250.209.171:20005	KISTI	203.250.209.171	20005	LITERATURE	과학	Korean	On Service
http://203.250.209.180:20004	KISTI	203.250.209.180	20004	TREND	과학	Korean	On Service

Source Server Information Management

URL*	http://203.250.209.180:20002	Admin ID	k24ever
IP*	203.250.209.180	Port*	20002
Server Name	KISTI-180	DB Name*	ANALYSIS
Organization*	KISTI	Email	ana@kisti.re.kr
Area	Science	Language	Korean
Status	On Service		

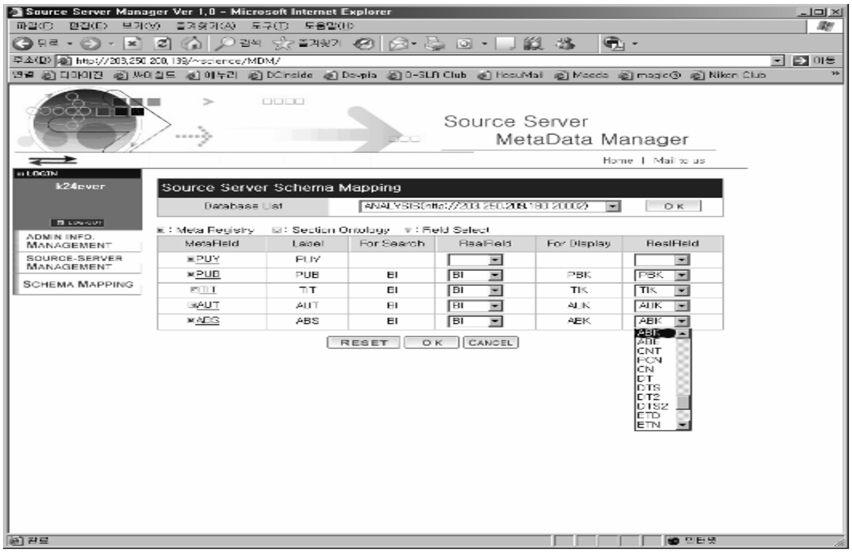
ADD MOD DEL CANCEL * Mandatory Fields

Figure 5. Web interface for source server registration

administrator can change information of the server or remove from the list. *Status* column on Source Server List represents three different status information of the source servers: *Holding, On Service, Unavailable*. Although the source server has been registered, it is not included for the integrated search until it is confirmed by the integrated server. During this period, the status of the source server is *Holding*. Once confirmed, the status of the source server is *On Service*. If the source server cannot service temporarily due to some reasons, for example, updating operating system of the source server, the source server administrator can change the status of the source server to *Unavailable*. When the source server returns to the normal condition, the administrator can change the status of the server to *On Service*.

4.2.2 Source Server Schema Mapping

Figure 6. shows the web interface for schema mapping for the source server. If multiple source servers are managed by an administrator, only those source servers whose statuses are *On Service* are listed on the web page. *MetaField* column represents integrated meta schemas loaded from metadata registry and *Label* column represents the names of the meta schemas. When values in *MetaFields* are clicked, the detailed information of the meta field is shown on a different window. *RealField* represents database schema of the source server mapped into the integrated meta schema. The values listed in the combo box are loaded directly from the source server.



Source Server Schema Mapping

Database List: ANALYSIS(http://203.250.208.180:2000/) OK

▣ : Meta Registry ▨ : Section Ontology ▼ : Field Select

MetaField	Label	For Search	RealField	For Display	RealField
▣PUY	PUY				
▣PUB	PUB	BI	BI	PBK	PBK
▣TIT	TIT	BI	BI	TIK	TIK
▨AUT	AUT	BI	BI	AUK	AUK
▣ABS	ABS	BI	BI	ABK	ABK

ABK
ABE
CNT
FCN
CN
DT
DTS
DT2
DTS2
ETD
ETN

Figure 6. Web interface for schema mapping

4.3 Integrated Server Manager

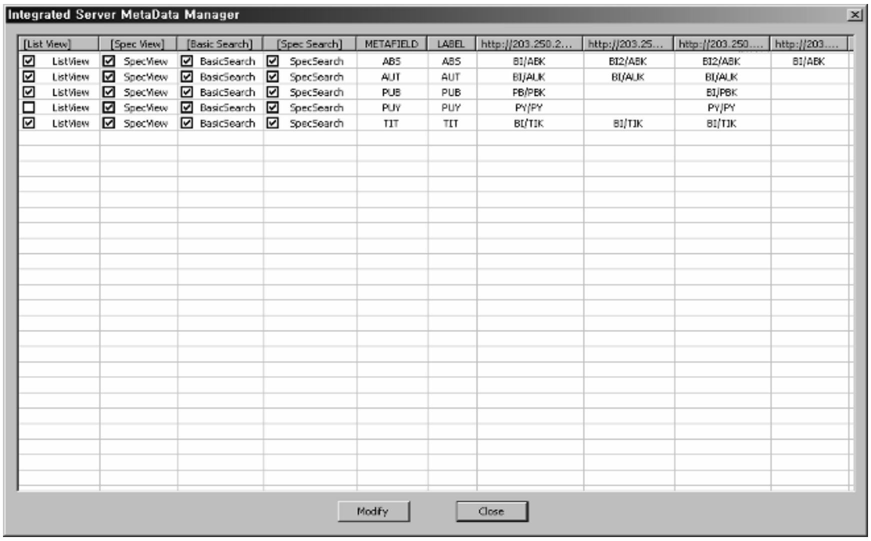
The role of the integrated server is to register metadata based on ISO/IEC 11179 and to maintain integrated meta schema and mapping information registered by the source servers. It also plays a role of entry point for the integrated search. Integrated server administrator determines which source servers will be included for the integrated search service and validity of schema mapping of the source servers. It monitors the status of the source servers included in the service. When it is detected that some source servers are not running correctly, the integrated server immediately exclude the servers from the source server list.

4.3.1 Source Server Configuration

Integrated server administrator is able to confirm the requests from the source server administrators who want to participate in the integrated search service. Once confirmed, the source server administrators are entitled to add their source servers into the service (see 4.2.1). For the source servers registered by the source server administrator, the integrated server administrator needs to decide whether or not to add them into the server list that will actually service end-users. Note that these two processes are different in that the former validates the source server administrators while the latter validates the source servers that the valid source server administrator registered.

4.3.2 Schema Mapping Management

Schema Mapping Manager of the integrated server determines integrated metadata schema and given schema mapping registered by the source server administrator (see 4.2.2), it checks the validity of the mapping (Figure 7). Each row in Figure 7 provides schema mapping information of all the source servers participating in the integrated search service. *ListView* and *SpecView* columns determine which meta fields will be shown to users after search completes. Typically, those fields marked as *ListView* will give rough information of the selected documents while those fields marked *as SpecView* will be used to provide more detailed information of the documents. *BasicSearch* and *SpecSearch* columns determine which meta fields must be compared against the given user query. We support two different modes for search operation: *Basic Search and Specific Search*. In *basic search*, user query will be compared to every target meta fields defined by the integrated server. However, in *specific search*, the user is able to determine which meta fields he/she wants to compare the query to.


Integrated Server MetaData Manager

[List View]	[Spec View]	[Basic Search]	[Spec Search]	METAFIELD	LABEL	http://203.250.2...	http://203.25...	http://203.250....	http://203....
☑ ListView	☑ SpecView	☑ BasicSearch	☑ SpecSearch	ABS	ABS	BI/ABK	BI2/ABK	BI2/ABK	BI/ABK
☑ ListView	☑ SpecView	☑ BasicSearch	☑ SpecSearch	AUT	AUT	BI/AUK	BI/AUK	BI/AUK	
☑ ListView	☑ SpecView	☑ BasicSearch	☑ SpecSearch	PUB	PUB	PB/PBK		BI/PBK	
☐ ListView	☑ SpecView	☑ BasicSearch	☑ SpecSearch	PUY	PUY	PY/PY		PY/PY	
☑ ListView	☑ SpecView	☑ BasicSearch	☑ SpecSearch	TIT	TIT	BI/TIK	BI/TIK	BI/TIK	

Figure 7. Source server schema mapping manager

When the actual fields of the source server is clicked, schema information of the source server is shown and the mapped fields for the given meta field are shown checked. By allowing multiple fields to be selected, we support ontology. When an administrator clicks *MetaField* column, the detailed information of the meta field, is loaded from meta data registry, is shown on a different window. The information shown in Figure 7 is the subset of this information.

4.4 Integrated Query Processing

4.4.1 Distributed Query Processing (DQP)

Given the user query, DQP re-generates queries for source servers using schema mapping information and sends them to the corresponding source servers. DQP

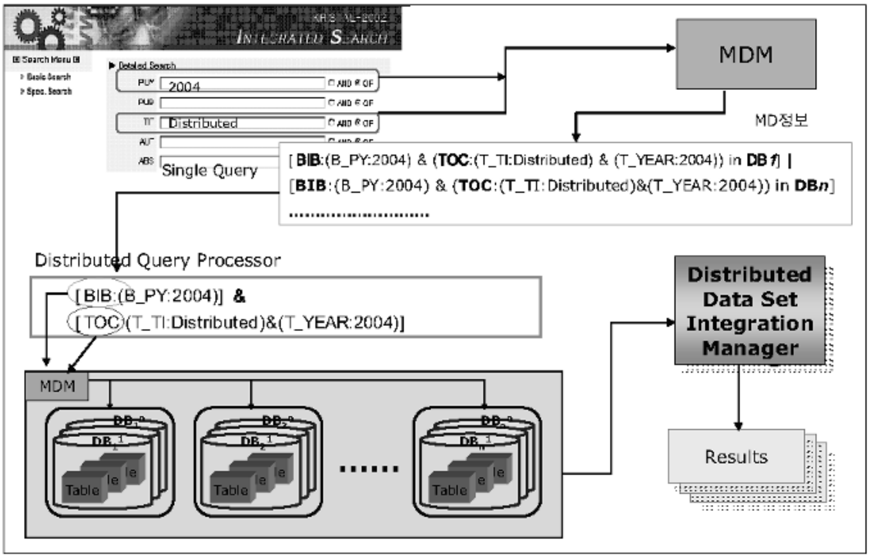

KRISTAL-2002
INTEGRATED SEARCH
Search Menu
Basic Search
Spec. Search
Detailed Search
2004
Distributed
Single Query
MDM
MD정보
[BIB:(B_PY:2004) & (TOC:(T_TI:Distributed) & (T_YEAR:2004)) in DB1] |
[BIB:(B_PY:2004) & (TOC:(T_TI:Distributed)&(T_YEAR:2004)) in DBn]
Distributed Query Processor
[BIB:(B_PY:2004)] &
[TOC:(T_TI:Distributed)&(T_YEAR:2004)]
Distributed Data Set Integration Manager
Results
Table

Figure 8. Relationship between DQP and other modules in the system

transforms user-input queries into Boolean or Vector queries. Figure 8. shows the relationship between DQP and other modules in the system. For the user-input entries, DQP first read schema mapping information of each source server. If the source server has mapped the meta fields used for the integrated search into the actual fields of the database it manages, DQP extracts the table names and the actual fields that correspond to the user-input entries and regenerates new query as follows: *[table name: (actual field name: user-input entry)]*. If user-input entries are connected with AND or OR operations, then sub queries of the new query must be connected with the same Boolean operators. The following shows an example.

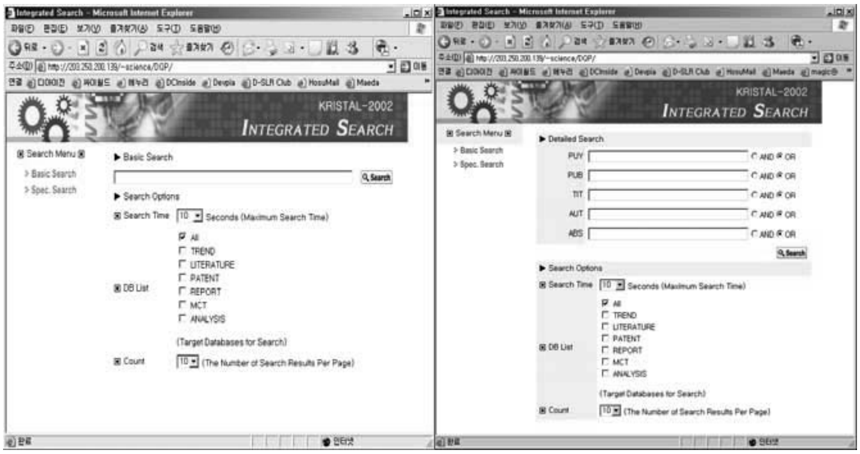
```
AND operation
[BIB: (B_PY:2004)] & [TOC:(T_TI:Distributed) & (T_YEAR:2004)]

OR operation
[BIB: (B_PY:2004)] | [TOC:(T_TI:Distributed) & (T_YEAR:2004)]
```

4.4.2 Distributed Data Set Integrated Manager (DDSIM)

DDSIM is responsible for collecting and displaying the search results to users (Figure 8). For the documents found by the source servers, it extracts those fields specified by schema mapping manager of the integrated server (See 4.3.2). Using the values, it generates HTML document.

5 Prototype Implementation

We have implemented and deployed the proposed integrated information retrieval system. We have included six sources servers in different areas that are currently servicing independently: Science Literature DB, Scientific Technical Trend DB, Scientific Technical Report DB, Scientific Technical Analysis DB, Patent DB and Human Resource DB.



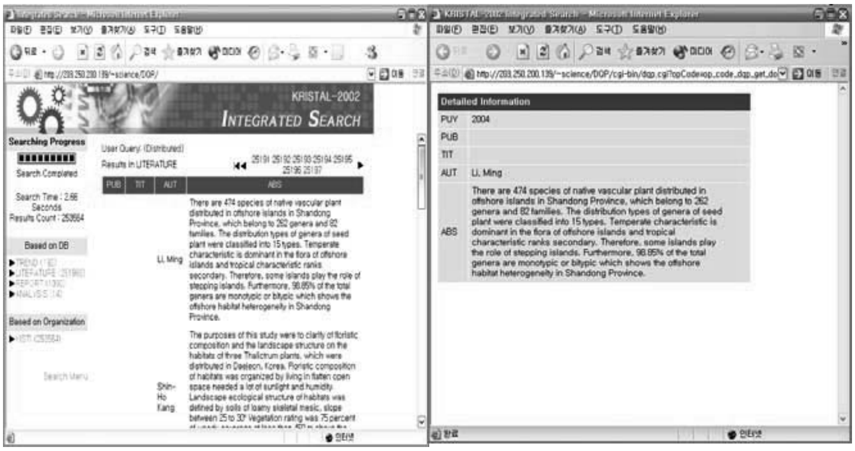
Search Time: 10 Seconds (Maximum Search Time)

DB List: [✓ All] TREND LITERATURE PATENT REPORT MCT ANALYSIS

Count: 10 (The Number of Search Results Per Page)

Figure 9. Web interface for integrated search service

Two types of search services in the current implementation are provided: *Basic Search* and *Specific Search* (Figure 9.). In basic search mode, users enter query words in a single text area of the web page. Users are able to select target source servers, the number of documents per page and the maximum search time. The user-input entries are compared against all of the meta fields marked for *BasicSearch* (see 4.3.2). When there are multiple meta fields for basic search, they



Search Time : 2.66 Seconds
Results Count : 253564

User Query: (Distributed)
Results in LITERATURE

25191 25192 25193 25194 25195 25196 25197

PUB	TIT	AUT	ABS
		Li, Ming	There are 474 species of native vascular plant distributed in offshore islands in Shandong Province, which belong to 262 genera and 82 families. The distribution types of genera of seed plant were classified into 15 types. Temperate characteristic is dominant in the flora of offshore islands and tropical characteristic ranks secondary. Therefore, some islands play the role of stepping islands. Furthermore, 98.85% of the total genera are monotypic or bitypic which shows the offshore habitat heterogeneity in Shandong Province.
		Shin-Ho Kang	The purposes of this study were to clarify of floristic composition and the landscape structure on the habitats of three Thalictrum plants, which were distributed in Daejeon, Korea. Floristic composition of habitats was organized by living in flatten open space needed a lot of sunlight and humidity. Landscape ecological structure of habitats was defined by soils of loamy skeletal mesic, slope between 25 to 30° Vegetation rating was 75 percent

Detailed Information	
PUY	2004
PUB	
TIT	
AUT	Li, Ming
ABS	There are 474 species of native vascular plant distributed in offshore islands in Shandong Province, which belong to 262 genera and 82 families. The distribution types of genera of seed plant were classified into 15 types. Temperate characteristic is dominant in the flora of offshore islands and tropical characteristic ranks secondary. Therefore, some islands play the role of stepping islands. Furthermore, 98.85% of the total genera are monotypic or bitypic which shows the offshore habitat heterogeneity in Shandong Province.

Figure 10. Search results

are connected with OR operators. Unlike basic search mode, in specific search mode, users are allowed to select target meta fields that will be compared to the user-input entries. These target meta fields are those meta fields that have been marked for *SpecSearch* by the integrated server administrator (see 4.3.2). In addition, the Boolean connectors also can be determined by users.

Figure 10. shows the search results. Once the searching process completes, DDSIM shows only the summarized information of the selected documents. If the user wants more detailed information, he/she clicks the document and can obtained detailed information. The actual fields of source servers used for displaying documents are defined dynamically by the source server administrator (see 4.3.2).

6 Conclusion

In this paper, we proposed an integrated information retrieval system that does not require modification of existing information sources. The proposed system consists of Source Server Manager, Integrated Server Manager, MetaData Registry Manager, Distributed Query Processor and Distributed Data Set Integrated Manager. We have implemented and deployed the proposed system and tested and verified it using six source servers that use different database schema.

Future work includes developing a system that does not assume that participating source servers are running the same database system.

References

1. Z39.50 Gateway, http://www.loc.gov/z3950/gateway.html.
2. ANSI/NISO Z39.50 Revision, January 2002. http://www.loc.gov/z3950/agency/revision/revision.html.
3. http://giis.kisti.re.kr/download/k-protocol-03-02-14.pdf.
4. D. Brickley, R. Guha, eds. Resource Description Framework Schema Specification. W3C Proposed Recommendation, March 1999.
5. Metadata Registry, http://metadata-stds.org/11179.
6. K. Beard, T. Smith. A Framework for Meta-Information in Digital Libraries, In Sheth A, Klas W (eds) Multimedia Data Management: Using Metadata to Integrate and Apply Digital Media. Mc Graw Hill: 341-365, 1998.
7. Online Computer Library Center, Inc. 1997 Dublin Core Metadata Element Set: Reference Description. 1997. Office of Research and Special Projects, Dublin, Ohio. http://www.oclc.org:5046/research/dublin_core.
8. Ontology, http://www.w3.org/2001/sw/WebOnt.
9. D Calvanese, G. Giacomo, and M. Lenzerini. Ontology of Integration and Integration of Ontology, Proceedings of the 2001 Description Logic Workshop (DL 2001), 2001.
10. D. Calvanese, G. Giacomo, M. Lenzerini, D Nardi, R. Rosati. Description Logic Framework for Information Integration, Proceedings of Principles of Knowledge Representation and Reasoning (KR'98), 1998.
11. Farquhar, R. Fikes and J. Rice. The Ontiliqua Server: A Tool for Collaborative Ontology Construction. International Journal of Human-Computer Studies, 1997.
12. D. Egnor and R. Lord. Structured Information Retrieval using XML, ACM SIGIR Workshop on XML and Information Retrieval, 2000.

13. M. Kobayashi and K. Takeda, Information Retrieval on the Web, IBM Research Report, RT0347, 2000.
14. Nishizawa Itaru, Takasu Atsuhiro, Adachi Jun. A Schema Integration and Query Processing Information Retrieval, IPSJ SIGNotes Database System Abstract, No.094009, 1993.
15. Intelligent Integration of Information, http://mole.dc.isx.com/13.
16. Knowledge Sharing Effort, http://www-ksl.stanford.edu/knowledge-sharing.
17. Robert M. Losee and Lewis Church Jr., Information Retrieval with Distributed Databases: Analytic Models of Performance, IEEE Transactions on Parallel and Distributed Systems, Vol.14, No. 12, 2003.
18. Soumen Chakrabarti, Martin van den Berg and Byron Dom, Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery, Elsevier Science, 1999.
19. Oren Zamir and Oren Etzioni, Web Document Clustering: A Feasibility Demonstration, SIGIR, 1998.
20. Hsinchun Chen, Haiyan Fan, Michael Chau, and Daniel Zeng, MetaSpider: Meta-Searching and Categorization on the Web, Journal of American Society for Information Science and Technology, Vol. 52, No. 13, 2001.
21. Luis Gravano, Chen-Chuan K. Chang, Hector Garcia-Molina, STARTS: Stanford Proposal for Internet Meta-Searching, Proceedings of ACM SIGMOD, 1997.
22. KRISTAL-2002 User's Manual V1.1, KISTI, 2002

A Domain Model for the IST Infrastructure

Jan B.M. Goossenaerts

Eindhoven University of Technology, Dept. of Technology Management, POBox 513, 5600 MB Eindhoven, The Netherlands J.B.M.Goossenaerts@tm.tue.nl

Summary. A model driven engineering (MDE) approach is positioned w.r.t. collaborations of multiple agents acquiring information society technology (IST) capabilities. Our focus is on the stakeholders of work systems, their adaptive cycles, and their aligned assets, computation independent models in particular. When referring to the state-of-the-art in the software application interoperability area, three "missing" fragments of the domain model for the IST Infrastructure are explored: a *Total Asset Methodology* (TAM), an *Extended Generic Activity Model* (EGAM), and a concept for *TOken-based eXecution In Knowledge Spaces* (TOXIKS).

1 Introduction

This paper positions a model driven approach in the context of cultural historical activity theory [20] and IT-reliant work systems [4]. It considers society, enterprises and persons as goal-oriented agents that acquire IT capability (also called IST instruments or IST infrastructure) [11]. Earlier preliminary results on architecture descriptions for an information infrastructure [12] are extended. Relevant state of the art is vast and a systematic recollection does not fit in the available pages for this paper. Interested readers are referred to INTEROP state of the art reports, or to the listed references. Focus in this paper is on fragments in the domain model that complement the current Model Driven Engineering "received view" and its modelling foundation. Particular focus is on the early phases of IST instrument acquisition in IT-reliant work systems. Total Asset Methodology (TAM) and Extended Generic Activity Model (EGAM) have a focus on generic requirements, for society and its members. The TOken-based eXecution In Knowledge Spaces (TOXIKS) execution concept has a focus on how TAM and EGAM can deliver.

2 Agents, IST Instruments, Infrastructure

An IST infrastructure consists of the information models, data, and information processing services and tools that are shared by the different autonomous agents that collaborate or interact in a community or society. The trend towards an ubiquitous information infrastructure builds on the connectivity and low-cost high-performance computing and communication facilities provided by computers, the Internet and wireless communications, ranging from Bluetooth to satellite-based. An IST infrastructure is defined for and embedded in a society to support (all) the society's members and communities.

The term *society* has the meaning of "all people, collectively, regarded as constituting a community of related, interdependent individuals". A *community* is "a group of people having interests or work in common, and forming a smaller (social) unit within a larger one." This definition thus covers enterprises, public bodies, municipalities, nations, sports clubs, schools, hospitals, etc. All members of the society are *persons* with equal rights. Each person may belong to several communities. A community has no member outside society. An ubiquitous IST infrastructure will support interactions that involve at least three kinds of agents and their IST instruments. At each level, one can apply the concepts of the IT-reliant work system. Humans or micro-agents use personal IST instruments. Their win conditions include a.o. empowerment, legal security, efficient operations, optimal propagation of change, minimal inconsistencies, data protection and privacy [6]. Meso-agents such as businesses, universities, or any other kind of organisation, have mission-oriented IST instruments. Their success depends on the support that members receive for their relevant actions, conform the processes or collaborations that have been enacted within it, conform the society's law or rules. E.g., the certification of a new type airplane by the relevant authorities, or the carrying out of tax payments and elections. Change must happen smoothly, without disrupting the community's processes, and with a minimal burden to its members.

Society, the (socio-industrial) eco-system in which micro-agents and meso-agents exist, has an IST infrastructure to share information, publicly, for certain missions, or in the context of contracts. The society as a whole pursues compliance to its enacted institutions and agreed upon policy goals (e.g. fair trade and protection of property in the global society). It could have goals such as rapid implementation of new "rules" or charters and it could use the subsidiarity principle to organize its institutions and ensure that each problem is addressed at the level at which it is common for all the lower-level stakeholders.

Each pair of agent and instrument has become a "software/data/knowledge intensive system" for which the standard IEEE 1471-2000[13] defines architecture.

Typically, each agent will deploy applications serving its interests. Maybury for instance, describes Collaborative Virtual Environments for distributed analysis and collaborative planning for intelligence and defense[16]. The DIISM conferences have been dedicated to the design of the information infrastructure systems for manufacturing and engineering enterprises [5]. Virtual communities in relation to Peer-to-Peer collaboration architectures are discussed in [15]. Section 3 proposes a synthetic change framework and adapted modelling primitives. Section 4 addresses missing fragments of the domain model for the IST Infrastructure.

3 Anchoring IST Instrument Acquisition by Models

The current state of the IST infrastructure is that physical view aspects of its architecture are better understood than the conceptual view aspects. A crisp problem statement can be found in DODAF [10,page 3-1]: *Requirements were often developed, validated, and approved as stand-alone solutions to counter those specific threats or scenarios, not as participating elements in an overarching system of systems. This approach fosters an environment in which DoD Components make acquisition decisions that, in a joint context are not fully informed by, or coordinated with other components. ..acquisition pipeline that is inefficient, time consuming, and does not support interoperability ...Additionally, acquisition management focuses solely on materiel solutions and does not adequately or fully consider the profound implications that changes in joint Doctrine, Organization, Training, Leadership & education, Personnel, and Facilities (DOTMLPF) may hold for the advancement of joint warfighting.*" This statement points at the broad-scope context in which interoperability problems emerge. Anchoring the IST instrument acquisitions w.r.t. available assets, including the goals, needs, domain models and context of work of the acquiring agents, is one of the drivers for the Total Asset Methodology (TAM). The TAM promise is that acquisition decisions can prevent the emergence of semantic interoperability problems, thus limiting the need for curative approaches [18] to legacy systems. The OMG-proposed Model Driven Architecture (MDA) puts the model, a specification of the system functionality, on the critical path of software development, prior to the implementation of that functionality on a specific technology platform. Beyond OMG-MDA, TAM aims for an end-to-end role of computation independent domain models in the re-engineering of work systems and the acquisition of IST instruments.

3.1 Models in Work System Change Projects

Intuitively, the vision of model driven engineering can be linked to a combination of Boehm's Win-win Spiral model [7] and Kruchten's 4+1 view model [14] of (software) systems architecture. This combination proved effective in several change projects in companies. The collaborative foundation of the Win-win spiral ensures that the end-users drive the IT capability acquisition. The 4+1 view model is adopted because the re-engineering of IT-reliant work systems are situated in an engineering context where a large portion of specifications (expressed as models), software systems and data, and hardware systems are (re-)used and/or have to inter-operate (in a software intensive system), and evolve over time.

For the specification of the domain of IT-reliant work systems we introduce *Activity Patterns* as a modelling formalism. It has traits of High Level Petri nets but adopts the concepts of Cultural Historical Activity Theory. UML Class diagrams are used for *entity modelling*. All models in the conceptual view are computational independent models (CIM) in the sense of MDA. The platform specific models (PSM) are part of the physical view.

Assume now that there is an existing work system (AS-IS) with identified stakeholder needs. Then the re-engineering spiral in Figure 1 is model enabled:

stakeholder needs are identified, analysis and design phases deliver an extended or new system specification, often with refinements in the logical view and the activity patterns. Development and implementation deliver the TO-BE physical realization meeting the identified needs. The logical view models, activity patterns and system specifications are soft assets, maintained and available for future change projects.

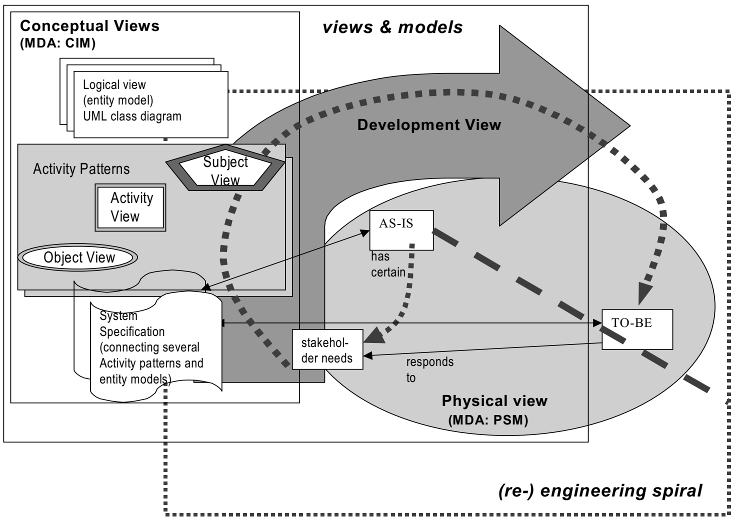


Figure 1. A re-engineering spiral anchored by views and models

3.2 Introducing Activity Patterns

Cultural-Historical Activity Theory (CHAT) is suitable to perform contextual analysis for cognitive processes in which the cognition is embedded in broader institutional structures and long-term historical trajectories of development and change[20]. Concepts and tools that the society has developed during its history culturally mediate interactions of the human in the world. For analyzing an *activity*, we must consider its *subject*, the entity performing the activity, and its *object,* the necessary entity that allows realizing the outcome. A tool can be anything used in the transformation process, including both material tools and cultural mediators. The cultural mediators or artifacts that individuals (subjects) use also carry the typical intentions and objectives of people in specific situations. CHAT regards enterprise and society development as a process of remediation: the substitution of old mediating artifacts (for instance sentences on papers and in documents) with new artifacts (including the IST instruments), which better serve the needs of the activity concerned. Remediation means that the external objects are seen in a new context.

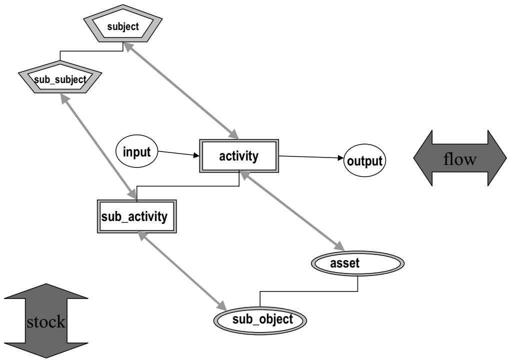


Figure 2. The building blocks and arcs of Activity Patterns

Drawing on the CHAT conceptual framework the Activity Pattern modelling formalism articulates three primitive building blocks: the *activity block* (rectangular), the *object block* (oval) and the *subject block* (pentagonal). Between blocks of the same kind, the arc (└) denotes a *sub-block* relationship (e.g., a sub-object is part of an object). An arc associating an activity block to an object block denotes an *involvement* (of the object in the activity). Directed arcs may be used to express that an object is the output or the input of an activity. An arc associating a subject block to an activity block denotes a *participates* relationship (active involvement, the subject performs the activity). A subject structure consisting of a hierarchical structure with several subject blocks, can be used to describe an organisational structure: in this case the subject blocks represent organisational elements or units. An activity structure (a hierarchical structure with several activity blocks) corresponds to a work break down structure. A product structure can be represented as an object structure (a hierarchical structure with several object blocks). Activities can take objects as input and produce outputs. For activities, the distinction between reliance on assets (stock) and the consumption or production of objects (flow) is represented by connecting the arc to a different side of the rectangle: left or right side for flow; bottom side for reliance on assets. Subject blocks are linked to the topside to express participation in the activity. In this paper we do not address the allocation of activities to subjects.

A first characteristic of the activity pattern is its generic aspect: all blocks are *open*, and can be refined at any time. For a given work system, multiple activity patterns must be specified. The transitions of asset tokens (e.g., a databases) or moves of flow tokens (e.g., a cases) are synthesized from the specifications of multiple activity patterns. While the Activity Pattern has similarities with many other models, its distinguishing features include symmetric treatment of object, activity and subject, compositional properties that reflect epistemic pluralism, and a decoupled token based execution concept explained later in the paper. In what follows we liberally build upon the semantic constructs of Colored Petri nets to explain TOXIKS.

4 Total Asset Methodology for the IT-reliant Society

To explain TAM for IT-reliant agents in an information society we use GERA phases (Generic Enterprise Reference Architecture, Annex A to ISO 15704) to describe the agent's adaptive cycle. Regarding the model enabled aspect of TAM, we use the OMG MDA terms. After briefly introducing relevant elements from these frameworks, two specific themes are highlighted to illustrate the value-focussed flexible inter-operation of agents: (i) EGAM as a decisional reference model that emphasizes the frequent need for change in the work systems, and (ii) TOXIKS as an execution concept that reconciles stability in operations with dynamism in the knowledge spaces.

4.1 Asset Alignment in the IT-reliant Society

GERA names the phases in the adaptive cycles in which the S-, E, and P-agents cope with new necessities. It is convenient to introduce S-GERA, E-GERA and P-GERA phases and name them: S-identification (S_I: Identification for Societal level), E-identification (E_I: Identification for Enterprise level) and P-identification (P_I: Identification for Person level), etc. The asset alignment activity is indicated for the objects supporting the respective GERA phases. All GERA phases for the society, enterprise and person agents may be ongoing. Each phase maintains specific assets to produce its outcome.

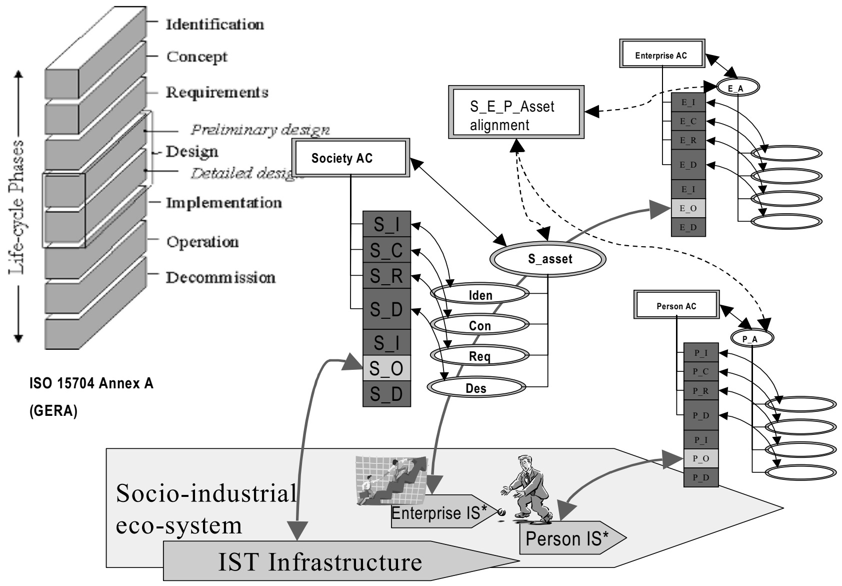


Figure 3. GERA phases, assets and IST instruments for society and its members

The activity patterns in Figure 3 are in the Conceptual View of Figure 1. The arrows in the lower part of Figure 3 give an abstract picture of the socio-industrial eco-system, its members and their IST instruments in the Physical View. The observed meso-agent variety in society is in the physical view. It is a result of "close-to-biological" evolution as comprehensively described by McCarthy et al. [17].

For any goal-oriented agent, the adaptive cycle is overseen by a planning process. Russell Ackoff defines planning as "a process that involves making and evaluating each of a set of interrelated decisions before action is required, in a situation in which it is believed that unless action is taken a desired future is not likely to occur, and that, if appropriate action is taken, the likelihood of a favourable outcome can be increased" [2].

4.2 TAM for IT-reliant work systems: Model Driven Engineering

The adaptation of IT-reliant work systems faces two huge hurdles [11]: socio-diversity and techno-diversity. To illustrate these hurdles, consider the society goal of reducing greenhouse gas emissions. The diversity at the operational layer is evident: gasses are emitted in a myriad of different situations. The people and businesses that are within the scope of any measure use multiple technologies and (software) solutions. This complicates the enactment of measurement and trading schemes such as proposed in Kyoto Protocol implementation schemes.

For overcoming similar hurdles, businesses have made explicit (externalised) their structure and operating procedures, especially with a focus on computer support for improved operations. These trends have already given rise to the large-scale use of enterprise models and the use of several dimensions to manage the complexity of enterprises applying ICT [3]. Enterprise Architecture tools are gaining importance in the market, and focussed architectural frameworks are being developed [10].

In the TAM road towards a knowledge society, the model and data assets will play a key role in designing and implementing policy measures in a calm manner, in accordance with the relevant legal principles, and for the available technology. As consolidated models are (becoming) available for the socio-technical contexts of work, any adaptive cycle (project) will deliver a "delta-specification" to realize a particular new scenario in a given socio-technical context. For a given subject (S, E or P) and its work system, the models at the three MDE layers (computation independent, platform independent and platform specific) result from different GERA phases, and are part of different asset layers. In the planning perspective each asset layer offers its own contribution to the reduction of risks [9] and to the system design. The Computation Independent Model (CIM) shows the system in the environment in which it will operate, and thus helps in presenting exactly what the system is expected to do (Concept). It is useful as an aid to understanding a problem and for communication with the stakeholders, it is essential to mitigate the risks of addressing the wrong problem, or disregarding needs. The use of platform independent and platform specific models (PIM and PSM) mainly matter when IST instruments are part of the solution.

4.3 A Decisional Reference Model

The model driven engineering (MDE) has no internal mechanism to identify problems in the work system. For the goal driven agent, IST instrument acquisition should be problem driven and asset enabled. In general, a problem refers to a situation, condition, or issue that is unresolved or undesired. In *society*, a problem can refer to particular social issues, which if solved would yield social benefits, such as increased harmony or productivity, and conversely diminished hostility and disruption. In *business* a problem is a difference between actual conditions and those that are required or desired. We assume that the values held by society are related to the so-called livelihood assets: *human, natural, physical, social* assets in addition to *financial* assets. Given an indicator system, performance objectives are expressed and evaluated for a work system (object system) that performs a function. The environment is the source of inputs and the sink (market) for the outputs. The model in Figure 4 is called an *Extended Generic Activity Model* (EGAM) because it also includes the reflective activities that influence the operations of an object system. The governance activity, the management activity and the analysis&design activity support reflective functions of determining/setting the artefacts (objectives, problem, etc.) linked from their right-hand side. A quantitative difference between objectives and performance data signals a problem to the management activity. In pull-based change, the management activity will call upon the analysis&design activity to analyse the problem of the object system, to create new designs (TO_BE model & technology), and to compare performance. Governance and management activities decide about the implementation and acquisition of new capability proposed by the analysis and design activity in a management or governance advice (m_advice or g_advice). In particular, the activities are defined as follows:

- The *object-system operation*: The operational processes that are performed by the object system, and for which performance *objectives* are expressed and evaluated,
- The *environment processes*: The processes of the environment in which the object system operates or performs a function – the environment is the source of *inputs* and the sink (market) for the *outputs*. Also *resources* originate from the envi-ronment, and *wastes* are deposited there (not in the Figure),
- The *governance activity*: The activity in which objectives (stylistic and performance) are expressed for the object system, taking into consideration relevant constraints (natural, social, etc.) that exist for the capital assets in the factory's environment, it gives a mandate to the management activity.
- The *management activity*: The activity in which the operations of the object system are monitored and controlled. If one or more performance targets are not met, a problem is signalled to the analysis & design activity,
- The *design and analysis activity*: In this activity, performance problems of the object system are analysed, redesigns performed and evaluated, and advices given to the management or governance activity deciding which solution to adopt.

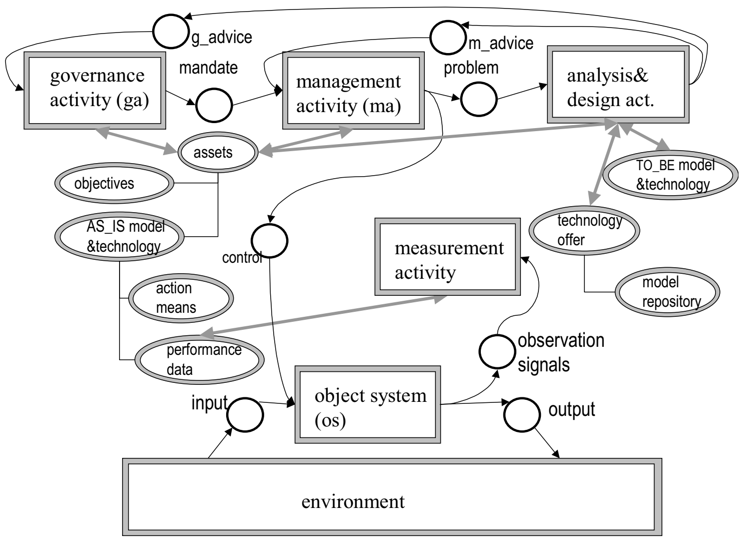


Figure 4. The Extended Generic Activity Model (EGAM)

4.4 Token based Execution in Knowledge Spaces (TOXIKS)

Work systems must be adaptive to survive. EGAM draws one possible picture of the adaptive cycle. GERA draws another one. In the vision of TAM, multiple models and data must be reused. These models must also evolve. How can work system operations be liberated from the tyranny of the models in systems modelling? This question generalizes the question on instances and classes asked by Parsons and Wand [19]. TOXIKS is a tentative answer that draws on Bunge's distinction between ontology and epistemology[8] and generalizes the emancipating guidelines of Parsons and Wand. In essence, the token or instance is seen as an ontological construct, and the activity pattern or class as an epistemic entity. Such interpretations are generalized to transitions (actions, as instances of "class-like" activities), and to subjects (actors or agents as instances of subject-classes). A *Knowledge Space* is composed of some domain models (in UML), some activity patterns, and system specifications that specify transitions and related token classes over the activity patterns and the domain models (as in Colored Petri nets). Hence, a knowledge space is an epistemic construct. It usually is shared within an enterprise, a community, a science discipline, or culture.

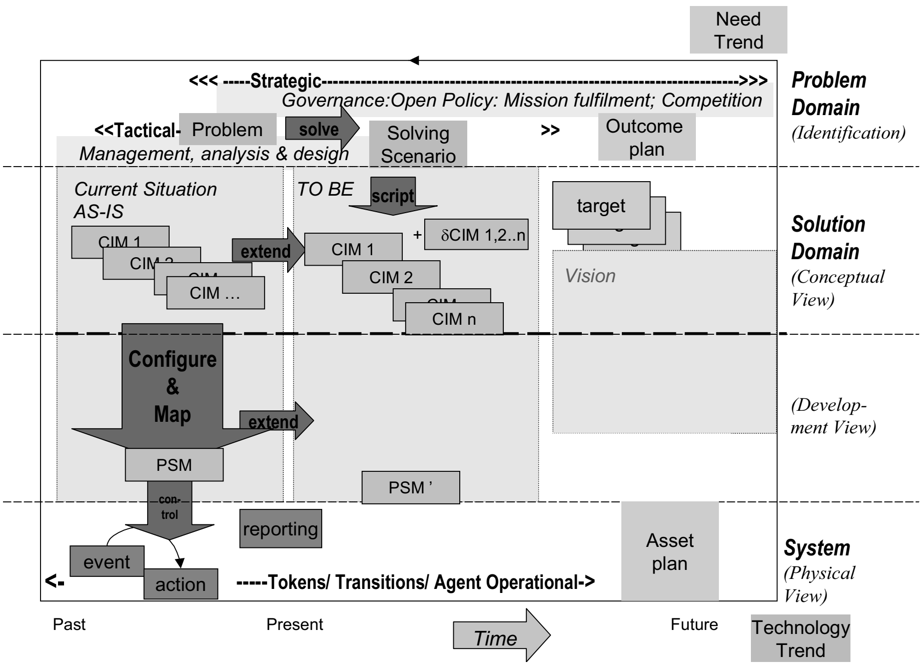


Figure 5. Operations under Changing Knowledge Spaces.

A Knowledge Space is expressed as a CIM model, and defines partial meanings for operations and tokens as sketched in Figure 5. The knowledge spaces allow epistemic pluralism: multiple knowledge spaces coexist for a single object of analysis or discourse. Events in the work system are interpreted differently in the knowledge spaces pertaining to business (management), multiple science disciplines, or societal monitoring systems, a.o. The workflow or work system operations (system row in Figure 5) is token based and has an ontological status. The token game is relatively stable and bound by laws in the hosting ontological stratum [1]. In contrast the knowledge spaces are highly adaptable, expandable and even dispensable for the operations. Their purpose lies in planning, though.

The development of the knowledge spaces is "driven" by the adaptive cycle initiated in a management or governance activity: a problem is identified. As part of an advice, the analysis&design activity proposes a scenario that solves the problem (pull style) or explains a promise (push style). The execution of the IT-reliant agent's MDE or GERA phases inludes the scripting of scenario's by specifying δCIM models w.r.t. the consolidated reference CIM models and by selecting suitable knowledge spaces (CIM'=CIM+δCIM); the δCIM is mapped to δPSM models; the δPSM models are included in the action prescriptions that will control the event flow and support the reporting demands. For instance, if there is a need for additional (new) measurements of ongoing event streams (ontologically, the same work system operations or primary process), then the measurements are defined as *δ-actions* for selected events. Information about these events is recorded in accordance with the (new) interpretative structures (knowledge spaces, epistemic commitments) defined in δCIM and mapped to δPSM.

5 Conclusion and Future Work

This paper has proposed consistent fragments of the computation independent models of the domain that goal-oriented agents and their IT-reliant work systems share. The proposed models may evolve into assets in an IST infrastructure. A Total Asset Methodology (TAM), an Extended Generic Activity Model (EGAM) and an execution concept (TOXIKS) have been proposed. TOXIKS reconciles ontological invariance with epistemic pluralism and dynamism, and thus is a necessary feature of TAM at work. The validity of the proposed models and concepts must be further demonstrated. MDE-GERA style planning and development processes must be performed with aligned assets, multiple scenario's must be scripted with respect to these assets, and TOXIKS must be tested in IT-reliant work systems.

References

1. Abramov V.A., J.B.M. Goossenaerts, P. De Wilde, L. Correia (2005) Ontological stratification in an ecology of infohabitants. In: E. Arai, J. Goossenaerts, F. Kimura, K. Shirase, (eds.) *Knowledge and Skill Chains in Engineering and Manufacturing: Information Infrastructure in the Era of Global Communications*, Springer, pp 101-109.
2. Ackoff, R.L. (1970) A Concept of Corporate Planning, Wiley-Interscience.
3. Aerts, A.T.M., J.B.M. Goossenaerts, D.K. Hammer & J.C. Wortmann, 2004, "Architectures in context: on the evolution of business, application software, and ICT platform architectures," *Information & Management*, vol. 41, no. 6.
4. Alter, S (2003) 18 reasons why IT-reliant work-systems should replace the "IT-artifact" as the core subject matter of the IS field. Communications of the Association for Information Systems, Vol. 12, 2003, pp. 366-395.
5. Arai, E., J. Goossenaerts, F. Kimura, K. Shirase, editors (2005) *Knowledge and Skill Chains in Engineering and Manufacturing: Information Infrastructure in the Era of Global Communications*, Springer.
6. Berkers, F., Goossenaerts, J., Hammer, D.K., Wortmann, J.C., 2001, Human Models and Data in the Ubiquitous Information Infrastructure. In: H. Arisawa, Y. Kambayashi, V. Kumar, H.C. Mayr, I. Hunt (eds) Conceptual Modeling for New Information Systems Technologies. LNCS 2465, Springer Verlag, pp 91-104
7. Boehm, B., Egyed,A., Kwan,J., Port,D., Shah,A., Madachy, R. (1998) Applying the WinWin Spiral Model: a Case Study, IEEE Computer, July 1998, pp 33-44.
8. Bunge, M. (1983). Epistemology & Methodology I: Exploring the World, Treatise on Basic Philosophy Vol. 5, Reidel, Boston.
9. Dick, J., J. Chard, 2003, Requirements-driven and Model-driven Development: Combining the Benefits of Systems Engineering, Telelogic White Paper, www.telelogic.com
10. DoDAF (2004) DoD Architecture Framework Working Group: DoD Architecture Framework Version 1.0, Volume 1: Definitions and Guidelines, 9 February 2004.

11. Goossenaerts, J.B.M. (2004) Interoperability in the Model Accelerated Society (2004) in: P. Cunningham and M. Cunningham (2004) eAdoptation and the Knowledge Economy: Issues, Applications, Case Studies. IOS Press Amsterdam, pages 225-232.
12. Goossenaerts, J.B.M. (2005) Architecting an ubiquitous and model driven information infrastructure. In: E. Arai, J. Goossenaerts, F. Kimura, K. Shirase, (eds.) *Knowledge and Skill Chains in Engineering and Manufacturing: Information Infrastructure in the Era of Global Communications*, Springer.
13. IEEE. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Std 1471-2000.
14. Kruchten, P., Architectural Blueprints - The “4+1” View Model of Software Architecture, IEEE Software, November 1995, 12 (6)
15. Lechner, U. (2002) Peer-to-Peer beyond File Sharing. In: Unger,H., Boehme,T.,& Mikler, A. Innovative Internet Computing Systems, LNCS 2346, pp. 229-249, Springer Verlag.
16. Maybury, M. (2001) Collaborative Virtual Environments for Analysis and Decision Support, Communications of the ACM, Dec. 2001, 44(12) pp. 51-54
17. McCarthy, I., K. Ridgway, M. Leseure, Fieller, N. (2000) Organisational diversity, evolution and cladistic classifications, Omega, The International Journal of Management Science, 28, pages 77-95
18. Park, J. & Ram, S. (2004) Information Systems Interoperability: What lies beneath? ACM Transactions on Information Systems, 22(4) pp. 595-632
19. Parsons, J. & Wand, Y. (2000) Emancipating Instances from the Tyranny of Classes in Information Modeling, ACM Transactions on Database Systems, 25(2) pp 228-268
20. Virkkunen, J. & K. Kuutti, "Understanding organizational learning by focusing on "activity systems"," *Accounting, Management and Information Technologies*, vol. 10, no. 4, pp. 291-319, Oct. 2000.

Towards a New Infrastructure Supporting Interoperability of Information Systems in Development: the Information System upon Information Systems

Thang Le Dinh

University of Geneva, CUI, 24 General Dufour, CH-1211 Geneva 4, Switzerland
Thang.LeDinh@cui.unige.ch

Summary. This paper addresses the issue of interoperability of information systems (IS) in development. Accordingly, an Information System upon Information Systems (ISIS) is proposed as a new infrastructure to manage and coordinate information resources used in IS development process. The proposed approach considers that conceptual specifications of information systems are the fundamental constituents of information resources; and therefore, the first challenge is to manage and coordinate conceptual specifications. Correspondingly, the conceptual framework for building and managing the ISIS is presented, including how to identify, represent, manage and coordinate conceptual specifications.

1 Introduction

Nowadays, interoperability of information systems (IS) is one of the most interesting challenges of the IS community. Consequently, various research and industrial approaches have been focused on providing more connectivity among information systems. Unfortunately, providing interoperability of systems and tools at the Informatics level is not sufficient because systems and tools weren't designed to interoperate at this level [1, 8, 9].
In our point of view interoperability of information systems should be managed at the higher level of abstraction, independent with technologies and choices of implementation. For this reason, we suggest a new layer called **Information layer,** which plays the intermediate role between the *Business layer* and the *Informatics layer* (Figure 1) to control and support interoperability of information systems.

The Information layer will rep-resent concepts of IS for effective computation and coordination. In other words, this layer represents the formal explicit specifications of concepts of IS which refer to an abstract model of how people think about objects in the information system context.

The objective of this paper is to propose a typical information system that supports the activities of the Information layer, in particular the activities concerning interoperability of information systems in IS development process. This information system is called the **Information System upon Information Systems** (ISIS) [4], which supports interoperability of information systems in development. The ISIS will participate in the IS infrastructure hierarchy of each enterprise as a new infrastructure and will coexist with other infrastructures.

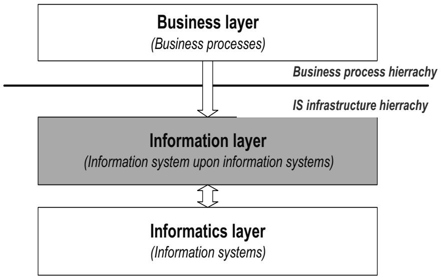

Business layer
(Business processes)
Business process hierrachy
IS infrastructure hierrachy
Information layer
(Information system upon information systems)
Informatics layer
(Information systems)

Figure 1. The information layer

Furthermore, we also argue that *conceptual specifications*, representing the concepts of IS, are bound to play an important role in the deliverables that are consumed or produced in the IS development process.. In our approach, specifications, especially conceptual specifications, are used to capture the knowledge about information systems and used to manage and operate interoperability of information systems at the Information level.

Indeed, conceptual specifications, which represent the semantic content of information at the Information level, are the *key resources* used in IS development process. Those key resources are often used to construct other information resources, which are produced and consumed by development activities. Besides, most technical specifications are the implementations of conceptual specifications. Consequently, once the interoperability of conceptual specifications is settled at the Information level, the interoperability of the technical specifications and business data at the Informatics level will be solved accordingly.

Accordingly, this paper continues with the *conceptual framework for building and managing the ISIS*, including how to identify, represent, manage and coordinate conceptual specifications of information systems.

The remaining of this paper is proposed as the followings: Section 2 presents how to identify different categories of conceptual specifications. Section 3 discusses about the representation of conceptual specifications. Section 4 concerns with the management of conceptual specifications. Section 5 describes how to coordinate specifications within and between development projects. Section 6 comes to end with conclusion and future works.

2 Identification of Conceptual Specifications

This paper uses the *M7 method* [2, 3, 4] as the tool for representing the framework. The M7 method proposes several models to represent different aspects of an IS such as the Static, Dynamic, and Integrity rule aspects.

The Static aspect represents the structure of information; the Dynamic aspect represents the transformation of information; and the Integrity aspect represents the coherence of information.

2.1 Conceptual Specifications of the Static Aspect

Conceptual specifications of the Static aspect describe what type of information exists, their structures as well as their interrelationships. There are the categories of conceptual specifications of the Static aspect such as Atomic-class, Tuple-class, Hyperclass, Attribute, Key, and Sub-hyperclass.

An object type and a set of objects of this type define a class. There are three kinds of classes: Atomic-class, Tuple-class and Hyperclass:
An **atomic-class** is defined as a primitive class, which is indecomposable. Objects of an atomic-class have a particular characteristic: their identifier is also their value.
A **tuple-class** contains objects having the same structure and the same behaviour. A structure of a tuple-class is characterized by a set of attributes. The behaviour of a tuple-class is represented by a set of methods.
A **hyperclass** is a subset of classes that all connected by navigation links to a *key class* without ambiguity [5]. We can work on a hyperclass as a tuple-class. Consequently, a tuple-class is a specialisation of a hyperclass.

The interrelationships between the classes' concepts lead to other concepts such as Attribute, Key and Sub-hyperclass:
An **attribute** of a hyperclass is a function that corresponds to every object of this hyperclass to a set of objects of the other class.
A **key** of a hyperclass is defined by a set of special attributes can be used to distinguish one object from other objects in the same hyperclass.
A hyperclass can define its **sub-hyperclasses**. The interpretation of a sub-hyperclass is exactly the set of all identifiers of the interpretation of its super-hyperclass for which the specialisation condition evaluates to be "true".

2.2 Conceptual Specifications of the Dynamic Aspect

In the M7 method, there are two levels of behaviour: *Local behavior* defined as the behaviour of objects of a hyperclass, and *Global behavior* defined as the behavior of the IS or a part of IS. The categories of conceptual specifications of the Dynamic aspect are Dynamic state, Method, Event, and Process.

The local behaviour is represented by the concepts of *Dynamic State* and *Method:*
Dynamic states of an object are modes or situations during which certain methods are "enabled" and other methods are "disabled".
A **method** of a hyperclass is used to transit between the dynamic states of the objects of this hyperclass. In a clear manner, a method transfers a set of dynamic states to another set of dynamic states of a hyperclass.

On the other hand, the global behaviour is represented by the concepts of *Event* and *Process*:

Event is remarkable phenomena outside of the information system that may provoke a change of its dynamic states. In fact, the event structure helps to define the interface of the IS with its environments.
A **process** is a feedback of the IS to the occurrence of an event. In fact, a process performs a transformation of a set of dynamic states of the IS.

2.3 Conceptual Specifications of the Integrity Rules Aspect

The categories of conceptual specifications of the Integrity rule aspect are Integrity rule, Scope, Primitive and Risk.
In fact, the Integrity rule aspect includes the concepts such as *Integrity Rule* and *Primitive* as well as their interrelationships such as *Scope* and *Risk* [6]:
Integrity rules (IR) of an IS often represent the business rules of an organization. An IR actually is a logical condition defined over tuple-classes that can be specified formally and verified by processes or methods.
Scopes of an IR represent the context of an IR including a set of tuple-classes on which the IR has been defined.
A **primitive** is a basic operation on a tuple-class such as such *Create*, *Update* and *Delete*. The execution of a primitive may violate the validation of an IR.
Risks are the possibilities of suffering the incoherence of information. In principal, a risk is defined on a scope and a primitive. In particular, especially in the case of the *Update* primitive, it is indispensable to specify the related attributes of a risk.

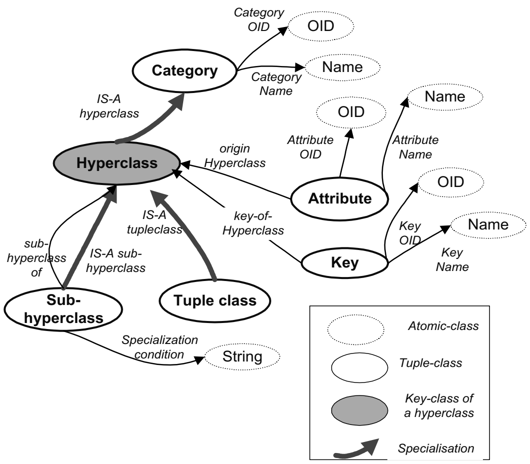


Figure 2. Structure of Hyperclass generic hyperclass.

3 Representation of Conceptual Specifications

This section concerns with how to represent the structure, the behavior and the coherence of conceptual specifications.

3.1 Structure of Conceptual Specifications

In the ISIS, a conceptual specification is represented by an object. Each category of conceptual specifications is represented by a hyperclass of the ISIS. Hyperclasses of the ISIS are called ***generic hyperclasses.*** Table 1 proposes the generic hyperclasses.

Table 1. Generic hyperclasses.

Aspect	***Generic hyperclass***	***Key class***	***Constituent classes***
Static aspect	Atomic-class	Atomic-class	Category
	Hyperclass	Hyperclass	Category, Tuple-class, Sub-hyperclass, Attribute, and Key
	Attribute	Attribute	Category, and Hyperclass
	Key	Key	Category, Hyperclass, and Attribute
Dynamic aspect	Dynamic state	Dynamic state	Sub-Hyperclass, and Hyperclass
	Method	Method	Dynamic state, Sub-hyperclass, and Hyperclass
	Event	Event	
	Process	Process	Event, Hyperclass, and Method
Integrity rule aspect	Integrity rule	Integrity rule	Integrity rule, Scope, and Risk
	Scope	Scope	Integrity rule, Tuple-class, and Hyperclass
	Risk	Risk	Scope, Primitive and Attribute

Notes: *Category* tuple-class is the generalization of *Atomic-class* tuple-class and *Hyperclass* tuple-class. An object of *Category* tuple-class can be an object of *Atomic-class* tuple-class or an object of *Hyperclass* tuple-class.

For illustrating, the next example presents the *Hyperclass* generic hyperclasses.

Example: Structure of *Hyperclass* specifications.
Hyperclass hyperclass is the generic hyperclass of the ISIS that represents the specifications of hyperclasses of information systems.

Hyperclass hyperclass is defined over its key class: *Hyperclass* tuple-class.

Objects of Hyperclass tuple-class are objects of *Category* tuple-class. In the same manner, objects of *Sub-hyperclass* and *Tuple-class* tuple-classes are also objects of Hyperclass tuple-class and therefore objects of *Category* tuple-class.
From an object of *Hyperclass*, one can navigate to an object of *Sub-Hyperclass* tuple-class (using its *sub-hyperclass-of* attribute), a set of objects of *Attribute* tuple-class (using the *origin-Hyperclass* attribute), and a set of objects of *Key* tuple-class (using the *key-of-Hyperclass* attribute).

3.2 Coherence of Conceptual Specifications

The coherence of conceptual specifications is guaranteed by the integrity rules of the ISIS. In our framework, the integrity rules of the Figure 2. Structure of Hyperclass generic hyperclass.
ISIS are called ***generic integrity rules***. Indeed, these integrity rules concerns with the conformity of conceptual specifications.

An object of a generic hyperclass of the ISIS is said: "to be conformed" if it is satisfied all the generic integrity rules. There are two types of generic integrity rules: *validity* and *completeness rules*.

The concept of **validity rule** is actually inherent to the concept of "integrity rule" at the meta-model level. Indeed, there is a set of rules coordinating with the meta-model to control the validity of every object of generic hyperclasses.

When designer modifies an object of a generic hyperclass, this modification may violate the validity rules, which are concerned about this object. If the modification violates one of those rules, the object is brought into the *invalid* state. On the contrary, it is in the *valid* state.

For instance, there is a validity rule related to objects of Hyperclass generic hyperclass such as: "*The dependent constituents of a hyperclass such as its attributes, keys, and sub-hyperclasses must be valid"*.

The concept of **completeness rule** is related to the perception about the organization and the real world that designers have to realize. In fact, the decision of designers about the completeness of a conceptual specification depends on the finish of works and the stability of the constituents of the real world, which are modelled with that specification. Therefore, designers who are responsible for managing completeness rules will decide their completeness status.

For example, "*a hyperclass must have all its attributes*" is a completeness rule. In this case, the designer, who decides whether all the "necessary" attributes of that hyperclass are already specified or not, will specify its completeness status.

3.3 The behaviour of Conceptual Specifications

The behaviour of conceptual specification is represented by the generic dynamic states of conceptual specifications and the corresponding object life cycles.

A ***generic dynamic state*** is a dynamic state of the ISIS that is common for all the object life cycles of categories of conceptual specifications. In other words, those dynamic states exist in all the object life cycles.

We propose the following dynamic states: *Ready to initialize, Initialized, Valid, Invalid, Completed, Uncompleted, Implemented,* and *Unimplemented.*

Firstly, every object of a generic hyperclass, which represents a conceptual specification, has two dynamic states: **Ready to initialize** before its initiation and **Initialized** after its initiation.

Secondly, when an object is initialized, it is said: "to be conformed" if it is satisfied all the conformity rules, including validity and completeness rules.

Therefore, the next dynamic states are proposed:
Valid / **Invalid** to state that an object is satisfied /dissatisfied all the concerned validity rules;

Completed / **Uncompleted** to state that an object is satisfied /dissatisfied all the concerned completeness rules.

Finally, the dynamic states to indicate that an object of a generic hyperclass is implemented (or not) are also necessary. Consequently, the two generic dynamic states: **Implemented** and **Unimplemented** are also proposed.

In the ISIS approach, the development process of a conceptual specification can be generally represented by a ***generic object life cycle***. A method of the ISIS that may change the generic dynamic states of an object is called a ***generic method.***

Figure 3 presents the generic object life cycle using the Petri-net [7].

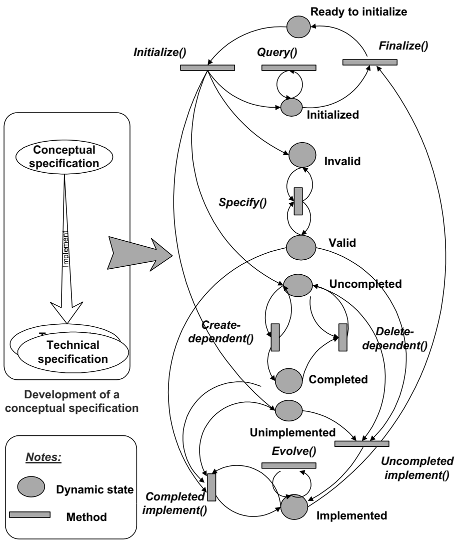


Figure 3. A generic object life cycle.

An object before its initiation is in the Ready to initialize state. After its initiation (using Initialize() method), the object is in the Initialized state and ready for processing by other methods. For instance, it is ready to be queried by the Query() method. When an object is created, it is also in the Invalid, Uncompleted and Unimplemented states.

A set of Specify() generic methods can be used to bring an object from the Invalid state into the Valid state or vice versa. Accordingly, a set of Create_dependent() generic methods can be used to bring an object from the Uncompleted state into the Completed states. On the other hand, the Delete_dependent() generic method may bring an object return to the Uncompleted state.

An object can be implemented when it is in *Valid* state. When an object is in the *Unimplemented* state, there are two possibilities:

This object is valid but has not yet completed, however, the responsible person decides to implement it. In that case, an *Uncompleted-implement()* method change its state into *Implemented*;

This object is *Valid* and *Completed*, and then a *Completed-implement()* method changes its state into *Implemented.*

Moreover, when an object is in the *Implemented* state, there are two possibilities:

- This object may be completed or not yet completed, therefore a *Completed-implement()* method or *Uncompleted-implement()* method can be executed. Those methods do not change its state;
- There is a need for an evolution. In that case, the *Evolve()* method can perform the evolution primitives. This type of methods does not change the state of object.

Finally, an object in the *Initialized* state, probably implemented or not implemented, can be finalized by using the *Finalize()* method. This method brings the object return to the *Ready to initialize* state.

4 Management of Conceptual Specifications

This section firstly present how the ISIS store conceptual specifications, then continues with the overall architecture of the ISIS.

4.1 Organizational Aspect of the ISIS

An excerpt of the meta-model of the organizational aspect of the ISIS presents how the ISIS store and manage the conceptual specifications (Figure 4).

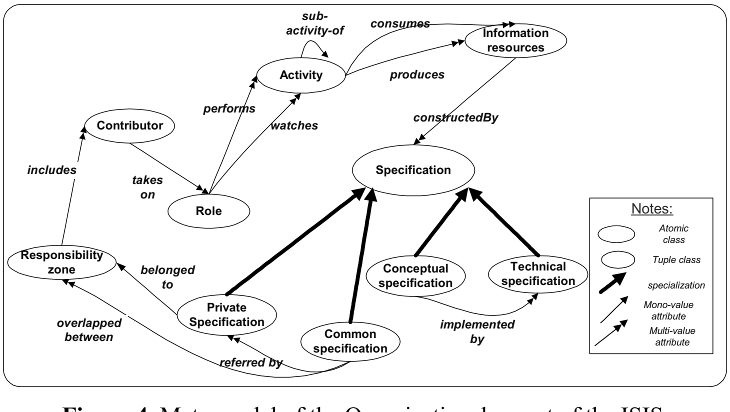


Figure 4. Meta-model of the Organizational aspect of the ISIS.

Concerning the levels of abstraction, there are two types of specifications: *Conceptual specification* and *Technical specifications*. Technical specifications represent the internal design of the information system, turned to achieve reasonable performance on the target platform. A conceptual specification can be implemented in one or several technical specifications.

An *information resource* is a package of specifications that are consumed or produced by the activities of the development process.

An *activity* is defined as a unit of work that may produce or consume certain information resources. Activities can be nested: one activity can expand into several activities at a lower level. An activity can be performed by a set of roles and can be watched by another set of roles.

Roles represent a set of necessary responsibilities, authorities and capabilities to perform the execution of activities or to watch (monitor) the execution of activities performed by the other roles.

A *contributor* is a person that participates in the IS development process. A contributor may take on several roles.

A *responsibility zone* (RZ) is a part of working environment that may carry out an IS development project. A RZ includes a set contributors associated together.

Concerning the interest of RZs, there are two specializations of specification: *Private specification* and *Common specification*. A private specification is belonged to only one RZ. Meanwhile, a common specification is overlapped between several RZs. Consequently, a common specification can be referred by several private specifications.

4.2 Overall Architecture of the ISIS

Figure 5 introduces the overall architecture of the ISIS. In the ISIS, specifications are stored in the ISIS repository. This repository includes two workspaces: *Private workspace* and *Common workspace*. Private workspace stores private specifications of responsibility zones. Meanwhile, common workspace stores common specifications.

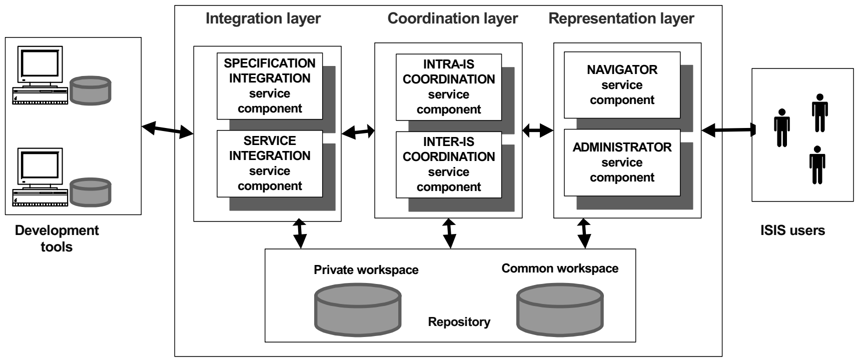

Integration layer
Coordination layer
Representation layer
SPECIFICATION INTEGRATION service component
SERVICE INTEGRATION service component
INTRA-IS COORDINATION service component
INTER-IS COORDINATION service component
NAVIGATOR service component
ADMINISTRATOR service component
Development tools
ISIS users
Private workspace
Common workspace
Repository

Figure 5. Overall architecture of the ISIS.

On the other hand, there are three layers that provide facilities to represent, validate, manage, integrate and coordinate specifications such as the Integration, Coordination and Representation layers.

Integration layer provides the facilities to integrate specifications stored in development tools and specifications stored in the ISIS. This layer includes:
Specification-integration service component: providing facilities to integrate specifications stored in tools with specifications stored in the ISIS;
Service-integration service component: including facilities to integrate services provided by tools and services provided by the ISIS.

Representation layer provides the facilities so that ISIS users, developers and administrators can work with specifications stored in the ISIS repository, including:
Navigator service component: providing the interface to allow the ISIS users to specify and complete the specifications;
Administrator service component: providing facilities to support the ISIS administrator.

Coordination layer provides the facilities to support interoperability of specifications within and between responsibility zones. This layer includes:
Intra-IS coordination service component: supporting interoperability of private specifications of the same RZ;
Inter-IS coordination service component: supporting interoperability of common specifications of different RZs.

5 Coordination of Conceptual Specifications

In the following, we discuss about various situations of coordination and illustrate how an ISIS may support different situations of coordination. Indeed, there are two general situations of coordination: Intra-IS coordination and Inter-IS coordination. *Intra-IS coordination* is the coordination of private conceptual specifications managed by a same responsibility zone. Meanwhile, *Inter-IS coordination* is the coordination of common conceptual specifications shared by different responsibility zones.

5.1 Intra-IS Coordination

In the ISIS, the development process of a conceptual specification is represented by an object life cycle of the corresponding generic hyperclass. Therefore, the interdependencies between conceptual specifications can be analyzed based on the interdependencies of dynamic states of those life cycles.

In fact, when a conceptual specification changes its dynamic states, this change may lead to the changes of dynamic states of other conceptual specifications, which has the interrelationship with that conceptual specification.

For instance, "when a conceptual specification of a hyperclass becomes Valid, it is assured that the all its sub-hyperclasses, attributes, and keys must be in Valid dynamic state".

The impact as mentioned above can be implemented using coordination rules. In other words, to support the coordination of object life cycles of generic hyperclasses, the ISIS needs to guarantee the coordination rules, which represent the impact of the changes of dynamic states of objects of generic hyperclasses.

For instance, the next table presents the coordination rules related to the Hyperclass generic hyperclass. Those coordination rules concerns with the coordination of dynamic states of objects of *Hyperclass* and its dependent generic hyperclasses (such as *Sub-hyperclass*, *Attribute* and *Key*).

Table 2. Coordination rules concerning the coordination of objects of the Hyperclass generic hyperclasses with objects of other generic hyperclasses**.**

Rule	***Generic hyperclass***	***Description***
S_Cn#1	Hyperclass	If a specification of a hyperclass is in Valid state then the specifications of its sub-hyperclasses, attributes, and keys must be also in Valid state.
S_Cn#2	Hyperclass	If a specification of a hyperclass is in Invalid state then the specification of its parent hyperclass is also in Invalid state.
S_Cn#3	Hyperclass	If a specification of a hyperclass is in Completed state then the specifications of its sub-hyperclasses, attributes, and keys must be in Completed state.
S_Cn#4	Hyperclass	If a specification of a hyperclass is in Uncompleted state then the specification of its parent hyperclass is in Uncompleted state.

5.2 Inter-IS Coordination

Inter-IS coordination concerns with common conceptual specifications, which reflect the interdependences between different responsibility zones. These interdependences can be represented by *overlap situations* and overlap situations are operated by *overlap protocols*.

An **overlap situation** occurs when there is at least one class or one process is common to several RZs. There are three types of overlap situations:
Distinct: there is no common class and no common process between RZs.
With borders: there are common classes, but no common process.
With overlaps: there are common classes, and common processes, which perform operations on those common classes.

An **overlap protocol** is a protocol that allows a RZ to perform its own processes locally and to monitor the processes in other RZs, which can influence its own processes. In fact, an overlap protocol includes a set of semantics, rules, and formats that conduct the coordination of different RZs. At the time being, we propose the following categories of overlap protocols:

Ownership-based overlap protocol appoints which RZ would play the role of the owner for each common object. The owner of an object takes the responsibility for defining, developing and maintaining it. The other RZs may communicate to the owner to obtain information about this object.

Service-based overlap protocol appoints which RZ would play the role of the provider for each common object. The provider of an object takes the responsibility for providing services related to this object. This protocol allows other RZs to send a request to perform a process to the provider. Normally, the provider will perform the requested process and return the result to the requested RZ.

Watch-based overlap protocol allows a RZ to monitor the consequences when other RZs performed a process, which is overlapped between them. Indeed, each RZ plays the role of the co-owner for each common object.

6 Conclusion

In this paper, we have shown up the important of a new infrastructure: the *Information layer* to control and support interoperability of information systems. Moreover, we have presented the *Information System upon Information Systems* (ISIS), which supports the activities of the Information layer concerning the interoperability of information systems. In particular, we have also presented a conceptual framework for building and managing the ISIS based on conceptual specifications.

The contribution of our work is to provide a unique and coherent framework to represent, coordinate and validate conceptual specifications of information systems. The perspective of this work is to provide an effective architecture that would be best suited for the interoperability of existing tools and systems used in IS development.

In future work, we will focus our research on designing and building a tool that supports the development of the ISIS, in particular the modelling phase. This tool will help the IS professionals to adapt and to build their own ISIS conforming to their enterprise. For instance, they can define and represent different categories of specifications conforming to their development methods, and select the overlap protocols conforming to their culture, organization styles, and existing technologies.

References

1. M. Stonebraker, “Integrating Islands of Information”, EAI Journal, Sept 1999.
2. Th. Estier, G. Falquet, J. Guyot, and M. Léonard, “ Six Spaces for Global Information Systems Design ”, Proc. of IFIP Working Conference on The Object-Oriented Approach in Information Systems, Québec, Canada, October 1991
3. Th. Estier, "Intégration des spécifications dans la conception des systèmes d'information, Thèse de doctorat de l'Université de Genève, Faculté des Sciences Economiques et Sociales, 1996.
4. T.Le Dinh, “Information System upon Information System: A conceptual framework“, Ph.D thesis, No 577, Faculté des Sciences Economiques et Sociales, University of Geneva, 2004.
5. Slim Turki, "Hyperclass: Towards a new kind of independence of the method from the schema", ICEIS, Ciudad Real Spain, 2002.
6. M. Léonard, “Information System Engineering getting out of classical SystemEngineering”, keynote lecture, 5th ICEIS, Angers – France, April 2003.
7. J.L. Peterson, "Petri net theory and the modeling of systems", Prentice Hall, 1981.
8. Wilhelm Hasselbring, “Information system integration“, Communications of the ACM, Volume 43 Issue 6, June 2000.
9. Jinsoo Park, Sudha Ram, “Information systems interoperability: What lies beneath?”, ACM Transactions on Information Systems (TOIS), October 2004

Design Solutions for Interoperability Using a Process Manager

Paul Johannesson[1] Erik Perjons[1], Benkt Wangler[2] and Rose-Mharie Åhlfeldt[1]

[1] Department of Computer and Systems Sciences, Stockholm University/KTH
Forum 100, S-164 40 Kista, Sweden, pajo@dsv.su.se, perjons@dsv.su.se

[2] Department of Computer Science, University of Skövde, S-541 28 Skövde, Sweden, benkt.wangler@his.se, rose-mharie.åhlfeldt@his.se

Summary. The healthcare domain is in urgent need for solutions to making clinical and administrative systems, possibly belonging to different healthcare units, interoperable and hence making them deliver timely and correct information as needed in particular situations. Process manager technology allows making all actors (humans or information systems) involved in healthcare processes communicate along these processes. This paper argues that process manager technology is essential for achieving interoperability in healthcare, but that some serious problems need to be overcome to realise its full potential. A number of design solutions to address these problems are proposed.

1 Introduction

Healthcare personnel, such as nurses, physicians, administrators and managers, need a vast amount of clinical and administrative information to fulfil their tasks and to deliver high-quality, secure and efficient patient care [1]. To achieve such goals, the information must be on time, correct, up-to-date, and presented in a format that facilitates interpretation. Information about the patient may, however, be spread over the organisation, not giving a complete picture in one single place of the care of the patient (i.e., the patient's medical history, current diagnosis, planned investigations and therapies). Therefore, integrated information systems are fundamental in healthcare. The trend in Europe towards specialised and distributed patient care has also made it important to integrate information from different healthcare units, such as home care, primary care, hospitals and laboratories.

Non-integrated information in healthcare may lead to:

- That the business operations are more costly than they would have to
- That information needed for treating patients does not reach units in time
- That information is false or outdated
- That information is distributed to non-authorized personnel
- That the source of information is not recorded

Therefore, there is a need for:

- Solutions that facilitate analysis and improvement of healthcare processes within a unit and processes that cross boundaries between healthcare units
- Solutions that facilitate integration of IT systems within and between units
- Solutions that facilitate analysis of security aspects (e.g., availability, confidentially, and accountability) as well as quality aspects of the information

A possible solution would be to introduce a process manager. Process manager technology transparently integrate existing IT system by executing graphical process diagrams that visualise the integration. The process diagrams executed in the process manager also reflect and support healthcare processes within and among healthcare units. Focus on healthcare processes can provide a more efficient care, by reducing unnecessary and redundant activities, and by placing the patient at the centre of healthcare, since, after all, the patient process is the most important healthcare process, see for example [2].

Despite the promises of process managers, they face a number of shortcomings that need to be addressed. In this paper, we will address three of these issues:

- *Business orientation.* Process diagrams are typically expressed through low level concepts like control flow structures (e.g., parallel split) and message passing. Such concepts are not easily understood by domain experts and users, who instead prefer to understand processes through business oriented terms like agents, roles and resource flows.
- *Traceability.* Constructing a process diagram includes taking a number of design decisions that affect the structure of the diagram. It should be possible to trace these design decisions back to explanations and motivations expressed in business terms.
- *Information correctness.* The process manager does not maintain control over external applications, which means that these applications can be updated without the process manager being notified. As a consequence, there may be incorrect and conflicting information in the distributed system.

The purpose of this paper is to propose a number of design solutions that will address the issues of business orientation, traceability, and information correctness. The solutions are based on experiences from two case studies, a project in the telecom domain (Process Broker project) [3] and a project in the healthcare domain (VITA Nova project) [4], [5], in which process manager technology was used.

The paper is structured as follows: Section 2 gives an overview of process manager technology. Section 3 introduces a method used in the healthcare case study (i.e., VITA Nova project) in order to provide a context for the design solutions introduced. Section 4 describes three design solutions. Section 5 concludes the paper by presenting how the design solutions address the identified issues and gives suggestions for further research.

2 Process Managers

Various approaches to managing the integration of IT systems have been developed such as distributed objects (CORBA), message brokers and Internet portals [6]. Recently, Web Services have appeared as a new approach to integrate IT systems through XML documents over Internet protocols [7]. However, with the exception of some recent developments of Web Services, such as BPEL4WS, these technologies mostly function as interfaces to existing IT systems and do not lend adequate support to the business processes. Therefore, a new type of process oriented integration architectures have been developed, referred to as *process managers* (sometimes *business process management systems*, *process brokers,* or *process automation systems*) [6], [8], which closely reflect the business processes (in healthcare: healthcare processes), see Figure 1.

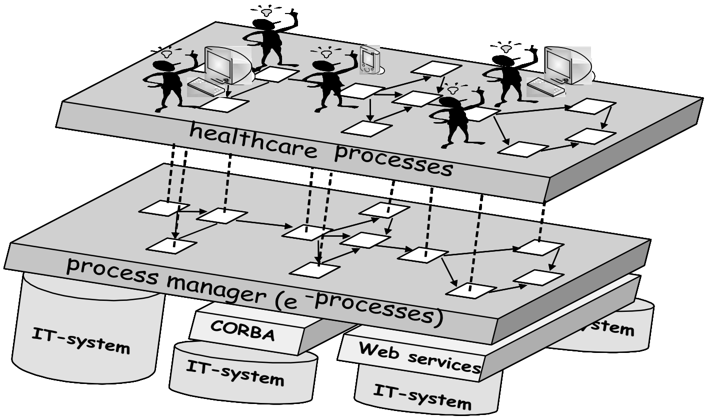


Figure 1. The process manager both visualises and manages the communication between the IT systems and human actors, which could use desktop computers and PDAs.

Somewhat simplified, the process manager functions as an extra abstraction layer (see the “process manager” layer in Figure 1) above the existing IT systems and integration technology, like Corba, Web Services or message brokers [6]. The process manager is a software product that visualises and executes the communication between the IT systems, and between the IT systems and the human actors. Therefore, we make a distinction between healthcare processes, and e-processes. This distinction facilitates the understanding of the process manager’s functionality and the relationship between the developed diagrams, described in Section 3.

By *healthcare process* we mean a partial order of manual tasks, which are performed by human actors and based on (business) rules. Examples of manual tasks are “treating a patient”, or “using a computer”. An example of a business rule is that “patients have to pay in advance of the examination”.

By e-process we mean a process that is executed in a process manager. An e-process is essentially a series of message exchanges, comprising the interaction logic between the process manager and various IT systems and human beings, as well as business rules.

The benefits of a process manager are many. The communication between the IT systems and between the IT systems and the human actors may be changed at any time, by changing the e-process, which is executed in the process manager. The process manager also offers specific tools for e-process design and change.

Since the process manager executes the graphical diagrams, it also facilitates monitoring the e-processes. That is, the process diagrams can be used to study the state of the instances of a single patient process while it is running. It is also possible to let the process manager simulate process scenarios in order to come up with an optimal e-process.

3 Method Support for Process Managers

In the VITA Nova projects, the Visuera Process Manager [9] was used, together with a method based on Business Model Language (BML), a graphical process modelling language for modelling and executing e-processes. BML was integrated with UML Activity diagrams, describing the healthcare processes, see Figure 2.

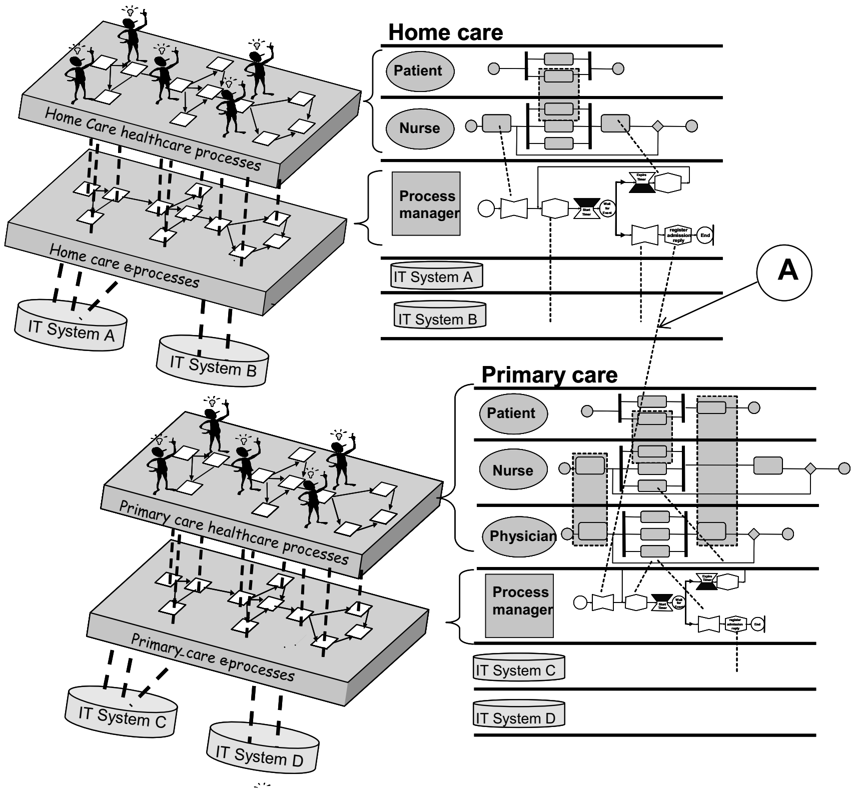


Figure 2. Healthcare processes, e-process and IT system in different swim lanes at two healthcare units.

BML [3] has similarities with Specification and Description Language (SDL), but is more adapted to IT systems integration. BML's metamodel consists of a limited number of concepts; the main are: "receive message", "send message", "automated task", "choice", "start timer" and "timer expired". In Visuera Process Manager, BML is used directly as an implementation language and replaces, to some extent, ordinary programming languages.

UML Activity diagrams are used for modelling the healthcare processes. By adding swim lanes (in UML 2.0 called partitions) to the method, diagrams of different healthcare processes can be related to each other, e.g., the patient's process, nurse's process and physician's process.

In Figure 2, two different swim lane models, the home care and the primary care, are specified. For each healthcare unit the different healthcare processes (e.g., patient's, nurse's and physician's processes) are specified using UML Activity diagrams. These diagrams are related to each other and to the e-processes, represented as BML diagrams. Finally, the e-processes are related to the IT systems. In Figure 2 the relations (i.e., message passing) between the healthcare processes, the e-processes and the IT systems are visualized as dotted lines. Note that IT system from different healthcare units also are integrated via the process manager (see A in Figure 2).

4 Design Solutions

In this section, we present a number of design solutions, i.e., solutions which are not yet implemented in industry. However, they have been partially validated in the Process Broker and VITA Nova project. The design solutions are described by the following form:

- Context – describe the situation/circumstances to situate the targeted interoperability problem
- Problem – an explicit statement of the interoperability problem concerned
- Solution – describe the implemented practice (proposed solution) to solve the problem defined
- Strength – the strong point of the solution

4.1 Value Based Process Modelling

Context

Conceptual models have become important tools for designing and managing complex, distributed and heterogeneous systems, e.g., e-commerce and e-health [10]. In e-commerce and e-health it is possible to identify two basic types of conceptual models: business models and process models. A business model focuses on the "what" in a system, i.e., identifying agents, resources, and exchanges of resources between agents. Thus, a business model provides a high-level view of the activities taking place in e-commerce and e-health. A process model, on the other hand, focuses on the "how" in an e-commerce or e-health system, specifying operational and procedural aspects of business communication. The process model moves into a more detailed view on the choreography of the activities carried out by agents [11].

Problem

In order to realize interoperability between services and systems, it is required to make their underlying processes interoperable. Designing and understanding the interrelationships between a numbers of complex interacting processes can be a difficult and time consuming task. Furthermore, it is often difficult to understand why a process model has been designed in a certain way and what consequences alternative designs would have.

Solution

Process models should be given a declarative foundation by basing them on business models that describe the exchange of value, see Figure 3. Two instruments for a declarative foundation of process models are process patterns and activity dependencies. A process pattern is a generic template for a set of interrelated activities between two agents, while an activity dependency expresses a sequential relationship between two activities [12].

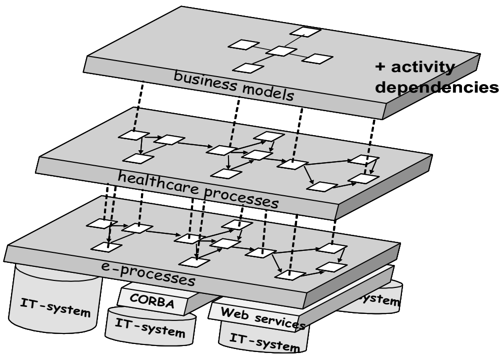


Figure 3. A business model layer added above the healthcare processes.

A business model specifies agents involved in the business process, the resources exchanged between them, and the activities through which these resources are exchanged.

Furthermore, there exist many different kinds of patterns for business processes. These include basic workflow patterns [13], transaction patterns as in ebXML [10] and more complex patterns like the Action Workflow pattern [14]. Such patterns provide a basis for a partial ordering of the activities taking place in a business process. However, the ordering derived from such process patterns only provide a starting point for designing complete business process diagrams, i.e., it needs to be complemented by additional interrelationships among the activities in the process diagram to be designed. These interrelationships should have a clear business motivation, i.e., every interrelationship between two activities should be explainable and motivated in business terms. We suggest formalizing this idea of business motivation by introducing the notion of activity dependencies. An activity dependency is a pair of activities, where the second activity for some reason is dependent on the first one. We identify four different kinds of activity dependencies, each one providing a particular reason expressed in business terms, for the interrelationship between the activities.

Flow dependencies. A flow dependency is a relationship between two activities, which expresses that the resources obtained by the first activity are required as input to the second activity. An example could be a retailer who has to obtain a product from an importer before delivering it to a customer. Another example could be that a healthcare unit needs to have a delivery of a certain resource type before a therapy could be performed.

Trust dependencies. A trust dependency is a relationship between two activities, which expresses that the first activity has to be carried out before the other one as a consequence of low trust between the agents. Informally, a trust dependency states that one agent wants to see the other agent do her work before doing his own work. An example could be a car dealer who requires a down payment from a customer before delivering a car. Another example could be that a patient's insurance company has to pay the healthcare unit before a therapy could be performed.

Control dependencies. A control dependency exists when one agent wants information about another agent before establishing a contract with that agent. A typical example could be a company making a credit check on a potential customer. Another example could be that a healthcare unit needs to check with another unit about a patient's status, before accepting to take care of a patient.

Negotiation dependencies. A negotiation dependency expresses that an agent is not prepared to establish a contract with another agent before she has established another contract with a third agent. One reason for this could be that an agent wants to ensure that certain resources can be procured before entering into a contract where these resources are required. Another reason could be that an agent does not want to procure certain resources before there is a contract for an activity where these resources are required. An example could be that a healthcare unit needs to establish a contract with another healthcare unit before promising to perform a therapy.

A business process diagram can be understood and designed based on process patterns and activity dependencies. Designing a business process diagram is not a trivial task but requires a large number of design decisions. In order to support a designer in this task, we propose an automated designer assistant (a software program/wizard) that guides the designer through the task by means of a sequence of questions. The designer specify the business model, the activity dependencies and process pattern by answering a set of predefined questions, and the assistant automatically create the process models based on the answers.

Strength

Process models that are given a declarative foundation based on business models and activity dependencies would provide justifications, expressible in business terms, for design decisions made in process modelling, thereby facilitating communication between systems designers and business users. Following the layered approach described in section 2 and 3, we have by introducing the value concept added a third level, the business model, describing the value foundation of the business. Graphically this can be viewed as in Figure 3.

4.2 View Based Process Modelling

Context
Process diagrams have become an important tool for specifying and analysing the business processes and the e-processes (constructed to be executed in process managers). However, when integrating different IT systems, many different aspects need to be taken care of. For example, exception handling makes up a large part of an integration specification, i.e., situations when IT systems do not respond in an expected way. The description of such exceptions easily obscures the main business logic. Furthermore, it is possible to distinguish among several kinds of stakeholders involved in system integration projects: domain experts such as business managers, business and technical designers, and operators that handle the day-to-day operations. Different stakeholders require different views of the process diagrams, while at the same time they need to be able to communicate with each other using common diagrams.

Problem
System integration specifications often results in highly unstructured and complex process diagrams, specifying the integration between the IT systems. These diagrams are also used by different stakeholders, which are only interested in viewing special parts of the diagrams.

Solution
Process diagrams visualizing the integration of systems should be structured through a series of views. These series should start with a customer oriented view, or some other actor's view, on the business level and add more and more details moving from a business perspective to a more technical perspective [3].

The different views of process diagrams are described below and illustrated by means of a healthcare case, in which a patient wants to order an investigation. The order is sent to an IT system, IT System A, which offers the patient date and time for the investigation and if the patient accept the date and time, update the order. However, if the patient does not pay tax in the region to which the healthcare unit belongs, a healthcare unit manager first has to accept the order.

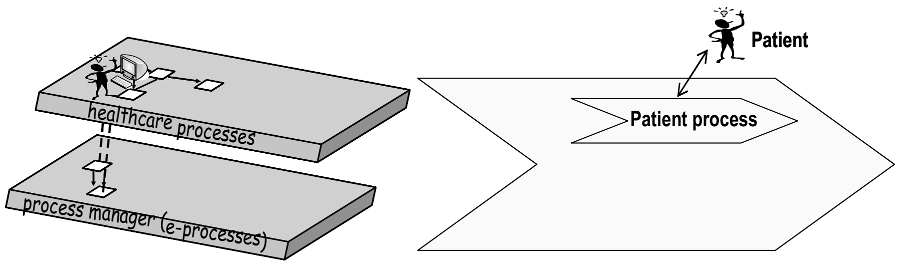


Figure 4. Patient interaction (View 1).

View 1. Patient interaction. This view models the interactions between the integrated system and the patient, i.e., the messages exchanged by the patient and the integrated system as well as the flow of control, see Figure 4. Note, the diagrams (to the right) in Figure 4 and 5 only visualise the process diagram from a static point of view, i.e., the control flows are not visualised. Process diagrams visualising the control flows are described in [3]. Note that the focus in view 1 does not need to be a patient - it could also be a nurse, physician or any other healthcare provider, or a customer in another domain.

View 2. System requests. This view is an extension of view 1 and describes how the integrated system produces the messages and sends them to the patient, i.e., answers the patient's requests. Therefore, this view must also model the interaction with the IT systems and human being to be integrated. For every IT system and human being integrated, an interface process needs to be constructed, see the two diagrams at the bottom of Figure 5 (right). The design rule is: one interface process describes exactly one IT system's or human actor's interaction with the system.

In the simplest case, only one system produces the message to the patient. In some cases, it is convenient to introduce two or more levels of sub processes. If the sub process to be specified requires interaction with several IT systems, the designer should construct a synchronization process, in the middle in Figure 5, which invokes its own sub processes and synchronizes these. Following Figure 5, if the patient is not a tax payer in the region, the Synchronization process sends a message to both Interface process a and b. Interface process b sends a massage to the Manager, which has to accept the order or not.

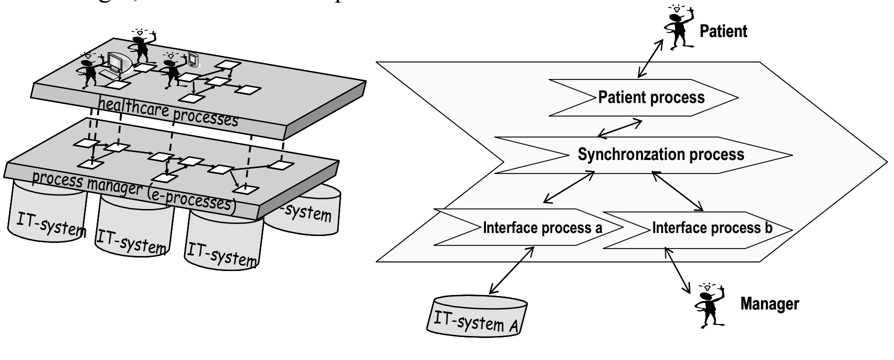


Figure 5. System requests (view 2).

View 3. Exception handling and resource releasing. First, this view specifies the exception handling. Views 1 and 2 specify only the normal course of events. In view 3, the process diagrams specify how to handle exceptions, i.e., situations where an external IT system or a human actor has not replied to a request within a pre-specified time limit. Secondly, this view describes resource releasing. When a process cannot be completed as intended, it becomes necessary to release the resources that have been reserved or booked during the process.

View 4. Notifications. This view adds all notifications messages. We have identified three situations where notification messages are required. First, when an exception has occurred, it is customary to inform an operator about this event so that he or she may take appropriate action. Secondly, a notification about essential states of affairs may be sent to a master storage which redundantly stores the information, see Section 4.3. Thirdly, a notification can be sent to a data warehouse, which uses the information for strategic analysis and decision, see Figure 6.

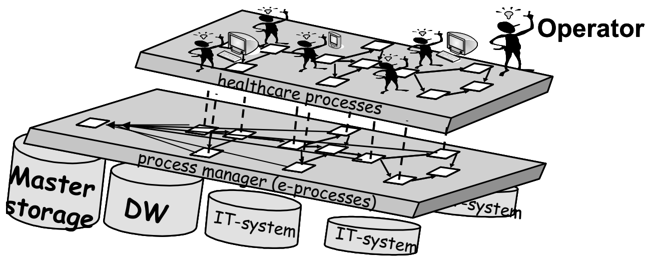


Figure 6. Notifications (view 4).

Strength
The views presented above can be used for two reasons. First, it could help the designer to construct the process diagrams. A designer could utilize the views by first constructing a process diagram according to view 1 and then gradually refining it until a set of diagrams in view 4 is obtained. Therefore, the views can be seen as steps in a method for designing process diagrams. Secondly, it can be used for presentation purposes to show or hide parts of process diagrams. Business oriented users can choose to see only the top view or views, while technical designers and implementers can proceed to the lower views. Even for the latter category of stakeholders, the layered views can help to understand a system by allowing the focus on an essential business perspective first and thereafter proceed to a technical perspective. Different categories of users, for example customers/healthcare providers, business and technical designers, have the possibility of suggesting input on the right level in the modelling process. Note that the views presented above may apply to the healthcare process as well as the e-processes.

4.3. Master Storage for Process Managers

Context
Integration of IT systems can be supported by different architectures. One architecture for integrating IT systems is the point-to-point solution where every IT system is directly connected to every other IT system. This solution could work for a small number of IT systems, but as the number of IT systems increases, the number of connections quickly becomes overwhelming. By using a central message broker or process manager, this complexity is reduced, i.e., the number of interfaces will be reduced. Furthermore, tools to facilitate the format conversions can be applied, and the visualisation added by the process manager enables different categories of people to take part in the process design.

Problem
Integration of IT systems using a central message broker or process manager is a complex task and it is seldom possible to integrate every IT system via the central

system at one occasion. Therefore, integrated IT systems still need communicate with external system which are not part of the integrated solution. These IT systems can be updated without the message broker or process manager being notified. For example, information about a patient can be changed by mistake when an external system communicates with IT systems connected to the integrated solution.

Solution
When integrating different IT system, for example by using a message broker or process manager, a master storage should be developed that duplicates part of the data in the integrated IT systems [3].

Strength
There are several reasons for maintaining redundant information in a master storage. First, it is possible that the integrated IT system can be updated without the message broker or process manager being notified. By using an internal storage the system has complete and correct data saved in one place. Secondly, if the patient or customer service quickly requires information about, for example, an order, the system does not have to query several IT systems; it only has to query the master storage. It is not unusual that information about a patients or customer is distributed over several IT systems. In that case, the internal storage is, from a performance perspective, an important improvement of a system. Thirdly, the opportunity to notify an internal storage of events that occur in the integrated system, makes it possible to create a data warehouse that is updated in real-time.

5 Conclusions

In section 1 we introduced the following issues that need to be addressed: Business Orientation, Traceability, and Information correctness.

Business Orientation is about describing the business processes in business terms. This is addressed through both the value and view based process modelling. These design solutions focus on business oriented concepts like resource, customer and healthcare provider, instead of going directly into procedural details. This allows a stakeholder to get a high level overview of the process expressed in familiar business terms. Furthermore, when the stakeholder wants to get deeper into the details of the process she can use the views of the models to still hide details irrelevant for her present purposes.

Traceability is about understanding and tracking the reasons for design decisions in a process model. This is also addressed through the value based process modelling. Reasons for process design are based on value concepts complemented by activity dependencies and process patterns. These allow a stakeholder to understand specific process designs in terms of higher level concepts, which also provides a basis for comparing alternative designs.

Information correctness is about maintaining a consistent and correct view of the information in the entire distributed system in spite of local autonomy. This problem is addressed by the master storage design solution, which offers an additional storage for ensuring information correctness.

We believe that the proposed design solutions will provide important benefits for process modelling and managers, thereby strengthening one of the most promising approaches for interoperability. Further work will include the identification of additional process patterns as well as empirical validation.

Acknowledgements

The authors would like to thank all the VITA Nova project members. The VITA Nova – Hemma project is sponsored by the Vinnova (Swedish Agency for Innovation Systems). The paper is also part of the joint research activities “Design principles for interoperability” (WP6) and “Enterprise interoperability architectures” (WP9) of the INTEROP Network of Excellence sponsored by EC.

References

1. Haux R., Winter A., Ammenwerth E., Birgl B., Strategic Information Management in Hospitals. An introduction to Hospital Information Systems, Health Informatics Series, Springer, 2004
2. Vissers, J. M. H., Health care management modelling: a process perspective, Health Care Management Science, 1, pp. 77-85, 1998
3. Johannesson P., Perjons E., Design principles for process modelling languages in enterprise application integration, Information System, 26, pp. 165-184, 2001
4. Perjons E., Wangler B., Wäyrynen J., Åhlfeldt R.-M., Introducing a Process Manager in Healthcare. An experience report, Health Informatics Journal, Vol 11(1), 2005
5. Wangler B., Åhlfeldt R.-M., Perjons E., Process oriented information systems architectures in healthcare, Health Informatics Journal, Vol 9(4), 2003
6. Linthicum S. D., B2B Application Integration – e-Business-Enable Your Enterprise, Addison-Wesley, 2001
7. Oellermann W. L., Architecting Web Services, Apress, 2001
8. Dayal U., Hsu M., Ladin R., Business Process Coordination: State of the Art, Trends, and Open Issues, Proc. of the 27th International Conference on Very Large Data Bases, September 11-14, Roma, Italy, pp. 3-13, 2001
9. Visuera Process Manager, Visuera Integration AB, [online] http://www.visuera.com [accessed Jan 2004]
10. UN/CEFACT Modelling Methodology (UMM), Revision 10, [online] http://www.untmg.org/doc_bpwg.html, [accessed April 2003], 2003
11. Gordijn, J., Akkerman J.M., Vliet J.C., Business Modelling, is not Process Modelling, Proc. of the 1th International workshop on Conceptual Modeling Approaches for e-Business (eCOMO’2000), 2000
12. Bergholtz, M., Jayaweera, P., Johannesson, P., Wohed, P., A pattern and dependency based approach to the design of process models, Proc. of ER 2004, Shanghai, 2004
13. van der Aalst W.M.P., ter Hofstede A.H.M., Kiepuszewski B., Baros A.P., Workflow Patterns, Distributed and Parallel Databases, 14(3), July, 2003
14. Winograd T., Flores F., Understanding Computers and Cognition: A New Foundation for design, Ablex, Norwood, N.J, 1986

Towards an Interoperability Framework for Model-Driven Development of Software Systems

Brian Elvesæter[1], Axel Hahn[2], Arne-Jørgen Berre[1] and Tor Neple[1]

[1] SINTEF ICT, P. O. Box 124 Blindern, N-0314 Oslo, Norway
{brian.elvesater, arne.j.berre, tor.neple}@sintef.no
[2] Wirtschaftsinformatik Universität Oldenburg, D-26111 Oldenburg, Germany
hahn@wi-ol.de

Summary. This paper presents an interoperability framework for model-driven development of enterprise applications and software systems. The framework provides a foundation for how to apply MDD in software engineering disciplines in order to support the business interoperability needs of an enterprise. The framework introduces reference models for conceptual integration, technical integration and applicative integration of software systems.

1 Introduction

Enterprise applications and software systems need to be interoperable in order to achieve seamless business across organisational boundaries and thus realise virtual networked organisations. IEEE [1] defines interoperability as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged”.

Model-driven development (MDD), and in particular OMG’s Model Driven Architecture® (MDA®[1]) [2], is emerging as the state of practice for developing modern enterprise applications and software systems. We believe that there is a need for an interoperability framework that provides guidance on how MDD should be applied to address interoperability.

In this paper we present initial results from ATHENA [3, 4] and INTEROP [5, 6] in defining an interoperability framework for model-driven development of enterprise applications and software systems. The framework provides a foundation, consisting of a set of reference models, for how to apply MDD in software engineering disciplines in order to support the business interoperability needs of an enterprise.

The paper is structured as follows: In section 2 we provide some background information. In section 3 we present the interoperability framework and its

[1] Model Driven Architecture® and MDA® are registered trademarks of the Object Management Group, http://www.omg.org/mda/

reference models for integration. Section 4 discusses the usage of the reference models in an example scenario focusing on inventory replenishment. In section 5 we describe some related work. Conclusions and future work are presented in section 6.

2 Background

Model-driven development can be viewed as a new architectural approach for developing software systems based on requirements derived from enterprise and business models. Interoperability solutions should be driven by business needs first and software solutions second.

Model-driven development represents a business-driven approach to software systems development that starts with a computation independent model (CIM) describing the business context and the business requirements. The CIM is refined to a platform independent model (PIM) which specifies services and interfaces that the software systems must provide independent of software technology platforms. The PIM is further refined to a platform specific model (PSM) which describes the realisation of the software systems with respect to the chosen software technology platforms. A model-driven framework should also address how to integrate and modernise existing legacy systems according to new business needs, known as Architecture-Driven Modernization (ADM[2]).

In order to structure the various models developed and used within an enterprise we adopt the recommendations of IEEE 1471 [7] which provide a terminology for structuring descriptions of systems according to viewpoints. A view can be represented by a set of visual models expressed using a modelling language such as UML [8, 9]. The following terms are adopted from the IEEE 1471 standard:

- *System*: A collection of components organised to accomplish a specific function or set of functions.
- *Stakeholder*: An individual, team, or organisation (or classes thereof) with interests in, or concerns relative to, a system.
- *Concern*: Those interests which pertain to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. (In this paper we will also use the term aspect as a synonym for concern.)
- *View*: A representation of the whole system from the perspective of a related set of concerns.
- *Viewpoint*: A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.
- *Model*: A representation of an entity in the real world.

[2] Architecture-Driven Modernization Task Force, http://www.omg.org/adm/

3 Interoperability Framework

The interoperability framework presented in this paper is designed to fulfil these design rationales:

- Identification of interoperability issues by interrelating software architectures and enterprise architectures.
- Identification of the relevant software architecture components.
- Integration of model-driven software development processes.
- Structuring of software technologies, frameworks and methodologies.

The interoperability framework itself is structured into three integration areas:

1. **Conceptual integration** which focuses on concepts, metamodels, languages and model relationships to systemise software model interoperability.
2. **Technical integration** which focuses on the software development and execution environments.
3. **Applicative integration** which focuses on methodologies, standards and domain models. It provides us with guidelines, principles and patterns that can be used to solve software interoperability issues.

For each of these three areas we developed a reference model to describe and support the application of model-driven development of interoperable software systems. We use the term model-driven interoperability (MDI) to describe the overall approach of applying MDD to design interoperable software systems.

3.1 Reference Model for Conceptual Integration

The reference model for **conceptual integration** (see Figure 1) has been developed from a MDD point of view focusing on the enterprise applications and software system. A computation independent model (CIM) corresponds to a view defined by a computation independent viewpoint. It describes the business context and business requirements for the software system(s). A platform independent model (PIM) corresponds to a view defined by a platform independent viewpoint. It describes software specifications independent of execution platforms. A platform specific model (PSM) corresponds to a view defined by a platform specific viewpoint. It describes the realisation of software systems. Figure 1 shows this relationship with respect to an enterprise system. It also shows how MDA and ADM could be perceived as a “top-down” and a “bottom-up” approach to software development and integration. The models at the various levels may be semantically annotated using reference ontologies which help to achieve mutual understanding on all levels. We also see the usage of interoperability patterns for horizontal and vertical integration.

We have identified four categories of system aspects where specific software interoperability issues can be addressed by conceptual integration. These four aspects can be addressed at all three CIM, PIM and PSM levels.

1. **Service aspects:** Services are an abstraction and an encapsulation of the functionality provided by an autonomous entity.

2. **Information aspects:** Information aspects are related to the messages or structures exchanged, processed and stored by software systems or software components.
3. **Process aspects:** Processes describe sequencing of work in terms of actions, control flows, information flows, interactions, protocols, etc.
4. **Non-functional aspects:** Extra-functional qualities that can be applied to services, information and processes.

All of the elements discussed above are integrated into Figure 1 where we look at horizontal and vertical integration between multiple enterprise systems, here exemplified with two enterprise systems A and B.

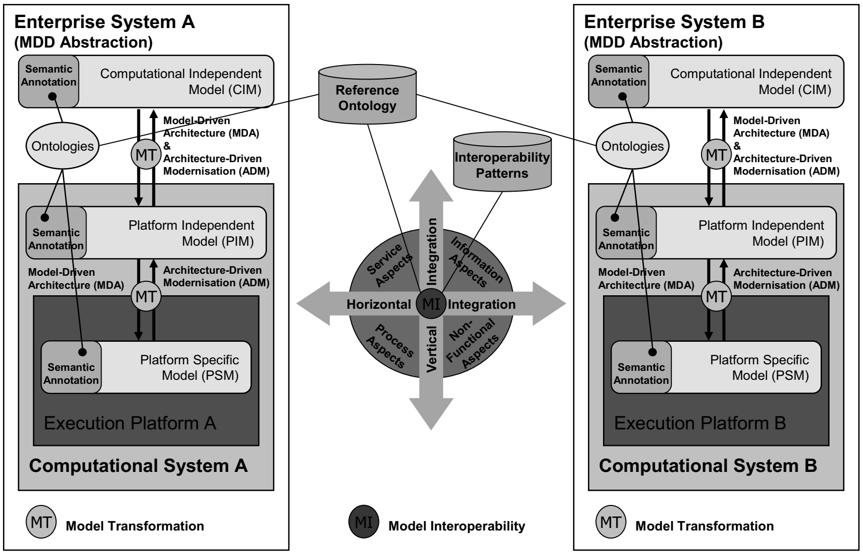


Figure 1. Reference model for conceptual integration

We will use this reference model to address model interoperability, where metamodels and ontologies will be used to define model transformations and model mappings between the different views of an enterprise system. In literature [10, 11] different dimensions of system design are identified:

- **System abstraction:** This dimension of system design reflects the abstraction in terms of implementation independency and is addressed by MDD.
- **Generality:** The applicability of a system design has impact on the adaptability and reusability of the system components.
- **Viewpoint:** System models represent a complex and strongly interrelated network of model entities. To address different issues and for complexity reduction different viewpoint on the model are used. This viewpoint may also be regarded for interoperability.

– **Composition:** Systems are iteratively composed in a hierarchy from individual objects to the system in the enterprise context. On each of this aggregation layers the entities have to be interoperable.
– **Time:** The system itself is modified in status, configuration and design.
– **Model abstraction**: Metamodels help to describe and analyse the used models.

These dimensions can be used to analyse software systems or help to structure the system modelling process and to catalyse design decisions. Each of these dimensions may support interoperability achievements or could represent a challenge of interoperability.

3.2 Reference Model for Technical Integration

The reference model for **technical integration** (see Figure 2) has been developed from a service-oriented point of view where a software system provides a set of services required by the businesses and users of the enterprise.

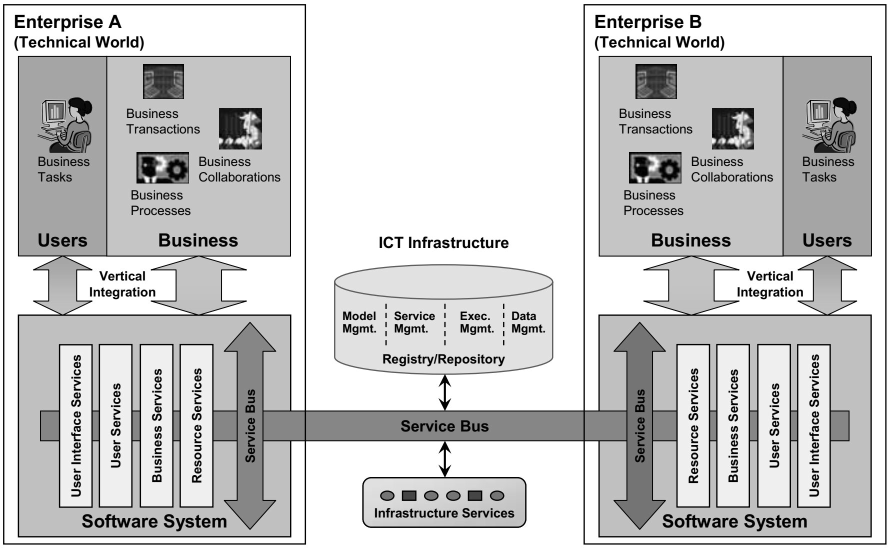


Figure 2. Reference model for technical integration

The architecture of the enterprise applications and software systems can be described according to a 4-tier reference architecture where each tier provides different software services required by the enterprise. The software system itself is coupled to an ICT infrastructure illustrated by a service bus that provides the necessary communication infrastructure. Infrastructure services such as composition, mediation, matchmaking and transformation that enables interoperability between software systems should be provided. We recognize the need for a model repository for managing models of various kinds, a service registry for managing naming, directory and location of services, an execution repository for managing information and state needed in the execution of software services and processes, and a data repository for managing results and traces of the executions.

Figure 2 shows how a service bus comes into play when integrating two (or more) enterprises systems. The service bus will make use of infrastructure services, and registry and repository.

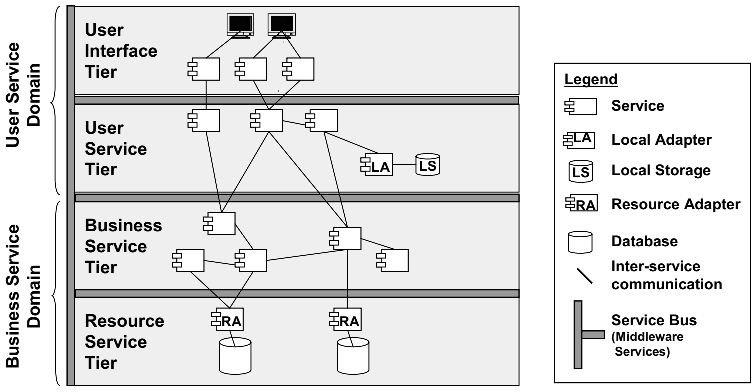


Figure 3. 4-tier reference architecture for software system architectures

We have defined a reference architecture (see Figure 3) that separates the architecture of a software system into four logical tiers. The reference architecture consists of a local user-space called the user service domain, and a shared transactional business-space called the business service domain. The four tiers are as follows:

1. **User interface tier** provides presentation and user dialog logic.
2. **User service tier** provides the user's model, which may include user session logic and user-side representations of processes and information. It is an abstraction for a set of business services.
3. **Business service tier** provides components that represent business functionality and pervasive functionality (vertical vs. horizontal services). This tier provides enterprise-level services, and is responsible for protecting the integrity of enterprise resources. Components in this tier can be process-oriented, entity-oriented or workflow-oriented.
4. **Resource services tier** provides global persistence services, typically in the form of databases. Resource adapters (e.g. JDBC or ODBC drivers) provide access, search and update services to databases and its data stored in a database management system (DBMS) like Oracle or Sybase.

In addition to these four tiers we need a service communication bus so that services deployed at the various tiers can interoperate both within a tier and across tiers.

3.3 Reference Model for Applicative Integration

The reference model for **applicative integration** (see Figure 4) has been developed based on work related to enterprise architecture frameworks and software architecture frameworks [12]. Enterprise and software models can be related in a holistic view, regardless of modelling language formalisms, by the use of metamodels. This is important in order to understand the dependencies between the different models and views to achieve interoperability.

The MDD methodology needs to follow a structured approach where interoperability requirements from business operations in a networked enterprise drive the development of software solutions. This means that MDD methodology needs to be related to enterprise architectures. A specific part needs to address how the MDD concepts and the technical software components are reflected in a model world of the enterprise. Figure 4 shows how the model world, reflecting the applicative integration, is related to the reference models for conceptual and technical integration. Enterprise and software models can be built to understand and analyse the interoperability requirements of an enterprise.

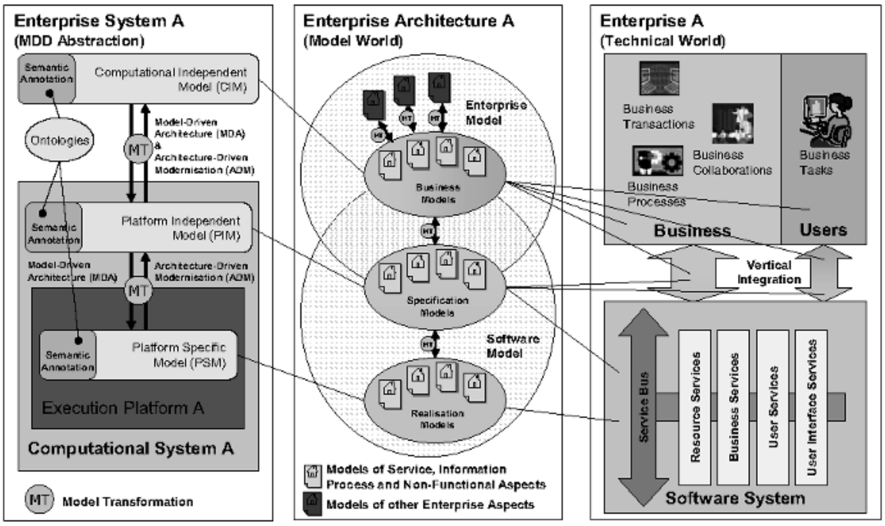


Figure 4. Reference model for applicative integration

An enterprise model describes a set of enterprise aspects, which includes business models capturing actors and stakeholders, business services, business information and business processes. These business models provide a context for the software solutions that needs to be developed and integrated.

Software models describe how software systems are used to support the businesses of an enterprise. The software models further refines the business models in terms of software realisation models. All these models should include descriptions of the four system aspects identified in the reference model for conceptual integration. The software models can be classified as CIM, PIM or PSM models according to a MDD abstraction.

In our interoperability framework we have identified a set of models that we see useful in achieving interoperability. However, we also acknowledge the fact that this is just a baseline. Different enterprises must be able to develop their own software views that they see purposeful. It is important that the applicative integration supports the development of a set of shared views amongst different stakeholders and provides means for managing the dependencies between these views. We believe a viewpoint-based integration approach must be chosen. This allows incorporating viewpoints, which are implicitly or explicitly defined by other enterprise or software modelling approaches into an applicative framework.

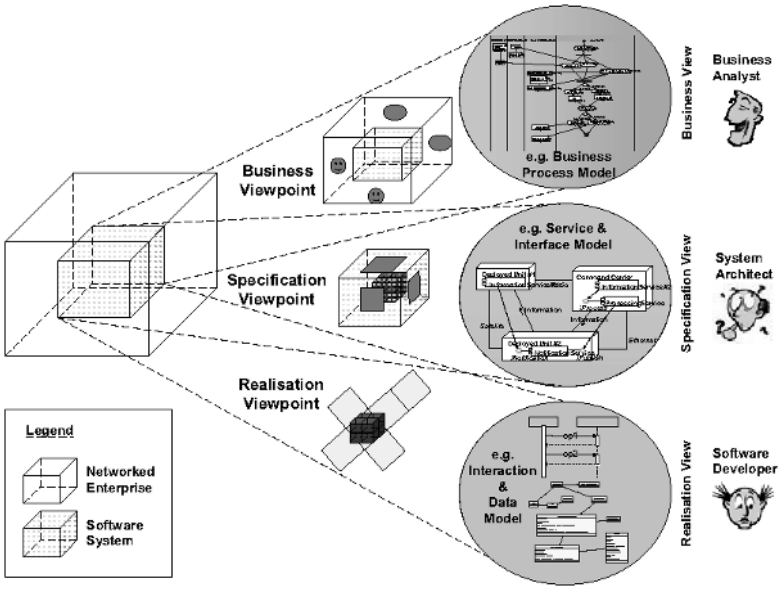


Figure 5. Illustration of the three basic viewpoints and their corresponding views

We have identified three basic viewpoints (see Figure 5) that can be used as a starting point for viewpoint-based integration of software systems; *Business viewpoint* focusing on the business context of the software system, *Specification viewpoint* focusing on the specification of the main components of the software system, and *Realisation viewpoint* focusing on the implementation of the software system. Figure 5 illustrates the viewpoint metaphor exemplified using the three basic viewpoints. A business analyst is concerned with aspects related to the business context of the software system within the networked enterprise. A system architect is concerned with aspects related to the specification of the software system. A software developer is concerned with aspects related to the realisation of the software system.

4 Example – Inventory Min/Max Replenishment

As a part of the work of defining the framework presented in this paper a student project was performed over a couple of months. The task was to take an example integration scenario and create a set of models that were sufficient to be used for further transformations into new models and code. The goal was to be able to generate a running business process performing the integration scenario. The target platform for execution was BPEL running on IBMs BPWS4J[3] server. In addition to creating models and code the students were to come up with observations of shortcomings in the different tools, methods and modelling languages.

The chosen scenario was "Inventory min/max replenishment" as defined by the Automotive Industry Action Group (AIAG) [13]. The suppliers see the inventory level of the customer with minimum and maximum levels of inventory for a specific inventory item. Based on this information the supplier can perform inventory replenishment so that the actual inventory level for an item is "automatically" kept between the minimum and maximum levels defined. The document defines the process and the information passed using the UN/CEFACT Unified Modeling Methodology [14].

At a high level one can say that there were three main models. Initially the partition into these three main model types was taken from [15], but indicated below, these are easily mapable to the aspects defined in this paper;

1. Activities, a model that defined the min/max process including actors, activities and information flow. This model describes the *process aspects* as defined in the interoperability framework.
2. Interactions, a model that defined the different computational services that would be part of the automated process and how they interact using defined interfaces. This model describes the *service aspects* as defined in the interoperability framework.
3. Information, a model that defined the information that was passed between the different activities and services in the process. This model describes the *information aspects* as defined in the interoperability framework.

No models describing the *non-functional aspects* were created in this exercise. All of the models created can be viewed as platform independent models.

One of the ideas behind the framework is that different languages or notations can be used to define the needed models. In order to discuss pros and cons of different methods and notations the students created two sets of models, one using the Business Process Modelling Notation (BPMN) [16] and one using the Business Process Definition Metamodel (BPDM) [17]. These would be used to generate the needed BPEL and WSDL descriptions using the UMT[4] tool for code generation.

BPMN models were created using the Metis[5] tool from Computas. BPDM has defined a UML 2.0 profile. In this exercise the Enterprise Architect[6] tool from Sparx Systems was chosen because it had support for UML 2.0 at the point in time that this exercise was carried out. Due to limitations in the chosen tools it was hard to export the model information to a format that could be used by the UMT tool. It would be possible to implement, but this would mean spending resources on something that would not really add value to the exercise. To get around this problem it was chosen to create a UML 1.4 model based on the BPDM and the

[3] BPWS4J, http://www.alphaworks.ibm.com/tech/bpws4j/
[4] UML Model Transformation Tool, http://umt-qvt.sourceforge.net/
[5] Metis, http://www.computas.com/metis/
[6] Enterprise Architect, http://www.sparxsystems.com.au/ea.htm

BPMN model using the ACE-GIS UML profile [18]. From this model information UMT was able to generate BPEL and WSDL information that had to be manually crafted to be executed by the execution engine. The resulting web services (as executed by the execution engine) are *business services* as defined in the business service tier of 4-tier reference architecture described in section 3.2.

The approach of modelling process, service and information aspects, as defined in the framework, does provide enough domain information in order to create platform specific models.

5 Related Work

The IDEAS Interoperability Framework, developed in the IDEAS project [19], provides four different areas for structuring interoperability issues of enterprise applications; *Business layer* focusing on business environment and processes. *Knowledge layer* focusing on organisational roles, skills and competencies of employees, and knowledge assets. *ICT systems layer* focusing on applications, data and communication components. *Semantic dimension*, cutting across the three identified layers, focusing on supporting mutual understanding on all layers.

The European Computer Manufacturers Association/National Institute of Standards and Technology (ECMA/NIST) has developed a reference model for distributed system integration [20] that separates integration into four different categories; *Data integration* addressing the degree to which tools are able to share common data and information. *Control integration* addressing the degree to which tools are able to interact directly with each other, by requesting and providing services. *Process integration* addressing the degree to which the user's working process and use of tools can be guided by a model of the work process and the methodology to be followed. *Presentation integration* addressing the degree to which a user-interface program might provide access to the functionality needed by the user through a common look and feel.

E-Commerce Integration Meta-Framework (ECIMF) defines recommended interoperability methodology, and the technical specification and base tools needed to prepare specific comparisons of concrete frameworks [21]. The proposed ECIMF methodology for analysis and modelling of the transformations between e-commerce frameworks follows a layered approach. In order to analyse the problem domain one has to split it into layers of abstraction, applying top-down technique to classify the entities and their mutual relationships.

The Reference Model for Open Distributed Processing (RM-ODP) [22] is an ISO standard focusing on open distributed processing systems. RM-ODP divides the specification of ODP systems into five different, but related, viewpoints; *Enterprise viewpoint* focusing on purpose, scope and policies. *Information viewpoint* focusing on information processing and relationships between information objects. *Computational viewpoint* focusing on functional specification and decomposition. *Engineering viewpoint* focusing on how to solve distribution issues. *Technology viewpoint* focusing on specific technology and solutions.

The interoperability framework presented in this paper addresses much of the same interoperability elements identified in the above framework and approaches, but relates them more closely to software models in a model-driven development environment.

6 Conclusions and Future Work

We will conclude this paper by addressing each of the objectives or design rationales for the development of the interoperability framework.

– The reference model for applicative integration clearly indicates that software solutions cannot be developed in isolation if the goal is to achieve interoperability between business units within and across virtual networked enterprises. The software development process and models must be related to the enterprise and business needs of an enterprise. We believe that a viewpoint-based integration approach provides a good foundation for relating different models and views in a holistic approach where software architectures and enterprise architectures can be related.
– The reference model for technical integration provides a 4-tier reference architecture for describing software systems as a set of services addressing business and user needs. These software services are connected using a service bus that provides infrastructure, registry and repository services for integrating software.
– The reference model for conceptual integration provides a basic understanding of MDD concepts that can be integrated into model-driven software development processes.
– Finally, we believe that the reference models provided can be used to structure software technologies, frameworks and methodologies which can be found today.

The reference models described will be used in the development of more structured methodologies focusing on different business domains as well as the development of infrastructure, registry and repository services to enable interoperability between software systems. In particular we will look into the specification of a set of UML 2.0 profiles for the four system aspects identified in the reference model for conceptual integration.

The work published in this paper is partly funded by the European Commission through the ATHENA IP (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications Integrated Project) [3] and the INTEROP NoE (Interoperability Research for Networked Enterprise Applications and Software Network of Excellence) [5]. The work does not represent the view of the European Commission, the ATHENA consortium or the INTEROP consortium, and the authors are solely responsible for the paper's content.

References

1. IEEE (1990) Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. Institute of Electrical and Electronics Engineers (IEEE), IEEE Std 610-1990.
2. OMG (2003) MDA Guide Version 1.0.1. Object Management Group (OMG), Document omg/2003-06-01.
3. ATHENA (2005) ATHENA Public Web Site. ATHENA IP. http://www.athena-ip.org
4. ATHENA (2005) Specification of a basic architecture reference model. ATHENA IP, Deliverable D.A6.1.
5. INTEROP (2005) INTEROP Portal. INTEROP NoE. http://www.interop-noe.org/
6. INTEROP (2004) State-of-the-art for Interoperability architecture approaches - Model driven and dynamic, federated enterprise interoperability architectures and interoperability for non-functional aspects, Version 1.0. INTEROP NoE, Deliverable D9.1.
7. IEEE (2000) IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Institute of Electrical and Electronics Engineers (IEEE), IEEE Std 1471-2000.
8. OMG (2003) UML 2.0 Infrastructure Specification. Object Management Group (OMG), Document ptc/03-09-15.
9. OMG (2003) UML 2.0 Superstructure Specification. Object Management Group (OMG), Document ptc/03-08-02.
10. C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wust, and J. Zettel (2002) Component-based Product Line Engineering with UML. Addison-Wesley.
11. D. F. D'Souza and A. C. Wills (1998) Object, Components, and Frameworks with UML - The Catalysis Approach. Addison Wesley.
12. B. Elvesæter, T. Neple, J. Ø. Aagedal, R. K. Rolfsen, and O. Ø. Stensli (2004) MACCIS 2.0 - An Architecture Description Framework for Technical Infostructures and their Enterprise Environment. Presented at Command and Control Research and Technology Symposium (CCRTS 2004), San Diego, USA.
13. AIAG (2003) Inventory Visibility and Interoperability Min/Max Replenishment. Automotive Industry Action Group (AIAG).
14. UN/CEFACT (2003) UN/CEFACT Modeling Methodology (UMM) User Guide. United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), CEFACT/TMG/N093.
15. Norsk EDIPRO (2002) Infrastructure for electronic commerce - Application readable models in an open infrastructure, Version 1.0. Norsk EDIPRO.
16. BPMI (2004) Business Process Modeling Notation (BPMN), Version 1.0. Business Process Management Initiative (BPMI).
17. OMG (2004) Business Process Definition Metamodel, Version 1.0.2. Object Management Group (OMG), Revised submission bei/2004-01-02.
18. R. Grønmo, D. Skogan, I. Solheim, and J. Oldevik (2004) Model-Driven Web Service Development. International Journal of Web Services Research (JWSR), vol. 1.
19. IDEAS (2005) IDEAS Roadmaps. IDEAS. http://www.ideas-roadmap.net/
20. ECMA (1993) Reference Model for Frameworks of Software Engineering Environments, 3rd ed. ECMA, Technical Report NIST 500-211, ECMA TR/55.
21. ECIMF (2001) E-Commerce Integration Meta-Framework - General Methodology (ECIMF-GM). CEN/ISSS/WS-EC/ECIMF, Draft Version 0.3.
22. ITU-TS (1995) Basic Reference Model of Open Distributed Processing - Part 1: Overview and guide to use the Reference Model. Rec.X901 (ISO/IEC 10746-1).

Syndicate Data Incorporation into Data Warehouses: Contrasting Consumer Problems with Supplier Viewpoints

Mattias Strand[1] and Björn Lundell[2]

[1] School of Humanities and Informatics, University of Skövde, Sweden
mattias.strand@his.se
[2] School of Humanities and Informatics, University of Skövde, Sweden
bjorn.lundell@his.se

1 Introduction

Organizations have problems with their initiatives on incorporating syndicate data into data warehouses (DWs) and therefore, they are not able to fully exploit the potential thereof [9,10]. For clarification, a DW is a: "*subject-oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions*" ([5], p.33). From a user organization perspective, these problems have been categorized and verified, whereas the supplier side of the problem, i.e. how they interoperate with the user organizations, and tensions between the suppliers and the consumers[1], is unexplored [11]. The syndicate data supplier (SDS) perspective is important, as the user organizations incorporate most of their external data from these specialized data suppliers [9]. Therefore, we present the results of an interview study towards SDSs, partly aimed at contrasting the enhanced list of syndicate data incorporation problems experienced by user organizations, with the viewpoints, experiences, and work routines of the SDSs.

Contrasting the problems experienced by the consumers, from a SDS perspective, is important for at least three reasons. Firstly, it allows for an evaluation of the interoperability between the user organizations and the SDSs. Syndicate data incorporation is a multi-facetted undertaking, covering organizational as well as technological aspects and therefore such an interoperability evaluation may cover all three layers given by Chen and Doumeingts [1]: the business layer, the knowledge layer, and the ICT systems layer. Secondly, it extends the current body of knowledge of syndicate data incorporation into DWs, by describing tensions and collaborations between the consumers and the SDSs. Thirdly, it contributes with important details when

[1] For the rest of this paper, *user organization* and *consumer* will be used interchangeably to denote an organization that buys syndicate data from one or several SDSs.

developing a hands-on support for organizations incorporating syndicate data into their DWs (as suggested by Strand and Wangler, [10]), by including details on e.g. how they establish business relationships with their customers and to what degree the SDSs allow the user organizations to tailor the syndicate data.

In articulating our findings, we identify a number of important collaborations and tensions between the consumers and the SDSs, indicating that the two parties are rather mature in their interoperability on the ICT-systems layer, whereas more attention should be given the interoperability on both the business- and knowledge layers.

The rest of the paper is organized as follows: Section 2 describes briefly some related work. In Section 3, the research method is described and motivated. Section 4 presents the analysis of the interview study. Section 5 gives the results of the paper. In Section 6, we summarize the paper and outline future work.

2 Related Work

Kimball [7] gives the first instances of the concepts *syndicate data* and *SDSs*. However, he does not define syndicate data. Therefore, we have adopted the following definition from Strand et al. [11]: "*Business data (and its associated metadata) purchased from an organization specialized in collecting, compiling, and selling data, targeted towards the strategic and/or the tactical decision making processes of the incorporating organization*".

Furthermore the syndicate data incorporation problems, experienced by the user organizations, are given in Table 1. The problems are categorized according to the activities of the external data incorporation process, i.e. identification, acquisition, integration, and usage. (For a more detailed description of the process activities we refer to Strand and Wangler [10]). These activities are not unique for syndicate or even external data (external data is a more general concept, including syndicate data as a subtype, due to its acquisition from specialized data suppliers. Other external data may be acquired from e.g. industry organizations or business partners [9]). On the contrary, they may be considered as rather generic, since they are also included in most data warehouse development processes, regardless of whether the data has an internal or external source (e.g. [5,3,4,2]). Still, the process of including syndicate data (or external data) differs from the process of incorporating internal data in two ways: 1) the data is acquired from outside the organizations and thereby crosses organizational boundaries, which may cause other types of problems than those experienced with internal data [8,10] and 2) syndicate data is bought from special suppliers and therefore have a monetary cost associated [6,7].

Table 1. The enhanced list of syndicate data incorporation problems (adopted from Strand et al. [11]).

Identification problems
Id.1 – Identifying new entrants
Id.2 – Overlapping suppliers' capabilities
Id.3 – Overlapping data or products/services
Acquisition problems
Ac. 1 – Acquiring incomplete data sets
Ac.2 – Varying data source stability
Ac.3 - The syndicate data is expensive
Integration problems
In.1 – Demanding to design and maintain transformation processes
In.2 – Diverging data representations and structures
In.3 – Assuring data consistency
In.4 – Missing data identifiers
In.5 – Diverging time-stamps
In.6 – Conflicting data from multiple sources
In.7 – Hiding data quality issues in commercial ETL-tools
In.8 – Varying source content
Usage problems
Us.1 – Misunderstanding the meaning of data
Us.2 – Missing metadata
Us.3 – Lacking routines for data quality assurance
Us.4 – Making decisions on outdated data
Us.5 – Trusting the data
Us.6 – Contradicting data from multiple sources
Us.7 – Ignoring syndicate data for DW purposes
Us.8 – Restricting laws and regulations

3 Research Approach

The research approach of this work is to conduct an interview study towards SDSs. The interview study partly focused on contrasting the syndicate data incorporation problems in Table 1, towards the SDSs' perspectives, experiences, and work-routines. For being able to contrast the answers from the suppliers, with these problems, the interview questions were constructed from an opposite perspective with respect to the problems. For example, one of the problems given in Table 1 is that organisations find it demanding to design and maintain transformation processes Problem – In.1. As a consequence, the question stated towards the SDSs was: *to what degree do you allow your customers to tailor the data so that it fits with their own needs?* (Question translated from Swedish). In addition, follow-up questions were also stated, when the respondents answered the questions too generally or in an avoiding manner, or to broaden the material and acquire illustrative examples. Therefore the interviews may be considered as semi-structured [12].

The interview study included 9 interviews covering 8 different SDSs. Due to the strong competition and strong relation between different companies, all respondents wanted to stay anonymous and they will only be denoted as Respondent 1-9. In addition, as all companies sell data for many different purposes, the respondents have answered the questions with respect to specific data or services/products targeted towards DW incorporation. Furthermore, Company B was covered by two respondents due to their broad coverage of different databases and services. In addition, in one of the interviews, two respondents were participating. There answers were therefore separated in the transcript and they are denoted as 8A and 8B, respectively. The interviews were conducted via telephone and the interviews were taped and transcribed. The interview lasted in an average of 90 minutes and the transcripts of the interviews ranged from 3560 to 7334 words (5356 words in average). The transcripts were then returned to the respondents for validation, allowing them to, for example, correct misunderstandings or complement their answers. When the transcripts had been validated, they were included in the analysis.

4 Contrasting the Problems

In this section, we contrast the problems in Table 1 with the viewpoints, experiences, and work-routines of the SDSs. Some of the problems were impossible to respond to by individual suppliers, e.g. *Problem 3.6 – Conflicting data from multiple sources*. (All questions and quotations are translated from Swedish). Instead, these problems have been contrasted with the authors summarized understand of the particular topic, based on the statements of all or some respondents.

4.1 Contrasting the Identification Problems

Id.1 – Identifying new entrants
All respondents claimed that the industry is under a strong competition, even though a few suppliers are benefiting from a monopoly situation. All respondents also stated that due to the strong competition, they exercise a strong supplier-push towards the consumers. Every supplier had internal resources or bought outsourced capacities for conducting sales initiatives towards prospects or established customers.

Id.2 – Overlapping suppliers' capabilities
The analysis of the suppliers' data, products or services shows that many suppliers are capable of delivering approximately the same data and services. As a consequence, a majority of the suppliers indicated that they naturally compete with data and services offered, but equally much with other, competitive means, like e.g. availability, degree of refinement, and support. Above all, a majority of the respondents claimed that the refinement aspect, i.e. the intelligent combination of the customers' internal data and novel, relevant syndicate data added by the suppliers, was considered as the main competitive edge for the suppliers.

Id.3 – Overlapping data or products/services
The analysis shows that most of the suppliers strived towards offering different standardized sets or packages of data, of which the user organizations must select one or several for procurement. Normally, the data sets or packages are overlapping and the different sets are arranged according to the amount of data contained. As a consequence, the consumers sometimes are forced to procure a more expensive and extensive data set, with respect to content, for being able to acquire a specific parameter or attribute. However, a majority of the suppliers also claimed that they allow a high degree of tailoring of the data, since the customers have such varying demands. To illustrate, Respondent 6 gave the following statement: *"We try to have standardized solutions, but it has been shown that our customers have very varying demands, so for being able to compete, flexibility is a key-word. Therefore, we try to build a generic platform that allows flexibility towards each and every customer"*.

4.2 Contrasting the Acquisition Problems

Ac. 1 – Acquiring incomplete data sets
The analysis shows that all respondents accounted for a high data quality awareness and claim that it is a prerequisites for being able to survive at the market, because *"without a high quality of the data, you are out of the market"* (Respondent 8b). Most suppliers also conducted tool-based automatic data verifications, e.g. verifying that every record identifier had a corresponding record or that the data sets are complete and a majority also conducted manual data verifications, e.g. contacting organizations if important data are missing.

Ac.2 – Varying data source stability
The analysis shows that the root of this problem is handled very differently by the suppliers. Some of the respondents acquired the internal data from their customer, refined it, and sent it back via FTP to the customers. With this approach, the problems should not appear, as the user organizations are not the active part going in and downloading data from e.g. a web-hotel or a FTP-mailbox. A majority of the suppliers also applied alternative distributed techniques, like CDs/DVDs or e-mail attachments, which also avoids this type of problem. However, a few organizations required the customer to be the active part and access FTP-mail-boxes or Web-hotels.

Ac.3 – The external data is expensive
The analysis shows that most suppliers claimed that the pricing of the external data is under a strong pressure and upon a question related to the competition of the market, Respondent 5 states: *"it is very hard [...] and the price for the raw data is fast approaching the marginal cost"*. Consequently, and naturally, the suppliers and the user organizations have varying opinions on whether the syndicate data is expensive or not. More interestingly, the suppliers indicated two other reasons for why the consumers may consider syndicate data as expensive: 1) most respondents claimed that the ordering competence is very varying among the user

organizations, which may result in data acquisitions that do not meet the actual needs or expectations, and 2) a majority of the suppliers claimed that the consumers do not exploit the full potential of the data they buy.

4.3 Contrasting the Integration Problems

Generally, one may claim that the suppliers have a very good understanding of the problems related to integration and some of the suppliers have taken another approach to the data transformation problem, by receiving the data from the user organizations and integrate and refine it on the supplier side, before returning it back to the user organization. A few respondents even claimed that they constantly kept mirrors of the users' data, which they were periodically updating, e.g. customer master dimensions. Still, a majority of the suppliers do apply the more traditional approach of data sales, i.e. the suppliers distribute the data to the consumers, which cater for the integration efforts, either as the only approach or upon the customers' choice. Therefore most issues or examples given further on are related to the traditional approach.

In.1 – Demanding to design and maintain transformation processes
An analysis shows that all suppliers allow for some degree of tailoring of the data, but one should also be aware that the degree of tailoring is vastly shifting. A majority of the suppliers claimed that they allow for a high degree of tailored acquisitions, but it should also be brought forward that most of these tailored acquisitions still are based upon standardized outtakes of data, which do delimit the customers' abilities to totally customize their acquisitions. Respondent 3 was even more outspoken and claimed that: "*The abilities to tailor the data are big, BUT, in practice we rarely do that because it is a question of how we should use our resources. It is always risky to tailor data acquisitions, as they create a maintenance need that becomes very costly for us*".

In.2 – Diverging data representations and structures
Some of the respondents exemplified on internal codes for e.g. arranging companies according to industries or business, which diverged from the governmentally established code. They proposed structuring and analysis issues for introducing novel codes. A majority of the suppliers also promoted the XML as a key standard, as it allows for more flexible structuring and restructuring of the data. For example, Respondent 5 stated that "*XML in combination with the Internet is, for us that have been writing communication protocols, like a dream come true. It is a complete dream*" Furthermore, A few respondents indicated that they cooperate, or planned to collaborate, with software suppliers on special certificates. The underlying idea was that SDSs and software agree upon different representations of data and thereafter certifies these representations, meaning that a user organization following the certificate, i.e. procure the software from the particular vendor and the data from the particular SDSs, do not have to transform the syndicate data being incorporated. Thereby, the user organizations drastically reduce the resource they have to spend on data transformations and integration.

In.3 – Assuring data consistency
This issue was very difficult for the suppliers to relate to, as they normally do not know the internal systems structure at the customer side. However, a few respondents briefly indicated that one may question if the consumers really have a full control of their internal systems or if they have designed their systems so that the syndicate data updates are reflected all-over.

In.4 – Missing data identifiers
Besides the general comment given in the introduction to this section, the analysis of the study gives no further details on why this problem occurs. Still, a majority of the suppliers have specifically pinpointed the importance of identifiers for their internal data quality verifications and storage.

In.5 – Diverging time-stamps
Since this is very much related to how the user organizations design their integration processes, the suppliers could not comment upon this. Still, in those cases where the non-traditional integration approach is applied, the suppliers should cater for a correct time-stamping.

In.6 – Conflicting data from multiple sources
Due to the reference to multiple suppliers or sources, the respondents could not contribute with any details to this problem. Still, one may imagine that organizations encounter this problem, if combining demographic data from one supplier and economic data from another, since some of these suppliers have overlapping data/services.

In.7 – Hiding data quality issues in commercial ETL-tools
Since this is very much related to how the consumers design their integration processes, the suppliers could not comment upon this. Still, in those cases where the non-traditional integration approach is applied, the SDSs should cater for a correct time-stamping.

In.8 – Varying source content
All respondents claimed that they are very careful with changing the data and most have contracts with the user organizations, regulating e.g. data content, data format, data structures. Still, based upon the answers of the respondents, one may derive that approximately 50 % of all DW customers acquire the syndicate data on-demand, instead of subscribing to it. Such customer may end up in problems, as they may have ETL-processes that have been out-dated due to the fact that the suppliers have changed the data formatting they allows or the data content they deliver.

4.4 Contrasting the Usage Problems

Us.1 – Misunderstanding the meaning of data
In order to prevent that the consumers misunderstand the meaning of the data, a majority of the suppliers worked together with the user organizations in projects,

when starting up a business relationship. Since the incorporation of syndicate data into DWs is a rather extensive undertaking, the project form was considered as a necessity for being able to establish all the needs of the user organizations. To exemplify, Respondent 6 stated that *"most important in such a project is to establish a specification of the customers' needs, since we may then use the specification as a validation instrument and show the customer how the data contributes and which needs it fulfil"*. However, some of the respondents gave the impression of working in more loosely coupled supplier-consumer constellations, where no detailed contacts are established between the two parties and in such cases one may assume that misunderstandings arise, especially since most respondents claimed the order competence of the consumers as strongly varying.

Us.2 – Missing metadata

All respondents claimed that metadata was included with the data that was distributed to the customers. In addition a few respondents claimed that they have a metadata service, which the users could use for verifying the meaning of the data incorporated.

Us.3 – Lacking routines for data quality assurance

As indicated previously, all respondents were shown to be very data quality aware and a majority of the suppliers spent a lot of resources on conducting automatic and manual data quality verifications. A few respondents even claimed that they hire external organizations, labelled data quality verifiers, to manually verify that the data is correct. For example, by phoning private persons and ask if their names are spelled correctly or by contacting companies and verify that a certain person still is the manager of a certain department or division.

Us.4 – Making decisions on outdated data

This problem transcends from the data sources of the SDSs and some of the respondents claimed that their sources are sometimes rather slow on generating procurable data. For example, it may take one and a half year before a SDS gets the annual accounts of an organizations or it may take more than a month before an established organization is registered in the SDSs' databases and possible for user organizations to get informed of. Respondent 8 further exemplified that: *"it happens that our customers inform us that they have found an organization or a part of an organization that we do not have in other systems"*. Still, a majority of the suppliers have not acknowledged this as a problem and that it mostly comes down to the routines of the user organization on whether they make decisions on out-dated data or not.

Us.5 – Trusting the data

The suppliers had problems to give any particular details to this, but they indicated on an average that 50% of their customers were subscribing the data and that the customers acquiring on-demand tended to be rather faithful, which could be seen as an indication that the consumers trust their suppliers. Naturally, the selection of the respondents effects the response regarding this problem, since the suppliers participating were established and with a good reputation. Still, most of the respondents claimed that they were actively working for having a constructive relationship with their customers and a majority of the respondents exemplified on novel data or products/services that were a result of long business relations.

Us.6 – Contradicting data from multiple sources
Due to the reference to multiple suppliers or sources, the respondents could not contribute with any details to this problem. Still, one may imagine that organizations encounter this problem, if combining demographic data from one supplier and economic data from another, since some of these suppliers have overlapping data/services.

Us.7 – Ignoring syndicate data for DW purposes
A few respondents indicated that many organizations have problems bridging different parts of the organizations as the systems are stow-piped, making the syndicate data updates isolated or fragmented. Respondent 1 gave the following illustrative example: "*An organization may have 7-8 registries that do not interact, so there are many integration benefits that may be achieved. For example, an organization may have a customer registry, a supplier registry, a prospect registry, and a competitor registry, which are not integrated in any way. If merging these together, into e.g. a DW, one may find that a supplier is also a customer and a supplier may also be a prospect in other occasions. There I think it is a lot that needs to be done*". A majority of the respondents also claimed that incorporating syndicate data into DWs is a rather novel concept, compared to acquiring syndicate data for more operative purposes, e.g. verifying the customer address of a single customer or acquiring the solidity of a particular organization. Thereby, many user organization are rather immature when it comes to incorporating syndicate data in DWs and therefore do not have the experience for being able to fully exploit the potential thereof. However, most of the respondents claimed that the syndicate data incorporation will inevitably steer into a DW direction, as the combination of DW technology and syndicated data caters for novel and powerful abilities to perform various types of business analysis. Furthermore, a majority of the respondents also claimed that they try to influence organizations to make better use of the syndicate data, by indicating the possibilities of incorporating into DWs, if the user organizations were not already exploiting that possibility.

Us.8 – Restricting laws and regulations
A majority of the respondents informed the consumers on what they may legally do with the data, whereas some of the suppliers clearly stated that it is up to the user organizations to stay updated on law and regulations. In addition, all respondents established contracts with the consumers, in which they, amongst other things, regulated for what purposes the syndicate data may be applied. For example, all suppliers stated in their contracts that the data they sell may not be applied for establishing competing business, i.e. the consumers are not allowed to sell the data in their turn. The laws and regulations, combined with the contracts established by the suppliers, may also be hinderers for the user organizations to fully exploit the potential of the data acquired (see problem Ac.3), as the consumers may not dare to apply the data for other purposes than those regulated

in the contracts, since the regulations and laws are very complex and requires a lot of resources to stay updated with.

5 Results

The analysis of the problems gives a lot of details related to the suppliers' viewpoints, experiences, and work-routines related to their part of the problem and thereby contextualize the problems from a SDS perspective. In addition, the analysis also reveals some collaborations and tensions on all three interoperability layers of the organizations, which facilitate or hinder interoperability between the user organizations and the SDSs. These collaborations and tensions are given below:

Collaborations:

- The user organizations and SDSs collaborate in projects, when incorporating syndicate data into DWs, as these initiatives are rather demanding undertakings, stating technological challenges, as well as organizational challenges.
- The consumers and data suppliers are collaborating on developing novel services, based upon the demands of the user organizations or based upon the competencies of the suppliers, knowing what types of data or services the organizations may benefit from.
- The suppliers are very data quality aware and collaborate with the consumers on identifying missing data.
- The suppliers know that the user organizations have problems with their data quality verifications and therefore have a strong focus on assisting the consumers with these issues.
- The user organizations seem to be rather pleased with their suppliers and satisfied with their services, as approximately half of the consumers are subscribing on syndicate data for longer time-periods and the suppliers consider their customers as rather faithful.
- The data suppliers collaborate with the user organizations in order to assist them in exploiting the potential of the syndicate data as efficiently as possible.

Tensions:

- The SDSs strive towards standardized data sets or packages, since it facilitates the maintenance thereof and is cost efficient, as tailored solutions become expensive for the SDSs to maintain, especially if every single customer would like to tailor their acquisitions of data. Unfortunately, the standardization sometimes forces the user organizations to procure more expensive and extensive standardized data sets, than they actually need, for being able to acquire a specific parameter or attribute.
- The suppliers are conducting a strong marketing and sales push towards the consumers for buying novel data, despite the fact that they know that the

user organizations do not exploit the potential of syndicate data already being incorporated.

- The suppliers regulates the consumers with contracts, which may hinder the user organizations to fully exploit the potential of the data acquired, as they may not dare to apply the data for other purposes than those exactly stated in the contracts, since the contracts are very complex and requires a lot of resources to stay updated with.
- The consumers and the suppliers have very diverging opinions on whether the syndicate data is expensive or not and pricing issues may naturally cause tensions between the user organizations and the supplier.
- The suppliers state that the ordering competency is very varying among the user organizations, but still some of them outsource their marketing and sales initiatives. Thereby, it becomes even harder for the consumers to actually acquire the most appropriate data, since this intermediate actors distances the user organizations from suppliers.
- User organizations spend a lot of internal resources on transforming syndicate data and pays for non-relevant data, as the suppliers strive towards standardized data sets.

It is interesting to reflect upon the tensions and collaborations as some of them contradict each other, e.g. on one hand the suppliers conduct a strong push for selling novel data and at the same time they assist the consumers to fully exploit the potential in already acquired data. In addition, five out of six tensions are relatable to the business and knowledge layers. Together these indicate that the two parties are rather immature in their interoperability on these layers, whereas on the ICT-systems layer, covered by the acquisition and identification problems, only one tension were identified, and it was a consequence of incorporating data according to the traditional approach, instead of letting the suppliers handle the integration and refinement. As another example for a higher maturity on ICT-systems layer, the mirroring of the customers data could be given, showing that the suppliers and the user organizations interoperate to produce the best possible result.

6 Summary and Future Work

As indicated in the introduction, a SDS contextualization of the syndicate data incorporation problems experienced by the consumers served several purposes. Firstly, the contextualization allowed for evaluating the maturity of the interoperability between the SDSs and the user organizations. As shown in the results, the more organizational oriented aspects need more attention, as most of the tensions were of such a kind. Therefore, it is reasonable to conduct further research aimed at improving the interoperability between them. Consequently, one should focus on the tensions and try to solve the causes generating these tensions, by e.g. trying to develop models for calculating and making the pricing of the syndicate data for DW incorporation more obvious. Secondly, the contextualization gave general tensions and collaborations between the two parties. Future work could be devoted to deepen and detail the descriptions of and causes for these tensions. Finally, the contextualization contributed with important details for a hands-on support for organizations incorporating syndicate data into their DWs (as suggested by Strand et al. [11]), by including details on e.g. how they establish business relationships with customers, to what degree they allow the organizations to tailor the data, and how they respond to suggestions for novel services. The development of such a hands-on support is now being undertaken and will be presented in the shape of detailed guidelines or heuristics, accompanied with descriptions of the problem context and possible solutions.

References

1. Chen, D. and Doumeingts (2003) European initiatives to develop interoperability of enterprise applications – basic concepts, frameworks and roadmap. Annual Reviews of Control (27) pp. 153-162.
2. Damato, G. M. (1999) *Strategic information from external sources: a broader picture of business reality for the data warehouse*. Available at Internet: http://www.dwway.com/file/20020726170552_get_ext_data.pdp [Accessed 03.02.20]
3. Hammer, K. (1997) "Migrating data from legacy systems", in *Building, using, and managing the data warehouse*, in Ramon Barquin and Herb Edelstein (Eds), New Jersey: Prentice Hall PTR, pp. 27-40.
4. Hessinger, P. (1997) "A renaissance for information technology" in *Data warehouse practical advice from the experts*, Joyce Bischoff and Ted Alexander (Eds), New Jersey: Prentice Hall PTR, pp. 16-29.
5. Inmon, W. H. (1996) *Building the data warehouse, 2nd edition.* New York: John Wiley & Sons.
6. Kelly, S. (1996) *Data warehousing: the route to mass customization.* New York: John Wiley & Sons.
7. Kimball, R. (1996) *The Data Warehouse Toolkit.* New York: John Wiley & Sons.
8. Oglesby, W. E. (1999) *Using external data sources and warehouses to enhance your direct marketing effort.* DM Review. Available at Internet: http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=1743 [Accessed 03.02.21].
9. Strand, M., Wangler, B. and Olsson, M. (2003) Incorporating external data into data warehouses: characterizing and categorizing suppliers and types of external data. *In Proceedings of the Americas Conference on Information Systems (AMCIS'03),* 4-6 August, 2003, Tampa, Florida, USA, pp-2460-2468.
10. Strand, M. and Wangler, B. (2004) Incorporating external data into data warehouses - problems identified and contextualized. Presented at the *7th International conference on information fusion (Fusion'04)*, June 28- July 1, Stockholm, Sweden.
11. Strand, M., Wangler, B., Lundell, B. and Niklasson, M. (2005) Syndicate data incorporation into data warehouses: a categorization and verification of problems. Submitted to an international conference.
12. Williamson, K. (2002) Research methods for students, academics and professionals. 2nd Edition, Thousand Oaks: Sage Publications, Inc.

Conformance Testing of Open Interfaces in Healthcare Applications - Case Context Management

Tanja Toroi[1], Juha Mykkänen[2] and Anne Eerola[1]

[1] University of Kuopio, Department of Computer Science, P.O.B 1627, FIN 70211 Kuopio, Finland, Tanja.Toroi@cs.uku.fi, Anne.Eerola@cs.uku.fi
[2] University of Kuopio, Information Technology Centre, HIS Research and Development Unit, P.O.B 1627, FIN 70211 Kuopio, Finland Juha.Mykkanen@uku.fi

Summary. In this paper we describe the conformance testing model of the open interfaces developed and applied in the PlugIT project in Finland during 2003-2004. Conformance testing is needed to integrate different software products without a vast amount of extra adaptation work, and to improve software inter-operability. The model has been developed and evaluated with co-operation of several healthcare software companies and hospital districts. The clinical context management interface specification is used as an example in this paper.

1 Introduction

The number of information systems and integration needs between these systems is large in healthcare organizations. For example, in Kuopio University Hospital there are more than 180 information systems which have to communicate with each other. At the moment, new systems are integrated to existing ones by tailoring them separately. This is extremely expensive in the long run. If the systems have open, standard-based interfaces their interoperability improves, introduction and integration become easier and less local adaptation work is needed. Also, conformance testing is needed to examine if the systems really conform to standards or other specifications. When certificates (brands) are issued to specification-based software, software developers can use them in marketing and documentation. Also, customers, such as healthcare organizations can benefit from the specifications and brands. They can append specifications to the call for tenders and require standard-based software. Software components, which conform to specification can be easily integrated with or replaced by other components.

ISO/IEC defines that conformance is the fulfilment of a product, process or service of specified requirements [8]. A conformance clause is defined as a section of the specification that states all the requirements or criteria that must be satisfied to claim conformance. Conformance testing is a way to verify implementations of the specification to determine whether or not deviations from the specifications exist [9]. Conformance testing is necessary, but not sufficient, for interoperability, because conformance clauses (or specifications) typically only cover some aspects of interoperability.

In conformance testing software products are black boxes; only the interfaces and their relationship to the specifications are examined. In other words, when the soft-ware, which offers functionality or services through an interface receives an input, it is tested if the interface can handle and respond to the input correctly.

In the PlugIT project (see http://www.plugit.fi/english/) we noticed that conformance testing of open interfaces in the healthcare software products need to be examined and elaborated. Thus, we have developed a conformance testing model to test open interfaces. The model consists of four phases: an initial phase, testing performed by the developer, testing performed by the testing lab, and certificate issuing (see Section 4). In the project a HL7 standard, CCOW, is adapted and only the most essential parts of it have been included to the minimum level specification (see Section 2). We have also developed reference implementations to test and evaluate the test cases for conformance testing. Although, we have studied healthcare information systems and their integration, similar integration needs can be found in, and our model is applicable to the other domains or integration models.

However, it should be noticed that conformance testing can not be used as verifica-tion. Conformance testing only increases the probability that applications are implemented according to the interface specification. Normal software inspections and testing processes must be performed by the developer before conformance testing.

2 Background

National Project to Secure the Future of Healthcare in Finland has given recommendation in spring 2002: ”The interfaces between healthcare systems shall be made obliging to all healthcare actors by degree by the Ministry of Social Affairs and Health by the year 2007.” The PlugIT project was one step forward by contributing to the implementation of the recommendation by developing solutions for common services and clinical context management, which can be nationally standardized. PlugIT (2001-2004) was a research project, which aimed at decreasing the introduction threshold of healthcare software applications by developing efficient and open standard solutions for integrating them in practice. We have noticed that conformance testing is needed to assure that the applications follow interface specifications so that the application integration and introduction can be done without any extra adaptation or development work. We have developed a model for testing the open interfaces. In the model the applications are given certificates (or brands) if they are in accordance with the interface specification.

Interoperability solutions have been defined in the PlugIT project by developing healthcare interface specifications for clinical context management (Context manager and Context data interfaces) and common services (User and access rights interface, Patient data interface, and Terminology interface). The developed context management interface specification (see Table 1) is a simplified specification of CCOW (Clinical Context Object Workgroup) standard. CCOW is a vendor-independent standard developed by the HL7 organization to allow clinical applications to share information at the point of patient care [3]. CCOW allows information in separate healthcare applications to be synchronized so that each individual application is referring to the same patient, encounter or user. However, CCOW was found to be too extensive and complex to be used in context management in healthcare information systems in Finland. That is why only the most essential and worthwhile parts of the CCOW standard, i.e. user and patient context management, are included in the minimum level specification.

Table 1. Minimum level context management interface specification

Interface	Method	Raises exceptions
ContextManager	JoinCommonContext(string applicationName)	AlreadyJoined, TooManyParticipants, GeneralFailure, NotImplemented
	LeaveCommonContext (long participantCoupon)	UnknownParticipant, GeneralFailure, NotImplemented
ContextData	SetItemValues(long participantCoupon, string[] itemNames, string[] itemValues)	UnknownParticipant, NameValueCountMismatch, BadItemNameFormat, BadItemType, BadItemValue, GeneralFailure, NotImplemented
	GetItemValues(long participantCoupon, string[] itemNames)	UnknownParticipant, BadItemNameFormat, UnknownItemName, GeneralFailure, NotImplemented

During the PlugIT project, reference implementations have been implemented and two companies have released commercial implementations of the context management solution. HL7 Finland Association has accepted minimum level context management interface specification as a national recommendation. The specification will be further developed in the SerAPI project [10] and HL7 Finland Common Services SIG (Special Interest Group).

At the moment conformance testing has been performed against the minimum level context management interface specifications, but in the future also the other interfaces (e.g., common service interfaces produced by the PlugIT project) will be covered. Examples in this paper deal with context management interfaces.

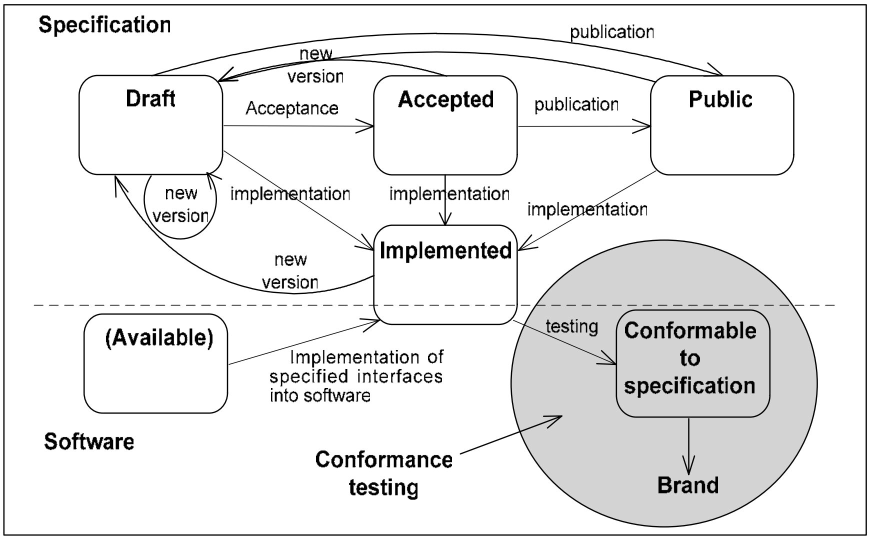


Figure 1. Relationship between software (below) and specification (above)

The relationship between the software and specifications, and the conformance testing model in the PlugIT project is illustrated in Figure 1. In the Figure, specification is above the dashed line and software is below. Specifications are in different states. At first, specifications are drafts. When a draft specification is accepted by management group it becomes an accepted version of the specification. If an accepted specification is published it becomes a publicly available specification. Software has three states, available, implemented or conformable to specification. Available state means that there exists the software but it does not necessarily have any implemented interfaces according to the specifications. Implemented state means that the specified interfaces have been implemented to software. When the developer has implemented interface implementations to the software, testing has to be performed to verify conformance to the specification. A Certificate issuer / Testing lab performs conformance testing and if the software passes the tests the software becomes conformable to specification state and the brand can be issued to the software. The model is described in more detail in Section 4.

3 Related Work

Integrating the Healthcare Enterprise (IHE) [7] promotes the use of standards by developing integration profiles. With the help of integration profiles standards can be implemented to the product at more accurate level.

Australian Healthcare Messaging Laboratory (AHML) [1] is part of the Collaborative Centre for eHealth (CCeH) at the University of Ballarat. AHML provides message testing, compliance and certification services for developers and vendors, system integrators and users within the healthcare industry. Clients can send electronic healthcare messages via Internet to the AHML Message Testing Engine. Messages are tested against healthcare messaging standards. Test cases are built using messaging standards and client-specific requirements, if any. Test history is saved and test reports can be viewed in detail or in summary level.

The Software Diagnostics and Conformance Testing Division (SDCT), part of NIST's (National Institute of Standards and Technology) Information Technology Laboratory is working with industry by providing guidance on conformance topics, helping to develop testable specifications, and developing conformance tests, methods and tools that improve software quality and conformance to standards [11]. SDCT has especially focused on developing XML conformance test suites, and has also written guidance on how to write better specifications.

Health Level 7 (HL7) Conformance SIG (Special Interest Group) attempts to im-prove interoperability and certification processes [5]. HL7 provides a mechanism to specify conformance for HL7 Version 2.X and HL7 Version 3 messages and provide a framework for facilitating interoperability using the HL7 standard. The problem with HL7 version 2.X standards is that they are complex to use in practice and the users are unable to ascertain compliance with standard against their specific needs [12]. As a solution HL7 has developed message profiles, which add specificity to existing messages and identify scenarios by providing a template for documenting particular uses of HL7 messages. In our model a HL7 standard, CCOW, is adapted to and only the most essential parts of it have been included to the minimum level specification.

Applications can use several interaction styles in application integration, such as direct database access, database-based interoperability adapters, interoperability tables or API-based interfaces with broker techniques [6]. Furthermore, for example, in a layered architecture integration can be performed in workspace, business logic or resource tiers, and every tier has its own special features. We use API-based, workspace tier interface integration in our conformance testing model.

4 Conformance Testing in the PlugIT Project

4.1 General

We have developed and used a conformance testing model in the PlugIT project during 2003-2004. Carnahan et al. [2] have introduced that interaction among roles and activities in conformance testing process can be illustrated by a diamond where each role (seller, certificate issuer, testing lab and control board) is in the corner of the diamond. In the PlugIT project certificate issuer and testing lab were not separate roles. Thus, we have adapted the idea and describe the interaction by a triangle. Interaction among roles and activities in conformance testing model in the PlugIT project is illustrated in Figure 2 (adapted from [2]). A buyer (customer) requires from a seller (developer) that a product conforms to the specification, and asks the developer to append the brand in the call for tenders. The developer develops an application with certain interface implementations and applies for the brand to the application. The testing lab/certificate issuer (=PlugIT project) performs interface testing and issues the brand if the application successfully completes the conformance testing. The control board is answering queries and disputes related to the testing process. The developer is responsible for the product even if it had got the brand and passed the conformance testing.

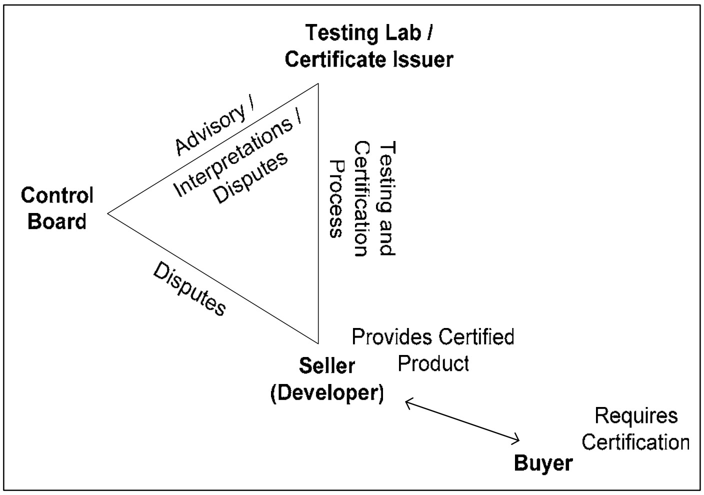

Testing Lab / Certificate Issuer
Advisory / Interpretations / Disputes
Testing and Certification Process
Control Board
Disputes
Provides Certified Product
Seller (Developer)
Requires Certification
Buyer

Figure 2. Interaction among roles and activities (adapted from [2])

The brand is issued to the product which conforms to the specification. To have the brand means that the product has passed the conformance testing process of the open interface specifications. The criterion for issuing the brand is that all the test cases (100%) for required features have to be passed. The brand also means that the product is supported by adequate documentation describing how to introduce, configure and use the integration solution in the product. The brands have been issued by the PlugIT project. Conformance testing does not remove responsibility from the developer. The developer has to perform normal system and integration testing and documentation before applying for the brand. Conformance testing assures that the interface operates in a tested environment with tested test cases. However, it can not guarantee operation in different environments, as the platform (or infrastructure) of the software is not covered by the integration specification. Developers can use brands in documentation and marketing. Customers can require brands in calls for tenders. The brand includes the following information:

- the software product and the version of the software product,
- the interface specification and the version of the specification,
- the level of conformance, if the specification has different conformance levels (e.g. minimum, basic, advanced),
- reference to the test report, and
- date and signatures.

4.2 Phases of the Model

The conformance testing model consists of four phases: An initial phase, testing performed by the developer, testing performed by a testing lab, and issuing the brand. The phases will be further discussed in the following subsections.

Initial Phase
In the initial phase, the developer informs the testing lab/certificate issuer that the integration solution is implemented into the product. The developer has to define to which product version the brand is being applied, which interface specifications and what level of conformance is implemented, and who is the contact person for the brand issuing process. The developer delivers the product interoperability description of the implementation to the certificate issuer. The product interoperability description describes major interoperability, integration, and introduction issues, but it does not include any inner details of the implementation. However, implementation-specific additions, expansions, deviations and removals of the specification, if any, must be stated clearly, so not causing contradictions when integrating the implementation with other software. The description includes also instructions for the introduction, installation and configuration. Additionally, it is recommended that the description has usage or configuration examples of the integration implementation.

Testing Performed by the Developer
The developer can start this phase when normal integration and system testing, and inspections have been performed and passed. The aim of the conformance testing performed by the developer is to assure conformance and to fix failures before applying for the brand. Developer tests the product with test cases received from the testing lab and with own test cases. It is essential that only the interfaces in the specification are tested, not the implementation-specific features. In the PlugIT project reference implementation of the context management client was developed to support the testing of the server part. Test cases (see Section 4.3) were developed using the black box testing method and the reference implementation. Test cases were sent to the developer for the self testing.

Testing Performed by the Testing Lab/Certificate Issuer
The aim of the testing in the testing lab is to complement testing, to ensure conformance to the specification, and to make sure that the implementation is functioning in practice. PlugIT project itself acted as both testing lab and certificate issuer. Testing was organized in practice using reference implementations. Testing environment was developer's development environment.

Not all the same test cases as in the previous phase are necessarily retested. Which of the test cases have to be retested depends on resources and situation. In our case, as the interface was simple, it was fairly easy to find adequate test cases with enough coverage. It is also possible to test the product together with the other products which have already gained the brand. If the testing is passed, a test report, which includes the summary of the test cases and their pass/fail information, is generated (see Section 4.4). Also the documentation is checked if it contains necessary information for the introduction of the integration solution.

Issuing the Brand

The last phase in the conformance testing model is the brand issuing. If all the tests have been passed and the documentation is adequate the certificate issuer issues the brand and delivers it to the contact person of the developer. Test reports are made public and published, for example, on the web page.

4.3 Test Cases

In this Section we give some examples of the context management test cases used in conformance testing in the PlugIT project. The basic context management operation flow follows, for example, the following path:

- A client application joins the context management (JoinCommonContext).
- The client application asks a user context (GetItemValues).
- The client application asks a patient context (GetItemValues).
- The client application changes the patient (SetItemValues).
- The client application updates the patient context (SetItemValues).
- The client application leaves the context (LeaveCommonContext).

Test cases are generated based on the previous path and on error situations. Each test case includes definition, precondition, input, and output information. Input is in URL format and is suitable for testing of the interfaces. Some of the values must be replaced with site-specific or application-specific parameters, such as IP address and application names. Some test cases can have an input value that is an output value of another operation, such as participantCoupon is an output value of JoinCommonContext and an input value for SetItemValues. Examples of the test cases in conformance testing are described in Tables 2-4. First there are two general context management operations: joining the context and getting the values from the context. Next there is a typical failure where the coupon, which is used in the identification of the session, is used after leaving the context.

Table 2. JoinCommonContext

Definition:	Client application joins the context management.
Precondition:	Application name must be accepted and another application has not joined the context with the same application name.
Input:	http://193.167.225.119/cm.pp?interface=ContextManager&method=JoinCommonContext&applicationName=LoginMaster
Output:	participantCoupon=11900200

Table 3. GetItemValues

Definition:	Client application asks the patient context
Precondition:	Application has joined the context, item has been set, participantCoupon has been received when joined.
Input:	http://193.167.225.119/cm.pp?interface=ContextData&method=GetItemValues&participantCoupon=11900347&itemNames=Patient. Id.NationalIdNumber
Output:	itemValues=Patient.Id.NationalIdNumber|220345-XXXX

Table 4. Coupon used after leaving the context

Definition:	Coupon used after leaving the context.
Precondition:	Application has left the context, participantCoupon has been expired.
Input:	http://193.167.225.119/cm.pp?interface=ContextData&method=GetItemValues&participantCoupon=11900347&itemNames=Patient. Id.NationalIdNumber
Output:	e.g. exception=GeneralFailure&exceptionMessage=General failure

4.4 Test Report

If conformance testing is passed the test report, which includes the information about the implementation under test, implementation-specific settings and special considerations, and test cases with the pass/fail information, is published. The information of the implementation under test describes the product and its version, the interface specification and its version, the role of the application in the integration, names of the testers, and date. Application-specific settings and special considerations could be, for example, accepted application and item names (optional features in context servers), address of the service, number of joining applications, and installation instructions. Additionally, any special information, which must be taken into consideration in integration is informed. Test cases are reported similarly as in Section 4.3. Inputs and outputs are written to the log file but they are not in the public test report.

4.5 Present State of the Model

The conformance testing model has been developed and evaluated in workshops with co-operation of several healthcare software companies and hospital districts. The evaluation has been performed for a context management server implementation. At the moment, there are several commercial implementations of the context manager, of which one has received a brand for the context management implementation. Conformance testing revealed different interpretations of the context management specification. They were corrected to the product in question, and clarified in further versions of the specification. At the moment, we are studying the conformance testing processes more and our model is further elaborated.

5 Discussion

The conformance testing model developed in the PlugIT project is on one hand a light and rapid model and on the other hand it assures that products conform to the specifications. However, the model is not complete. Some challenges, which have to be solved before applying the model further, are described here.

Software product versions are introduced in rapid cycles. Rosenthal et al. [9] have presented that a brand could be issued for a longer period (e.g. 2 years) if no errors are found in conformance testing. In our case, new software versions can be released, for example, once every two week. Thus, the brand has to be renewed much more often and re-evaluation has to be performed automatically using, for example, web-based testing services.

Some requirements of the solution (e.g. context server) are implicit. For example, the context management interface specification does not clearly state, that setting context from one workstation must not affect contexts set from other workstations. However, it is a basic requirement for server-based applications, which contain workstation-specific data, such as context management. Thus, integration specifications must contain enough information about the requirements for the solutions, in addition to mere interface signatures.

Some parameters are static for a given environment, but several are specific to the applications used or to the implementation of the server. A standard way of identifying and classifying this sort of parameters for test case definitions is needed. Furthermore, some requirements for the parameters can not be easily tested. For example, the specification states, that the participantCoupon parameters returned by the service must be unique during the execution of the service. Complete testing for such uniqueness would require far too many different test cases in practice.

Extension points in specifications (such as an optional feature for context servers to accept only some named applications), which are not required features must be tested, if the specification has conformance levels for such extensions. Thus, the integration specifications and standards should be developed to express clearly, which options are implementation-specific or optional. In addition, specifications should provide guidance on how should implementation-specific features be documented and used.

In our context management case, the contents of the parameters in test cases are quite simple and have well-specified semantics. However, when testing complex data structures (e.g. patient records), value sets for the parameters, and interpretations of different operations and data elements must be precise.

Versioning of these value sets (e.g. different versions of disease classifications) further complicates the interoperability and conformance testing.

If software customers coherently require standard-based ("branded") interfaces in calls for tenders, the quality and interoperability of solutions is improved. This does not prevent free competition, but promotes standardization in widely-used interfaces and reduces local “fixes”. However, customers need advice when gathering and setting their requirements. Interface specifications do not currently contain all the needed information, including basic requirements, conformance levels, and different types of parameters, which must be conformed to. Questions of the customers have to be answered and quality of interoperability specifications improved in order to get software customers to demand certified and interoperable software products.

As we can see, interface standards are only one part of interoperability. There can be products that conform to the specifications but are not interoperable. There can also be products that do not exactly conform to the specification but can be interoperable with other products [4]. Thus, when developing interface specifications and conformance testing models all the things influencing integration and interoperability have to be taken into consideration and specified. One step in this direction is integration profiles by IHE initiative (Integrating the Healthcare Enterprise) [7]. IHE integration profiles offer a common language that healthcare professionals and software developers can use in communicating requirements for the integration of products. Integration profiles describe real-world scenarios or specific sets of capabilities of integrated systems.

So, what are the lessons we learned? At first, more accurate and diverse test cases are needed. The test data can not be real patient (or production) data, but correspond as much as possible to the real (patient) scenarios. However, to be able to create such scenarios we need to have good domain knowledge. In our case, the testing model has to be flexible enough to cover both testing the client and server parts of context management solutions, and also other types of service interfaces, such as common services. The conformance testing model has to also support testing of the workflow. Therefore, test cases have to constitute a flow of test cases, in which the order of the execution matters. All these different types of solutions require different test cases. At second, software companies were not very keen on public conformance testing. The regulations by the authorities or demand on the market for certified interfaces are the only effective ways to make certification and interface "branding" common. Based on our initial experience in this limited setting, the conformance testing had positive effects both to the implementation and to the specification. At third, although our model is simple and the test cases as well as the specification are inadequate for comprehensive interoperability, the model is a good starting point to develop more efficient model for conformance testing. The model will be further elaborated in our next projects, OpenTE and SerAPI.

6 Acknowledgement

This paper is based on research in the PlugIT project (2001-2004), funded by the National Technology Agency TEKES, software companies and hospital districts. The authors thank all the involved participants in this project.

References

1. Australian Healthcare Messaging Laboratory. Referred October 12 2004. URL: http://www.ahml.com.au/
2. Carnahan L, Rosenthal L, Skall M (1998) Conformance Testing and Certification Model for Software Specification. ISACC '98 Conference
3. CCOW information for the healthcare industry. Referred October 12 2004. URL: http://www.ccow-info.com/
4. Eichelberg M, Riesmeier J, Jensch P (2002) DeNIA: Computer Supported Interoperability Assessment for DICOM Devices. In: Niinimäki J, Ilkko E, Reponen J (eds) Proceedings of the 20th EuroPACS annual meeting, Oulu university press 55-58
5. Health level 7 (HL7). Referred November 5 2004. http://www.hl7.org/
6. Herzum P, Sims O (2000) Business Component Factory. Wiley Computer Publishing, New York
7. HIMSS, RSNA (2002) Integrating the Healthcare Enterprise - IHE Technical Framework Volume I - Integration Profiles, Revision 5.3. HIMSS/RSNA
8. ISO/IEC Guide 2: 1996 (1996) Standardization and Related Activities: General Vocabulary
9. Rosenthal L, Skall M, Carnahan L (2001) White paper. Conformance Testing and Certification Framework NIST
10. Service-oriented Architecture and Web Services in Healthcare Application Production and Integration. Home page of the SerAPI project. Referred October 25 2004. URL: http://www.uku.fi/tike/his/serapi/english.html
11. The Software Diagnostics and Conformance Testing Division, Information technology Laboratory, National Institute of Standards and Technology. Referred Oct 12 2004. URL: http://www.itl.nist.gov/div897/
12. TNT Global Systems (2002) White Paper. National Electronic Healthcare Claims Standard, Conformance and Compliance Issues

Interoperability Research in the European Union

Arian Zwegers

European Commission, Avenue de Beaulieu 29, 1060 Brussels,
arian.zwegers@cec.eu.int

Increasingly, enterprises are cooperating with other enterprises. Not only large organisations set up cooperation agreements with other enterprises, but also SMEs are combining forces to compete jointly in the market. Nowadays, an enterprise's competitiveness is largely determined by its ability to seamlessly interoperate with others.

However, legacy enterprise applications often hinder cooperation endeavours. These applications were in many cases not designed to interoperate with other applications. Some estimates claim that around 40% of system implementation budgets are spent on integration with other (legacy) systems within an enterprise. These integration issues are increased when interoperation across enterprises is considered.

The interoperability landscape of enterprise applications has a number of characteristics. Integrations are often point-to-point using proprietary APIs. For instance, although many legacy systems and packaged applications 'speak XML', their data models and schemas are often quite different. The definition of common concepts such as an "order" or a "customer" may vary greatly among applications. Another obstacle is the lack of standards in a number of areas, for instance for describing and orchestrating business process flows across multiple systems.

The European Commission recognized these interoperability problems and launched a research initiative in this area. The objective of this targeted research initiative is to enable networked business by giving European enterprises the means to seamlessly and securely interoperate with each other. It will equip European industry with novel middleware and knowledge sharing solutions, as well as concepts and methods that provide for seamless interoperation both within and across enterprises.

The seven research projects that are described next are examples of ongoing work. They are the cornerstones of the European Community funded research in the area of enterprise interoperability.

The mission of the **Knoweledge Web** *Network Of Excellence* is to strengthen the European industry and service providers in one of the most important areas of current computer technology: Semantic Web enabled E-work and E-commerce, supporting the transition process of Ontology technology from Academia to Industry.

The **CrossWork** *project* focuses on process interoperability and semantic interoperability, seeing interoperability as a systemic property of the set of collaborating entities, arising in connection with their collaboration.

To **NO-REST** *project* analyses how standards, and their implementations, are subject to change incurred by the environment within which they are developed and implemented, respectively.

The **SPIDER-WIN** *project* aims to achieve efficient, simple and context-aware SME co-operation in supply networks with low-level local software requirements, focussed on the exchange of order status changes. This is achieved by an ASP platform with asynchronous data exchange between the platform and the enterprises.

The primary goal of the **INTEROP** *Network of Excellence* is the sustainable structuring and shaping of European research activities on Interoperability for Enterprises Applications and Software and the emergence of a lasting European Research Community that will influence standards, affect policy and solve recurrent problems in networked enterprises.

The **ATHENA** *Integrated Project* aims to enable interoperability by providing reference architectures, methods and infrastructure components. ATHENA takes a holistic approach to solving the Interoperability problem taking a technical as well as a business viewpoint into account.

Finally he **TERREGOV** *project* which aim is to enable local governments to deliver online services, specially in the Social Care environment, in a straightforward and transparent manner regardless of the administrations actually involved in providing those services.

The Knowledge Web Network of Excellence

Enrico Franconi

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy,
http://www.inf.unibz.it/_franconi/

Summary. In a nutshell, the mission of Knowledge Web is to strengthen the European industry and service providers in one of the most important areas of current computer technology: Semantic Web enabled e-work and e-commerce. We will concentrate our efforts around the outreach of this technology to industry. Naturally, this includes education and research efforts to ensure the durability of impact and support of industry. Therefore, the main objectives of Knowledge Web are: Outreach to Industry, Outreach to Education and Coordination of Research.

1 Summary of Activities

Knowledge Web (KW) is a 4 year Network of Excellence project funded by the European Commission 6th Framework Programme. Knowledge Web began on January 1st, 2004. Supporting the transition process of Ontology technology from Academia to Industry is the main and major goal of Knowledge Web.

The mission of KnowledgeWeb is to strengthen the European industry and service providers in one of the most important areas of current computer technology: Semantic Web enabled E-work and E-commerce. The project concentrates its efforts around the outreach of this technology to industry. Naturally, this includes education and research efforts to ensure the durability of impact and support of industry. Therefore, Knowledge Web devotes its efforts to the following main areas:

- Outreach to Industry. The main objective of Knowledge Web's outreach to industry area is to promote greater awareness and faster take-up of Semantic Web technology within Europe in full synergy with the research activity. This outreach will help to reduce time needed to transfer the technology to industry and to market.
- Outreach to Education. Knowledge Web aims to work towards the establishment of a Virtual Institute for Semantic Web Education (VISWE), which will act as the principal focus for educational activities on Semantic Web.
- Coordination of Research. The objective of Knowledge Web will be to ensure that the research as performed by the leading groups in this area will be sufficiently coordinated to avoid both duplication and fragmentation. Such coordination is particularly important for the Semantic Web: since it is an inter-disciplinary area, joint collaborations among and across various research

> communities is necessary. The objective of Knowledge Web is to coordinate the European research effort to make Semantic Web and Semantic Web Services a reality.

The Knowledge Web consortium is coordinated by the University of Innsbruck, Austria and consists of 18 leading partners in Semantic Web, Multimedia, Human Language Technology, Workflow and Agents.

At the end of its first year the Knowledge Web network is in good shape. Partners are doing integrated research on the themes defined in the JPA: scalability, heterogeneity, dynamics, web services and language extensions. The network has started an exchange program for researchers to work for a prolonged period at another institute. At this early stage there are already visible signs of impact, such as the W3C note on the semantic-web rule language SWRL. One look at the proceedings of the 2004 International Semantic Web Conference in Hiroshima (acceptance rate: 22%) makes clear that Knowledge Web partners are (the) key players in this field. Another sign is the fact that a Knowledge Web participant won the 2004 Semantic Web Challenge (a contest for semantic web applications, see http://challenge.semanticweb.org/), beating for example a large NASA project. In addition, the network is starting outreach activities, including setting up an growing industrial board. The outreach activity is focusing on realistic use cases and showcase applications. Network partners also play a key role in the new W3C Semantic Web Best Practices and Deployment Working Group, which is in the process of producing low-entry advisory notes for application developers. Finally, the network has been active in creating an educational infrastructure, such as a pool of learning modules and the organization of a highly-praised summer school in Spain. Overall, the network is in pole position to make the Semantic Web work.

2 Future Work

The network is currently performing a self-assessment which will lead to a new JPA. At this stage it is safe to say that we do not foresee major changes in the network. There will however be some natural focus changes. Whereas the research work packages have mainly worked internally in the first year, we expect to see more cross fertilization between work packages in the second year. Also, the exchange program is expected to grow significantly, possibly also including visits from employees of industry board members to network partners as well as exchanges with other IST networks. With respect to outreach, the network is expected to focus even more on showcase applications to demonstrate clearly the added value of semantic-web technology to foster uptake by newcomers. We expect more organizations to join the industry board, including nonprofit organizations. There will also be a steady flow of Knowledge Web output to the various standards bodies. The summer school will be continued but the network will consider organizing also educational activities for members of the industry board. In addition, educational modules will become available, supported by semantic-web technology. We are looking forward to the second year of the network.

Interoperability Contributions of CrossWork

Nikolay Mehandjiev[1], Iain Duncan Stalker[1], Kurt Fessl[2] and Georg Weichhart[2]

[1] School of Informatics, the University of Manchester, UK,
mehandjiev@manchester.ac.uk
[2] Profactor Produktionsforschung GmbH, Austria

Virtual organisations allow companies of all sizes to pursue immediate marketplace opportunities. This places substantial demands on supporting software to ensure seamless interoperability at the levels of communication, shared semantics and proc-esses. Interoperability issues are addressed by several EC-funded projects under the 6th Framework Programme. The focus here is on the specific contributions of one of them, CrossWork , which pursues the automated creation of cross-organisational workflows to support the formation of virtual organisations among members of Net-works of Automotive Excellence. The aim is to support effective process collaboration within the virtual organization allowing rapid response to business opportunities.

Achieving such levels of process interoperability relies on frictionless information exchange among the IT infrastructures of group members, or interoperability at the systems level, within the context of compatible legal and organisational structures, predicating interoperability at the business level. Any successful collaboration is informed and underpinned by a shared understanding and this reveals a fundamental need for semantic interoperability at the information level and beyond.

CrossWork focuses on process interoperability and semantic interoperability. We see interoperability as a systemic property of the set of collaborating entities, arising in connection with their collaboration. This has the following consequences for the mechanisms chosen to underpin CrossWork's approach to interoperability:

Semantic interoperability is based on ontologies and meaning negotiation mecha-nisms to find common ground and identify shared ontological commitments. This leads to a common core ontology and a number of peripheral ontologies specific to a sub-group or individual organisation, each extending the core. Novel negotiation protocols, informed by the theory of utility and supported by rigorous ontology mappings, enable the evolution of the core and the peripheral ontologies: for example, to incorporate new capabilities and services within the task group. Automatic reason-ing within this context is underpinned by the use of Formal Concept Analysis [2] and Lattice Theory [1]. The resultant model [4], which we call devolved ontologies, integrates centralised and distributed approaches to ontology engineering.

Process interoperability in CrossWork relies on the use of software agents to achieve the goal-driven formation of both team and its workflow, because of agents' deliberative reasoning and goal-driven behaviour. Agents represent collaborating organisations, providing software representations for business autonomy, information hiding, self-interested behaviour and coalition formation. Agents are also a natural choice in terms of our approach to semantic interoperability, as they employ sophisticated communication mechanisms based on explicit declaration of intent and ontological commitment, and provide the negotiation and reasoning capabilities necessary to maintain a devolved ontology model.

CrossWork is based on the outcome of the EC-funded project MaBE (Multi-agent Business Environment), a distributed, service oriented middleware, including basic facilities for evolving semantic interoperability in open business environments. The MaBE consortium has decided to distribute the MaBE kernel via a public board under an open source license model (LGPL). The MaBE middleware is based on a widely established open source agent platform named JADE (Java Agent Development Environment), which is also organised via a public board using an LGPL licence model. Both initiatives work closely with the agent standardisation organisation FIPA.

Crosswork extends MaBE to enable interoperability at the process level, whilst serving as an application of MaBE in the domain of automotive manufacturing. The team and workflow design [5] is based on Gero's FBS framework [3] and combines distributed planning and pattern-based workflow composition. This takes into account local rules and processes of collaborating partners whilst preserving their business autonomy and allowing information hiding regarding their business processes. Standard process modeling framework XRL, based on Petri Nets, provides the formal basis needed to enable reasoning and predictive analysis regarding model consistency.

The dynamics of the resulting system involving humans and computers require novel approaches to dynamic user interface generation, providing necessary and suffi-cient information to the individual users at different locations. Role-based enactment of business logic and security is used to provide the necessary level of abstraction and ease of maintenance. These core contributions provide the necessary complement of techniques for seamless collaboration and process integration within the target context of distributed and dynamic coordination of work.

References

1. Birkhoff, G.: 'Lattice Theory (3rd Ed.)'. AMS Colloquium Publ., Providence, RI (1967)
2. Ganter& Wille, 'Formal Concept Analysis. Mathematical Foundations.' Springer (1999)
3. Gero, J. S. : 'Design protoypes: A knowledge representation schema for design', AI-Magazine, 26–36 (1990)
4. Stalker, I D.: 'Dynamic Ontologies for Evolving Domains and Multiple Contexts', The Potential of Cognitive Semantics for Ontologies, Workshop of (FOIS-04), Turin (2004)
5. Stalker, I D, Mehandjiev, N., Weichhart, G, Fessl, K.: 'Agents for Decentralised Process Design – Extended Abstract', OTM Workshops,: OTM 2004, Agia Napa, Cyprus (2004)

Networked Organisations – Research into Standards and Standardisation

Knut Blind[1] and Kai Jakobs[2]

[1] The Fraunhofer Institute for Systems and Innovation Research, FhG-ISI
Knut.Blind@isi.fhg.de
[2] Aachen University, CoSc Dept, Informatik IV, Kai.Jakobs@cs.rwth-aachen.de

1 Background

The project focuses on the evolution of standards, their implementation in a dynamic environment, and on the mutual influences between them. To this end, NO-REST analyses how standards, and their implementations, are subject to change incurred by the environment within which they are developed and implemented, respectively. Moreover, the origin of standards (i.e., SDOs, consortia, etc.) will be analysed with respect to the impact it may have on a standard's market success. The implementation of a standard needs to be adaptable to changes in such an environment, and will thus change itself over time. NO-REST will analyse these dynamics, and will devise an analytical framework for a causal model of such changes. This, in turn, will allow for the derivation of conclusions for developing standards in the future and possible mechanisms to feed back these changes into dynamic standards building.

Once the dynamic elements in the life cycle of a standard have been understood, the project will develop a methodology for an integrated impact assessment of a standard.

2 Project Tasks – A Brief Outline

2.1 Demand for and Supply of Standards

The internal processes and rules of standards setting bodies (SSBs, including SDOs, consortia, industry fora) have been analysed, as well as various other characteristics which may have an impact on the uptake and performance of a standard in the market place. Since we observe an increasing competition between the different suppliers of standards products and related services, new organisations evolved and traditional or-ganisational models are substituted by new ones. For example, by now even national SDOs have introduced remote access to standardisation processes, as well as new prod-ucts. Also, aspects such as the 'credibility' of a standards-setting organisation in a specific technical domain will be analysed.

To complement the above, the project also analyses the impact of the environment within which networked organisations operate. Derived from their objectives, and the relevant framework conditions, we derive their demand for standards. Work will also include a study if, and how, the business model of the implementing organisation has an impact on an implemented standard.

2.2 The Dynamics of Standards

The various factors that impact the emergence and the implementation of a standard are being identified and analysed. These activities also integrate the work described above and the outcome of a critical review of the existing relevant literature. Moreover, the gradual modification of (the implementation of) a standard through its adaptation to a specific environment has been examined, as well as the resulting feedback to the standardisation organisations and possible further generations of the former standard (if any). Taken together, these research results will yield a good understanding of the dynamics of standards, i.e. how their implementations change over time due to external influences. The new model about the dynamics of standards reminds of the discharge of the linear model of innovation by non-linear models taking into account various feedback loops. Appropriate measures for preserving the compatibility and interoperability of the various different implementations will be devised.

2.3 Impact Assessment

The above three working steps provide the basis for the development of a dynamic model of standardisation and standards. Although in the dynamic evolution of standards we have already considered the impacts on the standardisation process itself, the objective here is to assess the impact of (the implementations of) standards and their dynamics on both private and public networking organisations at the micro-level, and their comprehensive impact on the systems or macro-level. In a first step, the relevant impact dimensions have been identified. In a second step, we are designing assessment tools for both ex-post and ex-ante impact assessments. Finally, we will select appropriate examples to perform an impact assessment in practice. The feasibility, methods, and results of this impact assessments will be evaluated. Based on these experiences, final guidelines for tools for an impact assessment will be proposed and distributed among the relevant .stakeholders in standardisation processes.

Methods for the Analysis of Supply Network Processes at European SMEs

Markus Rabe

Fraunhofer IPK, Pascalstr. 8-9, 10587 Berlin, Germany,
markus.rabe@ipk.fraunhofer.de

Summary. Business Process Modelling (BPM) is a well-understood method to analyse enterprise processes. As today more and more essential processes are conducted across the enterprise borders, this induces additional challenges in terms of different languages, process types and ontology. This paper indicates techniques which support such cross-enterprise BPM. The work reported has been conducted in three company networks including SMEs within different European regions, identifying potentials and constraints for a more sophisticated supply chain control.

1 Introduction

The SPIDER-WIN project (with financial contribution of the European Commission, IST Project 507 601) [1] aims to achieve efficient, simple and context-aware SME co-operation in supply networks with low-level local software requirements, focussed on the exchange of order status changes. This is achieved by an ASP platform with asynchronous data exchange between the platform and the enterprises.

In order to prepare the definition of the required functionalities and interfaces, three supply networks from different European regions have been analysed. In the preparation phase, reference models and documents have been developed as the base for the local studies. In the analysis phase, similarities between the different workflows have been identified.

2 Modelling Techniques Applied

According to the process orientation of the project's subject, the Integrated Enterprise Modelling (IEM) Method [2] has been applied. Through the object oriented approach of the IEM the use of reference classes is very efficient, simplifying the task of defining common terms, structures and attributes.

Reference models are a very efficient means to increase the efficiency of modelling, to raise the quality level of the developed models, and to improve the reusability of the models. Such reference should include class structures, template models and a manual which describes the correct and efficient use of the reference models as well as the validity and constraints of the model. Especially for distributed systems, where different persons perform the modelling task at different locations, reference models can improve the work, significantly [3]. The Supply Chain Operations Reference Model (SCOR) is a reference model, too, as it incorporates additional elements like standard descriptions of processes, standard metrics and best-in-class practices.

Before starting interviews, a *reference model* based on IEM methodology and SCOR terminology was developed, in order to guarantee that all information and requirements detected can be systematically documented within one single, consistent model. A *Guideline Document Suite* includes a description of the processes, variables and metrics to be considered as well as supporting documents and document templates. The guidelines supported the structure and completeness of the interview as well as the comparison of the results from the different supply networks.

3 Results

First, the single company models have been established. They have then been merged to models of the three different supply networks, thereby identifying additional potentials and challenges at the company interfaces. In total, 103 sub-models ("levels") have been established, with a total of 1852 process elements.

Based on the study results, a "general model" of the as-is-situation could be extracted from the three network models, which describes general and specific process elements, systematically documented within one single, consistent model. It contains the SCOR compliant process names, specific "information categories", relations between processes and information categories and further application rules. Therefore, by comparison of a specific supply chain model with the general model, the specifics of the supply chain can be identified.

The study has demonstrated, that a well-adapted reference model is an important base for the conduction of cross-enterprise business process studies. The IEM Method turned out to be a very efficient means for this purpose, allowing to switch the terms between two languages (the native interview language and English). The reference class trees significantly improved the development of models with comparable structures, without urging the interviewers into pre-defined processes. SCOR was a good base to establish common understanding between the coaches.

References

1. IST 507 601 SPIDER-WIN project (2005) www.spider-win.de
2. Mertins, K.; Jochem, R. (1999) Quality-oriented design of business processes. Kluwer, Boston
3. Rabe, M.; Jaekel, F.-W. (2003) The MISSION project – demonstration of distributed supply chain simulation. Enterprise inter- and intra-organizational integration. Kluwer, Boston Dordrecht London 235-242

The INTEROP Network of Excellence

Jean-Paul Bourrières

University of Bordeaux I, Laboratoire d'Automatique et de Productique, Signal et Images, 351 cours de la Liberation, F-33405, Talence cedex, France,
bourrieres@lap.u-bordeaux1.fr

1 Project Main Goals

Research in the interoperability domain in Europe remains insufficiently structured, is fragmented, sometimes overlapping. There is no global vision of research consistency, no co-ordination between European research centres, university laboratories or other bodies. This situation is not only true for research, but also in the training and education areas. Consequently, the primary goal of INTEROP is the sustainable structuring and shaping of European research activities on Interoperability for Enterprises Applications and Software and the emergence of a lasting European Research Community that will influence standards, affect policy and solve recurrent problems in networked enterprises.

The INTEROP Network of Excellence consortium is composed of 50 members coming from 15 countries (13 EU Member States, Norway and Switzerland). The network co-ordinates around 180 researchers and 100 Doctoral Students.

2 Key Issue

The Interoperabilty is seen as the capability of a system or a product to work with other systems or products without specific effort from the user. For INTEROP, it means the capacity of an enterprise software or application to interact with others.

Today enterprises and organizations must be able to adapt to market changes through efficient outsourcing and collaboration strategies. Collaborative business requires reliable exchange of commercial, financial and technical data as well. Legacy ERP, SCM, LCM and CRM enterprise applications commonly manage the information required for collaboration, but the software itself was for the most part conceived and programmed to be run within specific organizational boundaries.

Even if many applications use unified technologies, business and data models remain heterogeneous. Despite standardisation efforts, describing and orchestrating business processes across multiple systems is at best a semi-manual process.

Meanwhile, IT budgets are shrinking and new intrinsically interoperable systems are unfeasible for most IT directors.

In Europe, the cost of non-interoperability is estimated to 40% of enterprises IT budget. Reducing this factor of cost is one of the keys of European industry's competitiveness.

3 Technical Approach

The originality of the project is to take a multidisciplinary approach by merging three research areas supporting the development of Interoperability of Enterprise Applications and Software:

- Architecture & Platforms: to provide implementation frameworks (A&P),
- Enterprise Modelling: to define Interoperability requirements and to support solution implementation (EM),
- Ontology: to identify Interoperability semantics in the enterprise (ONT).

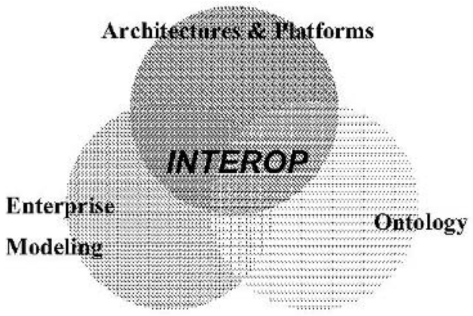


Figure 1. Knowledge integration for Interoperability

The INTEROP work programme comprises the following activities :
Integrating Activities

- INTEROP Knowledge map
- INTEROP method of work and collaboration platform
- Mobility of researchers
- Method for Scientific Integration and Assessment

Joint research activities

- Common Enterprise Modelling Framework in distributed environments
- Generation of customised Enterprise Software from Enterprise)
- Ontology-based Integration of Enterprise Modelling and Architecture & Platforms
- New architectures and platforms for Interoperability

Spreading of Excellence activities

- Training by e-learning
- Dissemination and Communication
- Transfer of research to Industry

4 Expected Achievements / Impact

The main expected achievement of INTEROP is the emergence of a durable European research community on interoperability of enterprise applications and software by setting up a virtual laboratory on Enterprise Interoperability with maximum research and industrial audience.

The provision of e.learning services and the set up of an European Master on Interoperability will impact the awareness of Interoperability requirements and spread excellence in the field.

To ensure a significant industrial impact, in particular through future standardisation, INTEROP specifically and explicitly interacts with FP6 IST ATHENA Integrated Project “Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications” (www.athena-ip.org).

From a scientific point of view, the value-added by INTEROP is the achievement of the knowledge integration process which will turn three initial disciplinary components (Architectures and Enabling technologies, Enterprise Modelling and Enterprise Ontologies) into a new multi-disciplinary

ATHENA - Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications

Rainer Ruggaber

SAP Research, SAP AG, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany, rainer.ruggaber@sap.com

Summary: Organisations are engaging in more and more sophisticated business networks to improve collaboration. These business networks can range from more static relationships like Supply Chains to very dynamic networks like virtual organisations. A prerequisite to enable business networks is the interoperability of the participants systems and applications. ATHENA [1] is an Integrated Project funded by the European Commission under Framework Programme 6 that addresses Interoperability of Enterprise Systems and Applications proposing a holistic approach. ATHENA will provide technical results like reference architectures, methodologies and infrastructures complemented by business results that provide ROI calculations and impact predictions for new technologies.

1 Introduction

One of the trends in the global market is the increasing collaboration among enterprises during the entire product life cycle. This trend requires, that enterprise systems and applications need to be interoperable in order to achieve seamless business interaction across organisational boundaries, and realise networked organisations.

ATHENA takes a holistic approach to solving the Interoperability problem taking a technical as well as a business viewpoint into account. Previous activities in that space led to fragmented solutions addressing only part of the problem. From a standards viewpoint in the B2B space there is rather a proliferation than a lack of standards.

2 ATHENA Integrated Project

The ATHENA IP (Integrated Project) [1] aims to enable interoperability by providing reference architectures, methods and infrastructure components. In ATHENA Research & Development will be executed in synergy and collaboration with Community Building: research will be guided by business requirements defined by a broad range of industrial sectors and integrated into Piloting and Technology Testing as well as Training.

Scenarios play an important role in ATHENA as source of requirements and for the validation of results. Scenarios of Supply Chain Management, Collaborative Product Development, e-Procurement and Portfolio Management are investigated.

In ATHENA six research topics/projects were defined for the first stage of the IP. The definition of these R&D projects was based on the roadmaps elaborated by the IDEAS roadmapping project [2]:

- *Enterprise Modelling in the Context of Collaborative Enterprises* aims at developing methodologies for management and modelling of situated processes, flexible resource allocation and assignment. Furthermore, it investigates methodologies for work management and execution monitoring.
- *Cross-Organisational Business Processes* deals with modelling techniques to represent business processes from different organisations on a level that considers the privacy requirements of the involved partners. Such models need to be executed through IT systems and need to operate efficiently in an architectural environment that adapts to particular business scenarios.
- *Knowledge Support and Semantic Mediation Solutions* aim at the development of methods and tools for the semantic enabled enterprise. A key objective is to build an integrated software environment that is able to manage the semantics of different abstraction levels that can be found in an enterprise.
- *Interoperability Framework and Services for Networked Enterprises* is concerned with the definition of reference architectures and infrastructures supporting interoperability of enterprise systems and applications.
- *Planned and Customisable Service-Oriented Architectures* is to develop the understanding, tools and infrastructures required for service-oriented architectures which can be achieved more easily through the planning and later customisation of solution.
- *Model-driven and Adaptive Interoperability Architectures* s to provide new and innovative solutions for the problem of sustaining interoperability through change and evolution, by providing dynamic and adaptive interoperability architecture approaches.

References

1. ATHENA - Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications, FP6-2002-IST-1, Integrated Project – Annex I, 2004, Homepage: http://www.athena-ip.org.
2. IDEAS Project Deliverables (WP1-WP7), Public Reports, http://www.ideas-road map.net, 2003.

This paper is partly funded by the European Commission through the ATHENA IP. It does not represent the view of the European Commission, and authors are solely responsible for the paper's content.

Terregov: eGoverment Interoperability on a Semantically Driven World

Santos Vicente, María Pérez, Xavier García, Ana Gimeno and Javier Naval

GFI Informática. Serrano Galvache, 56 28033 Madrid, terregov@gfi-info.com

1 Introduction

Public services within the European countries are demanding increasingly more interconnections among them nowadays. Integration and distributed responsibilities of the agencies involved in the government administrations are required very intensively. In this context of different administrations procedures integration and distributed responsibilities, the capacity of interaction between different agencies regardless of in home nomenclature or language would be a must. Semantic Interoperability Technologies will allow this kind of multicultural environments by the fact of standardising the concepts under a business area and after that, providing definitions based on these concepts for the different procedures provided for all the entities on the system. Once all the procedures on a system are defined according to these standardised concepts, automatic search based on concepts would be a take off point for enabling non supervised information systems interaction and automatic service discovery.

2 Terregov

TERREGOV (IST-2002-507749) is an European Union funded project which aim is to enable local governments to deliver online services, specially in the Social Care environment, in a straightforward and transparent manner regardless of the administrations actually involved in providing those services.

Semantic enrichment of web services is the philosophy developed by TERREGOV in which web services and plain text documents are described using an ontology. Such an ontology provides a language-independent mechanism to support the automatic discovery of an answer to a citizen's request regardless of administrations or organization. A constraint is that all information exchanged has to be achieved through the translation of data into the unique ontology for TERREGOV and always based on semantic knowledge of information in accordance with rules for interconnectivity and dynamic discovery.

Access to web services could be transparent but with restrictions because of the different legislative constraints and authorization limits in the different administrations involved.

TERREGOV´s requirements include the use of standards for implementing web services; use of ontologies for structuring knowledge, allowing to describe web services unambiguously, implementing human-machine interfaces, indexing and retrieving information; use of natural language processing for automating partially the ontology developments and improving the human-machine dialogs.

Overall, the goal of the project is to deliver eGoverment services in a transparent manner.

The TERREGOV solution provides ontologies used for organizing the relevant knowledge, and indirectly driving the business processes. They will be useful in different tasks, namely: (i) structuring knowledge for knowledge management, leading to the enrichment of web services meta-data with semantic descriptors; (ii) driving Human-Machine dialogs; (iii) semantic indexing/retrieval of text and documents.

The implementation of the project takes place in different European pilots in which administrations in the context of social care are involved.

The most expected use case of TERREGOV is that a civil servant, point of contact between citizen and public administration, is asked about a problem. In order to solve it and to report the solution to the citizen, the civil servant, through the use of TERREGOV solution, can collaborate with other civil servants and experts to get hints; get access to the specific knowledge base, and search for information; discover the best service for the specific citizen's case and invoke the execution of the selected service and monitor its execution.

The citizen can call on a range of services helped by an administrative agent or civil servant to make requests to public services. With search tools based on a semantic approach and natural language, the civil servant or the citizen will be able to call e-procedures at a local, county, regional or national level.

Each administration is responsible for defining the web services it wants to offer. The functionalities include the dynamic discovery of the Web Service using semantic description and searching. It also includes an interface that allows electronic registration and entry of citizen's data. This interface will include complete tools to extended with semantic capabilities like an electronic agenda for notes and appointments, an email connection service with other civil servants for information exchange, automatic printing of documents or automatic generation of letters for citizens and connection with database systems.

A registry of citizens' personal data and their completed requests are stored in databases. Then the TERREGOV web service will automatically offer a specialized referent based on a citizen's own criteria and will update the citizen's record. The file is stored in the database and is sent electronically to a buffer zone. Storing the queries made to the system could help TERREGOV in learning about the efficiency of requests, optimum indexing of web services and strengths/weaknesses of connection among administrations. The answer is generated through workflows to get dynamic discovery of services.

TERREGOV adopts the principles of a service oriented architecture (SOA[19]) based on interoperable components with dynamic support for finding services. The idea is strengthened by the fact that information, services and administrations are spread over several information systems. The architecture contains a set of collaborative tools for eGovernment web services semantically enriched.

The solution is built on a multi-layer approach where citizens from any location and using different languages can access the services. This view is in line with the Clearing House approach, where a global server providing different applications over a common back-office is used to solve interoperability and semantic problems.

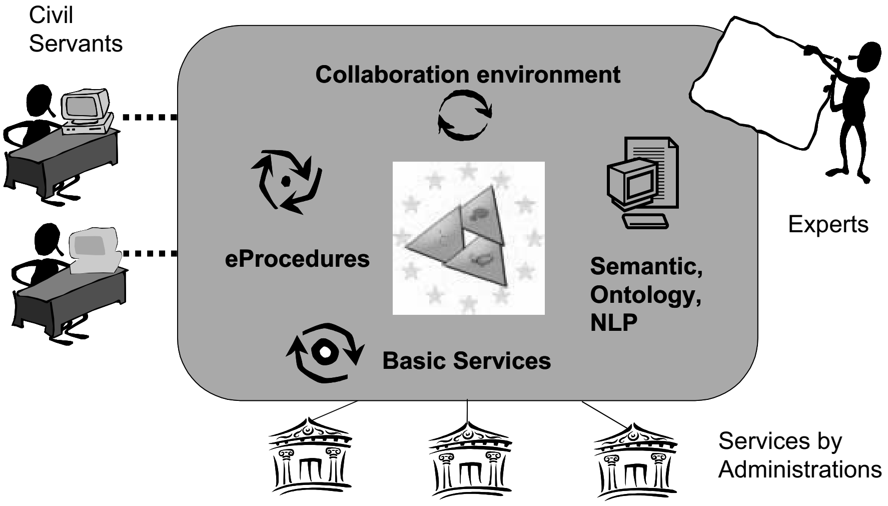


Figure 1. Terregov project: environment diagram

References

1. Terregov, D1.1 Technological State of Art v1, http://terregov.eupm.net/, April 2005.
2. Terregov, D4.2&4.3 Definition of TERREGOV Prototype, http://terregov.eupm.net/ , Sep 2005.
3. Europa , portal site of the European Union http://www.europa.eu.int/>, Jan 2005.

Author Index

Printing: Krips bv, Meppel
Binding: Stürtz, Würzburg