

ACT4SOC 2010

Marten van Sinderen and
Brahmananda Sapkota (Eds.)

Architectures, Concepts and Technologies for Service Oriented Computing

Proceedings of ACT4SOC 2010
4th International Workshop on
Architectures, Concepts and Technologies for Service Oriented Computing
In conjunction with ICSOFT 2010
Athens, Greece, July 2010

Marten Van Sinderen and
Brahmananda Sapkota (Eds.)

Architectures, Concepts and Technologies for Service Oriented Computing

**Proceedings of the
4th International Workshop on
Architectures, Concepts and Technologies for Service Oriented
Computing
ACT4SOC 2010**

In conjunction with ICSOFT 2010
Athens, Greece, July 2010

SciTePress
Portugal

Volume Editors

Marten Van Sinderen
University of Twente / CTIT / IICREST
The Netherlands

and

Brahmananda Sapkota
University of Twente
The Netherlands

4th International Workshop on
Architectures, Concepts and Technologies
for Service Oriented Computing
Athens, Greece, July 2010

Copyright © 2010
SciTePress
All rights reserved

Printed in Portugal

ISBN: 978-989-8425-20-1
Depósito Legal: 312841/10

Foreword

This volume contains the proceedings of the Fourth International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC 2010), held on July 23 in Athens, Greece, in conjunction with the Fourth International Conference on Software and Data Technologies (ICSOFT 2010).

Service-Oriented Computing (SOC) has emerged as a new computing paradigm for designing, building and using software applications to support business processes in heterogeneous, distributed and continuously changing environments. The architectural foundation for SOC is provided by the Service-Oriented Architecture (SOA), which states that applications expose their functionality as services in a uniform and technology-independent way such that they can be discovered and invoked over the network. Claimed benefits of SOC include cheaper and faster development of business applications through repeated aggregation of services, better reuse of software artifacts and legacy applications through service wrappings, and easier adaptation to changes in the business environment through replacement and reconfiguration of services.

In order to realize these benefits routinely with SOC, for realistic business settings with complex IT environments, many challenges still need to be addressed. For example, supporting business processes and collaborations in an open service-oriented world requires a better understanding of integration problems along different dimensions. First of all, alignment between business demands and application functions has to be achieved. This requirement for vertical integration should drive the aggregation of services, from basic IT services to rich business services, to achieve the desired or given business processes. Secondly, horizontal integration has to be considered if business collaborations span multiple organizations. In such cases, interoperability between the services has to be ensured at different levels (syntactic, semantic and pragmatic) and on different aspects (information and behavior). Thirdly, we have to assume that business demands as well as IT capabilities will change over time. This evolution will impact existing solutions, and thus require the adaptation, management and maintenance (e.g., versioning, replacing, updating) of services and service compositions. Moreover, changes that occur at one level or on one aspect have to be propagated to other levels and aspects in order to keep the consistency of the

integration solution. And finally, all of the above challenges not only exist at design-time, but at run-time as well. Service composition may be on-demand, driven by an end-user service creation activity, and running instances of composite services are subject to changes concerning, for instance, the availability of resources. This implies that service level agreements and associated quality-of-service need to be negotiated, monitored, and controlled in multi-party and heterogeneous environments.

The goal of the workshop is to focus on the fundamental and practical challenges related to SOC, to discuss what theoretical, architectural or technology foundation is needed, and how this foundation can be supported or realized by new or enhanced infrastructures, standards and/or technologies. The workshop aims at contributing to the dissemination of research results, establishment of a better understanding, and identification of new challenges related to SOC/SOA, by bringing together interested academic and industrial researchers.

This year 10 papers out of 30 submissions were selected, based on a thorough review process, in which each paper was reviewed by at least 3 experts in the field. Due to the number of reviews and the professionalism of the authors, program committee members and reviewers, we are confident that all selected papers are of high quality. The selected papers are also a good illustration of the different challenges mentioned above. They have been grouped in four presentation sessions during the workshop, named "SOA applications – logistics and online advertising", "Design and analysis of service-oriented systems", "Web services composition and "SOA applications – homecare and emergency support".

We like to take this opportunity to express my gratitude to all people who contributed to ACT4SOC 2010. Our thanks go to the authors, who provided the main content for this workshop, and to the program committee members and additional reviewers, who provided constructive comments that contributed to the quality of the content. We also like to thank the ICSOFT secretariat, especially Mónica Saramago and Vitor Pedrosa, for the excellent organizational support. Finally, we appreciate the opportunity given by the ICSOFT co-chairs, José Cordeiro and Maria Virvou, to allow the organization of this workshop in conjunction with ICSOFT 2010.

We wish all presenters and attendees an interesting and productive

workshop, and a pleasant stay in the historical city of Athens.

July 2010,

Marten Van Sinderen

University of Twente / CTIT / IICREST, The Netherlands

Brahmananda Sapkota

University of Twente, The Netherlands

Workshop Chairs

Marten van Sinderen
University of Twente / CTIT / IICREST
The Netherlands

and

Brahmananda Sapkota
University of Twente
The Netherlands

Program Committee

Marco Aiello, University of Groningen, The Netherlands
Markus Aleksy, University of Mannheim, Germany
Colin Atkinson, University of Mannheim, Germany
Sami Bhiri, Digital Enterprise Research Institute, Ireland
Barrett Bryant, University of Alabama at Birmingham, USA
Christoph Bussler, Saba Software, Inc., USA
Kuo-Ming Chao, Coventry University, UK
Remco Dijkman, University of Eindhoven, The Netherlands
Cléver Ricardo Guareis de Farias, University of São Paulo, Brazil
Walid Gaaloul, Télécom SudParis, France
Armin Haller, CSIRO ICT Centre, Canberra, Australia
Manfred Hauswirth, Digital Enterprise Research Institute, Ireland
Juan Miguel Gomez, Universidad Carlos III de Madrid, Spain
Ivan Ivanov, SUNY Empire State College, USA
Dimitris Karagiannis, University of Vienna, Austria
Haklae Kim, Samsung, Korea
Adrian Mocan, SAP, Germany
Michael Parkin, University of Tilburg, The Netherlands
Dick Quartel, Novay, The Netherlands
Dumitru Roman, SINTEF, Norway
Tony Shan, Keane Inc., USA
Boris Shishkov, University of Twente, The Netherlands
Ioan Toma, STI Innsbruck, Austria
Ken Turner, University of Stirling, Scotland
Tomas Vitvar, University of Innsbruck, Austria
Michal Zaremba, Seekda, Austria

Additional Reviewers

Pavel Bulanov, University of Groningen, The Netherlands

Wassim Derguech, Digital Enterprise Research Institute, Ireland

Alejandro Rodríguez González Universidad Carlos III de Madrid,
Spain,

Enrique Jiménez-Domingo Universidad Carlos III de Madrid, Spain,

Eirini Kaldeli, University of Groningen, The Netherlands

Alexander Lazovik, University of Groningen, The Netherlands

Table of Contents

Foreword	iii
Workshop Chairs	vii
Program Committee	vii
Additional Reviewers	viii

SOA Applications - Logistics and Online Advertising

Enterprise Interoperability Ontology for SOC applied to Logistics	3
<i>Wout Hofman</i>	
A Diffusion Mechanism for Online Advertising Service Over Social Media	16
<i>Yung-Ming Li and Ya-Lin Shiu</i>	

Design and Analysis of Service-oriented Systems

Specifying Formal executable Behavioral Models for Structural Models of Service-oriented Components	29
<i>Elvinia Riccobene and Patrizia Scandurra</i>	
Optimizing Service Selection for Probabilistic QoS Attributes	42
<i>Ulrich Lampe, Dieter Schuller, Julian Eckert and Ralf Steinmetz</i>	
From i* Models to Service Oriented Architecture Models	52
<i>Carlos Becerra, Xavier Franch and Hernán Astudillo</i>	

Web Services Composition

An Evaluation of Dynamic Web Service Composition Approaches	67
<i>Ravi Khadka and Brahmananda Sapkota</i>	

Model Checking Verification of Web Services Composition ..	80
<i>Abdallah Missaoui, Zohra Sbaï and Kamel Barkaoui</i>	

Semi-automatic Dependency Model Creation based on Process Descriptions and SLAs	93
<i>Matthias Winkler, Thomas Springer, Edmundo David Trigos and Alexander Schill</i>	

SOA Applications - Homecare and Emergency Support

Service Tailoring: Towards Personalized Homecare Services	109
<i>Mohammad Zarifi Eslami, Alireza Zarghami, Brahmananda Sapkota and Marten van Sinderen</i>	

Enabling Publish / Subscribe with Cots Web Services across Heterogeneous Networks	122
<i>Espen Skjervold, Trude Hafsføe, Frank T. Johnsen and Ketil Lund</i>	

Author Index	137
--------------------	-----

**SOA APPLICATIONS - LOGISTICS
AND ONLINE ADVERTISING**

Enterprise Interoperability Ontology for SOC applied to Logistics

Wout Hofman

TNO, P.O. Box 5050, 2600 GB, Delft, The Netherlands
Wout.hofman@tno.nl

Abstract. The Service Oriented Architecture (SOA, [1]) can be applied to enterprise integration. It creates an Internet of services for enterprises. Service science [2] defines services in term of value propositions of enterprises to customers. Both service science and the Internet of services require a form of mediation between customer requirements and service capabilities like for instance identified in [3]. However, mediation is not yet automated, thus can not be applied real time. Furthermore, many business areas already have agreement on semantics and interaction sequencing based on existing business documents, thus, mediation is not always required for a certain business area. This paper presents an ontology to support business services (Ontology Web Language (OWL), [18]). The concepts shared at business level are based on existing approaches like Resource, Event, Agent used for auditing and control [4] and builds upon service frameworks like OWL-S [13]. The ontology will be specialized to logistics and compared with other approaches that might be applied to enterprise integration.

1 Introduction

Over the past decades, services have become the most important part of economies [5]. Basically, the service economy refers to the service sector. It leads to more sophisticated forms of cooperation, or what is called value co-creation [2]. As Spohrer also points out, service systems are dynamic configurations of resources that interact via value-proposition-based interactions with governance mechanisms.

Each value proposition can be supported by a service. One could distinguish business services that directly relate to a value proposition, and IT services, mostly known as web services [6]. From an IT perspective, IT services can be defined by each individual service provider with its own semantics specified as ontology, thus requiring mediation to match user requirements and leading to mash ups for enterprise integration [3]. Mediation not only requires the matching of customer requirements to provider's capabilities, but also semantic mappings, and mapping of interaction sequencing (the latter is also called 'choreography', [7]). An alternative is to develop a shared ontology at business level for a particular class of business services common to more than one actor, e.g. logistics, customs, etc. The basic research question of this paper is on the feasibility of enterprise interoperability ontology common to all business services or phrased otherwise: do all enterprises use identical concepts in their business. To answer this question, we apply concepts from accountancy [4] to

interoperability, since these concepts already focus on value propositions and value exchange. This ontology will then be the basis for defining application area specific ontology. Firstly, we define the interoperability ontology, and secondly specialize it to logistics. Finally, we discuss the relation between our interoperability ontology and other approaches.

2 Concepts for Enterprise Interoperability

This section defines enterprise interoperability ontology. The concepts of this ontology are based on value propositions, their supporting electronic business documents and accountancy concepts. The concept ‘actor’ is used in this section to represent ‘enterprise’, ‘organizational unit’, or ‘government organization’. Although we discuss class - and data constraints, these are not formally shown in this paper. Furthermore, we did not have a tool to visualize classes and their properties, we have drawn such a visualization where classes are visualized by (blue) ellipses; the instances as (grey) rectangles. Class constraints are visualized by arrows between a domain and a range. First, the classes are defined and, second, briefly explained. Finally, some guidance on the construction of a business system semantic model is given.

2.1 Classes Representing Interoperability Concepts

The following classes represent enterprise interoperability concepts at business level:

- *Business System Interoperability Model*: a model of all business activities and a business system semantic model for modelling interoperability in a given business system or application area, e.g. logistics, customs, and supply chains.
- *Business Activity* (or *activity*): a generic activity provided by one or more actors that is able to change the state of a business system for customers. ‘*Capability*’, ‘*event*’ and ‘*state transition*’ are synonyms for business activity as they express state change that can be induced by an actor to the outside world.
- *Business System Semantic Model*: semantic model of all classes and their constraints that are common to all business activities in a business system.
- *Value proposition*: a value offered by a particular actor based on a business activity of that actor [2].
- *Business Service*: the constraint between an actor and a value proposition of that actor. This definition implies that value propositions are the main concepts from an actor’s viewpoint.
- *Business Process*: the internal process of an actor to provide a business activity.
- *Business Activity Choreography*: an ordered set of event types for the execution of a business activity. A synonym is *business activity protocol*. A business activity choreography can be common to more than one business activity (see lateron).
- *Event Type*: a means to exchange data between a customer and a service provider within the context of a business activity, see. REA [4]. *Interaction type* is a synonym.

- *Business Transaction*: information exchange between a customer and provider for actual value exchange referring to a particular business activity or value proposition. A business transaction behaves according the business activity choreography of the referred business activity.
- *Customer*: an actor that consumes a business activity (or business service) by initiating a business transaction.
- *Service Provider*: an actor providing a particular business activity with related value propositions.
- *Resources*: two types of resources are distinguished, namely operand and operant resources [2]:
 - *Operant resources (also called t-resources)*: the persons and/or means to execute a business process.
 - *Operand resources (also called d-resources)*: the actual resources that are exchanged between a customer and a service provider. Three types of resources are distinguished: goods, services, and rights [4].
- *Resource Allocation*: the mechanism to assign operant resources to one or more business transactions.
- *Bound resources*: those operant resources that are assigned to one or more business transactions.
- *Business Transaction Phases*: the different phases in the initiation and execution of a business transaction. Together, these phases compose a business activity choreography.

We will discuss these classes in more detail in the next section. The classes and their class constraints are shown in the following figure; an OWL representation of this model is constructed with Protégé. The concepts all comprise a business system interoperability model. The business system semantic model is not visualized, see later.

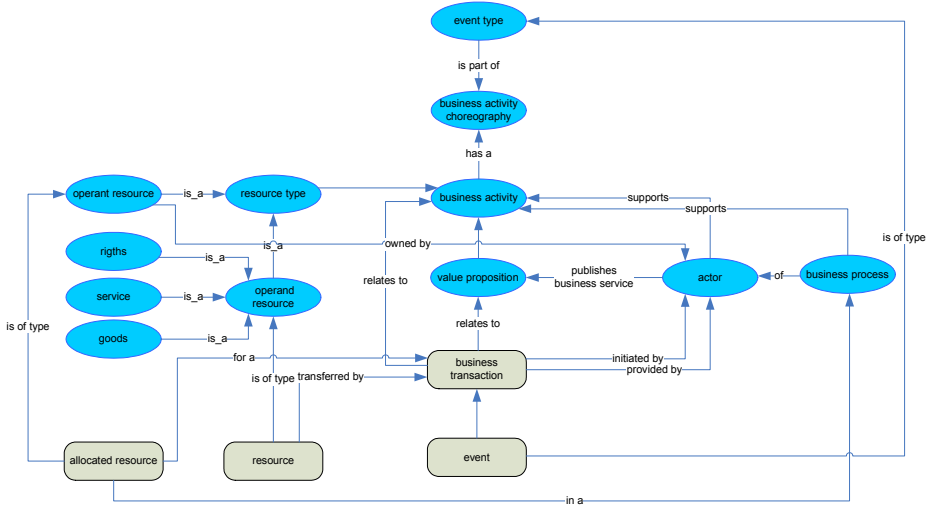


Fig. 1. Interoperability classes and their class properties.

2.2 Explaining the Classes

The classes need further explanation. This section elaborates on the difference between classes representing dynamic and static concepts, class constraints, and clarifies the core concept ‘business activity’.

Firstly, we distinguish between static and dynamic concepts, meaning that dynamic concepts change more frequent than static ones. Static concepts are shown in blue; dynamic ones in grey. It implies for instance that a business activity of an actor will change less frequent than business transactions referring to those business activities, i.e. the core business activity of an enterprise will probably not change during years.

Second, we will define class constraints. Business activity, value proposition, business transaction, business activity choreography, event type, and event are the central concepts. A business activity has choreography of event types. Choreography of event types is not modelled in this figure; we will discuss this aspect later on. A business transaction consists of a specific sequence of events that are of a type and need to be exchanged according to the business transaction choreography. The right part of the figure shows actors and their support of business activities. An actor supports a business activity and can define value propositions for those business activities. These value propositions are published as business services. An actor has a business process that supports one or more activities and their value propositions. As a business process supports a business activity, it must also support the related business transaction protocol and its choreography. The left part of the figure shows the resource types, which can be the operand and operant resource types. The operant resources are owned by an actor and can be allocated at a specific time and location to a business process for the execution of one or more business transactions. These business transactions control the exchange of the so-called operand resources: goods, service or rights. It implies that all information exchanged in a business transaction must be sufficient to exchange the goods, service or right.

Thirdly, the core concept ‘business activity’ needs clarification. We have defined a business system interoperability model that defines all real world aspects shared by actors and modelled by business activities provided by those actors, e.g. the transportation of containers from one place to another, financial risk management, and insurances against risk. These real world aspects, that either have a physical or a virtual nature, are considered as business activities that are provided by actors. The semantics that those business activities have in common are given by the business system semantic model. A business activity, or in short activity, is able to change the state of the real world by exchanging a resource (an operand resource defined according to [4]) according to a particular value proposition. Thus a state transition of a business system can be induced by executing a business activity. Thus, a business system interoperability model consists of a state space, modelled by a business system semantic model, and state transitions on that state space, modelled by business activities.

State transitions, and thus activities, in general consist of [8]:

- *Pre-conditions*: a set of conditions (or predicates) that must always be true before the execution of an activity.
- *Post-condition*: the actual result of the execution of an activity. The result can be defined as the state in case an activity is executed successfully.

- *Firing rule*: the (ordered) set of rules that are executed when all pre-conditions are met and result in the post-condition. A firing rule actually changes the state of the world. In our interoperability framework, a firing rule behaves according business activity choreography.

Pre- and post-conditions can be expressed in terms of the state space, i.e. the real world effects that can be changed by their activity. These pre- and post-conditions can be expressed at different abstraction levels that require different knowledge. An abstract specification by logical expressions can for instance not be understood by business persons, whereas they are the ones that specify the activities and thus the pre- and post-condition.

Finally, we have stated that firing rules between pre- and post-conditions are specified by business activity choreography. The latter can consist of different transaction phases as for instance [11] and [12] distinguish a negotiation and an execution phase after activity discovery. In the negotiation phase, all data needs to be exchanged to allow that pre-conditions can be met, whereas in the execution phase all required data for actual firing an activity is required. One could state that an activity can logically be decomposed into two activities that reflect transaction phases. Each of these phases consists of ordered set of event types exchanged between customer and service provider. The phases need to be completed by a rollback phase, see [11] and [12]. One could argue that business activity choreography always consists of the same set of phases. However, one could also distinguish between choreography for public and commercial services. The main difference would be that in public services there is no negotiation on prices and conditions of a value proposition, but a public service provider has to state officially that pre-conditions are met and a firing rule can be executed (in Dutch: ‘ontvankelijkheidsverklaring’). Execution results in a decision (e.g. an ordinance or a grant; Dutch ‘beschikking’), e.g. a building permission, permission for transportation of goods, and the reception of unemployment benefit. A decision can always be followed by an official objection or a different viewpoint. A decision is always according to the request, negative, or partly positive.

2.3 Refining Pre- and Post-conditions for an Application Area

In this section of the paper, we try to combine physical business systems and more abstract systems by introducing the concept of ‘place’. This latter concept represents either a physical location or a state. A place is always connected to a business activity and a place can contain zero, one or more resources at a given time. In our framework, a business activity equals a state transition with pre- and post-conditions. Pre-conditions consume so-called operand resources that are produced as a post-condition according to firing rules and their choreography utilising operand resources. For instance, in case a business activity is the assembly of cars as output resources, the input resources are its parts and assembly machines and personnel the operand resources. This example illustrates that all its parts need to be present at a given time to assemble a car at a given time.

We take the approach formulated in [12] and [14] and state that:

- A pre-condition of a business activity specifies the types and number of operand resources that are required for its execution and the types and

number of operant resources that are input to the activity. Pre-conditions are connected to a place by an input connector.

- A post-condition of a business activity specifies the types and number of operant resources that are produced as a result of the business activity, including the duration between consumption of the input operand resources and the output operand resources. Post-conditions are connected to a place by an output connector.
- Between two actors, a business transaction shares the availability of resources of type operant resources in a place connected to a pre-condition and the availability of resources, also of type operant resources, that are produced by a post-condition in a place connected to that post-condition via an output connector.

The concepts ‘place’, ‘connector’ and ‘availability’ to our ontology (see next figure) support the ability to express availability of resources in a given place at a given time. Time is expressed as ‘availability’ that refers to a place (or state) in a Petrinet and resources. At any given time, a place can contain various resources belonging to different business transactions for a business activity. In this approach, an input connector relates to a pre-condition and an output-connector to a post-condition. This extension expresses that a business transaction must always consider ‘time’, ‘resource’ and ‘place’ to be shared by two actors that participate in that business transaction. Place can represent a physical location or a state in a state space shared by two (or more) actors.

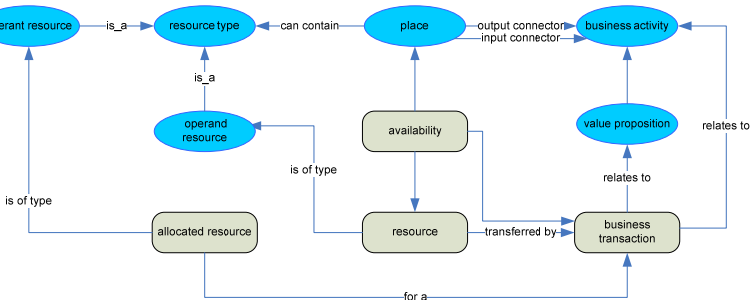


Fig. 2. Adding place and time to ontology.

According to Abstract State Machines [8], the relation between pre- and post-conditions is not expressed, which is however the case according to [14] by places and connectors. The latter means that a Petrinet of processors, places, and connectors exists and each place can contain object types. In our approach, these object types are resources and a business activity is equal to a processor. In some application areas, it might be worthwhile to specify such a network explicitly with a Petrinet, e.g. in transport a network of ports exists between which sea transport is possible. In other application areas like government, the network can become very complex due to the large number of business activities, each of which changes part of the state space, and an approach based on ASM will suit better. In the latter case, a Petrinet has to be constructed to validate that state transitions are really executable (although it can be complex to visualize due to the large number of states, see [15]).

3 Specialization to Logistics and Transport

Logistics and transport consists of several actors that utilize each others resources to transport goods from one physical location to another. The transportation process can be quite complex, involving a variety of actors. International container transport by sea for instances consists of transport by truck to a port (pre-carriage), loading a container in a vessel, sea transport, discharge of the container in the port of destination, short sea shipment from that port to another port, and final transport by truck to the final destination. In other occasions, large shippers have regional warehouse in close to a port, in which the goods are stored for regional distribution. This paper only presents ‘transport’ as a business activity, whereas others like ‘transshipment’ and ‘storage’ are other business activities. First of all, we define ‘transport’ as activity and secondly we present a semantic model for this particular activity.

3.1 The activity ‘transport’

‘Transport’ is a specialization of ‘business activity’. Quite a number of actors offer ‘transport’ as a value proposition, e.g. a number of shipping lines offer transport of containers between for instance specific ports in Europe and the Asian Pacific. One organization can also offer a variety of transport services, e.g. express delivery (the same day), air cargo and an international parcel service. Although each transport service will have a different price, is offered for a different type of cargo, and has different durations, all value propositions for ‘transport’ have common elements. A specialization of ‘business activity’ called ‘transport’ defines those common elements: the physical movement of cargo between two places with duration, cost, and a specific transport means as an allocated resource. Specific transport means have a so-called transport mode and will give restrictions to the type of goods that can be transported, e.g. a container vessel can for instance only carry 20 or 40 feet containers and an airplane requires the particular containers that fit that type of plane. Thus a particular transport activity can be expressed by the number and type of operand resources that can be transported using a particular operand resource. The relation between operand and operant resources is expressed by the business activity, although it can also be expressed as a constraint between operand and operant resources. Duration also gives restrictions to type of cargo, e.g. heavy cargo needs more attention and will take more time. The pre- and post-conditions for ‘transport’ are derived from the above mentioned parameters and can be expressed quite simply:

- Pre-conditions are defined as in terms of the operant resources that can be consumed via input connectors:
 - The cargo offered by a customer is part of the set of cargo types defined by a service provider.
 - The weight of the cargo offered must be between the minimal and maximal weight defined by a service provider.
 - The physical dimensions of the cargo must be between a minimal and maximal dimension defined by a service provider.

- Post-condition: the cargo offered by a customer is transported by a service provider to a place of delivery according to the agreed duration and costs. It is said to be produced in a place connected to the post-condition by an output connector.

Although the values of pre-conditions differ per value proposition, type and number of cargo are the only concepts requiring a value. Considering that ‘transport’ activity can be this simple, we still have to connect an activity to places with resource types. This connection can be specific for each value proposition of each service provider, although of course different service providers can use the same places (e.g. in terms of ports or other types of hubs for transshipment). By connecting places to ‘transport’ activities, a so-called transport network is defined. Such a network can be defined in various ways, e.g.:

- A transport activity is within a certain country, implying it is only national transport. Transport can in principle take place between any physical location within that country.
- A transport activity is in a region covering various countries or part of countries, e.g. between the Netherlands and northern part of Italy via Austria and Germany or between the EU and the Asian Pacific. The EU can also be an example of a region.
- A transport activity is defined from any place in a region and a particular location, e.g. a port. This transport activity is normally offered as inland transport in combination with sea-transport (see next example).
- A transport activity is defined between two physical locations, e.g. between two ports. By combining transport activities between various ports, a so-called voyage is defined. In this particular case, various value propositions between different ports can be defined on the same voyage¹.
- A similar approach to the last is to construct a so-called hub network. Between the hubs, that are physical locations, a transport activity is defined at a regular basis (e.g. daily between those hubs). Thus, a distribution network can be defined.

Additionally, a transport activity utilizes particular resources with a particular transport mode, e.g. vessels for sea transport and trucks for road transport. Whilst value propositions may differ for each transport mode, a customer may request a particular transport mode to be used. Considering a transport network, a business transaction has to contain the following information:

- The place of acceptance and delivery as indicated by a customer are in the geographical area that is connected to a pre-condition by an input connector and a post-condition by an output connector.
- The required transport mode is in the set offered by a service provider.
- The difference between the expected date and time of delivery and acceptance must be greater than or equal to the duration of the requested business activity.

¹ In sea-transport, a voyage most often has a port at which it starts and a port at which it ends, related to a particular time of start and end. In the meantime, the voyage passes different ports. A voyage is independent of the vessel used as an operand resource; that vessel can pass a port with different voyages in a given time period.

Thus, one could basically state that pre- and post-conditions can be simple, whereas complexity is in the network of places, connectors, activities, and resource types in those places. The structure is for all actors providing a ‘transport’ activity identical, but values will differ. The firing rule of ‘transport’ activity requires exact data of the cargo, e.g. the number and type of packages, container numbers, and other actors involved.

3.2 Transport Interoperability Ontology

Transport interoperability ontology is the specialization of the concepts introduced in section 2. Basic classes are already discussed in the previous part. The OWL document of the business system interoperability model is imported in an OWL document for transport and extended with the following concepts and class constraints for transport (see also the next figure):

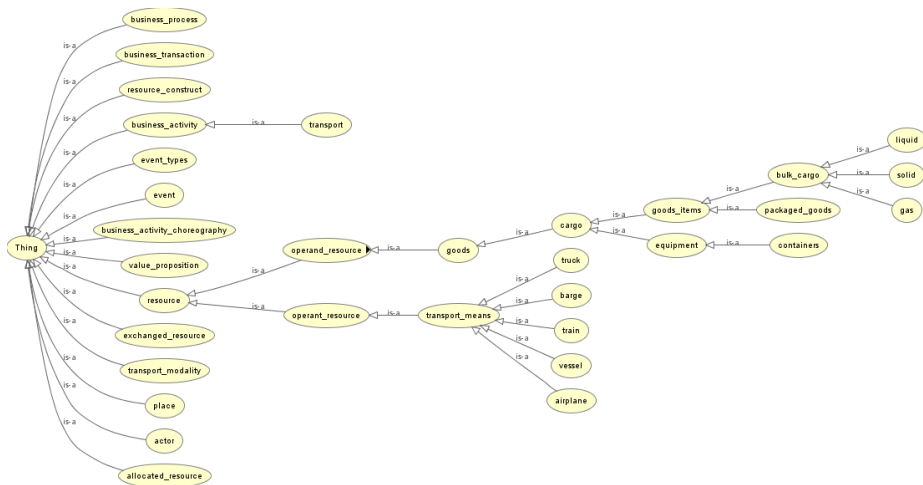


Fig. 3. Classes of transport ontology.

- Specialization of operant resources to transport means and packaging types. Transport means are further specialized to vessel, truck, barge, airplane, and train. Each of these transport means has a specific transport modality and its own characteristics, e.g. a truck has a license plate and a vessel a Radio Call Sign. Packaging types can be further specialized in those that are used once (e.g. cartons and boxes) and those that are used several times (e.g. containers and pallets). Note that these operant resources can also take the role of operand resources in which case the use is given to another actor (see for instance REA that defines ‘use’ [4]).
- Specialization of operand resources ‘goods’ to cargo. Cargo can be further specialized to ‘goods items’ and ‘equipment’. Goods items are further specialized to packaged goods and bulk cargo (either liquid or solid). Equipment, e.g. containers, can be both operant and operand resources: they can be used to facilitate transport and have to be allocated, and they are cargo.

- At the highest level, the concept transport modality representing a particular transport mode (sea, air, road, inland waterways) is introduced. Sea can be further decomposed into deep-sea and short sea.
- A number of specific properties is introduced like a property that a transport means has a modality (a specialized transport means like a truck should have modality 'road'). Additional properties can be defined like the fact that a certain transport means can only be used for container transport.

Fig. 3 shows the specialization of the overall model to transport. The figure only shows the classes, not the required class and data properties. It needs further extension for the support of for instance dangerous and temperature controlled cargo. Introducing these specializations also gives a number of additional properties like stowage restrictions to dangerous cargo. Further work needs to be done to complete this model.

4 Reference to Other Approaches

We have combined various concepts of other approaches to construct our ontology: Resource, Event, Agent (REA, [4]), Semantic Markup for Web Services (OWL-S, [13]), Web Service Modeling Ontology [3], and timed, colored Petrinets [14]. We will briefly discuss their relation to our framework.

REA is an accountancy framework to describe exchange of goods, money and rights between two organizations. The latter are called agents. Goods, money, and rights are called resources: a resource is a collection of rights associated with it: ownership, usage, and copy rights. An economic event is the transfer of rights associated with a resource from one economic agent to another. Resources, agents, and events can be of a type. Resources and their types are part of our model since they are exchanged between any two agents. An agent type is modeled as an actor and an event type as a business activity. Whereas REA distinguishes different event types like 'produce', 'consume', and 'use', those are part of pre- and post-conditions of 'business activity', e.g. a pre-condition consumes one or more resources and a post-condition produces them. In case there is no post-condition, a resource is said to be 'consumed'. The term 'event' in our model is restricted to information exchanged within the context of a business transaction. 'Business transaction' in our ontology seems to be identical to 'event' in REA. The approach taken in REA is to specialize these basic concepts to an application area, e.g. the production and consumption of pizza's or lending of books as illustrated in [4].

Semantic Markup for Web Services, OWL-S, defines a semantic model of all concepts required for a web service, or more generic, a service. A service has a profile (what it does), a grounding (how to access it), and a model (how it works). In our model, we do not describe grounding. A service profile defines aspects like pre- and post-conditions, input and output and a process. The process describes how to interact with a service; it can be constructed with a number of constructs like 'split' and 'join'. One of the core concepts in our ontology is 'business activity' with pre- and post-conditions, input and outputs, and a choreography. In this approach, a business activity can be compared with 'service'. However, we have taken the approach that

many actors can have identical business activities, whereas each actor can have its particular value proposition. The combination of ‘business activity’ and ‘value proposition’ seems to be identical to ‘service’. Inputs and outputs are in our model defined as resource types that can be consumed or produced in places, and choreography describes the way to interact with a business activity.

Web Service Modeling Ontology is based on Abstract State Machines (ASM [8]). The core concepts are ‘goal’ to represent a customer requirement and ‘capability’ to represent a service, including their grounding to technical standards like XML Schema. Whereas OWL-S does not give guidance on the specification of input and output semantics, WSMO does by applying OWL. Goal and capability are expressed by pre- and post-conditions of an ontology. Furthermore, goal and capability have interfaces with choreography. Mediation is meant to match a goal and a capability. Thus, WSMO uses similar concepts as defined in OWL-S, but slightly different. We will not discuss differences, but one could state that similar to OWL-S our concepts ‘business activity’ and ‘value proposition’ are identical to ‘capability’. They all reflect the concept of ‘state transition’ within ASM.

Finally, we have introduced concepts from timed colored Petrinets. We define actors in terms of the business activities they can provide according to a value proposition. A business activity is identical to a Petri net ‘process’ that has connectors to places. These places can contain tokens with a color that can be modeled by ontology. The concept ‘state’ is expressed by all tokens in all places at a given time. Thus, the concept ‘state’ is defined different from that in Abstract or Finite State Machines, where it refers to a specific object (e.g. the ‘state’ of a container). Whereas in Petrinets all processes need to be connected to places, Abstract State Machines specify the individual state transitions (the ‘processes’) on states which are arbitrary data structures. In ASM, ‘state’ can be derived from any pre- or post-condition, whereas in timed colored Petrinets, this part of state is reflected by the color of tokens that can be consumed or produced. Whereas Petrinets present (graphically) a complete model connected processes via places, ASM needs chaining of state transitions based on logical expressions defining pre- and post-conditions.

5 Conclusions and further Research

We have presented enterprise interoperability ontology for Service Oriented Computing at business level and applied it to logistics. The ontology is based on an accountancy model called REA, extended with concepts from existing service frameworks like WSMO, and defines concepts that can be shared amongst different classes of business services. It seems obvious that business transactions and their events contain all relevant information for executing a business activity, e.g. the availability of resources in a particular place. We have not discussed business activity or value proposition discovery, which differs from service discovery. It is described in more detail for a particular case [10].

There are still a number of issues for further research:

1. We have taken the basic concepts of timed colored Petrinets and applied them to enterprise interoperability for transport. However, other cases are better modeled with Abstract State Machines, see for instance [8]. Both approaches

offer identical functionality and a combination of approaches needs to be considered.

2. The concepts consider both process and semantic aspects, e.g. it models choreography as a concept and semantics of exchanged information. Choreography should be further refined, e.g. by applying process modeling approaches as defined for instance in OWL-S [13] or YAWL [16].
3. Relevant aspects like organizational issues and grounding need to be considered further. As grounding is clear, organizational issues consider the maintenance of models, service directories, etc. Furthermore, integration of semantic models of different application areas has to be considered. This is especially the case in for instance government interoperability where each government organization is responsible for (part of the) government data. Is a common model of enterprise interoperability meeting these requirements? Ontology import might offer some solution. Is it also allowed for actors to define their specific models based on a generic model? What is the impact of actor-refined models on an execution environment?
4. An execution environment should combine dynamic service mediation supported by user interaction, with automatic service mediation based on rules. What is the role of enterprise interoperability concepts in an execution environment? What would be the architecture and functionality of such an execution platform? Can this functionality be offered by for instance the DynamiCoS framework [17] or the SESA execution environment [3]?
5. There are some practical issues to consider when defining semantic models for an application area. One can for instance define a complete model for international logistics or a separate model for each modality. Another approach is to define separate models for each activity. Again, practical issues are the integration of different models to create different views on logistics, e.g. a customs view may differ from a transporters view.
6. Currently, practical applications still need the specification of the information exchanged in particular events that are of a type. Each event type only contains that specific information that is required by a particular value proposition. Further research supported by a Proof-of-Concept is required to investigate the impact in practice regarding interoperability without specific semantic models for each event type.
7. We have only applied the model to logistics in this particular paper. However, we also have examples of the application of our proposed enterprise interoperability ontology to government services [10].

References

1. T. Erl, *Service-Oriented Architecture – concepts, technology, and design*, Prentice Hall, 2005.
2. J. Spohrer. and S.K. Kwam, *Service Science, Management, Engineering and Design (SSMED) – An emerging Discipline – Outline and references*, International Journal on Information Systems in the Service Sector, May 2009.
3. D. Fensel, M. Kerrigan, M. Zaremba (eds.), *Implementing Semantic Web Services – the SESA framework*, Springer-Verlag, 2008.

4. P. Hruby, *Model-Driven Design using Business Patterns*, Springer-Verlag, 2006.
5. J. Heineke, M. Davis, The emergence of service operations management as an academic discipline, *Journal of Operations Management* 25 (2007) 364–374.
6. Y-H Tan, W.J. Hofman, J. Gordijn, J. Hulstijn, A framework for the design of service systems, Springer-Verlag (to appear in *Service Sciences*).
7. *Business Process Modelling Notation (BPMN)*, version 1.2, january 2009.
8. E. Börger: "High Level System Design and Analysis Using Abstract State Machines", *Proceedings of the International Workshop on Current Trends in Applied Formal Method: Applied Formal Methods*, p.1-43, October 07-09, 1998
9. M. Holtkamp, W.J. Hofman, *Semantic Web Technology – state of the art report*, Internal TNO report, 2009.
10. W.J. Hofman, M. van Staalduinen, *Dynamic Public Service Mediation*, eGov2010.
11. J.L.G. Dietz, *Enterprise Ontology, Theory and methodology*, Springer-Verlag, 2006.
12. W.J. Hofman, *A conceptual model of a business transaction system*, Uitgeverij Tutein Nolthenius, 1994.
13. *OWL-S, Semantic Markup for Web Services*, W3C member submission, 2004.
14. K.M. van Hee, *Information systems engineering: a formal approach*, Cambridge University Press, 1994
15. H.M.W. Verbeek, A.J. Pretorius, W.M.P. van der Aalst ... [et al.], *Visualizing state spaces with Petri Nets*, Eindhoven : Technische Universiteit Eindhoven, 2007.
16. A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, N. Russel, *Modern business process automation: YAWL and its support environmnet*, Springer-Verlag, 2009.
17. E. Silva, L. Ferreira Pires, M. van Sinderen, *Supporting Dynamic Service Composition at Runtime based on End-user Requirements*, 1st International Workshop on User-Generated Services, 2009.
18. P. Hitzler, M. Krotzsch, S. Rudolph, *Foundation of semantic web technologies*, CRC Press, 2009.

A Diffusion Mechanism for Online Advertising Service Over Social Media

Yung-Ming Li and Ya-Lin Shiu

Institute of Information Management, National ChiaoTung University, Hsinchu, Taiwan
{yml, ayaa.iim97g}@mail.nctu.edu.tw

Abstract. Social media has increasingly become a popular platform for diffusing information, through the message sharing of numerous participants in a social network. Recently, companies attempt to utilize social media to expose their advertisements to appropriate customers. The success of message propagation in social media highly depends on the content relevance and closeness of social relationships. In this paper, considering the factors of user preference, network influence, and propagation capability, we propose a social diffusion mechanism to discover the appropriate and influential endorsers from the social network to deliver relevant advertisements broadly. The proposed mechanism is implemented and verified in one of the most famous micro-blogging system-Plurk. Our experimental results shows that the proposed model could efficiently enhance the advertising exposure coverage and effectiveness.

1 Introduction

In recent years, social media has been flourished and raised high much popularity and attention. Social media provides a great platform to diffuse information through the numerous populations. Common social media marketing tools include Twitter, YouTube, Facebook and so on. An overwhelming majority (88%) of marketers are using social media to market their businesses, and a significant 81% of marketers indicate that their efforts in social media have generated effective exposure for their businesses, according to a social media study by Michael Stelzner[12]. In 2010, one of the most popular micro-blogging websites, Twitter, announced an innovative advertising model, "Promoted Tweets". Promoted Tweets makes tweets as ads, which are distinctive from both traditional search ads and recent social ads. They measure the advertising performance and the payment of sponsored tweets by "resonance" - the interactions between users and a particular sponsored tweet such as retweet, reply, or bookmarking [13]. How to choose the right people to deliver the information, how to take the advantage of the social media, and how to design an ads diffusion mechanism to widen the spreading coverage are crucial issues in the online advertising campaign. In this paper, to address these issues, we design a social media diffusion mechanism, based on the concepts of content recommendation and network routing. Once the appropriate messages and diffusion paths are identified, message can be effectively delivered with support of the generated information. Considering the factors of user preference, network influence, and propagation capability, our system can

effectively identify the most appropriate nodes in the social network for delivering the relevant ads and recommend the friends for information sharing for an intermediate node.

2 Related Literature

2.1 Social Media

Social media are Internet platforms designed to disseminate information or messages through social interaction, using highly accessible and scalable publishing techniques. Social media is composed of content (information) and social interaction interface (intimate community engagement and social viral activity). With its emerging trend and promising popularity, researchers have put academic efforts in analyzing the characteristics and functionalities of social media. For example, Kaplan and Haenlein [6] examine the challenges and opportunities of social media and recommend a set of ten rules that companies should follow when developing their own social media strategy. To effectively communicate with customers, researchers engaged in analyzing marketing trends and social relations. Gilbert and Karahalios [4] develop a predictive model that maps data of social activity to tie strength so as to improve design elements of social media. To better figure out the users' behaviors, many researchers analyze the social influence, social interactions, and information diffusion in social media [3]. Comparing to the existing works, the study of information diffusion mechanism design of social media is apparently rare and new.

2.2 Online Advertising

The issue of online advertising has aroused much academic interests and been spotlighted for decades. Online advertising usually could be categorized into two types: 1) targeting advertising, which deliver the ads based on user's preference profiles, 2) social advertising, which deliver the ads the influential users determined by social relationship [11]. Targeted advertising usually applies the content-based and collaborative-based approaches to discover users' personal preferences. Compared with the traditional online advertising, social advertising is a form of advertisement that addresses people as part of a social network and uses the social relations and social influences between people for selling products [14]. Some researchers measure the influential strength by analyzing the number of network links and users' relation and interaction in the network to identify the influential nodes for social advertising [11, 14]. In this paper, considering the factors of user preference, network influence, and propagation capability, we propose a social diffusion mechanism to identify the appropriate and influential endorsers from the social network to deliver relevant advertisements broadly.

2.3 Information Diffusion and Social Routing

Researchers analyze information diffusion in the social network based on individual's characteristics. Some based on the bond percolation, graph theory or probabilistic model to extract the influential nodes, considering the aspect of dynamic characteristics, such as distance, time, and interaction and so on [7-8]. By revealing influential factors and realizing the processes of the information diffusion, marketers can predict when and how the information spreads over social networks to maximize the expected spreading result [5]. In this paper, we include static and dynamic factors dimensions to evaluate the propagation ability of nodes in social network. The design of social diffusion mechanism is conceptually similar to that of computer network routing process in selecting paths to switching the packet. Routing directs packets to be forwarded from their source toward their ultimate destination through intermediate nodes; hardware devices usually called routers. However, in the context of social network, the links in social networks are formed by social relations and interaction and researchers focus on the study of the issue: delivering the right information to the right nodes and spreading widely. The goal can be achieved by implementing feasible approaches to discover the influential nodes and leveraging the social relations to diffuse the information between users further. In our paper, we incorporate the concept of network routing to develop a social endorser engine to generate "social routing tables" to support the information diffusion in a social network.

3 System Architecture

Analogous to the routing process in computer networks, we design a social diffusion mechanism which sends a recommended list of users to our initial nodes with suitable path for information diffusion. The recommendation lists suggest the users who have strong propagation abilities in social networks. The users are referred as social endorsers potentially willing to transmit the information to all his/her friends.

Notice that the proposed social diffusion mechanism is different from spamming. We recommended those friends based on their preferences, social influence, and propagation abilities via quantitative measurement. The advertising message will be guided to right people by user's judgments with the support of the system recommendation. If users deliver ads to their friends, it means that users also think their friends like the ads. The mechanism takes the advantage of content relevance and social relation to reduce the negative impression of the advertisement and gain the advertising effectiveness. Social media provided us the source data of individual's preference, social relation and social influence. Preference is an important issue in target advertising. The social influence between users of the social media happened when they are affected by others. It is likely that we are usually influenced by our friends or our family. The social relation is a crucial factor to empower the social influence. If a user frequently interacts with someone, to some extent, there are more similarities and closeness between them. Therefore, we incorporate these components into our proposed social endorser discovery engine.

3.1 Social Endorser Discovery Engine

Effective information diffusion on social networks is grounded on the relevance of individual preference and closeness of social relations. Therefore, the main functionality of the proposed social endorser discovery engine is to identify the nodes with the strong propagation ability in disseminating relevant messages as wider as possible. In order to identify the appropriate social endorsers so as to achieve a better diffusion performance, in this research, we not only consider the static factor, but also dynamic factor in the evaluation of nodes' diffusion capability - transmit information towards the most suitable friends and spread widely further. Figure 1 displays the main components and procedures of our social endorser discovery engine.

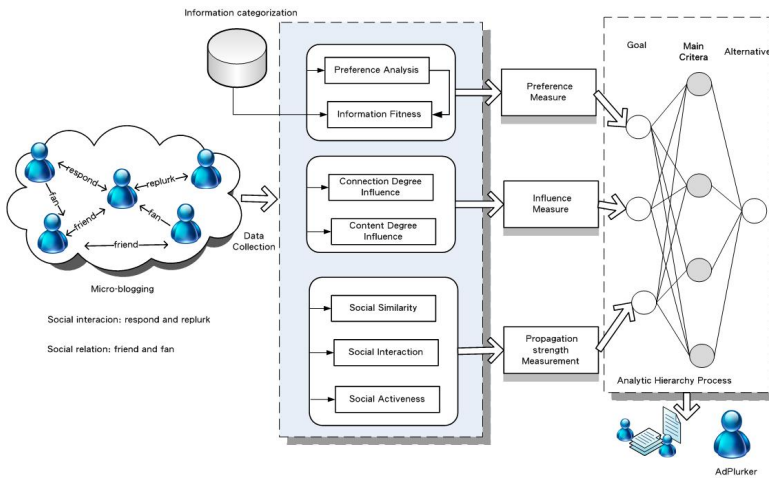


Fig. 1. Architecture of endorser discovery engine.

User Preference Analysis Module

As people tend to share information interesting to the receiver, discovering users' preference is an important factor to be considered in the social endorser discovery. By analyzing users' preference, we can better understand what kinds of information are suitable to be shared between the users. In order to realize user's affinity levels of information categories, we adopt a tree-like structure to categorize a set of information and users' preference. Tree-like structure is practically employed in many researches, such as product taxonomy [1, 9]. Besides, we utilize a distance-based approach to calculate the similarity between user categories and information categories. The preference of a user and the type of an advertising message are described by a catalog node they belong. Assume C_1 and C_2 stand for the category 1 (a user's preference) and catalog 2 (an advertisement type) belong to respectively and C_{fn} represent the catalog of the first mutual parent nodes of catalogs 1 and 2. The fitness degree of the ads to a user can be calculated by the following formula:

$$Sim_P(C_1, C_2) = \frac{2D_{fn}}{D_1 + D_2 + 2D_{fn}} \quad (1)$$

Network Influence Analysis Module

Connection Degree Influence. For the purpose of evaluating the relative importance of user position in the whole network, social network analysis is applied. Degree centrality is defined as the number of direct connections/links upon a node. Specifically, in-degree is a count of the number of connections directed to the node, and out-degree is the number of connections that the node directs to others. In this research, first, we consider the spammer or bots attempt to follow many people in order to gain attention. Secondly, based on in-degree or out-degree ignores the ability for a user to interact with content in the social network. Therefore, we use mutual relation (friendship) to measure the degree centrality as in practice, mutual degree represents the number of friends a user has. Mutual degree for user i is measured as

$$MD(i) = \sum_{j=1}^n E_{ij} \quad (2)$$

where E_{ij} is 1 if an edge from node i to j exists and an edge from node j to node i exists, too, otherwise it is 0.

Content Degree Influence. Content Degree Influence is used to evaluate the popularity of what a user posts. We measure the content degree of a user by the count of the total responses and message forwards by people. We denoted as the total number of the elements in a set. The formula for content degree measure can be expressed as:

$$CD(i) = \frac{|\Phi_{reponse(i)}| + |\Phi_{forwad(i)}|}{|\Phi_{post(i)}|} \quad (3)$$

where $\Phi_{post(i)}$ stands for a set of the messages posted by user i , $\Phi_{reponse(i)}$ represents the set of the responses on user i 's posts, and $\Phi_{forward(i)}$ is the set of i 's posts forwarded by others. The aggregate network influence score is the sum of the mutual degree value $MD(i)$ and the content degree value $CD(i)$.

Propagation Strength Analysis Module

Social Similarity. Social similarity aims to measure the similarity of two people from implicit social structure and behaviors, such as "friend-in-common" and "content-in-common". The more friends-in-common of two people generally reflects the higher connection level between them. If two people have more common friends, their interests should be more similar and the possibility that a people will forward a message he feels interesting to the other becomes higher. Denote $F(i)$ as a set of user i 's friends. The similarity of friend-in-common between user i and friend j , is measured as:

$$Sim_{CF}(i, j) = \frac{|F(i) \cap F(j)|}{\text{Max}(|F(i)|, |F(j)|)} \quad (4)$$

In addition, the more content-in-common posted by two people, the higher similarity degree between them. Semantic analysis can be use to evaluate the social similarity measure in the aspect of content-in-comment and to discover potential preference

of users [2]. Specifically, traditional information retrieval (IR) technology can be used to analyze the semantics of content. To examine the semantic similarity among posts, we use CKIP Chinese word segmentation system to parse and stem the crawled contents and apply the analysis of frequency-inverse document frequency (TF-IDF) weight to measure how important a word is to a document in a collection or corpus.

$$tf_{i,j} = \frac{freq_{i,j}}{\max_l(freq_{l,j})} \quad (5)$$

where $freq_{i,j}$ as the raw frequency of term i appear in post j and $\max_l(freq_{l,j})$ is the number of times the most frequent index term, l , appears in post j . The inverse document frequency for term i is formulated as

$$idf_i = \log \frac{N}{n_i} \quad (6)$$

where N is the total number of posts and n_i is the number of posts in which the term i appears. The relative importance of term i to post j can be obtained by calculating. Then, we measure the similarity degree between people by a cosine similarity metric. The similarity of corpus between user i and friend j is defined as:

$$Sim_{CC}(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{|\vec{i}| |\vec{j}|} \quad (7)$$

where \vec{i} and \vec{j} are the two vectors in the m dimensional user space which is the keywords to a person in a collection or corpus.

Finally, the total social similarity (SS) score is the sum of “friend-in-common” and “content-in-comment” value.

Social Interaction. Social interaction is different from social similarity since social interaction explicitly catches the factor of dynamic actions between people. It can be used to evaluate the intimacy between two users. For instance, it is common for a user to respond or forward someone’s message. It is reasonable to assume that more interaction activities would lead to a higher probability to transmit the information as they are usually interested in mutual messages. Given two users i, j , social interaction between them can be formulated as:

$$SI(i, j) = \frac{|\Phi_{response(i,j)}|}{|\Phi_{response(i)}|} + \frac{|\Phi_{forward(i,j)}|}{|\Phi_{forward(i)}|} \quad (8)$$

$\Phi_{response(i,j)}$ is the set of responses generated by user j to user i ’s posts and

$\Phi_{forward(i,j)}$ is the set of forwards conducted by user j to user i ’s posts.

Social Activeness. Social activeness is used to calculate the activity intensity of a user. A user with higher activeness indicates a larger level the user is engaged in information sharing or discussion with others and a higher probability to transmit the

information. We calculate the activeness of a user by the count of post records during a period of time in a social platform. The formula is defined as below:

$$SA(i) = \frac{\sum_{t=1}^T |\Phi_{messages(i,t)}|}{T} \quad (9)$$

where $|\Phi_{messages(i,t)}|$ is the total number of messages posted by user i at time period t .

The propagation strength measurement is used to evaluate the user whose network propagation capability of a user. The propagation strength of a user is measured by aggregating the propagation strength and can be computed in a recursive way. To enhance advertising efficiently, it is important to pay attention on next layer's propagation capability. Though advertisers delivered advertisements to a social endorser with high propagation capability, they can't ensure the social endorser's friends with high propagation capabilities equally. Therefore, we thought friends' propagation capabilities would affect a social endorser's propagation capability. Friends' propagation capabilities became a dimension to measure a social endorser propagation capability. In other words, individual's propagation capability is affected by their friends.

The propagation strength is formulated as below:

$$PA(i) = SA(i) + \sum_{j \in F(i)} PA(j) \cdot (SS(i, j) + SI(i, j)) \quad (10)$$

where $F(i)$ denotes a set of user i 's friends.

4 Experiments

Micro-blogging service has become one of the top tools for social media marketing. Compared to traditional blogging, micro-blogging allows users to publish brief messages make people easy to read and repost. These characteristics: brief messages, instant, easy to read and easy to share make micro-blogging become a good platform to conduct social media marketing. Therefore, in this research, we apply and validate our proposed mechanism in micro-blogging systems. We conduct our experiments in Plurk. According to Alexa, 2010, the user of Plurk is more than Twitter and 34.4% of Plurk's traffic comes from Taiwan. Users of Plurk can connect with their friends via lots of functions such as updating instant messages, sharing image or video to your friends and responding friends' messages. Besides the well-constructed network structure, another important reason of choosing Plurk as our platform is they provide the APIs for developers to easily request the data of users and networks which is helpful to crawl more complete data to conduct our experimental work.

4.1 Data Description

In our mechanism, AdPlurker will send private messages with ads to users whom we discover by different approaches. Users who receive the messages, which include

brief information of the ads and a recommended list of users who are also interested in the ads and have higher propagation capabilities in their network. User can click the hyperlinks of brief information to get detailed information and the click-through record will be collected for evaluation work. Also, users can share the messages with their friends who are recommended by the system. The transmitted message records will be collected. To preventing click fraud, we recorded one click for each individual user for the same advertisement. We conduct the experiment during the period of 11 April to 13 May.

In the preference module, we collect target users' explicit preferences by questionnaires. Besides, in order to better realize users' preferences, we collect implicit preferences from the behaviors in Plurk. In the Plurk, "become fans" is a function for users to follow others' plurks and also declare their preferences for information type. We use these data to match with the hierarchy of product category of Rakuten, one of the famous online shopping mall, to structure the preference category tree of each user. In the influence module, the out-degree measure and social popularity are taken into consider. The friend links usually are the strongest links and imply the structural influential in the network. A user is attention-getting since his/her plurks is often replurked and responded by others. It also means he/she is influential in content. In propagation strength measurement, we analyze users' occurring activities during the recent six months: daily plurks, responses and replurks as the active and social interaction measure, the similarity in friends and content as the social similarity measure. We calculate the statistics of these as the propagation strength measurement.

We develop a Plurk robot named as AdPlurker and invite users who are active and have used Plurk for a long time to join the experiments. Until April 2009, There are 107 users (55% male, 45% female) aging between 20-50. To simulate a real network structure, our target users are formed with different locations and careers. There are 121,837 users and 971,014 plurks in our database. We collect data from the target users to 3rd-4rd layers, due to the degrees of separation is limited the layer to 3rd-4rd layer[10], and crawl their plurks, responses, and interactions with friends that happened within six months.

4.2 Experimental Results

In order to evaluate the performance of our proposed mechanism, we used the click-through rate (CTR) [20] and repost-through rate (RTR) [13]. The former is a practical statistics about advertising effectiveness; the latter is an effective means to evaluate the eventual spread of the advertisement. Also, the two performance indicators are the key measures in promoted tweets which is newly social advertisement platform proposed by Twitter. The CTR formula is defined as:

$$CTR = \frac{\Phi_{clicks}}{\Phi_{delivered}} \quad (11)$$

where Φ_{clicks} is the total number of clicks and $\Phi_{delivered}$ is total number of ads delivered. The RTR formula is defined as:

$$RTR = \frac{\Phi_{repost}}{\Phi_{delivered}} \quad (12)$$

where Φ_{repost} is the total number of repost and $\Phi_{delivered}$ is total number of ads delivered. We compare four online advertising approaches, which are commonly used in micro-blogging. These different approaches are described as follows.

In-degree. It is the most common measure used to evaluate the influence of micro blogging by the number of fans. This measure is currently employed by many other third-part services, such as twitterholic.com and wefollow.com

Ratio-degree. The measure is similar to the ratio between the number of a user's followers and the number of other people that the user follows. It was proposed from the Web Ecology Project, an interdisciplinary research group based in Boston, Massachusetts.

Preference + Out-degree. Discovering the topic-influential nodes for delivering advertising message by taking the advantage of the target advertising and social influence.

Social Diffusion. The approach we proposed in this study. We applied analytic hierarchy process (AHP) to realize the final weight combinations of three components.

Figure 2 shows the results of different advertising strategies. According to the database, the advertisements of in-degree approach got total 0.157 CTR in 2042 delivery times. Ratio-degree approach got 0.176 CTR in 3922 deliveries. The hybrid approach of preference and out-degree got 0.217 CTR in 4596 deliveries. Our social diffusion mechanism got 0.299 CTR in 7356 deliveries. Comparing the diffusion performance in the four advertising approaches, we can observe that our proposed social diffusion approach has the best coverage and exposure in advertising campaign.

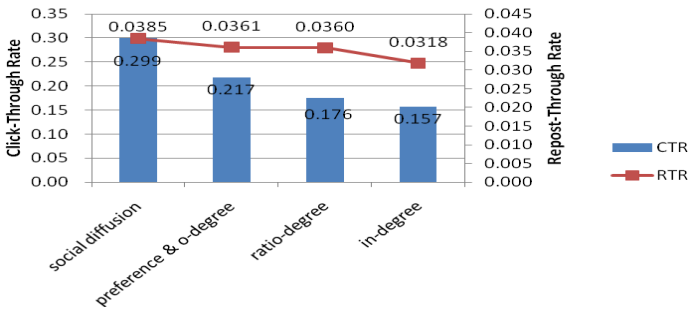


Fig. 2. Performance comparisons of various endorser discovering strategies.

5 Conclusions

In this paper, we propose a social diffusion mechanism to discover the nodes with the strong propagation capability in delivering advertising information and recommend each intermediate node a list of nodes with the high prior propagation so as to enhance the efficiency and effectiveness of spreading advertising message. We combine the static factor, which includes individual preference and link structure of relation-

ship and the dynamic factor, which includes social interactions and social similarity between the nodes, to develop our model. Our experimental results get positive outcomes in both click-through rate and repost rate, and reveal some implicit connections between the components in the framework. A better CTR reflects that our mechanism can raise the visibility of advertising information. And a higher RTR indicates a higher exposure of the advertising and reveals that users are interested in the advertisement shared by friends and willing to share them with others. Our proposed mechanism can widely extend the diffusion coverage of ads. It provides the advertising sponsors a powerful vehicle to successfully conduct advertising diffusion campaigns.

References

1. Albadvi, A., Shahbazi, M. A hybrid recommendation technique based on product category attributes. *Expert Syst. Appl.*(2009).
2. Berendt, B., Navigli, R. Finding your way through blogspace:Using semantics for cross-domain blog analysis. 2006.
3. Delre, S. A., Jager, W., Bijmolt, T. H. A. and Janssen, M. A. Will it spread or not? The effects of social influences and network topology on innovation diffusion. *Journal of Product Innovation Manageme*(2010).
4. Gilbert, E., Karahalios, K. Predicting tie strength with social media. In *Proceedings of the Proceedings of the 27th international conference on Human factors in computing systems*(2009) ACM.
5. Iribarren, J. L., Moro, E. Information diffusion epidemics in social networks. *Physics and Society*(2007).
6. Kaplan, A. M., Haenlein, M. Users of the world, unite! The challenges and opportunities of Social Media. *Business Horizons* 59-68.
7. Kempe, a., Kleinberg, J.,Tardos, É. *Influential Nodes in a Diffusion Model for Social Networks*. Springer Verlag2005.
8. Kimura, M., Saito, K., Nakano, R., Motoda, H. *Finding Influential Nodes in a Social Network from Information Diffusio*. Springer US 2009.
9. Leung, C. W.-k., Chan, S. C.-f.,Chung, F.-l. A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. *Knowl. Inf. Syst.*(2006) 357-381.
10. Li, Y.-M., Chen, C.-W. A synthetical approach for blog recommendation: Combining trust, social relation, and semantic analysis. *Expert Systems with Applications*(2009) 6536-6547.
11. Li, Y.-M., Lien, N.-J. An endorser discovering mechanism for social advertising. In *Proceedings of the Proceedings of the 11th International Conference on Electronic Commerce*(2009) ACM.
12. Stelzner, M. *Social Media Marketing Industry Report*. 2009.
13. *Twitter Promoted Tweets*. 2010.
14. Wen, C., Tan, B. C. Y. and Chang, K. T.-T. Advertising Effectiveness on Social Network Sites: An Investigation of Tie Strength, Endorser Expertise and Product Type on Consumer Purchase Intention. 2009.

DESIGN AND ANALYSIS OF SERVICE-ORIENTED SYSTEMS

Specifying Formal executable Behavioral Models for Structural Models of Service-oriented Components*

Elvinia Riccobene¹ and Patrizia Scandurra²

¹ DTI - Università degli Studi di Milano, Milan, Italy
elvinia.riccobene@unimi.it

² DIIMM - Università degli Studi di Bergamo, Bergamo, Italy
patrizia.scandurra@unibg.it

Abstract. This paper presents a behavioral formalism based on the *Abstract State Machine* (ASM) formal method and intended for high-level, platform-independent, executable specification of *Service-oriented Components*. We complement the recent *Service Component Architecture* – a graphical notation able to provide the overall and the components structure – with an ASM-based formalism able to describe the workflow of the service orchestration and the services internal behavior. The resulting *service-oriented component model* provides an ASM-based representation of both the structural and behavioral aspects of service-oriented systems, like service interactions, service orchestration, service tasks and compensation. The ASM formal description of a service-oriented system is suitable for rigorous execution-platform-independent analysis.

1 Introduction

The Service-Oriented paradigm is emerging as a new way to engineer applications that are exposed as *services* for possible use through standardized protocols. Services are loosely coupled, interoperable, evolvable, computational components available in a distributed environment. On top of these services, business processes and workflows are used to compose services as *service orchestration*. The Service-Oriented Architecture (SOA) is the architectural foundation for the Service-Oriented paradigm. SOA states that applications expose their functionality as services in a uniform and technology-independent way such that they can be discovered and invoked over the network. This new programming style relies on interface-based design, composition and reusability. It also requires specific modeling notations able to support the service-oriented system engineering with intuitive and easy to adopt design and implementation techniques.

Recently, the Service Component Architecture (SCA) [21] project is proposed to implement service construction based on the SOA principles. SCA provides a metamodel-based visual notation to construct and assemble service components in a platform-independent manner. The SCA initiative is divided into several specification documents, such as the SCA assembly model specification, the SCA policy framework, etc. The assembly model specifies the concept of service components and focuses on the relationship between service components in a particular assembly. However, the SCA

* This work was partially supported by the Italian Government under the project PRIN 2007 D-ASAP (2007XKEHFA)

assembly model lacks of a precise definition. As a service programming model, it is not enough for SCA to provide informal definition. A rigorous semantic model for SCA is necessary to specify the dynamic behavior of a service-oriented system, which can provide a formal foundation for the service component assembly and support to verify the compatibility of the assembled components. Moreover, the use of the SCA notation should be integrated within a precise engineering methodology for SOA, which, for high-level analysis purposes, requires a formal counterpart of the SCA description. Indeed, service-based systems usually have requirements such as service availability, functional correctness, protection of private data, etc. Implementing services satisfying these requirements demands the use of software engineering methodologies that encompass all phases of the software development process, from modeling to deployment, but also exploit formal techniques for qualitative and quantitative verification of systems.

This paper presents a behavioral formalism based on the Abstract State Machine (ASM) [5] formal method and intended for the specification and analysis of service-oriented systems at a high level of abstraction and in a technology agnostic way (i.e. independently of the hosting middleware and runtime platforms and of the programming languages in which services are programmed). This is a first result of our ongoing work towards the development of an ASM-based *back-end* framework, for high-level specification and analysis of SCA descriptions of service-oriented component systems. ASMs expressiveness and executability allow for the definition and analysis of behavioral aspects of services (and complex structured interaction protocols) in a formal way but without overkill. Moreover, the ASM design method is supported by a set of tools (developed through model-driven engineering technology), the ASMETA toolset [13, 2], useful for validation and verification (essentially simulation, scenario-based validation, model-based testing, and model-checking) of ASM-based models of services.

A *service-oriented component model* is introduced to provide an ASM-based representation of both the structural and behavioral aspects of service-oriented systems like service interactions, service orchestration, service tasks and compensation. In particular, the component model integrates the orchestration modeling with the specification of service behaviors, so integrating *intra-* and *inter-* component behavior in one formalism; this is especially useful for analysis purposes to verify “global properties” that depend on “local properties”. We start from the SCA standard [21] for the structural aspects of service-oriented components, and we complement the graphical view of a service-oriented system with a formal description which is then enriched with the executable specification of the services internal behavior and services orchestration. In particular, for modeling services behavior, the ASMs provide atomic (zero-time) parallel execution of entire (sub)machines – used to model service tasks – whose computations, analyzed in isolation, may have duration and may access the needed state portion, thus combining the atomic black box and the white box view of service-oriented components. For modeling services interaction, we exploit high-level communication patterns defined in [26] and adapted from [3]. They model in terms of the ASMs complex interactions of distributed service-based (business) processes that go beyond simple request-response sequences and may involve a dynamically evolving number of participants.

This paper is organized as follows. Some background concerning the SCA standard and the ASM formal method are given in Sect. 2 and 3, respectively. The ASM-based

service-oriented component model is presented in Sect. 4, while an illustrative case study is reported in Sect. 5. Sect. 6 provides a description of related work along the same direction and outlines some future directions of our work.

2 Service Component Architecture

The Service Component Architecture (SCA) [21] is an XML-based metadata model that describes the relationships and the deployment of services independently from SOA platforms and middleware programming APIs (such as Java, C++, Spring, PHP, BPEL, Web services, etc.). SCA is also supported by a graphical notation (a metamodel-based language developed with the Eclipse-EMF environment) and runtime environments (like Apache Tuscany and FRAScaTI) that enable developers to create service components, assemble them into composite applications, and run/debug them.

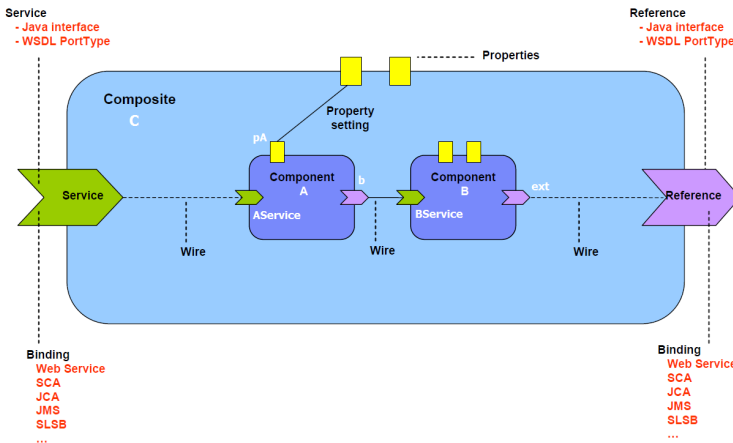


Fig. 1. An SCA composite example (adapted from the SCA Assembly Model V1.00 spec.)

To get an overview of the architecture of SCA, we will now look at its basic building blocks and their (inter-)relations. Fig. 1 shows an *SCA composite* (or *SCA assembly*) as a collection of SCA components using the SCA graphical notation. Following the principles of SOA, loosely coupled service components are used as atomic units or building blocks to build an application.

An *SCA component* is a configured piece of software that has been configured to provide its business functions (operations) for interaction with the outside world. This interaction is accomplished through: *services* that are externally visible functions provided by the component; *references* (functions required by the component) wired to services provided by other components or to references of the composite component containing the component; *properties* allowing for the configuration of a component implementation with externally set data values; and *bindings* that specify access mechanisms used by services and references according to some technology/protocol (e.g. WSDL binding to consume/expose web services, JMS binding to receive/send Java Message Service, etc.). Service and references are typed by *interfaces* that describe the

business function (operation). A particular business function is typically grouped with a set of other related operations, as defined by an interface, which as a whole make up the service offered by a provider component. Each invocation by a client component on a reference operation causes one invocation of the operation on one service provider. The provider may respond to the operation invocation with zero or more messages. These messages may be returned synchronously or asynchronously to the requester client.

As unit of composition and hierarchical design, assemblies of service components deployed together are supported in terms of *composite* components. A composite consisting of: properties, services, service implementations organized as sub-components, required services as references, wires connecting sub-components.

3 Abstract State Machines

Abstract State Machines (ASMs) are an extension of FSMs [6], where unstructured control states are replaced by states comprising arbitrary complex data. Although the ASM method comes with a rigorous mathematical foundation [5], ASMs provides accurate yet practical industrially viable behavioral semantics for pseudocode on arbitrary data structures. This specification method is tunable to any desired level of abstraction, and provides rigor without formal overkill.

The *states* of an ASM are multi-sorted first-order structures, i.e. domains of objects with functions and predicates (boolean functions) defined on them, while the *transition relation* is specified by rules describing how functions change from one state to the next. Basically, a transition rule has the form of *guarded update* “**if Condition then Updates**” where *Updates* are a set of function updates of the form $f(t_1, \dots, t_n) := t$ which are simultaneously executed³ when *Condition* is true.

There is a limited but powerful set of *rule constructors* that allow to express simultaneous parallel actions (*par*) of a single agent *self*, either in an atomic way, *Basic ASMs*, or in a structured and recursive way, *Structured or Turbo ASMs*, by sequential actions (*seq*), iterations (*iterate*, *while*, *recwhile*), and submachine invocations returning values. Appropriate rule constructors also allow non-determinism (existential quantification *choose*) and unrestricted synchronous parallelism (universal quantification *forall*). Furthermore, it supports a generalization where multiple agents interact in parallel in a synchronous/asynchronous way, *Synch/Asynch Multi-agent ASMs*.

Based on [5], an ASM can be defined as the tuple:

(*header*, *body*, *main rule*, *initialization*)

The *header* contains the *name* of the ASM and its *signature*⁴, namely all domain, function and predicate declarations. Function are classified as *derived* functions, i.e. those coming with a specification or computation mechanism given in terms of other functions, and *basic* functions which can be *static* (never change during any run of the machine) or *dynamic* (may change as a consequence of agent actions or *updates*).

³ f is an arbitrary n -ary function and t_1, \dots, t_n, t are first-order terms. To fire this rule to a state S_i , $i \geq 0$, evaluate all terms t_1, \dots, t_n, t at S_i and update the function f to t on parameters t_1, \dots, t_n . This produces another state S_{i+1} which differs from S_i only in the new interpretation of the function f .

⁴ *Import* and *export* clauses can be also specified for modularization.

Dynamic functions are further classified into: *monitored* (only read, as events provided by the environment), *controlled* (read and write), *shared* (read and write by an agent and by the environment or by another agent) and *output* (only write) functions.

The *body* of an ASM consists of (static) domain and (static/derived) function definitions according to domain and function declarations in the signature of the ASM. It also contains declarations (definitions) of transition rules. The body may also contains definitions of *axioms* for invariants to assume over domains and functions of the ASM.

The (unique) *main rule* is a transition rule and represents the starting point of the machine program (i.e. it calls all the other ASM transition rules defined in the body). The main rule is *closed* (i.e. it does not have parameters) and since there are no free global variables in the rule declarations of an ASM, the notion of a move does not depend on a variable assignment, but only on the state of the machine.

The *initialization* of an ASM is a characterization of the initial states. An initial state defines initial values for domains and functions declared in the signature of the ASM. *Executing* an ASM means executing its main rule starting from a specified initial state.

A *computation* of an ASM M is a finite or infinite sequence $S_0, S_1, \dots, S_n, \dots$ of states of M , where S_0 is an initial state and each S_{n+1} is obtained from S_n by firing simultaneously all of the transition rules which are enabled in S_n .

A lightweight notion of module is also supported. An *ASM module* is an ASM without a main rule and without a characterization of the set of initial states.

In addition to its mathematical-based foundation, a general framework, the *ASMETA tool set* [14, 2], based on the Eclipse/EMF modeling platform is also available for developing, exchanging, simulating, testing and model checking ASM models.

4 Modeling Service-oriented Systems in ASMs

A *service-oriented system* is a distributed system: a system made of collection of distributed computational components (computers, software applications, devices, etc.) perceived by a user as a single system. However, compared with classical distributed systems, service-based systems are rather non predictable as many parts may be unknown at a given time. Indeed services are volatile distributed entities; they may be searched, discovered, and dynamically linked with the remained part of the system environment, and unlinked at a later moment. A business process may be provided that acts as an orchestrator, i.e. an active entity that invokes available services according to a given set of rules to meet some business requirements. A service orchestration is a composition specification showing how services are composed in a workflow.

We represent in ASM a service-based system exploiting the notion of *distributed multi-agent ASMs*. Essentially, each business participant (or partner role) has an associated ASM agent with a *program* (a set of transition rules) to execute. A service-oriented component is an ASM endowed with (at least) one agent able to be engaged in conversational interactions with other external agents by providing/requiring services to/from other (partner) service-oriented components. Moreover, in a service *assembly* component (a composite component made of other internal or external service-oriented components), an agent may act as “orchestrator” by executing (as part of its own program)

<pre> module A import STDL/StandardLibrary //domains and functions for standard data types import STDL/CommonBehavior //predefined rules for services interactions import AService //provided services (interface) import BService //required services (interface) export * //all functions and rules are exported signature: //Property shared pA: Agent -> D //D is a domain for a data type //Reference shared b: Agent -> BService //Client agent to which the component's agent will be linked to shared client: Agent -> Agent //Other user-defined domains and functions (if any) controlled rcv: Agent -> String ... definitions: //Axioms (if any), i.e. assumptions and constraints on functions ... //Rule for the provided business function getPA in the AService interface rule r_getPA(\$a in AService, client in String) = seq ... //Do something for the client getPA(\$a,client) := ... //setting of the out business function location endseq //Other utility rules ... //Agent's program (life cycle): receive a request and handle it rule r_A () = seq r_wreceive(client(self),"getPA",rcv) r_getPA(self,rcv) //direct service invocation r_wreplay(client(self),"getPA",getPA(self,client)) endseq //Constructor rule (invokable by the container composite) macro rule r_init(\$a in AService) = ... //do initial properties settings and other </pre>	<pre> module AService import STDL/StandardLibrary export * signature: //decl. for business roles and functions signature: domain AService subsetof Agent out getPA: Prod(Agent,String) -> D ... </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Fig. 2. ASM modules for an SCA component A and its provided service interface AService.

an ASM rule capturing the behavior of the orchestration workflow. The resulting system is therefore an asynchronous multi-agent ASM that will behave accordingly to the behavior of each service (ASM agent) involved in. This main ASM also provides the necessary initialization (such as appropriated component *bindings*) and initial startup of all agents' programs (in the main ASM rule) to make the system model executable.

4.1 Service-oriented Components and their Assemblies

A transformational semantic mapping is provided to transform SCA descriptions of service structures into ASM-based formal descriptions. Listings in Fig. 2 and in Fig. 3 report the templates of an ASM module corresponding to an SCA component (like the component A in Fig. 1) with its provided service interface (like the AService interface provided by the A component) and to an SCA composite (like the composite C in Fig. 1), respectively, using the AsmetaL notation of the ASMETA toolset.

Appropriate transformation rules map the SCA key modeling elements (components, properties, services, references, wires, and composites) into ASM concepts. Essentially, an SCA service-oriented *component* is mapped into an ASM *module* endowed with (read: provides a type declaration for) at least one agent able to interact with other

```

module C
import A,B //import of ASM modules for subcomponents
export *
signature:
//Agents of the sub-components
static compA: AService
static compB: BService
//Properties
shared pA: D //D is a domain for a data type
...
shared ext: Agent //external reference
shared client: Agent -> Agent //Client agent to which the component's agent A will be linked to
//Other user-defined domains and functions (if any)
...
definitions:
... //Axioms (if any), i.e. assumptions and constraints on functions
//Constructor rule
rule r_init =
  par
    //wires setting
    client(compA) := client
    b(compA) := compB
    ...
    //Properties setting
    pA(compA) := pA
    ...
    //Agents program assignment
    program(compA) := r.A()
    program(compA) := r.B()
    //execution of agents initialization routines
    r_init(compA)
    r_init(compB)
  endpar

```

Fig. 3. ASM template for an SCA composite C.

external service-oriented components. Each service-oriented component with its business role has, therefore, an associated ASM and an ASM agent with a *program* to execute. An SCA component's *property* is straightforwardly mapped into an ASM function. An *interface* is a description of business functions. Services and references of a component are typed by interfaces. An interface is mapped into an ASM module containing only a collection of declarations of signature elements (domains and functions) for the business roles, declared in terms of subdomains of the predefined ASM *Agent* domain, and business functions, declared as parameterized ASM *out* functions. This ASM module is imported (through *import clauses*) by both the “provider” ASM module and the “requester” ASM module in order to “provide” (by giving definitions for those elements), respectively to “require” (by exposing an explicit reference typed by the declared agent subdomain), the declared business functions.

The ASM module A shown in the left of Fig. 2 (corresponding to the component A in Fig. 1), for example, provides definitions for the business functions declared in the imported *AService* ASM module (corresponding to the provided *AService* interface) shown in the right of Fig. 2. The A module also provides declarations for the property *pA*, the reference *b* to a *BService* agent, a reference *client* to a generic client agent, and other functions. The agent domain *AService* declared in the *AService* module and the rule *r_A* characterize the agent associated to the component A.

The notion of *service* operation provided by a component is captured by a named ASM *turbo rule*. It models the notion of submachine computation in a black-box view,

hiding the internals of the subcomputation by compressing them into one step. The name of such a rule – it is a convention – is the same name of the out business function declared in the typing service interface. In case of a return value, the body of such a rule must contain, among other things, an update of such out function (location); the value of such location denotes the value to be returned to the client. See, e.g., the `r_getPA` rule in the ASM module `A` in Fig. 2 and the occurrence within it of the business function `getPA` (declared in the `AService` module) on the left-side of an update-rule.

Services can be accessed through *references* in SCA. These are abstract access endpoints to services that will be possibly discovered at runtime. In the ASMs, references are represented in terms of functions that have as codomain a subset of the *Agent* domain named with the name of the reference’s typing interface (see, e.g., the reference `b` to a `BService` agent in the ASM module `A` in Fig. 2). This domain is declared in the ASM module corresponding to the reference’s typing interface, and the ASM module corresponding to the component exposing the interface has also to import the ASM module for the interface. In this way we identify (even if it is not known at design time) the partner’s business role (i.e. the agent type).

An SCA *composite* component (made of an assembly of components) is represented by a composite ASM module that embeds (through import clauses) the ASM modules corresponding to the sub-components of the SCA composite. Communication links between components are denoted in SCA by appropriated *wires* as configured by the assembly. These links are created in the initial state or in an initialization (constructor) rule of the ASM corresponding to the assembly component in terms of function (reference) assignments. The ASM module `C` shown in Fig. 3 (corresponding to the composite `C` in Fig. 1), for example, imports the ASM modules for the sub-components `A` and `B`, and declares two references `compA` and `compB` to the agents of the subcomponents. It also carries out in the constructor rule `r_init` the wires setting, properties setting, agents’ program assignment, and initialization of the sub-components. A “top-level” composite containing the overall assembly is mapped into composite ASM (read: the *main ASM*) with a possible ASM initial state to initialize the ASM modules and their agents as dictated by the configured sub-components.

We abstract from the SCA notion of *binding*, i.e. from several access mechanisms used by services and references (e.g. WSDL binding, JMS binding, etc.). We assume that components communicate over the communication links through an abstract asynchronous and message-oriented mechanism (see next subsection), where a message encapsulates information about the partner link and the referenced service name and data.

4.2 Service Behavior: Orchestration and Interactions

The behavior of a service-oriented system is the description of the involved service activities composed in a workflow (orchestration). Here we adopt a simple service composition technique. We compose services by embedding more than one service component into a top-level composite component (the main ASM). A component embeds an ASM agent executing (as its own program) an appropriate interactive behavior or a “piece” of orchestration workflow. The overall orchestration is, therefore, spread throughout the

internal components⁵ and consists of the patterns of interactions (or communication).

For modeling service orchestration, basic control-flow constructs are easily supported in the ASMs by rule constructors such as the *seq-rule* for executing activities sequentially, the *par-rule* for synchronous parallel split of activities, the *conditional rule* for alternative flows, etc.. Other control flow patterns (not reported here) can be easily supported in ASM as formalized in [7]. For example, the “fork” and “merge” nodes (using the same terminology of the UML activity diagrams) can be used separately; a fork node is to be intended as an *asynchronous parallel split* [7] that spawns finitely many sub-agents using as underlying parallelism the concept of asynchronous ASMs. As another example, the *choice rule* can be used to define non deterministic selection patterns [7]. Moreover, more complicated workflow patterns like those introduced in the recent OMG initiative *Business Process Management Notation (BPMN)*[24] on business process modeling can be captured by ASM rule-patterns as well (some formalization work for BPMN has been already done; see for example [8]).

In addition to control-flow patterns, we define three basic kinds of service activities:

- (i) *functional activities*: they deal with data manipulation (assignments);
- (ii) *fault activities*: they deal with faults or exceptions, and error recovery (by compensation or exception handlers);
- (iii) *communication or interaction activities*: they deal with message exchange between services to interact. Activities (i) and (ii) do not require a special treatment as they can be intuitively captured by means of ASM rules with no special rule constructor or rule patterns. *Compensation handlers* can be, for example, specified in terms of named ASM rules associated to certain services to be executed in case of faults. Communication activities (iii) deserve more explanation, as better explained below.

Services are invoked (i.e. interact) through communication activities. To this purpose, we take advantage of the precise high-level models for eight fundamental service interaction patterns, given by Barros and Boerger in [3] in terms of the ASMs. They define turbo ASM rules $SEND_s$, $RECEIVE_t$, $SENDRECEIVE_{s,t}$ and $RECEIVESEND_{s,t}$ to capture the semantics of both asynchronous and synchronous message passing (the non-blocking and blocking mode) and the semantics of service interactions beyond simple request-response sequences by involving acknowledgment, resending, etc. All these variants are denoted by parameters $s \in \{noAck, ackNonBlocking, ackblocking, noAckResend, ackNonBlockingResend, ackBlockingResend\}$ and $t \in \{blocking, buffer, discard, noAckBlocking, noAckBuffer, ackBlocking, ackBuffer\}$.

Therefore, we capture the semantics of common interaction actions *send*, *receive*, *send&receive*, and *replay* by the following ASM submachines (turbo rules):

- $WSEND_{noAck}(lnk, op, snd)$: sends data *snd* without blocking to the partner link *lnk* in reference to the service operation *op*.
- $WRECEIVE_{noAckBlocking}(lnk, op, rcv)$: receives data in the location *rcv* from the

⁵ For the specification of the externally visible behavior of service components as provided to or required from a partner, some proposals (such as [23]) adopt a (declarative) *Protocol State Machine* formalism to specify which interaction a component can be engaged in which state and under which condition. Similarly, in ASM the (unknown) behavior of an external required component may be captured by a class of ASMs, named *control-state ASMs*, that specifies in an abstract way the external partner agent’s life cycle when engaged in service interactions.

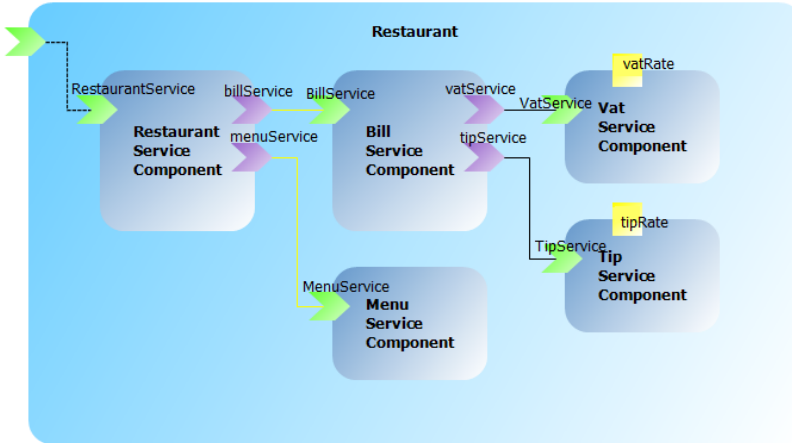


Fig. 4. SCA structure of the Restaurant case study.

partner link lnk in reference to the service operation op ; it blocks until data are received.

- $WREPLAY_{noAck}(lnk, op, snd)$: returns some data snd to the partner link lnk , as response of a previous op request received from the same partner link.

- $WSENDRECEIVE_{noAck, noAckBlocking}(lnk, op, snd, rcv)$: in reference to the service operation op , some data snd are sent to the partner link lnk , then the action waits for data to be sent back, which are stored in the receive location rcv .

These submachines have been already defined in [26] as “wrappers” of the general patterns originally presented in [3]. Each of these communication rules describes one side of the interaction and relies on a dynamic domain *Message* that represents message instances managed by an abstract message passing mechanism.

Note that additional communication patterns can be supported in ASM (e.g. for *multi-party interactions*) as specializations of the more abstract patterns formalized in [3], allowing, therefore, more expressiveness in the service interactions specification.

5 Running Case Study

Fig. 4 shows the SCA assembly of the *Restaurant case study* taken from the SCA distribution [21]. The *Restaurant* composite is a composition of five components: *RestaurantServiceComponent* that allows a client to see the menus proposed by the restaurant and also to compute the bill for a particular menu; *MenuServiceComponent* that provides different menus; a *Menu* (as data type) is defined by a description and the price without taxes; *BillServiceComponent* that computes the price of a menu with the different taxes; *VATServiceComponent* that computes the VAT (Value Added Tax); and *TipServiceComponent* that computes the tip.

As example, Fig. 5 reports the ASM module for the *BillServiceComponent*. The bill agents’ program and the service rule are a small orchestration example for the coordination of the two helper services VAT and Tip.

```

module BillServiceComponent
import STDL/StandardLibrary
import STDL/CommonBehavior
import BillService //provided interface
import TipService //required interface
import VatService //required interface
export *
signature:
shared vatService: Agent -> VatService //reference
shared tipService: Agent -> TipService //reference
shared clientBillService: Agent -> Agent //Client agent to which the component is linked to
//Other functions used for internal computations
controlled priceWithTaxRate: Agent -> Real
controlled priceWithTipRate: Agent -> Real
controlled menuprice : Agent -> Real
definitions:
//Rule for the provided service operation getBill
rule r_getBill($a in Agent, $menuPrice in Real) =
seq
r_wsndreceive(vatService($a),"getPriceWithVat",$menuPrice,priceWithTaxRate($a))
r_wsndreceive(tipService($a),"getPriceWithTip",priceWithTaxRate($a),priceWithTipRate($a))
getBill($a,$menuPrice) := priceWithTipRate($a) //setting of the out business function location
endseq
rule r_BillServiceComponent == //Agent program (life cycle)
seq
r_wreceive(clientBillService(self),"getBill",menuprice(self))
r_getBill(self,menuprice(self)) //direct service invocation
r_wreply(clientBillService(self),"getBill",getBill(self,menuprice(self)))
endseq
//Constructor rule
macro rule r_init($a in BillService) = skip //do nothing

```

Fig. 5. ASM module of the BillServiceComponent.

6 Related Work and Future Directions

On the formalization of the SCA component model, some previous works, like [9, 10] to name a few, exist. However, they do not rely on a practical and executable formal method like ASMs. In [18], an analysis tool, *Wombat*, for SCA applications is presented; their approach is similar to our as their tool is used to perform simulation and verification tasks by transforming each SCA module into one composed Petri net. We are, however, not sure that their methodology scales effectively to large systems.

Lightweight visual notations for service modeling have been proposed such as the OMG SoaML UML profile [20]. The SoaML profile, like the SCA initiative, is more focused on architectural aspects of services.

Another UML extension for service modeling, named UML4SOA [23], has been developed within the EU project SENSORIA [19]. The UML4SOA language is fo-

cused on modeling service orchestrations as an extension of UML2 activity diagrams. In order to make UML4SOA models executable, some code generators for low level target languages (such as BPEL/WSDL, Jolie, and Java) already exist [22]; however the target languages do not provide the same rigor and preciseness of a formal method necessary for early design exploration and analysis.

Within the EU project SENSORIA, another modeling notation specific to the SOA domain, named SRML [25], has been developed. SRML is a declarative modeling language for service-oriented systems with a computation and coordination model. We believe it is worth to study the feasibility of defining an encoding from UML4SOA⁶ (or SRML) into ASMs, but we leave it as a challenge for future work. The goal of this activity would be the definition of an executable operational semantics of UML4SOA (SRML) models in terms of the ASMs and then explore ASM-based analysis tools.

Several *process calculi* for the specification of SOA systems have been designed (see, e.g., [17, 15, 16, 4]). They provide linguistic primitives supported by mathematical semantics, and verification techniques for qualitative and quantitative properties. In particular, in [11] an encoding of UML4SOA in COWS (Calculus for the Orchestration of Web Services), a recently proposed process calculus for specifying services while modeling their dynamic behavior, is presented. Compared to these notations, the ASMs have the advantage to be executable and formal without mathematical overkill.

Within the ASM community, the ASMs have been used in the SOA domain for the purpose of formalizing business process modeling languages and middleware technologies related to web services, like [8, 7, 12, 1] to name a few. Some of these previous formalization efforts are at the basis of our work.

As future work, we propose to complete the proposed ASM-based service-oriented component model towards different directions. We have been developing several case studies, some taken from the SCATuscany distribution and some other from the EU SENSORIA project [19], in order to assure the approach scales effectively to large and different systems. We have been also extending the Eclipse-based SCA Tools and exploiting the Tuscany runtime that allows extension modules to be plugged in, to provide a direct support of the ASM-based component model and automate the transformation from SCA to ASMs. We aim also at defining and developing synthesis patterns to generate code automatically (at least for some critical parts) from ASM models of services.

We plan to revise our component model (if necessary) to take in consideration also the changes recently made to the SCA Assembly specification [21] to introduce some extensions for Event Processing. Moreover, since service-oriented components can be discovered and bound to other components at run-time to produce configurations, we want to address the behavioral aspects of *service discovery* (for the lookup of service provider interfaces and service locations) and *self-adaptability* by extending the service-oriented component model in ASMs with specific “roles” of service agents.

In the future, we aim also at specifying and reasoning about “classes of properties” of services through the ASMETA analysis tools, for example, to verify the compatibility of the assembled components and check that the services resulting from a composition meet desirable properties without manifesting unexpected behaviors.

⁶ Such an encoding would be natural to carry out for the UML4SOA since we also inspired from the UML4SOA communication activities for our interaction patterns.

References

1. M. Altenhofen, A. Friesen, and J. Lemcke. Asms in service oriented architectures. *J. of Universal Computer Science*, 14(12):2034–2058, 2008.
2. The ASMETA tooset website. <http://asmeta.sf.net/>, 2006.
3. Alistair P. Barros and Egon Börger. A compositional framework for service interaction patterns and interaction flows. In *ICFEM'05 Proc., LNCS 3785*, pages 5–35. Springer, 2005.
4. M. Boreale, R. Bruni, R. De Nicola, and M. Loreti. Sessions and pipelines for structured service programming. In *FMOODS Proc., LNCS vol. 5051*, pages 19–38. Springer, 2008.
5. E. Börger and R. Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer Verlag, 2003.
6. Egon Börger. The ASM method for system design and analysis. A tutorial introduction. In *Frontiers of Combining Systems, 5th International Workshop, FroCoS 2005 Proc., LNCS vol. 3717*, pages 264–283. Springer, 2005.
7. Egon Börger. Modeling Workflow Patterns from First Principles. In C. Parent, K.-D. Schewe, V. C. Storey, and B. Thalheim, editors, *ER, LNCS vol. 4801*, pages 1–20. Springer, 2007.
8. E. Brger, O. Srensen, and B. Thalheim. On defining the behavior of or-joins in business process models. *J. of Universal Computer Science*, 15(1):3–32, 2009.
9. Zuohua Ding, Zhenbang Chen, and Jing Liu. A rigorous model of service component architecture. *Electr. Notes Theor. Comput. Sci.*, 207:33–48, 2008.
10. Dehui Du, Jing Liu, and Honghua Cao. A rigorous model of contract-based service component architecture. In *CSSE (2)*, pages 409–412. IEEE Computer Society, 2008.
11. F. Tiezzi F. Banti, R. Pugliese. Automated verification of UML models of services. Submitted for publication, 2009.
12. R. Farahbod, U. Glässer, and M. Vajihollahi. A formal semantics for the business process execution language for web services. In Savitri Bevinakoppa, Luís Ferreira Pires, and Slimane Hammoudi, editors, *WSMDEIS*, pages 122–133. INSTICC Press, 2005.
13. A. Gargantini, E. Riccobene, and P. Scandurra. Model-driven language engineering: The ASMETA case study. In *Int. Conf. on Software Engineering Advances, ICSEA 2008*.
14. Angelo Gargantini, Elvinia Riccobene, and Patrizia Scandurra. A metamodel-based simulator for ASMs. In Andreas Prinz, editor, *14th Int. ASM Workshop Proc.*, 2007.
15. C. Guidi et al. : A calculus for service oriented computing. In Asit Dan and Winfried Lamersdorf, editors, *ICSOC, LNCS 4294*, pages 327–338. Springer, 2006.
16. I. Lanese, F. Martins, V. Thudichum Vasconcelos, and A. Ravara. Disciplining orchestration and conversation in service-oriented computing. In *SEFM*, pages 305–314. IEEE, 2007.
17. A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. In *LNCS*, pages 33–47. Springer, 2007.
18. Axel Martens and Simon Moser. Diagnosing sca components using wombat. In *Business Process Management Proc., LNCS 4102*, pages 378–388. Springer, 2006.
19. EU project SENSORIA, ist-2 005-016004 www.sensoria-ist.eu/.
20. OMG. The SoaML Profile, ptc/2009-04-01
21. OSA. Service Component Architecture (SCA) www.osoa.org.
22. P. Mayer, A. Schroeder, and N. Koch. A model-driven approach to service orchestration. In *IEEE SCC (2)*, pages 533–536. IEEE, 2008.
23. P. Mayer et al. The UML4SOA Profile. *Tech. Rep., LMU Muenchen*, 2009.
24. OMG, Business Process Management Notation (BPMN). www.bpmn.org/, 2008.
25. SRML: A Service Modeling Language. <http://www.cs.le.ac.uk/srml/>, 2009.
26. E. Riccobene and P. Scandurra. An ASM-based executable formal model of service-oriented component interactions and orchestration. Workshop on Behavioural Modelling - Foundations and Application (BM-FA 2010), ACM DL Proc. ISBN 978-1-60558-961-9

Optimizing Service Selection for Probabilistic QoS Attributes

Ulrich Lampe, Dieter Schuller, Julian Eckert and Ralf Steinmetz

Multimedia Communications Lab (KOM)
Technische Universität Darmstadt
Rundeturmstr. 10, 64283 Darmstadt, Germany
{firstname.lastname}@KOM.tu-darmstadt.de

Abstract. The *service selection problem* (SSP) – i.e., choosing from sets of functionally equivalent services in order to fulfill certain business process steps based on non-functional requirements – has frequently been addressed in literature considering deterministic values for the Quality of Service (QoS) attributes. However, the usage of deterministic values does not reflect the uncertainty about the actual value of an attribute during execution, thus ignoring the risk of QoS violations. In the paper at hand, a simulative step, based on stochastic QoS attributes, is performed as complement for optimally solving the SSP using linear programming methods. With this two-step approach, uncertainties in the selected set of services can be explicitly revealed and addressed through repeated selection steps, thus allowing to prevent the violation of QoS restrictions much more effectively.

1 Introduction

In Service-oriented Architectures (SOA), business processes can be realized by composing loosely coupled services. Depending on their granularity, these services provide a more or less complex functionality [1]. Thereby, the services are not necessarily located only within the boundaries of the own enterprise. In the *Internet of Services*, multiple service providers offer their services at various service marketplaces [2]. If services with substitutable functionalities are available at different cost and quality levels, service requesters have the opportunity to decide which services from which service providers to select, based on their preferences regarding Quality of Service (QoS). This *service selection problem* (SSP) respectively its solution recently attracted a lot of attention in the literature [3–6].

In this problem, an abstract representation of a workflow is assumed to be given (e.g., in *Business Process Modeling Notation* – BPMN), as well as a list of functionally equivalent services which are able to accomplish the tasks of the respective workflow steps. The aim is to assign each workflow step exactly one service from the respective set of functionally equivalent candidate services, so that the overall (workflow) QoS is optimized and the requesters' end-to-end QoS requirements are satisfied. In order to compute an (optimal) solution, almost exclusively deterministic values for the QoS attributes are considered at planning time in the literature. However, these values do not reflect the uncertainty that is associated with an attribute during execution. E.g., response times – i.e. the elapsed time period between the service invocation to the response arrival

– may fluctuate due to varying network or computational load, thus resulting in a violation of the requester’s QoS requirements in the actual workflow execution.

Therefore, we propose to perform an additional simulation step that takes stochastic distributions for the QoS attributes into account after having computed the optimal solution to the SSP (considering only deterministic values). This simulation step allows to detect potential violations of QoS restrictions in the actual execution, based on the respective probability of such events. Depending on the requester’s preferences, the outcome of the simulation may trigger repeated optimization steps using additional restrictions. As a proof-of-concept, we implemented and evaluated a simulation for the QoS attribute *response time*.

The remainder of this work is structured as follows: In Section 2, we will present our approach for optimally solving the SSP using linear programming, based on deterministic QoS values. In Section 3, the potential drawbacks of deterministic optimization will be outlined. Based on the findings, a simulation process that relies on stochastic QoS attributes will be presented and evaluated using a prototypical tool. The paper closes with a conclusion and an outlook of our future work in Section 4.

2 Optimal Service Selection for Complex Workflows

In this section, we present our approach for the computation of an optimal solution to the SSP. For this, we formulate a linear optimization problem, which can be solved optimally – if a solution exists – using (mixed) integer linear programming (MILP) techniques from the field of operations research [7]. The optimization problem consists of a target function and a set of constraints. We perform a worst-case analysis – instead of an average-case analysis – by applying our aggregation functions proposed in [8] in order to make sure that all restrictions are satisfied at planning time. Performing an average-case analysis would have led to a solution, where the restrictions are satisfied only *in average*.

For the optimization, we consider the QoS attributes response time e (elapsed time from the service invocation until the response arrival), costs c (costs for the invocation of a service), reliability r (the probability that the service successfully provides the requested results), and throughput d (number of parallel service invocations), although the mentioned simulation step will only be performed for response time e . With these QoS attributes – in fact with a subset of these attributes – the aggregation types summation, multiplication and the min/max operator are covered. The integration of further aggregation types is straightforward.

In the paper at hand, we concentrate on the workflow patterns sequence, parallel split (AND-split), synchronization (AND-join), exclusive choice (XOR-split), simple merge (XOR-join), and arbitrary cycles (Loop), which only form a subset of all workflow patterns (cf. [9]). The patterns can be combined to create complex workflows. An example for such a complex workflow is given in Figure 1.

We consider an abstract workflow (e.g., in BPMN), consisting of n tasks respectively process steps PS_i . For each PS_i with $i \in I = \{1, \dots, n\}$, a set J_i of m_i services $j_i \in J_i = \{1, \dots, m_i\}$, able to realize PS_i , exists. Each process step PS_i thereby is realized by exactly one service j_i . This is indicated by the demand for (binary) decision

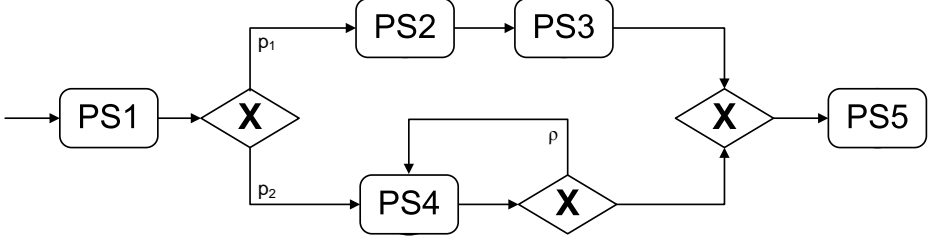


Fig. 1: Example abstract workflow.

variables $x_{ij} \in \{0, 1\}$ (cf. condition (14)). The logical order of the process steps is depicted from the abstract workflow as follows: in case PS_k is a direct successor of PS_i , we add $PS_i \rightarrow PS_k$ to a set $DS = \{PS_i \rightarrow PS_k | PS_k \text{ direct successor of } PS_i\}$. DS_s is the set of *start* tasks, i.e., the tasks that need to be executed first in the workflow. In addition, we define DS_e as the set of *end* tasks, i.e., tasks with no direct successor. To give an example, we refer to Figure 1. Here, PS_3 is a direct successor of PS_2 . We therefore add $PS_2 \rightarrow PS_3$ to DS .

With respect to XOR-splits and XOR-joins, we define a set $L = \{1, \dots, o\}$ of o path numbers for the paths within the XOR-split and -join – and name these paths *XOR-paths*. Thereby, $l \in L$ represents the respective XOR-path number. The process steps PS_{i_l} within an XOR-path are assigned to a set W_l , $PS_{i_l} \in W_l = \{PS_{i_l} | PS_{i_l} \text{ in XOR-path } l\}$, and their respective process step numbers i_l are assigned to the set IW_l , $i_l \in IW_l = \{i | PS_i \in W_l\}$. Further, $S = \{PS_1, \dots, PS_n\} \setminus (W_1 \vee \dots \vee W_o)$ represents a set of the remaining process steps PS_i when removing process steps PS_{i_l} from a set of *all* process steps. $IS = I \setminus (IW_1 \vee \dots \vee IW_o)$ denotes the set of the corresponding process step numbers.

Within an XOR-path, we assume a sequential arrangement of the process steps and label the first and last process steps with $PS_{i_l}^1$ and $PS_{i_l}^e$. The respective start times for these process steps are labeled analogously with $t_{i_l}^1$ and $t_{i_l}^e$. The probability that XOR-path l is executed, is indicated by p_l . We demand $\sum_{l=1}^o p_l = 1$.

Regarding the workflow pattern Loop, I_{loop} represents the set of process step numbers i with a Loop. Further, ρ_i denotes the respective probability that this Loop is followed (cf. PS_4 in Figure 1). Thereby, ρ is independent of whether the Loop was followed or not before. If a Loop is followed multiple times, the respective process steps are executed multiple times, too. As this affects the regarded, aggregated QoS values, we define e_{ij}^* in (1), c_{ij}^* in (2), and r_{ij}^* in (3) in dependence of a boundary value consideration of ρ (cf. [8]). The throughput d_{ij} is not effected by a Loop.

$$e_{ij}^* := \begin{cases} \frac{1}{1-\rho_i} e_{ij} & , \text{ if } i \in I_{loop} \\ e_{ij} & , \text{ else} \end{cases} \quad (1)$$

$$c_{ij}^* := \begin{cases} \frac{1}{1-\rho_i} c_{ij} & , \text{ if } i \in I_{loop} \\ c_{ij} & , \text{ else} \end{cases} \quad (2)$$

$$r_{ij}^* := \begin{cases} \frac{(1-\rho_i)r_{ij}}{1-\rho_i r_{ij}} & , \text{ if } i \in I_{loop} \\ r_{ij} & , \text{ else} \end{cases} \quad (3)$$

Based on our aggregation functions in [8], we propose Model 1 to perform the proposed worst-case analysis. Here, QoS restrictions are labeled with b (bounds).

Model 1: Optimization Problem.

Objective Function

$$\text{minimize } F(x) = \sum_{i \in I} \sum_{j \in J_i} c_{ij}^* x_{ij} \quad (4)$$

s.t.

$$t_i = 0 \quad \forall i \in I | PS_i \in DS_s \quad (5)$$

$$t_i + \sum_{j \in J_i} e_{ij}^* x_{ij} \leq t_k \quad \forall i \in I | PS_i \rightarrow PS_k \in DS \quad (6)$$

$$t_i + \sum_{j \in J_i} e_{ij}^* x_{ij} \leq b_e \quad \forall i \in I | PS_i \in DS_e \quad (7)$$

$$\max_{l \in L} \{ (t_{i_l}^1 + \sum_{i \in IW_l} \sum_{j \in J_i} e_{ij}^* x_{ij}) \} \leq t_k \quad \forall i \in I | PS_{i_l}^e \rightarrow PS_k \in DS \quad (8)$$

$$\max_{l \in L} \{ (t_{i_l}^1 + \sum_{i \in IW_l} \sum_{j \in J_i} e_{ij}^* x_{ij}) \} \leq b_e \quad \forall i \in I | PS_{i_l}^e \in W_l \quad (9)$$

$$\sum_{i \in IS} \sum_{j \in J_i} c_{ij}^* x_{ij} + \max_{l \in L} \{ \sum_{i \in IW_l} \sum_{j \in J_i} c_{ij}^* x_{ij} \} \leq b_c \quad (10)$$

$$\left(\prod_{i \in IS} \sum_{j \in J_i} r_{ij}^* x_{ij} \right) \cdot \left(\min_{l \in L} \left\{ \left(\prod_{i \in IW_l} \sum_{j \in J_i} r_{ij}^* x_{ij} \right) \right\} \right) \geq b_r \quad (11)$$

$$\min_{i \in IS} \left\{ \min_{j \in J_i} \left\{ \sum d_{ij} x_{ij} \right\} \right\}, \min_{l \in L} \left\{ \min_{i \in IW_l} \left\{ \sum_{j \in J_i} d_{ij} x_{ij} \right\} \right\} \geq b_d \quad (12)$$

$$\sum_{j \in J_i} x_{ij} = 1 \quad \forall i \in I \quad (13)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J_i \quad (14)$$

Regarding Model 1, it has to be noted that the workflow patterns AND-split and AND-join are already covered in (8) to (12) (cf. [8]).

To compute an optimal solution using MILP techniques, a *linear* optimization problem is required. As the min/max operator as well as the multiplication are non-linear aggregation types regarding the decision variables x_{ij} , we apply the approximation (15) to (11) – which is very accurate for values z_{ij} close to 1 (like reliability) [10] – and exchange constraints (8)–(12) for (16)–(20). To explain this (second adaptation step), it has to be noted that if the minimum (maximum) of a set of values has to be higher (lower) or equal to a certain bound, each element of this set needs to satisfy this constraint.

$$\prod_{i=1}^n \sum_{j=1}^{m_i} z_{ij} x_{ij} \approx 1 - \sum_{i=1}^n \left(1 - \sum_{j=1}^{m_i} z_{ij} x_{ij} \right) \quad (15)$$

$$t_{i_l}^1 + \sum_{i \in IW_l} \sum_{j \in J_i} e_{ij}^* x_{ij} \leq t_k \quad \forall l \in L, \forall i \in I | PS_{i_l}^e \rightarrow PS_k \in DS \quad (16)$$

$$t_{i_l}^1 + \sum_{i \in IW_l} \sum_{j \in J_i} e_{ij}^* x_{ij} \leq b_e \quad \forall l \in L, \forall i \in I | PS_{i_l}^e \in W_l \quad (17)$$

$$\sum_{i \in (IS \vee IW_i)} \sum_{j \in J_i} c_{ij}^* x_{ij} \leq b_c \quad \forall l \in L \quad (18)$$

$$1 - \sum_{i \in (IS \vee IW_i)} \left(1 - \sum_{j \in J_i} r_{ij}^* x_{ij}\right) \geq b_r \quad \forall l \in L \quad (19)$$

$$\min_{i \in I} \left\{ \sum_{j \in J_i} d_{ij} x_{ij} \right\} \geq b_d \quad (20)$$

Having conducted these substitutions, an optimal solution can be obtained by applying MILP techniques.

3 Stochastic Simulation of Complex Workflows

In the previous section, we have outlined how an optimal set of services can be selected for the process steps in a complex workflow, based on given QoS constraints. Because the underlying optimization problem is solved using MILP, the usage of deterministic QoS attributes is required. These fixed values commonly represent a lower or upper bound that is guaranteed by a service provider with respect to a certain QoS attribute in terms of a Service Level Agreement (SLA).

However, the usage of deterministic values does not reflect the *uncertainty* (or risk, which we use as a synonym) that may be associated with QoS attributes. Response time, e.g., is ultimately a stochastic variable that depends on various random determinants, such as network and computational load. Consider two sets of services for the same business process, where the second set has a slightly higher average response time for each service. However, the variance in response time is much lower for the second set, e.g., due to the usage of load-balancing techniques. While the first set is optimal with respect to the objective of minimal (average) response time, it exhibits a much more fluctuating behavior with respect to this attribute. This may lead to an increased risk of exceeding certain response times threshold, which is undesired. Thus, we believe that the notion of optimality in service selection needs to regard two aspects: the average outcome of an QoS attribute as well as its fluctuation.

Accordingly, we propose to extend the representation and computation of QoS attributes in a manner that appropriately incorporates uncertainty. Our approach adapts a methodology suggested by Dawson and Dawson in the domain of project planning [11]. They introduce the notion of *generalized activity networks* [12]. Such networks consist of nodes and edges. Nodes represent activities (or tasks); edges represent precedence relationships and thus paths between the activities, where each task may have one or more incoming and outgoing incident edges. For additional details and an example, we refer to Dawson and Dawson [12]. Notably, the duration for each activity is given as stochastic distribution, rather than a deterministic value, in generalized activity networks. This is a well-known principle that has been applied in traditional planning techniques, such as PERT, which was devised in the early 1960s [13]. Furthermore, if more than one edge results from an activity, all edges are annotated with an execution probability. These execution probabilities may also be correlated between edges.

Following the findings by Schonberger [14], who states that traditional planning techniques such as PERT commonly underestimate the overall duration of an activity

network, Dawson and Dawson utilize simulation as a means of analyzing generalized activity networks [11]. I.e., the activity network is virtually executed a selected number of times; in this process, the duration of each activity and choice of path execution is drawn as a random variable. The individual durations of all executed activities are then aggregated into an overall duration in each iteration. From the distribution of aforementioned overall durations, conclusions can be drawn about the characteristic of the activity network in actual execution. Most importantly, the probability that a set of activities exceeds a certain threshold due to the fluctuations in duration can be inferred.

The notion of generalized activity networks can easily be transferred to workflows as a special application domain. In this scenario, services then correspond to activities, while splits (joins) constitute dummy activities with multiple outgoing (incoming) edges. Depending on the type of split (AND, XOR, or Loop), the execution probabilities of the edges and respective correlations will differ. E.g., in the case of AND-splits, each edge will be assigned a probability of 1, due to the fact that each edge is certainly executed.

Because services have multiple non-functional attributes, we not only adapt, but also extend Dawson and Dawson's approach. Namely, we allow for an *arbitrary* number of random variables, representing QoS attributes, being associated with each activity (i.e. service) apart from duration (which, in the context of workflows respectively services, translates into response time). In our proposed methodology, each QoS attribute for each service is modeled as an independent random variable adhering to some probability distribution. This loosely relates to the idea of *soft contracts* in Web service orchestration, as proposed by Rosario et al. [15].

The probability distribution may essentially be determined in two ways. The first option is to infer it, based on historic execution data of a service. This requires the installation of proper monitoring mechanisms. After a relevant sample has been collected, a QoS attribute such as response time may, e.g., be represented through a normal distribution. The second option is that a service provider explicitly specifies a probability distribution for each QoS attribute.

In order to infer execution probabilities for each path, three options exist. The first is mining from historical data again. However, this requires that a workflow (or at least a workflow segment) that is identical to one being simulated has previously been executed and monitored. The second option is to have an user manually assign the probabilities, based on his or her knowledge about the underlying business process. The third and final option is to utilize conservative default values, assuming that either each path (in case of AND-splits) or the worst path with respect to each individual QoS attribute (XOR-splits) will be executed.

Figure 2 depicts an example workflow for which a set of services (S1 through S5) has been selected. In addition, the random variables and respective probability distributions for each service, as well as execution probabilities for each edge, are illustrated. For reasons of simplicity, solely the random variables for the QoS attribute *response time* are included. For service S1, e.g., the response time is given by $X_{e;1}$, which is normally distributed (N) with a mean value of 6.3 seconds and a standard deviation of 1.5 seconds. For the XOR-split, the probability of executing the top and bottom path is 0.3 and 0.7 respectively. Accordingly, for the Loop construct, the probability of looping and thus repeatedly executing S4 is 0.25.

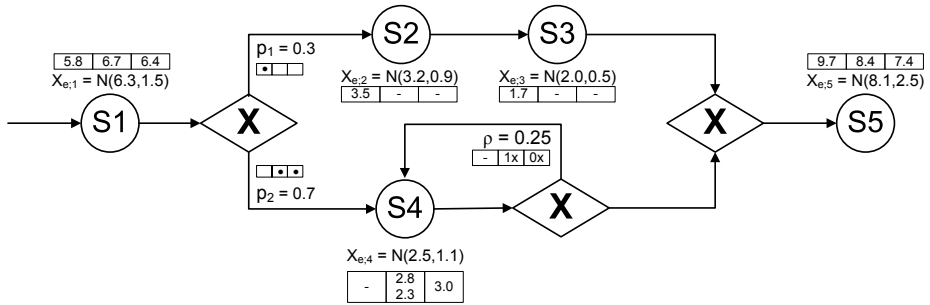


Fig. 2: Example workflow including simulation outcomes.

Figure 2 further depicts three exemplary simulation runs for the sample workflow. For every service, the randomly drawn response times are depicted in the boxes next to the random variables. For the XOR-split, the pursued path is indicated by a bullet; for the Loop construct, the number of additional executions (repetitions) of S4 is depicted. As can be seen, each run results in a different outcome for each service with respect to response time and in varying paths being executed. E.g., in the first iteration in the example, services S1, S2, S3, and S5 have response times of 5.8, 3.5, 1.7, and 9.7 seconds respectively. The lower path is not executed, and thus, S4 and the consecutive Loop construct are omitted. Accordingly, the overall response time for the first iteration is 20.7 seconds (and 20.2 and 16.8 seconds for the second and third iteration respectively). Once the process is repeated multiple times, a representative distribution for each QoS attribute can be obtained.

Service selection and workflow simulation serve as a mutual complement: In the first step, a set of services is selected by solving a linear optimization problem. This provides an optimal result with respect to the objective of minimizing total cost and allows to make statements about the workflow characteristics in theory. In the second step, the resulting workflow is simulated, ideally based on historic execution data, which allows to anticipate the workflow characteristics in practical execution. If the uncertainty in the workflow is found to be unacceptable with respect to given constraints, the selected set of services is discarded. This may, e.g., be the case if a specified response time constraint is not met with a certain probability. Consecutively, the process of computing an optimal solution is repeated with further restrictions. A manifest strategy is to explicitly exclude one or more services with the highest standard deviation in a critical QoS attribute from the set of candidate services.

To assess the principal benefits and effectiveness of our approach, we have implemented a prototypical workflow simulation tool in Java. The tool allows to specify complex workflows, consisting of services and their structure, using an XML-based format¹. For each service, an arbitrary number of QoS attributes, along with the respective probability distributions, may be specified and freely parameterized.

A simulation with one million iterations has been conducted for the example workflow in Figure 2 using the aforementioned tool. Additionally, the workflow has been

¹ A sample listing is available from <http://www.kom.tu-darmstadt.de/lampeu/icsoft-2010/workflow.xml>

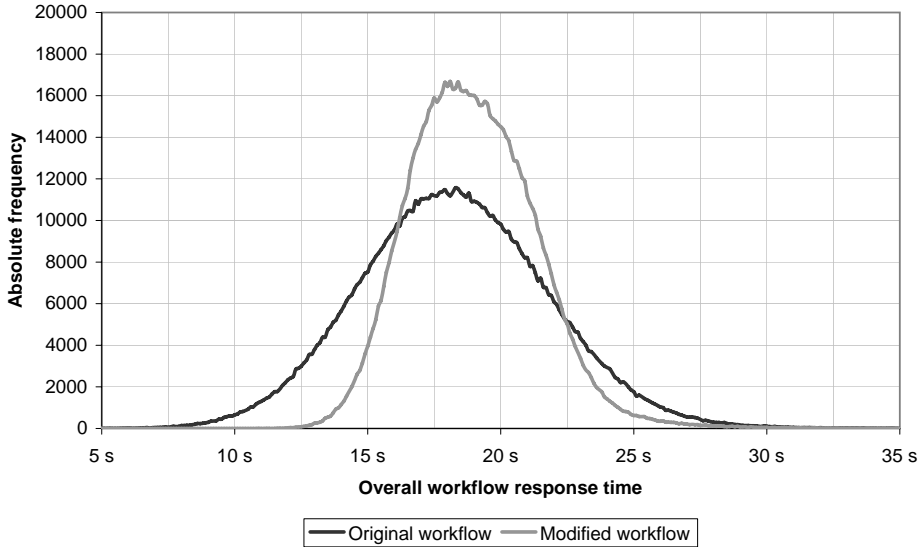


Fig. 3: Distribution of the overall response time for two workflows.

modified for a second simulation. In detail, the mean of the response time probability distribution for each service was incremented by 0.2 seconds, and the standard deviation was set to half of its original value. I.e., each initially selected service has been replaced by a variant that is less optimal on average, but also shows less fluctuation in terms of response time. In practice, this process would be iteratively conducted for one service at a time.

The resulting distributions of the workflows' overall response times are depicted in Figure 3, where the *absolute frequency* refers to clusters (or classes) of outcomes that were identical up to the first decimal place. While the modified workflow responds slower on average, it can be seen that it is significantly more favorable once a strict response time constraint of approximately 20 seconds or more has been specified. This figure is fairly close to the average response time of 18.2 and 18.9 seconds for the original and modified workflow respectively. In these cases, the original workflow is much more likely to break the constraint than the modified workflow. E.g., a response time restriction of 22.5 seconds is violated with a probability of 11.15% by the original workflow – for the modified workflow, the probability is only 6.25%, i.e. roughly half. Differently stated, an increase in average response time (and cost) is traded against a decrease in uncertainty – namely of breaking an overall response time constraint – by replacing the original services through their alternative counterparts.

4 Conclusions

In the work at hand, we have presented two complimentary approaches to the problem of QoS-aware service selection for complex workflows. As foundation, we have outlined how an optimal set of services can be identified under given QoS constraints using

linear programming. However, this process is based on deterministic values, which insufficiently reflect the uncertainty associated with a QoS attribute in actual execution. E.g., response times may heavily fluctuate due to network and computational load, thus leading to QoS violations in the actual execution of a workflow.

As a solution, we have adapted an existing methodology for the simulation of generalized activity networks to the specific field of workflows in SOA. This simulation process allows to assess the expected characteristics of a workflow, most importantly the likelihood that a QoS constraint will be violated, in more detail. Depending on a requester's preferences, the outcome of the simulation process can be utilized to repeatedly conduct the service selection procedure, thus minimizing the probability of QoS violations more effectively. The practical applicability and benefit of our approach has been proven using a prototypical implementation of a workflow simulation tool.

In our future work, we aim at combining the currently separated steps of service selection and workflow simulation into an integrated tool. We will further investigate the issue of mining probability distributions from historic service execution data as a prerequisite of more realistic simulation. In this context, QoS attributes besides response time will also be explicitly addressed.

Acknowledgements

This work has partly been sponsored by the E-Finance Lab e. V., Frankfurt am Main, Germany (<http://www.efinancelab.de>).

References

1. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall PTR, Upper Saddle River, NJ, USA (2004)
2. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson Education Limited, Harlow, England (2008)
3. Anselmi, J., Ardagna, D., Cremonesi, P.: A QoS-based Selection Approach of Autonomic Grid Services. In: International Conference on Service-oriented Computing. (2007) 1–8
4. Menascé, D.A., Casalicchio, E., Dubey, V.: A Heuristic Approach to optimal Service Selection in Service-oriented Architectures. In: Workshop on Software and Performance. (2008) 13–24
5. Mabrouk, N.B., Georgantas, N., Issarny, V.: A Semantic end-to-end QoS Model for Dynamic Service-oriented Environments. In: Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service-oriented Systems. (2009) 34–41
6. Huang, A.F.M., Lan, C.W., Yang, S.J.H.: An Optimal QoS-based Web Service Selection Scheme. Information Sciences 179 (2009) 3309–3322
7. Domschke, W., Drexl, A.: Einführung in Operations Research. Springer Verlag, Heidelberg (2007)
8. Schuller, D., Eckert, J., Miede, A., Schulte, S., Steinmetz, R.: QoS-Aware Service Composition for Complex Workflows. In: International Conference on Internet and Web Applications and Services. (forthcoming 2010)
9. van der Aalst, W.M., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
10. Heckmann, O.: A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers. PhD thesis, TU Darmstadt, Fachbereich Informatik (2004)

11. Dawson, R.J., Dawson, C.W.: Practical Proposals for Managing Uncertainty and Risk in Project Planning. *International Journal of Project Management* 16 (1998) 299–310
12. Dawson, C.W., Dawson, R.J.: Generalised Activity-on-the-Node Networks for Managing Uncertainty in Projects. *International Journal of Project Management* 13 (1995) 353–362
13. Miller, R.W.: How to Plan and Control with PERT. *Harvard Business Review* 40 (1962) 93–104
14. Schonberger, R.: Why Projects are "always" late: a Rationale based on Manual Simulation of a PERT/CPM Network. *Interfaces* (1981) 66–70
15. Rosario, S., Benveniste, A., Haar, S., Jard, C.: Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations. *Transactions on Services Computing* 1 (2008) 187–200

From i* Models to Service Oriented Architecture Models

Carlos Becerra^{2,3}, Xavier Franch¹ and Hernán Astudillo²

¹ Universitat Politècnica de Catalunya (UPC), C. Jordi Girona, 1-3 (Campus Nord, C6)
E-08034 Barcelona, Spain

franch@essi.upc.edu

² Universidad Técnica Federico Santa María, Avda. España 1680, Valparaíso, Chile
{cbecerra, hernan}@inf.utfsm.cl

³ Universidad de Valparaíso, Avda. Gran Bretaña 1091, Valparaíso, Chile
carlos.becerra@uv.cl

Abstract. Requirements engineering and architectural design are key activities for successful development of software systems. Specifically in the service-oriented development systems there is a gap between the requirements description and architecture design and assessment. This article presents a systematic process for systematically deriving service-oriented architecture from goal-oriented models. This process allows generate candidate architectures based on i* models and helps architects to select a solution using services oriented patterns for both services and components levels. The process is exemplified by applying it in a synthesis metadata and assembly learning objects system.

1 Introduction

Service-oriented architecture (SOA) is a flexible set of design principles used during the phases of systems development and integration [5]. A deployed service or architecture provides a loosely-integrated suite of services that can be used within multiple business domains. SOA defines how to integrate widely disparate applications for a world that is Web-based and uses multiple implementation platforms. Rather than defining an API, SOA defines the interface in terms of protocols and functionality.

One of the main problems facing architects of service-oriented systems is the gap between requirements description and architecture design and assessment.

This article presents a systematic process for deriving and evaluating service-oriented architectures from goal-oriented models. This process generates candidate architectures from i* [20] models and helps architects to select a solution, with the SOA patterns using. The i* models are used because: facilitates reasoning about the purpose of a proposed solution; i* models can be analyzed to demonstrate which goals realize other goals and which goals conflict or negatively contribute to other goals; demonstrates the contribution of the proposed and designed solution to the actual need [22].

The article is structured as follows: Section 2 presents related work; Section 2.2 describes the service oriented approach based on i*; Section 3 describes the service oriented architecture representation; Section 4 presents the Learning Objects (LOs) case study; Section 5 describes the service-oriented architecture generation process; and Section 6 summarizes and concludes.

2 Related Work

2.1 Requirements to Architectural Design

Several authors have proposed systematic approaches to obtain an architectural design from requirements description. Liu and Yu [9] proposed to explore the combined use of goal-oriented and scenario-based models during architectural design; the Goal-oriented Requirement Language (GRL) supports goal and agent-oriented modeling and reasoning, and the architectural design process; and Use Case Maps (UCM) are used to express the architectural design. The combined use of GRL and UCM enables the description of both functional and non-functional requirements, both abstract requirements and concrete system architectural models, both intentional strategic design rationales and non-intentional details of temporal features.

Chung et al. [10] proposed the NFR Framework, which uses Non-Functional Requirements (NFRs) as goals to systematically guide selection among architectural design alternatives; during the architectural design process, goals are decomposed, design alternatives are analyzed with respect to their tradeoffs, design decisions are made rationalized, and goal achievement is evaluated.

Brandozzi and Perry [11] proposed the use of Preskriptor, a prescriptive architectural specification language, and of its associated process, the Preskriptor process. Architectural prescriptions consist of the specification of the system's basic topology, constraints associated with it, and its components and interactions. The Preskriptor process provides a systematic way to satisfy both the functional and non-functional requirements from the problem domain, as well as to integrate architectural structures from well known solution domains.

Van Lamsweerde [12] presented a systematic incremental approach to deriving software architecture from system goals; it is grounded on the KAOS goal-oriented method for requirements engineering, with the intent of exploring the virtues of goal orientation for constructive guidance of software architects in their design task. It mixes qualitative and formal reasoning towards building software architectures that meet both functional and non-functional requirements.

Lucena et al. [13] presented an approach based on model transformations to generate architectural models from requirements models. The source and target languages are respectively the i^* modeling language and Acme architectural description language [21]. Gross and Yu [14] proposed a systematic treatment of NFRs in descriptions of patterns and when applying patterns during design. The approach organizes, analyzes and refines non-functional requirements, and provides guidance and reasoning support when applying patterns during the design of a software system.

Grau and Franch [2] explored the suitability of the i^* goal-oriented approach for representing software architectures. For doing so, they compared i^* 's representation concepts against those representable in common Architecture Description Languages and defined some criteria to close the gap among these representations. They clarified the use of the i^* constructs for modeling components and connectors: actors and dependencies provide an architecture-oriented semantics to help the process; added the notions of role, position and agent models in order to help traceability of the architectural representation; proposed adding of attributes to actors and model dependencies

to store information for later analysis; and suggested the use of structural metrics to analyze the properties of the final system.

All of these approaches offer systematic processes to derive requirements from architectural designs, but are not appropriate for service-orientation, because they do not provide guidelines to describe basic structures or interfaces between services. Several SOA specific characteristics demand a special approach to map requirements to architectural design alternatives, namely: 1) reuse, granularity, modularity, composability, componentization and interoperability; 2) standards-compliance (both common and industry-specific); 3) Services identification and categorization, provisioning and delivery, and monitoring and tracking. To our knowledge, only Estrada [1] has done so, proposing to address the enterprise modeling activity using i^* . Estrada's [1] approach is based on using business services as building blocks for encapsulating organizational behaviors, and proposes a specific business modeling method in accordance with the concept of business service. The use of services as building blocks enables the analyst to represent new business functionalities by composing models of existing services. He proposed starting activity elicitation, the actual implementations of the services offered and requested by the analyzed enterprise, are used as basis to play a very relevant role in the discover process, and for a formal definition of the basic concepts and process design SOAs. Unfortunately, this proposal only went so far as business services, and said nothing about architectural components and connectors.

2.2 Estrada's Approach Service-orientation from i^*

Estrada [1] aimed to define service-oriented architectures that address the complexity of large i^* models in real-life cases. The proposed architecture distinguishes three abstractions levels (services, process and protocols) and a methodological approach to align the business models produced at these abstraction levels.

The approach includes : a) a conceptual modeling language, based on i^* , which defines the modeling concepts and their corresponding relationships; b) a service-oriented architecture specific for the i^* models that define the service components and the modeling diagrams. c) a business modeling method to represent services at the organizational level.

The key idea of the approach is to used business services as building blocks that encapsulate internal and social behaviors. Complementary models allow to reify the abstract concept of service to low level descriptions of its implementation.

The business service architecture is described by three complementary models (see Figure 1) that offer a view of what an enterprises offers to its environment and what enterprise obtains in return:

- **Global Model.** The organizational modeling process starts with the definition of a high-level view of the services offered and used by the enterprise. The global model permits the representation of the business services and the actor that plays the role of requester and provider. In this model are defined basic and compound services.
- **Process Model.** Once business services have been elicited, they must be decomposed into a set of concrete processes that perform them. This is done with a process model that represents the functional abstractions of the business process for a

specific service; this model provides the mechanisms required to describe the flow of multiple processes.

- **Protocol Model.** Finally, the semantics of the protocols and transactions of each business process is represented in an isolated diagram using the i* conceptual constructs. This model provides a description of a set of structured and associated activities that produce a specific result or product for a business service.

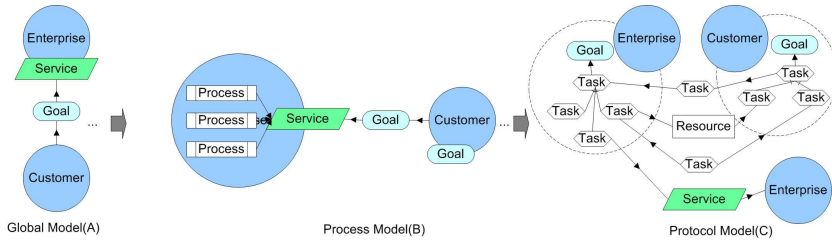


Fig. 1. A Service Oriented Approach for the i*.

The proposed approach enables the analyst to reuse the definition of protocols by isolating the description of the processes in separate diagrams. In this way, the process model represents a view of the processes needed to satisfy a service but without giving details of its implementation. Each business process is detailed through a business protocol. The detailed description of the protocols is given in the protocol model.

3 Representing Service-oriented Software Architectures

Mapping requirements to architectural design demands formalized architecture model as target, must include the notions of services, components and interfaces at different abstractions levels.

The i*-SOA Process is based on previous work by Grau and Franch [2] that defined several intentional component abstraction levels, for both services and components:

- **Service:** a set of related software functionality and the policies that control their usage. A service is accessible over standard communication protocols independent of platforms and programming languages.
- **Service Capabilities:** the operations set defined for each service [5] independently of their implementation. Therefore, this notion is especially useful during service modeling stages when the physical design of a service has not yet been determined
- **Service Components:** represents a specific component that can be integrated into the service, to implement a capability.

Connectors are described according to their abstraction level; the following types are proposed:

- **Intentional Relationships:** involve human or organizational actors and are present in the requirements models; they represent the intentional needs of the actors upon the system:
 - **Goal Dependencies:** functional requirement over the system.

- **Resource Dependencies:** flow of concepts, or a concept relevant to the domain that does not physically exist.
- **Architectural Relationships:** occur among service components or services, as follows:
 - **Service Interfaces:** describe relationships among services. The dependencies definition encapsulates (hides) the deployment properties, making it vendor-programming-language and technology-independent. Service interfaces are described with Web Services Description Language (WSDL) [7].
 - **Service Component Interfaces:** describe components relationship within a service. they are described with the notation proposed by Han [8].

Since a pattern services concept is required to apply this in different abstraction levels, Erl's [5] set of patterns is used:

- **Services Design Patterns:** functional service contexts are defined and used to organize available service logic. Within technology-independent contexts, service logic is further partitioned into individual capabilities.
- **Composition Design Patterns:** provide the means to assemble and compose together the service logic that is successfully decomposed, partitioned, and streamlined via the service definition patterns.

Based on these definitions, the i*-SOA Process models the architecture at two different levels:

- **Service Pattern View:** In this model we apply service-oriented design patterns to describe the system architecture. There are two model sub-views:
 - **Service Design Pattern View:** Shows the structure of components and connectors for each service, based on services design patterns (e.g. redundant implementation, service data replication, message screening, etc).
 - **Composition Design Pattern View:** Shows the structure and the dependencies of services that form the system under development (e.g. service messaging, service agent, asynchronous querying, etc.).
- **Services Component View:** States the different components that exist in the service architecture (i.e. specific software component that can be integrated into the service architecture and fulfill with de capabilities). This model represents the dependencies among components within a service.

4 Introducing the Case Study

The approach reusable learning content, by combining Learning Objects (LOs) [6] has emerged in educational technology and computer science research. The approach associated with the LOs delivery rigor to the educational materials development, making the content cheaper to obtain and easy to reuse. LO are educational resources designed to generate and support learning experiences. One of the main activities to be developed in this area is to prepare courses, programs and activities based on these LO. According to this idea LO can be used by different instructors and each instructor can be reused in different learning materials.

The i^* -SOA Process approach has been tried and evaluated with a Learning Object (LO) management system. The original motivation to the case study is the community need for services to improve existing LO descriptions [18, 19] and generate LO assemblies automatically. Currently, teachers and trainers have a large amount of resources (digital or not) to prepare educational materials, update their content and develop educational activities. The evolution of content distribution models from a centralized topology toward a decentralized and distributed one, has led to a scheme in which digital resources are widely and freely available. This wealth, rather than an asset, can be disadvantageous, since it adds a complexity level for users when discriminating good quality and relevant resources for specific applications.

Using LO requires collecting related information, enabling search, index and reuse. The main problem is associated with the information that user finds about a LO, which is often imperfect (imprecise, incomplete and unreliable). since many LO are not clearly classified for specific domains, search results are too general and with many possible answers list, which is not practical for users.

Thus, there are two problems to solve and whose solutions must be integrated:

- Automatically generate LO assemblies (e.g. presentations, courses, classes) from simple resources, via aggregation or composition and considering imperfect information.
- Improving LO Descriptions, gathering and synthesizing metadata from different sources.

This LO management system will be developed based on a service-oriented architecture, making available as web services the algorithms that solve the problems of LOs generation and assembly.

5 Goal Oriented Models to Service-oriented Architectural Design Process

The i^* -SOA Process extends the approach by Estrada [1] described in Section 2.2. The main objective is to derive architecture at implementation level using additional model called Deployment Model. The i^* -SOA Process original stages are also improved to give more semantic to dependencies intra- and inter- business services and processes. The i^* -SOA Process generates alternative architectures that meet user requirements.

The method has been structured into four main activities that may iterate or intertwine as needed (see Figure 2). Section 5.1 explain the alternative SOA architectures generation process using i^* -based models.

5.1 Defining the Global Model

Two complementary views of the service global model have been generated at this first phase (A).

- Abstract view of the global model: focused on representing a simple view of the offered business services (see [1]).

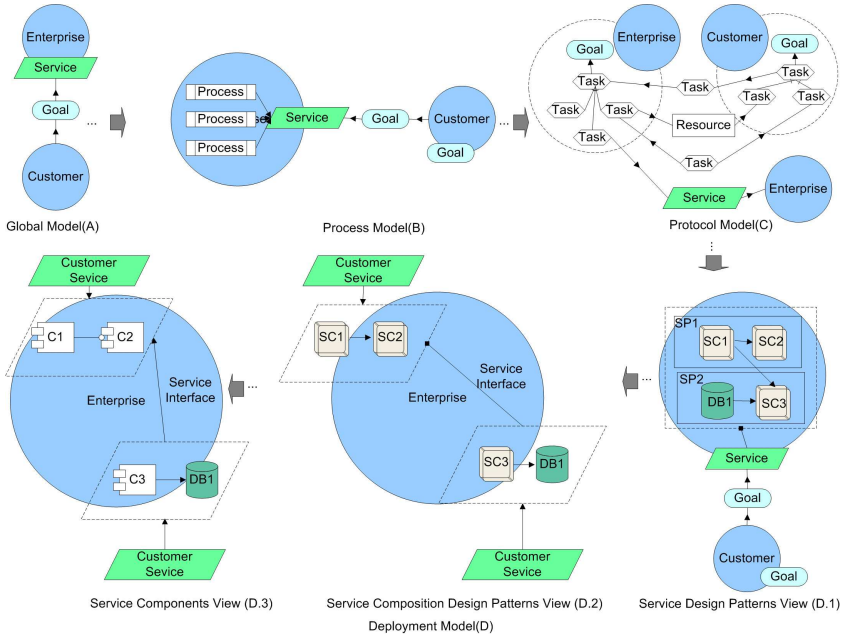


Fig. 2. The i*-SOA Process.

- Detailed view of the global model: focused on detailing the goals that are satisfied by the offered business services.

The Detailed view introduces the dependency relationships among services; specifically intentional dependencies (goal or resource dependencies) between basic services.

Figure 3 exemplifies the Global Model Detail View for the case study LO System, the main services associated with the LO Management System are:

- Learning Objects Management Service: creates new LOs descriptions from experts intentionally categorical metadata; it also allows search, update and delete of existing LOs.
- Metadata Retrieval Service: retrieves the LO descriptions dataset, to generate the initial database for the expert community.
- Metadata Synthesis Service: synthesizes and improves the LO metadata using several evidence sources.
- Learning Objects Assemblies Generation Service: using the learning objectives description provided by teachers, this service generates candidates LO assemblies that meet the requested learning objectives.

The dependencies among basic services are represented for the LO Metadata Resource Dependency and Querying LO Databases Goal Dependency.

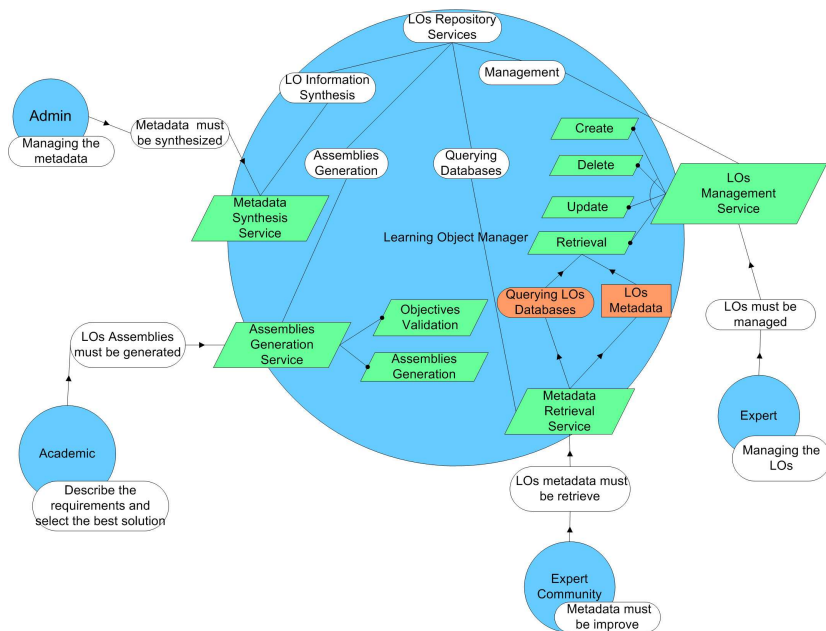


Fig. 3. Global Model Detail View.

5.2 Defining the Process Model

For each service a process model using the approach proposed in [1]. The i*-SOA Process adds the notion of dependency among processes, making necessary to specify the dependency flow and to describe the resources dependencies (resources or information). For each milestone present among processes (if required) we must specify the resource or information which helps to achieve that relationship. Figure 4 shows a LO Management Service Process Model (the resources dependencies among services processes are represented for the LO Metadata and New LO Metadata resources dependencies).

5.3 Defining the Protocol Model

The protocol model is generated based on the same process specified in [1]. Figure 5 shows a LOs Management Service Protocol Model.

5.4 Defining the Deployment Model

The method to define the deployment model has three sub-phases:

D.1: For each service identified in phase A:

- Based on the Process Model, identify the service design patterns (e.g. Figure 6 shows Contract Centralization, Contract Desnormalization, Concurrent Contract, Service Faade and Agnostic Capability Patterns applied in the Assemblies Generation Service) that fit the processes. For each pattern identified in the service, are specified the service components.

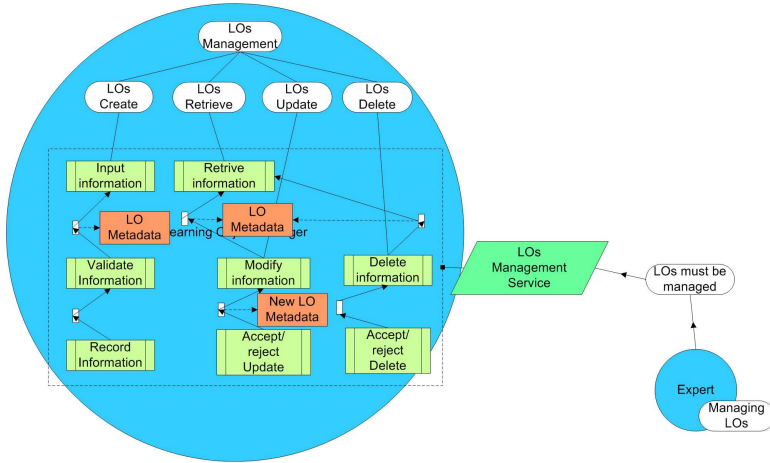


Fig. 4. LOs Management Service Process Model.

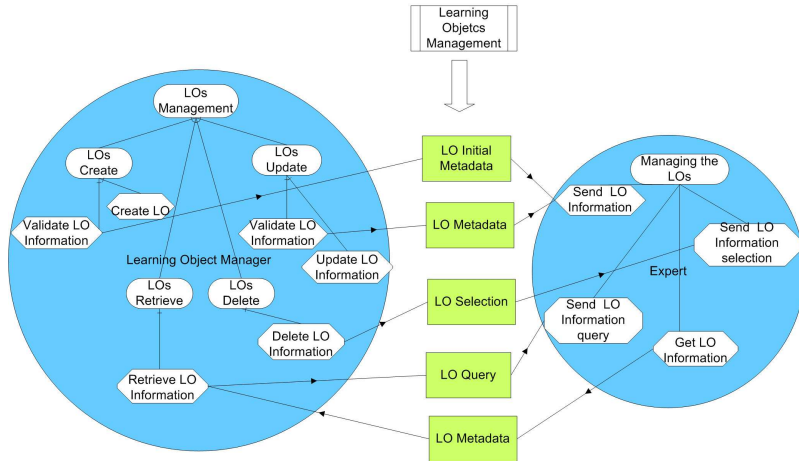


Fig. 5. LOs Management Service Protocol Model.

- For each service component specify the operations (capabilities) and the service component interfaces, which are obtained from the current Process Model activities and dependencies. Service interfaces among components are described using the notation defined in Section 3.

Identifying this pattern yields the Service Design Pattern View, which contains the service components, service components interfaces and services capabilities description for each pattern. Figure 6 shows Service Design Pattern View for one service in the running example.

D.2: Services are joined to generate the complete system architecture:

- From Service Design Pattern Views, apply service composition design patterns and structure the system, at the level of its services, services customers and services

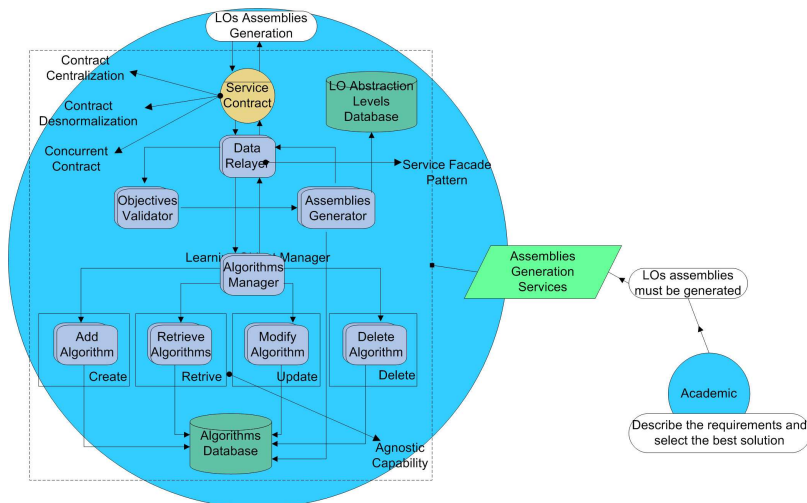


Fig. 6. LOs Assemblies Generation Service Design Pattern View.

interfaces. The interfaces among services and services customers are taken from the Protocol Model described for each service. The system service general structure and interfaces among services are taken from the Global Model. Services interfaces are described using the notation defined in Section 3.

- This sub-phase yields the Service Composition Design Pattern View. Figure 7 shows the LOs Service Composition Design Pattern View for the running example.

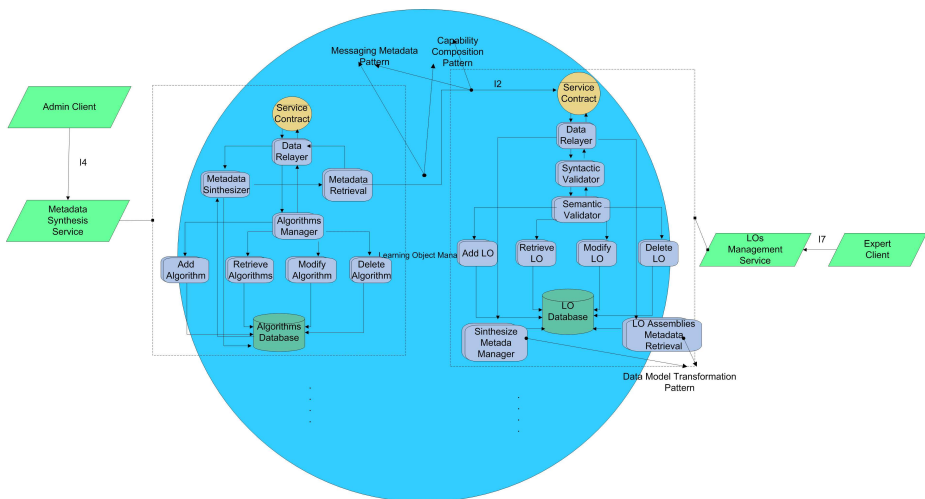


Fig. 7. LOs Service Composition Design Pattern View.

D.3: The services pattern description to specific components that implement each services capability and their interfaces. This yields the Services Component View.

6 Conclusions and Future Work

In this paper was proposed a systematic process for deriving and evaluating service-oriented architecture from goal-oriented models. This process allows to generate candidate architectures based on i^* models. The main contributions are: 1) definition of basic constructs for describing a SOA architecture using i^* ; 2) development enables derivation of service-oriented architectures from requirements description, up to a components and connectors level; 3) description of a systematic process that applies SOA patterns in the SOA design alternatives generation.

Overall, we have proposed a systematic generation method for SOA architectures, which allows mapping requirements (specified with i^*) to architectural design alternatives.

Future work will extend this proposal up to a technological solutions level, associated with the architectural design. We are also developing a method to select and evaluate SOA alternatives design, including models and metrics to generate and evaluate the solutions. We are developing automated support and/or adopting and possibly extending existing tools for this proposal, and validate the efficiency an effectiveness of this proposal with an experimental study (after and before implement an automatic support).

Acknowledgements

This work has been partially supported by the Spanish project TIN2007-64753.

References

1. Estrada, H.: "A service oriented approach for the i^* framework". Universidad Politcnica de Valencia Phd. Thesis, 2008. Thesis Director Oscar Pastor Lpez.
2. Grau, G. and Franch., X.: "On the Adequacy of i^* Models for Representing and Analyzing Software Architectures". *Advances in Conceptual Modeling Foundations and Applications*, 2007, pages 296-305.
3. Rud, D., Schmietendorf, A., Dumke, R.: "Product metrics for service oriented infrastructures". In *Proceedings of the 16th International Workshop on Software Measurement and DASMA Metrik Kongress (IWSM/MetriKon 2006)*, pp. 161-174, November 2-3, 2006, Potsdam, Germany.
4. Aier, S. and Ahrens, M., and Stutz, M., and Bub, U.: "Deriving SOA Evaluation Metrics in an Enterprise Architecture Context". *Service-Oriented Computing - ICSOC 2007 Workshops: ICSOC 2007, International Workshops, Vienna, Austria, September 17, 2007, Revised Selected Papers*, 2007.
5. Erl. T.: "SOA Design Patterns". Prentice Hall/PearsonPTR, , Upper Saddle River, NJ, USA, 2009
6. IEEE. draft standard for learning object metadata - proposed standard. Technical report, IEEE, Piscataway, 2002.
7. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer, W3C Working Draft 3 August 2005, <http://www.w3.org/tr/2005/wd-wsdl20-primer-20050803/>

8. Han, J.: "A Comprehensive Interface Definition Framework for Software Components". APSEC '98: Proceedings of the Fifth Asia Pacific Software Engineering Conference 1998, IEEE Computer Society.
9. Liu, L. and Yu, E.: "From Requirements to Architectural Design - Using Goals and Scenarios". First International Workshop From Software Requirements to Architectures (STRAW 01), 2001, Toronto, Canada.
10. Chung, L., Nixon, B., and Yu E.: "Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design". Proc. 1st Int. Workshop on Architectures for Software Systems, 1994, pp. 31-43.
11. Brandozzi, M., Perry, D.E.: "From goal-oriented requirements to architectural prescriptions: the preskriptor process". Second International Software Requirements to Architectures Workshop (STRAW'03), 2003, pp. 107-113.
12. Van Lamsweerde, A.: "From system goals to software architecture". Formal Methods for Software Architectures, 2003, pages 25-43.
13. Lucena, M., Castro, J., Silva, C., Alencar, F., Santos, E. and Pimentel, J.: "A Model Transformation Approach to Derive Architectural Models from Goal-Oriented Requirements Models". OTM '09: Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems: ADI, CAMS, EI2N, ISDE, IWSSA, MONET, OnToContent, ODIS, ORM, OTM Academy, SWWS, SEMELS, Beyond SAWSDL, and COMBEK 2009, Vilamoura, Portugal, pp. 370-380.
14. Gross, D., and Yu, E.: "From Non-Functional Requirements to Design through Patterns". Requirements Engineering, Volume 6 (1), 2001, pp. 18-36.
15. Liu, Y. and Traore, I.: "Complexity Measures for Secure Service-Oriented Software Architectures". PROMISE '07: Third International Workshop on Predictor Models in Software Engineering, 2007.
16. Qian, K., Liu, J., and Tsui, F.: "Decoupling Metrics for Services Composition". ICIS-COM SAR '06: 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse, 2006, pp. 44-47.
17. Hirzalla, M., Cleland-Huang, J., Arsanjani, A.: "A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures". ICSOC 2008 Workshops: ICSOC 2008 International Workshops, Sydney, Australia, December 1st, 2008, pp. 41-52.
18. Chan, L.M.: "Inter-Indexer Consistency in Subject Cataloging". Information Technology and Libraries, 1989. 8(4): p. 349-358.
19. Currier, S., Barton, J., O'Beirne, R., and Ryan, B.: "Quality Assurance for Digital Learning Object Repositories". Issues for the Metadata Creation Process. ALT-J, research in Learning Technology, 2004. 12(1): p. 6-20.
20. Mylopoulos, J., Chung, L., Yu, E.: "From Object-Oriented to Goal-Oriented Requirements Analysis"; Commun. ACM 42(1): 31-37 (1999).
21. Garlan, D., Monroe, R. and Wile, D.: "Acme: An Architecture Description Interchange Language"; Proceedings of CASCON97, 1997, 169-183.
22. Quartel, D.A.C., Engelsman, W., Jonkers, H., and van Sinderen, M.J. "A goal-oriented requirements modelling language for enterprise architecture". Thirteenth IEEE International EDOC Enterprise Computing Conference, EDOC 2009, 1-4 Sep 2009, Auckland, New Zealand. pp. 3-13. IEEE Computer Society Press.

WEB SERVICES COMPOSITION

An Evaluation of Dynamic Web Service Composition Approaches^{*}

Ravi Khadka¹ and Brahmananda Sapkota²

¹ Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands

² Information System Group, University of Twente, Enschede, The Netherlands
r.khadka@student.utwente.nl, b.sapkota@utwente.nl

Abstract. Web Services composition has received much interest from both the academic researchers and industry to support cross-enterprise application integration. Promising research projects and their prototypes are being developed. At the same time the web service environment is getting more dynamic as numerous web services are being published by the service providers in the Internet. To meet the users requirements regarding on-demand delivery of customized services, dynamic web service composition approaches have emerged. But still many compositional issues have to be overcome like dynamic discovery of services, compositional correctness, transactional supports etc. In this paper we discuss some of these issues and then investigate some of the representative dynamic web service composition approaches. We evaluate those approaches on the basis of the issues and present how the future research can benefit by addressing those issues of dynamic web service composition.

1 Introduction

In recent years Web services have received much interest as an emerging technology for Business-to-Business Integration (B2Bi) of heterogeneous applications over the Internet [1]. Many enterprises are transforming their business model in Internet with web services because web services technology enables integration of heterogeneous applications regardless of implementation platforms. The full potential of web services as a means for B2Bi solutions will only be realized when existing services and business processes are able to integrate into a value-added composite service. A composite service [2] is a service developed by aggregating the existing services to realize a new value-added functionality. The aggregating process is called (web) service composition. There are two approaches of web service composition: static and dynamic composition.

In static compositions, the aggregation of the services is done at design time. The composition of all the necessary components is determined, bound together and then deployed. This type of composition is more suitable if the business partners involved in the process are fixed and their offered functionalities or the requirements are unlikely to change in short term. Static web service composition is not flexible in the sense that it is not adaptive to the runtime changes when it is in execution and is too restrictive to unavoidable changes in the service requirements [3]. Dynamic web service composition

^{*} The work is done in the context of DySCoTec Project.

aims at overcoming the problems which are apparent in static web service composition. A dynamic service composition provides flexibility for modifying, extending and adapting changes at runtime [4].

The web service environment is highly dynamic in nature. The number of web service providers is constantly increasing leading to the availability of new services in daily basis [5]. In such a dynamic environment, realizing dynamic web service composition is not so easy because of the following reasons:

- composition process should discover the appropriate components in a composition that is able to transparently adapt the environmental changes.
- composition process should adapt to the customer requirements with minimal user intervention.
- composition process should have transactional support.
- composition process should guarantee composition correctness.
- composition process should also consider Quality of Service (QoS) properties.

With some of above stated issues of dynamic web service composition, we aim at evaluating available approaches of dynamic web service composition with respect to the issues like transaction support, compositional correctness, and QoS support etc. Based on the evaluation we believe that understanding those issues and their importance will pave way for future research directions in dynamic web service composition. A significant number of evaluation papers have been published in literatures for web service composition approaches. The survey of [5] extensively evaluates the various available web service composition approaches based on classifications and compositional issues. In [1], the paper focuses more on web service composition languages like WS-BPEL with WSDL, OWL-S with Golog/Planning. In [6], the authors evaluate the web service composition frameworks based on the level of automation of the service composition process. [7] discusses Web Service Composition and Execution (WSCE). The framework are categorized as interleaved, Monolithic, Staged, and Template-based service composition and execution. We believe there is no evaluation paper specifically on dynamic service composition. So, the paper summarizes the currently available dynamic web service composition approaches, evaluate the approach according to the framework and then gives an outlook to essential future research works.

This paper is structured as follows: In Section 2, we define a framework used in the evaluation of the different dynamic service composition approaches. In Section 3, we present some of the most relevant dynamic service composition approaches on the basis of a evaluation framework. In Section 4 we evaluate the approaches against the evaluation framework. Finally, we conclude the paper in Section 5.

2 Evaluation Framework

We propose an evaluation framework for the dynamic service composition approaches that are discussed in Section 3. The evaluation framework includes some of the key composition issues and requirements that are identified based on the analysis of [5] and [8]. We do not guarantee the completeness of all the necessary requirements and issues in this evaluation framework. However, we have included some of the important ones in the evaluation framework.

2.1 Transaction Support

Traditional transaction is based on Atomicity, Consistency, Isolation, and Durability (ACID) properties and implicitly assumes closely coupled environment with short-duration activities. But web services often involve loosely coupled systems with long run transactional activities. Using ACID-based transactions in such long running transactional activities could result in undesirable effects like locking of resources for longer time. So, in web service composition it is necessary to provide a flexible transactional supports for the long-running activities in order to guarantee consistency and reliable execution of composite web services that can support coordination, recovery and compensation [9, 10]. WS-Transaction standard, WS-TXM standards are examples of transactional feature support in web service environment. In this paper, we identify whether the transaction support is available in a composition approach being evaluated.

2.2 Compositional Correctness

An important aspect of web service composition is to maintain the correctness of behavior of the composite service. We consider two ways of exploiting compositional correctness of a service composition to preserve the behavioral properties like deadlock free, liveness, safety etc. The first way is to design the composition algorithm in such a way that it preserves the behavioral properties. The other way is to formalize the composite service and then verify the behavioral properties formally i.e. using some formalization like pi-calculus, petri-nets e.t.c and verify the properties. We believe the later mechanism can be more useful in those cases where the business goal is already achieved but the workflow still might have deadlocks in the path that are yet to be traversed. Using the later mechanism, we can reason about the preservation of the behavioral properties in all possible paths of the workflow of the composite logic. So, formalizing the compositional logic enables us to verify the behavior of the composite service completely. Recently, a variety of concrete proposals from the formal methods community have emerged in order to verify the correctness of the web service composition which is based on state action models or process models [11]. In this paper, we identify whether the approach supports the verification of compositional correctness. In case if it supports, we also identify by which mechanism does it support.

2.3 QoS Support

QoS is used for expressing non-functional properties like performance, reusability, maintainability, security, reliability and availability [12]. In case of loosely coupled environment of web services, the QoS properties can vary greatly because web services can be provided by various third parties and invoked dynamically over the internet [13]. Therefore service compositions must be QoS-aware in terms of understanding and respecting one another's policies, performance levels, security requirements [14]. To support QoS in dynamic service composition, a web service description language that supports QoS description, QoS estimation approach of the composite web service and QoS monitoring is required [15]. In this paper, we identify whether the QoS support is available or if there is partial support in a composition approach being evaluated.

2.4 Automated Composition

We define automated composition as a process of generating an executable process that communicates and binds with a set of existing web services for web service composition and publish itself as a web service with higher level of functionalities. One of the main aims of web service composition approaches is to achieve automated service composition because it enables faster application development and reuse [8]. In absence of automated web service composition, the end user/agent will have to specify the business goal and manually select the available services after the discovery of services matching the requirements and then compose them. In this paper, we identify whether the automated composition feature is available in a composition approach being evaluated.

2.5 Composition Logic Formulation

We define composition logic as the way how the interactions among the participating services take place in the composition process. Under this requirement we investigate how the control flow and the data flow of the composition mechanism are represented. The classification will be either process-driven or rule-driven or hybrid. In process-driven mechanism, the composition uses process definitions where business logic is expressed as a process model to specify possible interactions among the participating web services [16, 17]. In rule-driven approach, business rules are used as composition logic. A business rule¹ is defined as a statement that defines or constrains some aspect of the business. The main features of rule-driven composition approach are that it is purely declarative, highly adaptive and integrated in a truly service oriented approach to business rule management [18]. In the hybrid approach [19], the composition logic is broken down into core part (business processes) and independently evolving, well modularized business rules. In this paper we investigate if the composition approach being evaluated is process-driven, rule-driven or hybrid approach.

3 Dynamic Service Composition Approaches

This section presents a brief overview of some of the prominent dynamic service composition approaches namely: eFlow [20, 21], METEOR-S [22], WebTransact [23, 24], DynamiCoS [25, 26] and SeGSeC [27]. We choose these approaches on the basis of high references in literatures for dynamic service composition. These approaches are then evaluated based on the framework presented in Section 2. Through this evaluation, we identify which of the features are supported by these approaches.

3.1 eFlow

eFlow [20, 21] is a system that supports the specification, enactment, monitoring and management of composite e-services where the composite e-services are modeled as business processes. E-services and web services share commonalities [28] so we interchangeably use those terms in this paper. eFlow runs on the top of E-services Platforms

¹ www.businessrulesgroup.org

(ESPs), such as HP e-speak or Sun Jini which allow the development, deployment and secure delivery of e-services to business and customers. The eFlow model and the overall system provide a flexible, configurable and an open approach to the service composition [29]. The adaptive and dynamic eFlow process model allows processes to adapt the changes in the running environment, perform necessary service execution according to the need of the customer. E-service composition is modeled by a graph that defines the order of execution among the nodes in the process. The graph is created manually but it can be updated dynamically. The graph may include services (represent the invocation of WS), decisions (specify the alternatives and rules controlling the execution flow) and event nodes (enable service processes to send and receive several types of events). Arcs in the graph denote the execution dependency among the nodes.

eFlow includes the notion of transactional region and supports ACID service-level transaction. A transactional region enforces to maintain service level of atomicity. According to the definition of transaction support in Section 2, only preserving the ACID property of transaction is not sufficient to maintain transaction support in web services environment. Hence, transactional feature is not supported in eFlow. eFlow supports the modification of the process for dynamic service composition, but it can not guarantee the correctness of the output. eFlow does not provide QoS modeling capabilities. Service processes in eFlow are able to transparently adapt to environmental changes and dynamically configure at runtime. However, the limitation of the eFlow is that it needs too much manual participation to concretize the generic service nodes (a node in eFlow that supports dynamic process definition for composite services) at execution phase and it does not support the automatic generation of composition for the generic nodes [30]. So, there is no support for the automatic composition. In eFlow, the composite services are modeled as processes that are enacted by a service process engine [29]. So, the composition logic is process-driven and a composite service is described as a process schema that composes other basic or composite services.

3.2 METEOR-S

METEOR for Semantic web services (METEOR-S) is a dynamic web service composition framework developed at the University of Georgia which incorporates workflow management for semantic web services. METEOR-S is the follow-up research of Managing End-To-End Operations (METEOR). METEOR-S uses semantics for the complete life-cycle of the semantic web services. Its annotation framework is an approach to add semantics to current industry standards such as WSDL. METEOR-S uses techniques from the semantic Web, semantic Web services and the METEOR project to deal with the problems of semantic Web service description, discovery and composition. The Figure 1 depicts the architecture of METEOR-S which has two parts: front end and back end. The front end of METEOR-S is related with annotation and publication of service specifications. The abstract process designer which is related with dynamic composition is a component present at the back end of the METEOR-S. The composition process is initiated by creating the flow of process using the control flow constructs provided by WS-BPEL. The requirements of each service in the process is represented by specifying the service template, which allow to either specify semantic description of the web services or a binding to a known web services. Then, the process

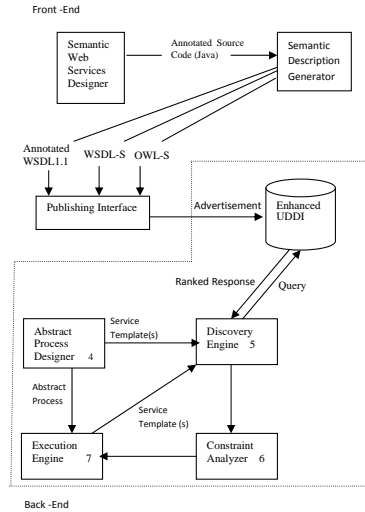


Fig. 1. METEOR-S architecture [22].

constraints for optimization is specified. The details of dynamic service composition in METEOR-S is available on [31].

In METEOR-S architecture, the The constraint analyzer deals with correctness of the process based on QoS constraints. The support for state machine based verification of WS-BPEL process also contributes on the existence of compositional correctness. METEOR-S uses an extensible ontology to represent the generic QoS metrics and domain specific QoS metrics. The cost estimation module of constraint analyzer represents the QoS support. There composition process is not fully automated. The METEOR-S uses process-driven approach for composition. The coordination of the composite service is based on a BPEL-like centralized process engine.

3.3 WebTransact

WebTransact [23] provides necessary infrastructure for building reliable, maintainable, and scalable web service composition. It is composed of a multilayered architecture, an XML-based language named Web Service Transactional Language and a Transaction model. The multi-layered architecture containing a Service Composition Layer, a Service Aggregation Layer, an Integration layer, and a Description Layer are depicted in Figure 2. Based on [24], we provide a brief explanation of web service composition in WebTransact. Application programs interact with the composite mediator services written by composition developers. Such compositions are defined through transaction interaction patterns of mediator services. Mediator services provide a homogenized interface of semantically equivalent remote services. Mediator services also integrate web services providing the necessary mapping information to convert messages from the particular format of the web service to the mediator format.

In WebTransact, an XML-based language named Web Service Transaction Language (WSTL), is used for describing the transaction support. The transaction model

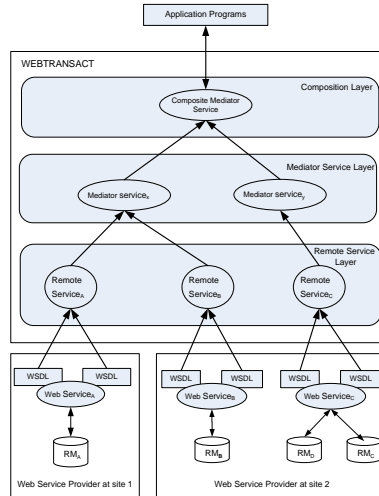


Fig. 2. WebTransact Architecture [24].

of WebTransact provides an adequate level of correctness guarantees when executing the web services composition built with WSTL. Hence, there is transaction support in WebTransact. The transactional model of the WebTransact exploits the dissimilar transaction behavior of web services and guarantees the correct and safe execution of mediator compositions. The notion of correctness of composition execution is based on both the user needs (composition specification) and the *2L-guaranteed-termination criterion* (a weaker notion of atomicity that considers the needs of web service environments). Hence, WebTransact has compositional correctness feature. WebTransact does not provide QoS modeling. The WebTransact approach does not support the dynamic discovery and integration of web services. The Web Services are statically integrated in WebTransact by a developer who plays the role of Web service integrator [24]. So automatic composition is not supported. In WebTransact, web service composition is modeled as composite task by WSTL where a composite task is the combination of atomic task or another composite task. Tasks are identified by its signature, execution dependencies, links and rules. Here rules specify the conditions under which certain event will happen and can be associated with dependencies or to data links and finally evaluating either true or false based on execution [23]. Hence, WebTransact follows the rule-based logic formulation.

3.4 DynamiCoS

In [26, 25], an approach for automated and dynamic service composition named Dynamic Composition of Services (DynamicCoS) is proposed primarily to alleviate the complexity of service composition from the end-users. Upon specifying the service specifications by the users as per their requirements, automatic discovery, matching and the composition of set of services that together fulfill the user's requirement is done by DynamiCoS. The results are then presented to the user who can select the best suited composition. DynamiCoS represents services in language neutral formalism. A service

is represented as a seven-tuple $S = \langle ID, I, O, P, E, G, NF \rangle$, where ID is the service identifier, I is the set of service inputs, O is the set of service outputs, P is the set of service preconditions, E is the set of service effects, G is the set of goals the service realises, NF is the set of service non-functional properties and constraint values. DynamiCoS approach consists of following modules: service creation, service publication, service request, service discovery and service composition. Figure 3 depicts the service composition framework of DynamiCoS. To perform composition, it first organizes the set of services discovered in service discovery phase in a Casual Link Matrix (CLM). The CLM stores all possible semantic connections, or causal links, between the discovered services input and output concepts. Then the graph-based algorithm approach [26] finds a composition of services that fulfil the service request.

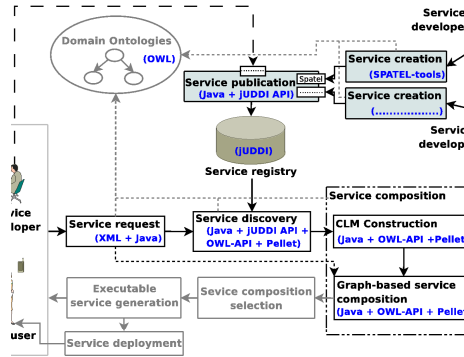


Fig. 3. DynamiCoS Architecture [25].

The main aim of DynamiCoS is to develop dynamic service composition mechanism to support the end-users requirements. Considerable interest is not given in transaction support in DynamiCoS. The service composition module of DynamiCoS builds compositions from service requests and the compositions are correct-by-construction. The algorithm for composition checks for deadlock and also verify if the composition is in accordance with the goals. In DynamiCoS some simple QoS characteristics can be represented and considered in service compositions. Among four ontologies, Non-Functional.owl in DynamiCoS defines non-functional properties for services and hence partially supports QoS modeling. DynamiCoS enables service creation and publication by service developers at design-time, and automatic service composition by end-users at runtime. The composition is based on semantic graph based composition algorithm, so the composition logic formulation is process-driven.

3.5 SeGSeC

In [27, 32], the authors present a semantic-based dynamic service composition architecture named Semantic Graph-Based Service Composition (SeGSeC) in which the user requests the service in a natural language and the request is then converted into machine-understandable format, i.e. a semantic graph. Based on this semantic graph, SeGSeC composes services. In order to achieve semantic-based dynamic service composition, modeling of the service components and the service composition mechanism

itself must support semantics. To satisfy this requirement, SeGSeC is supported by Component Service Model with Semantics (CoSMoS) and Component Runtime Environment (CoRE). The semantic support in the component modeling is achieved by CoSMoS which integrates the semantic and functional information into semantic graph representation. CoRE functions as middleware and provides functionality to discover and convert different component implementations into a single semantic graph representation. Figure 4 depicts the architecture of SeGSeC.

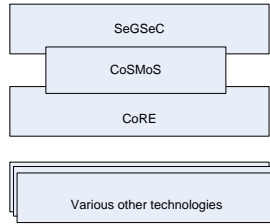


Fig. 4. SeGSeC Architecture [27].

Upon receiving the service request from a user in a natural language, SeGSeC generates the execution path or workflow. The execution path represents the order plan specifying which operations of which component should be accessed in what order. Additionally, SeGSeC also performs the semantic matching to confirm the semantics of the execution plan matches the semantic of the user request.

SeGSeC does not provide transactional support and does not guarantee the compositional correctness. Given the user requirements in the natural language the overall approach generates the workflow such that it satisfies the semantics of the requested services. Starting from the service request from the user to the final composition the process is automated. SeGSeC does not provide QoS modeling capabilities. Upon receiving the user request in the form of semantic graph from CoSMoS, the Service-Composer component of SeGSeC discovers the components and creates the workflow. In later stage of composition the semantic retrieval rules are applied onto the semantic graph such that the graph models the semantics of the workflow. Hence, SeGSeC has hybrid compositional logic formulation because the workflow is process-driven and later the rules are imposed in composition.

4 Summary and Evaluation

This section presents the overall summary of the comparison and evaluation of the dynamic service composition approaches based on the evaluation framework presented in Section 2. Table 1 presents the summary of the approaches evaluated according to the evaluation framework.

4.1 Transaction Support

Due to the inherent autonomy and dissimilar capabilities of web services, maintaining the transaction support is challenging but still the web service composition requires

Table 1. Summary of the dynamic web service composition approaches.

Approaches	Transaction Support	Compositional correctness	QoS Support	Automated composition	Composition logic formulation
eFlow	No	No	No	No	Process-driven
METEOR-S	No	yes	Yes	No	Process-driven
WebTransact	Yes	Yes	No	No	Rule-driven
DynamiCoS	No	Partial	Partial	Yes	Process-driven
SeGSeC	No	No	No	Yes	Hybrid

an advanced transactional management solution for reliable, consistent and recoverable composition. In this aspect WebTransact approach appears to be better than others since it is supported by a transactional model. The eFlow approach includes the notion of transactional regions and ACID level transactional support which is insufficient in the loosely-coupled web service environment. The remaining three approaches do not ensure the transaction support.

4.2 Compositional Correctness

The compositional correctness ensures the verification of behavior of the composite services. In order to preserve compositional correctness various formal methods techniques can be integrated with the framework. METEOR-S has a support for state machine based verification of WS-BPEL process. However, WebTransact provides another approach to reason about the compositional correctness. It uses compositional specification provided by the user and *2L-guaranteed-termination* criterion to verify the compositional correctness [24]. With this feature WebTransact and METEOR-S appear to be the promising one among the others.

4.3 QoS Support

From the evaluation it shows that most of the approaches do not include the QoS properties of the web service composition. Only METEOR-S provides the framework for modeling QoS. There is partial support of QoS modeling in DynamiCoS which is done by extending the OWL-S. Even the industry based approach eFlow neglects such an important aspect of the composition. In [12], the author mentions two ways to include QoS property namely developing a new language based on semantics like XL and OWL-S and to extend web service description languages like WSDL to support more QoS description. The later one seems to be promising to those approaches where QoS modeling is not included because all of them are based on WSDL.

4.4 Automated Composition

Automated composition enables faster application development without the intervention of the users. From the evaluation and comparison presented earlier, we conclude that use of semantic descriptions or ontology can help in automating the composition process as in DynamiCoS, METEOR-S, and SeGSeC. Including the semantic approach and automating the composition in eFlow can be helpful to the industry.

4.5 Compositional Logic Formulation

Process-driven approach is more suitable for those collaborative businesses where there are few changes in the requirements of business process workflow. It appears to be more effective in terms of managing the workflow because seldom changes are required. But in those collaborative businesses where there are constant changes in the requirements of the business then rule-based systems are advantageous. Hybrid composition approach has its own benefits as it reduces the complexity and avoids monolithic composition [19]. Each part of composition logic in hybrid approach is expressed in the more suitable way either in a business rules or as a process activity providing higher degree of flexibility.

With this evaluation WebTransact and METEOR-S appear to be the most promising approaches. The WebTransact framework is a multidisciplinary work that is related with many other areas such as e-service composition, transactional process coordination, workflow management system and distributed computing systems [24]. However, WebTransact lacks semantic descriptions for automation of the composition process and QoS support. The METEOR-S lacks transactional support and is semi-automated but supports QoS modeling. eFlow is the industrial product and another promising framework but it significantly lacks the transactional support, semantic descriptions for automation and QoS support.

5 Conclusions

Because of the growing trend of transforming existing business models to the Internet with web services and the possibility of creating new web services dynamically, various developments are on-going in the dynamic web service composition. The research and development ranges industry-based products like eFlow to academic researches like WebTransact, DynamiCoS to name a few. In all these developments, many different standards and mechanisms have been proposed but there is still a lack of an overall agreement. In this paper, we presented different dynamic service composition frameworks, ranging from industry based product to the academic research products and compared them in terms of some important requirements and features.

The evaluation shows that transactional support is still missing in most of the approaches which, is one of the most important requirements. So, researches leading to include transactional supports in those approaches and in currently ongoing research approaches will be an important step. The verification of the compositional correctness is also missing in most of the approaches. It is an important aspect however, on the other hand integrating formal techniques in such frameworks is not so easy. In order to verify the correctness, compositional specifications have to be modeled into mathematical formalisms like pi-calculus, process algebra such that the safety and liveness properties can be explored to detect deadlocks and data consistency in the composition process. The easier way is to include compositional correctness features in the composition algorithm of the framework. QoS support is also the important requirement of the web service composition. In most of the approaches QoS support is not available so research on including QoS modeling is also an open research direction. The researches on the features like especially automated composition of the web service composition

have gained some maturity by using semantics. The use of semantics and ontology in composition has resulted in the automated composition but is still limited in academic researches like DyanmiCoS, METEOR-S, and SeGSeC approaches. The use of semantic descriptions in order to automate composition in industry approaches like eFlow is still an awaiting result in web service composition field.

The dynamic web service composition problem is likely to be around with some open issues like supporting transactional support, verification of compositional correctness etc. In future research, addressing such issues will surely be beneficial and is desirable.

References

1. Srivastava, B., Koehler, J.: Web Service Composition - Current Solutions and Open Problems. In: ICAPS 2003 Workshop on Planning for Web Services. (2003) 28–35
2. Alonso, G.: Web services: concepts, architectures and applications. Springer Verlag (2004)
3. Shen, L., Li, F., Ren, S., Mu, Y.: Dynamic composition of web service based on coordination model. *Advances in Web and Network Technologies, and Information Management* (2007) 317–327
4. Tasic, V., Mennie, D., Pagurek, B.: Dynamic Service Composition and Its Applicability to E-Business Software Systems–The ICARIS Experience. *Advances in Business Solutions* (2002) 93–104
5. Dustdar, S., Schreiner, W.: A survey on web services composition. *International Journal of Web and Grid Services* 1 (2005) 1–30
6. Rao, J., Su, X.: A survey of automated web service composition methods. *Semantic Web Services and Web Process Composition* (2005) 43–54
7. Agarwal, V., Chafle, G., Mittal, S., Srivastava, B.: Understanding approaches for web service composition and execution. In: *Proceedings of the 1st Bangalore annual Compute conference, ACM* (2008) 1
8. Milanovic, N., Malek, M.: Current solutions for web service composition. *IEEE Internet Computing* 8 (2004) 51–59
9. Papazoglou, M.: *Web services: principles and technology*. Addison-Wesley (2008)
10. Papazoglou, M.: Web services and business transactions. *World Wide Web* 6 (2003) 49–91
11. ter Beek, M., Bucchiarone, A., Gnesi, S.: A survey on service composition approaches: From industrial standards to formal methods. Technical report, Technical Report 2006-TR-15 (2006)
12. Chen, Y., Li, Z., Jin, Q., Wang, C.: Study on qos driven web services composition. (*Frontiers of WWW Research and Development-APWeb* 2006) 702–707
13. Ran, S.: A model for web services discovery with QoS. *ACM SIGecom Exchanges* 4 (2003) 10
14. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. *Computer-IEEE computer society-* 40 (2007) 38
15. Sun, H., Wang, X., Zhou, B., Zou, P.: Research and implementation of dynamic web services composition. *Advanced Parallel Processing Technologies* (2003) 457–466
16. Benatallah, B., Sheng, Q., Dumas, M.: The self-serv environment for web services composition. *IEEE Internet Computing* 7 (2003) 40–48
17. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.: Quality driven web services composition. In: *Proceedings of the 12th international conference on World Wide Web, ACM* (2003) 421

18. Weigand, H., van den Heuvel, W.J., Hiel, M.: Rule-based service composition and service-oriented business rule management. In Vanthienen, J., Hoppenbrouwers, S., eds.: *Proceedings of the International Workshop on Regulations Modeling and Deployment (ReMoD'08)*, ACM (2008) 1–12
19. Charfi, A., Mezini, M.: Hybrid web service composition: business processes meet business rules. In: *Proceedings of the 2nd international conference on Service oriented computing*, ACM (2004) 30–38
20. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: eFlow: a platform for developing and managing composite e-services. Technical Report HPL-2000-36, HPL (2000)
21. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: Adaptive and dynamic service composition in eFlow. In: *Advanced Information Systems Engineering*, Springer (2000) 13–31
22. Aggarwal, R., Verma, K., Miller, J., Milnor, J.: *Dynamic Web Service Composition in METEOR-S*. Technical report, LSDIS Lab, Univeristy of Georgia, Athens (2004)
23. Pires, P., Benevides, M., Mattoso, M.: Building reliable web services compositions. *Web, Web-Services, and Database Systems (2002)* 59–72
24. Pires, P.F.: WEBTRANSACT: A Framework for Specifying and coordinating reliable web services compositions. Technical Report ES-578/02, Federal University of Rio De Janerio (2002)
25. Lécué, F., Silva, E., Ferreira Pires, L.: A framework for dynamic web services composition. *Emerging Web Services Technology II (2007)* 59–75
26. Silva, E., Ferreira Pires, L., van Sinderen, M.J.: Supporting Dynamic Service Composition at Runtime based on End-user Requirements. In: *Workshop at the International Conference on Service Oriented Computing (ICSOC) 2009*, Stockhome, Sweden. (2009) 22–27
27. Fujii, K., Suda, T.: Semantics-based dynamic Web service composition. *International Journal of Cooperative Information Systems* 15 (2006) 293–324
28. Tiwana, A., Ramesh, B.: E-services: Problems, opportunities, and digital platforms. In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, 2001. (2001) 8
29. Casati, F., Ilnicki, S., Jin, L.J., Krishnamoorthy, V., Shan, M.C.: An Open, Flexible, and Configurable System for E-Service Composition. Technical Report HPL-2000-41, HPL (2000)
30. Li, G., Hai, H., Hu, Z.: A Flexible Framework for Semi-automatic Web Services Composition. In: *IEEE Asia-Pacific Services Computing Conference, 2008. APSCC'08.* (2008) 1258–1262
31. Sivashanmugam, K., Miller, J., Sheth, A., Verma, K.: Framework for semantic web process composition. *International Journal of Electronic Commerce* 9 (2005) 71–106
32. Fujii, K., Suda, T.: Dynamic service composition using semantic information. In: *Proceedings of the 2nd international conference on Service oriented computing*, ACM (2004) 39–48

Model Checking Verification of Web Services Composition

Abdallah Missaoui¹, Zohra Sbai¹ and Kamel Barkaoui²

¹ Ecole Nationale d'Ingénieurs de Tunis, BP. 37 Le Belvédère, 1002 Tunis, Tunisia
{abdallah.missaoui, zohra.sbai}@enit.rnu.tn

² Conservatoire National des Arts et Métiers, Rue Saint Martin, 75141 Paris, France
barkaoui@cnam.fr

Abstract. Web services composition is becoming very important in today's service oriented business environment. Different services frequently have semantic inconsistencies which may lead to the failure of the services composition. In order to verify the correctness of the Web Services composition, we present a method for analyzing and verifying interactions among web services. We model web service composition based on special class of Petri nets: open workflow nets. We translate this composition to Promela, a source language of SPIN model checker, designed to describe communicating distributed services. At the requirements level, model checking is used to validate the specification against a set of formulae specified into LTL which are used to verify constraints satisfaction of web services composition.

1 Introduction

Founded on standard protocols, Web service is a kind of software interface and platform, which is described, published and invoked through the Internet. Web services have the capability of implementing distributed applications. Recently, integrating various services distributed becomes a valuable issue in current research on Web services.

The behaviour of a service composite may become very complex due to the complex of the communication between partners. The tasks that are performed within a service generally depend strongly on the interaction that has taken place. Thus, it is very important to decide whether all services interact properly. At present, many industry and academia researchers are paying attention to find some methods to detect these behaviour problems such deadlock.

Model checking is a verification technique that explores all possible system states in a brute-force manner. A model checker, the software tool that performs the model checking, examines all possible system scenarios in a systematic manner. In this way, it can be shown that a given system model truly satisfies a certain property. There are many powerful model checkers such as NuSMV [4], BLAST [5] and SPIN [6, 7].

This paper proposes to use the software model checking techniques for the verification of web services composition. We adapt Petri nets to describe services interaction and use SPIN model checker as a verification engine. First, a composed system model is written in Promela (PROcess MEta LAnguage) that describes the behaviour of the web services composition. Then, correctness properties that express requirements on

the system's behaviour are specified in Linear Temporal Logic (LTL). By efficient exploration of the complete set of states generated from the Promela specification, SPIN verify LTL formulae corresponding to properties of web service composition.

The rest of this paper is organized as follows. Section 2 presents preliminaries. Section 3 focuses on Petri net model of web service composition as well as our Promela implementation of this composition. In section 4, We formulate in LTL some soundness properties to be verified by SPIN. In section 5, we study an example of web service composition. We discuss in section 6 some related work. Section 7 concludes the paper.

2 Preliminaries

2.1 Workflow Process Modelling

Petri nets are a well founded process modeling technique that have formal semantics. They are one of the best known techniques for specifying business processes in a formal and abstract way. The semantics of process instances ensuing from process models described in Petri nets are well defined and not ambiguous.

We choose to adopt open workflow nets [17] for web service modelling. Open workflow nets result from the application of Petri nets to workflow management. In fact, Petri nets have been used for their formal semantics, graphical nature, expressiveness, analysis techniques and tools.

In the rest of this section, we present first the basic definitions and notations of Petri nets [2] used in this work. Then we highlight the notion of open workflow nets.

Petri Nets

A Petri net is a 4-tuple $N = (P, T, F, W)$ where P and T are two finite non-empty sets of places and transitions respectively, $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, and $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is the weight function of N satisfying $W(x, y) = 0 \Leftrightarrow (x, y) \notin F$.

If $W(u) = 1 \forall u \in F$ then N is said to be ordinary net and it is denoted by $N = (P, T, F)$.

For all $x \in P \cup T$, the preset of x is $\bullet x = \{y | (y, x) \in F\}$ and the postset of x is $x^\bullet = \{y | (x, y) \in F\}$.

A marking of a Petri net N is a function $M : P \rightarrow \mathbb{N}$. The initial marking of N is denoted by M_0 .

A transition $t \in T$ is enabled in a marking M (denoted by $M[t]$) if and only if $\forall p \in \bullet t : M(p) \geq W(p, t)$. If transition t is enabled in marking M , it can be fired, leading to a new marking M' such that: $\forall p \in P : M'(p) = M(p) - W(p, t) + W(t, p)$.

The firing is denoted by $M[t]M'$. The set of all markings reachable from a marking M is denoted by $[M]$.

For a place p of P , we denote by M_p the marking given by $M_p(p) = 1$ and $M_p(p') = 0 \forall p' \neq p$.

Petri nets are represented as follows: places are represented by circles, transitions by boxes, the flow relation is represented by drawing an arc between x and y whenever (x, y) is in the relation, and the weight function labels the arcs whenever their weights

are greater than 1. A marking M of a Petri net is represented by drawing $M(p)$ black tokens into the circle representing the place p .

Open Workflow Nets (oWF-nets)

In this paper, we choose to adopt a special class of Petri-nets which is open workflow-nets (oWF-net), for web service modeling. It generalized the classical workflow nets [1] by introducing an interface for asynchronous messages with partners. A petri net $N = (P, T, F)$ is called an oWF-net, if:

- i. P is composed from disjoint sets: internal places P_N , input places P_I and output places P_O ;
- ii. For all transition $t \in T : p \in P_I$ (resp. $p \in P_O$) implies $(t, p) \notin F$ (resp. $(p, t) \notin F$);
- iii. F does not contain cycles.

In all examples in this paper, an oWF-net are initially one token in start place (no tokens in other places including the interface places).

Open workflow nets allow diverse analysis methods of business process. Moreover, the explicit modeling of interface allows the verification and behaviour analysis of web service composition.

2.2 Model Checking Techniques

Model checking is a verification technique that explores all possible system states in a brute-force manner. Similar to a computer chess program that checks possible moves, a model checker, the software tool that performs the model checking, examines all possible system scenarios in a systematic manner. In this way, it can be shown that a given system model truly satisfies a certain property. It is a real challenge to examine the largest possible state spaces that can be treated with current means, i.e., processors and memories.

There are many powerful model checkers such as NuSMV, BLAST and SPIN.

The SPIN model checker is a system that can verify models of computerized systems. The name SPIN was originally chosen as an acronym for Simple Promela INterpreter. It can be used in two basic modes: as a simulator and as a verifier. In simulation mode, SPIN can be used to get a quick impression of the types of behavior that are captured by a system model, as it is being built. Some optimization techniques, e.g., partial order reduction and graph encoding, are available to help reduce the usage of CPU time or memory space.

2.3 Linear Temporal Logic

SPIN checks properties formulated with Linear Temporal Logic. An LTL formula f may contain any lowercase propositional symbol p , combined with unary or binary, Boolean and/or temporal operators. Propositional and temporal operators are presented in figure 1.

The semantics of a formula is given in terms of computations and the states of a computation. The atomic propositions of temporal logic can be evaluated in a single state independently of a computation.

Propositional Operators	
&& conjunction	disjunction
- > implication	! Negation
Temporal Operators	
□ always	◇ eventually
	⊔ until

Fig. 1. Propositional and Temporal Operators.

3 Web Services Composition

In this section, we begin with a specification of services in terms of open Workflow net and Promela language. Then, we model the composition of web services and we give a verification method by means of model checking techniques.

The formalism of open Workflow net is meant to model complex, distributed computing systems with well defined semantics. Web services composition can be checked against requirements such as the mentioned above. The properties that are to be checked need to be formulated based on temporal logic, relate to the Promela specification of workflow representation of web service composition. SPIN Model checker will evaluate these LTL formulae.

3.1 Modeling Web Service Composition

Using oWF-nets, we can model business process. We can furthermore compose two or many oWF-nets and describe the web services interaction with partners.

The communication between two oWF-nets is based on interface places (unidirectional). Given two oWF-nets N1 and N2, their composition is the result of merging one or many input places of one oWF-net with suitable output net of the other place and vice versa. Each oWF-net shares only interface places with other oWF-nets.

Fig 2.a shows the composition of two oWF-nets: Producer and Consumer. Each of which has one input place and one output place. Shared places which model the interfaces are represented by dashed lines. The consumer of the left side sends an order message to the producer in order to execute its task. In the right side, the producer executes the task and then sends a response to consumer.

3.2 Promela Implementation

Like structured programming languages, in Promela we can use variables and subroutines such as *proctype*. The *proctype* construct is used for the declaration. It can be used to declare process behavior is to be executed completely deterministically. Each process behavior must be declared before process instantiation. This instantiation can be done either with the *run* operator, or with the prefix *active* that can be used at the time of declaration. The types of variables are sets of integer in order to guarantee that the program has finite state space. This section presents a method to use the SPIN model checker for the representation of the web services composition and analysis. The basic idea is just to translate a composition of oWF-nets into Promela source program. web services models communicate through interface places modeled by variables. We

adopt an approach of oWF-net introduced by [17] to model web services composition. Workflow module defines its internal behaviour without interface places that are used to represent message flow between partners.

Based on the definition of workflow module, we present a web service specification in Promela language. This specification is described in terms of the marking of places and the firing count of transitions. Hence, we use the following conventions:

- Places are represented by an array PL of integers with length equal to the number of places. This array contains initially zero in each element.
- Transitions are modelled by an array TR of integers (initialized to zero) with length equal to the number of transitions.

The behaviour of a web service can be described in terms of system states and their changes. A marking is initialized to M_i and it is changed according to the following transition rule: A transition t is enabled if each input place p of t is marked. Moreover, an enabled transition t may or may not fire, and a firing of t removes one token from all $p \in \bullet t$ and adds one token to each output place of t . These concepts are translated in the corresponding Promela description of a workflow module as follows: The firing of a transition consists on decreasing by 1 the integer corresponding to each place $p \in \bullet t$ in the array PL and increasing by 1 the elements of PL corresponding to the output places ($p \in t \bullet$). Although, we increase by 1 the element of TR corresponding to the transition t to mark that t is fired once.

These modifications are made by the macro $add1, add2, \dots, addS, remove1, remove2, \dots$, and $removeK$ where S is the maximum number of possible input places and K is the maximum number of possible output places.

To fire a transition t which has I input places and J output places, we call these macros with the appropriate arguments in order to achieve the following actions:

1. $removeI(p_1, p_2, \dots, p_I)$ destructs one token from each input place. It tests if these input places are marked ($PL[index\ of\ p_j] > 0, 1 \leq j \leq I$) and if this condition is satisfied, it removes one token from each of its parameters.
2. We increase the element corresponding to t in TR .
3. And finally $addJ(p_1, p_2, \dots, p_J)$ creates one token in each output place.

Complex inter-organizational business processes are structured as a set of communicating elementary services. A service represents a self contained software unit that offers an encapsulated functionality over a well defined interface.

Several languages, such as BPEL [3], are used to describe the composition of services. The behaviour of a service composite may become very complex due to the nature of the communication between partners. The tasks that are performed within a service generally depend strongly on the interaction that has taken place. Thus, analyzing the behaviour of a service on the one hand and the verification of its interaction with partners on the other hand are very important to perform and improve inter-organizational business processes.

Web services composition is modeled based on oWF-nets. Interface places are represented, in our specification, by an array I of integers with length equal to the number of places in web services composite. Each element of this array, that initialized to zero,

perform the communication between services partners, and for each place, when one service increment element of I (add token in interface place), another partner consume a token and decrement this element by 1. Figure 2.a shows an example of web service

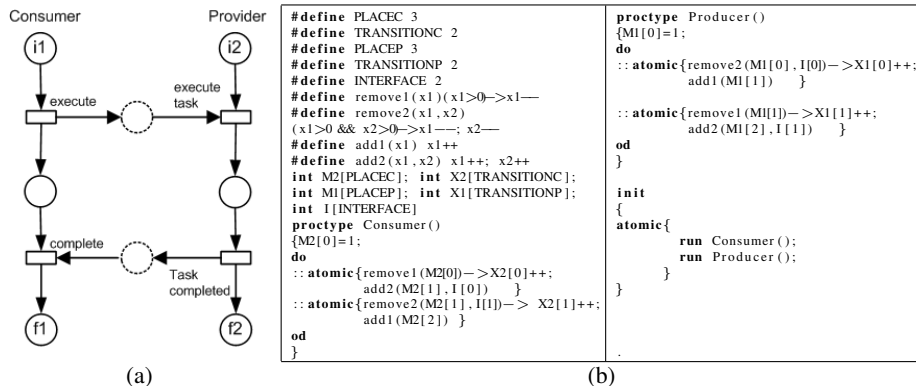


Fig. 2. The process with two services.

composition based on two oWF-nets. The consumer communicates with producer by means of two interface places. The figure 2.b represents the Promela description of this composed system.

4 Verification

SPIN allows Promela systems to be simulated, either step-by-step or randomly. Additionally, it is possible to execute an exhaustive simulation by generating the complete state-space of the system. This allows us to verify if the system requirements are satisfied.

In web service architecture, by composing various services, complex activities (e. g., inter-organizational business processes) can be realized. Hence, the correct interplay of distributed services is crucial to accomplish a common goal. The flow should have safety properties such as deadlock freedom. At the same time, the flow should satisfy some properties since it is executed in an environment.

In this section, we discuss how to verify a given composed oWF-net specification using the SPIN model checker. The input language of SPIN is Promela, which is a language for modeling web services composition. SPIN verify LTL properties of Promela specification. We implement the verification process in three steps : (1)we model the composed web services based on oWF-net; (2)we translate this composed system to Promela processes with communication based on an interface array; (3)we define properties in LTL and we check if the system satisfies or not these properties.

The requirement needs to be formulated as LTL formulas. All logical variables used in the formula must be defined by *define* macro. Using logical variables integrated in TLT formulae, all properties are verified on the fly. It is not necessary to generate the whole state space to detect errors. However, to verify that properties are satisfied, the whole state space may be generated.

4.1 Termination

Several Technologies [9–11] have been proposed in order to automatically compose Web services to perform some desired task. Regardless of how the compositions originated, we are interested here in describing and verifying properties of these services by simulating their executions under different input conditions. Deadlock is a condition which states that the composed system is deadlocked, i.e. it is neither terminated nor more activities could be executed.

The language source used by SPIN (Promela) describe the behaviour of a web services composition based on interactions between the service and its partners. SPIN can simulate and validate a model in Promela language and it can accept constraints described in LTL formulae.

```

Never-claim :
never {
  /* !(⟨>p) */
  accept_init:
  T0_init:
    if
    :: (! ((p))) -> goto T0_init
    fi;
}

```

Fig. 3. Never-claim corresponding to F .

The formula $F : \diamond p$ (p is declared in Promela source as $(M1[index\ of\ f1] \geq 1 \ \&\& \ M2[index\ of\ f2] \geq 1)$) allows us to check if the process, given in figure 2, completes successfully. First, SPIN translates a negative form of formula into a never-claim statement. Second, SPIN validates the model S with this statement. The never-claim (generated by SPIN) corresponding to formula F is given in figure 3.

Never claim is the Promela model of the Büchi automaton (figure 4) corresponding to the LTL formula F . It is used to represent a property that should never be satisfied during the execution of a model. Then, if S contains an acceptance cycle then a counterexample to F exists and this proves that the model does not satisfy the property.

4.2 Soundness

Composing web services and guarantying the correctness of the design is an important and challenging problem in web services.

An oWF-net is a more suitable model for analysis and verification of web services composition. Several Correctness criteria, like liveness and termination, should be valid for single process and for the combined model. In this section, we give a method to verify some properties of web services composition. For each of them, we define the property of the composed oWF-nets. We also give an LTL formula that allows to verify it based on corresponding Promela specification of web services composition.

A workflow process is sound if, for any case, termination is guaranteed, there are no dangling references, and deadlocks and livelocks are absent. This dynamic property is decidable and it can be checked in polynomial time. Soundness property is proved in [18] equivalent to liveness and boundedness, thus it can be verified by standard Petri net methods.

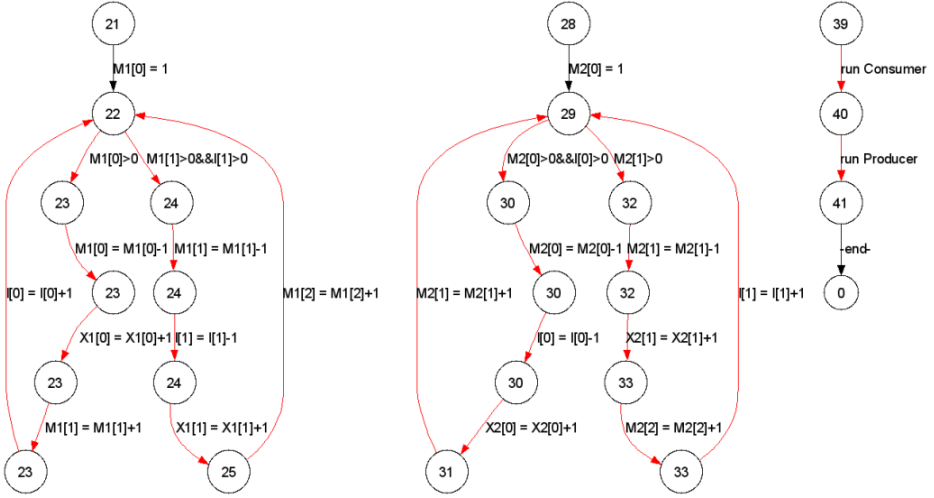


Fig. 4. A Büchi automata.

The extension of this soundness definition in the context of web service composition via oWF-nets is given in the following definition 1. The overall net describing the composition can be seen as a classical petri net with a specific initial marking and its soundness is given in terms of states evolution and transitions firing. Thus the verification of this property is decidable.

Definition 1. (*Soundness*)

Let $N_1 = (P_1, T_1, F_1)$, $N_2 = (P_2, T_2, F_2), \dots$, and $N_x = (P_x, T_x, F_x)$ be x oWF-net. The composed oWF-nets $WC = (P, T, F)$ from N_1, N_2 and N_x , is called sound if and only if the following three requirements are satisfied:

- i Termination: For each reachable marking (reachable from $M_0 = [i_1, i_2, \dots, i_x]$), the final marking $M_f = [f_1, f_2, \dots, f_x]$ is reachable;
- ii Proper completion: For each reachable marking, if $M \geq M_f$ holds then $M = M_f$;
- iii No dead transition: It makes certain that starting from the initial state (just one token in every initial place), it is always possible to reach the state with one token in every final place.

We can define LTL formulae based soundness of web service composition by studying the three sub-properties: Completion, Proper Completion and Deadlock-freedom:

- i $\diamond (\bigwedge_{i=1}^{i=x} M[\text{index of } f_i] \geq 1)$
- ii $\square (\bigwedge_{i=1}^{i=x} M[\text{index of } f_i] \geq 1) \Rightarrow (M = [f_1, f_2, \dots, f_x])$
- iii $\neg(\square (\bigvee_{t \in T} X[\text{index of } t] == 0))$

To verify soundness of web service composition, we have to check the three preceding formulae.

Weak soundness properties are introduced by Martens [8] in the context of web service composition. We can allow service that exhibits a proper behaviour but not all tasks. The overall Composed system is weak sound, because the final state can be reached from each state reachable from the initial state, and when the final state M_f is reached, no other token remains in the net.

Using Petri nets notation, we give the following definition of weak soundness.

Definition 2. (*weak soundness*)

a composed system $N = (P, T, F)$ is called weak sound if and only if:

- i. For each reachable marking (reachable from M_0) the final marking f is reachable.
- ii. For each reachable marking, if $M \geq M_f$ holds then $M = M_f$.

Based on LTL formulae, corresponding to Termination and Proper Completion, defined in this section, we can verify weak soundness properties of the web service composition.

5 Case Study

Figure 5 presents an example of web service composition. The overall flow of this example is as follows: A traveler will plan her trip. When the traveler is finished with specifying the details of the trip plan, she sends this information together with the prepared details (example of details concern the list of participants) to the travel agent. Next, the traveler will await the submission of the electronic tickets as well as the final itinerary for the trip. When the agent receives the traveler's trip order, he will determine the legs for each of the stages, which includes an initial seat reservation for each of the participants as well as the rate chosen. To actually make the corresponding ticket orders the agent submits these legs together with the information about the credit card to be charged to the airline company.

Then, the agent will wait for the confirmation of the flights, which especially includes the actual seats reserved for each of the participants. This information is completed into an itinerary, which is then sent to the traveler.

When the airline receives the ticket order submitted by the agent, the requested seats will be checked. After that, the credit card will be charged, and the updated leg information is sent back to the agent as confirmation of the flights. After that, the airline sends the electronic tickets (by e-mail) to the traveler. Information about the recipient of the tickets is passed to the agent as well as to the airline.

In this example, we propose to draw its Promela model as follows: we begin with specifying each process as a single process defined by *proctype* construct. The processes describing the work of the traveler, the agent and the airline are given respectively in the following source code.

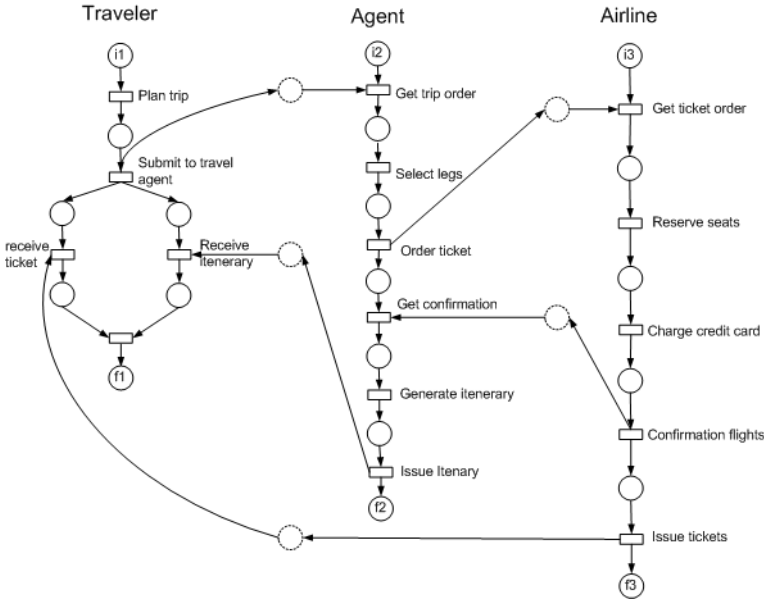


Fig. 5. Travel Plan Example.

```

proctype Traveler ()
{PL1[0]=1;
do
::atomic{inp1(PL1[0]) -> TR1[0]++;out1(PL1[1]) }
::atomic{inp1(PL1[1]) -> TR1[1]++;out3(PL1[2],PL1[3],I[0]) }
::atomic{inp2(PL1[2],I[4]) -> TR1[2]++;out1(PL1[4]) }
::atomic{inp2(PL1[3],I[1]) -> TR1[3]++;out1(PL1[5]) }
::atomic{inp2(PL1[4],PL1[5]) -> TR1[4]++;out1(PL1[6]) }
od }
proctype Agent ()
{PL2[0]=1;
do
::atomic{inp2(PL2[0],I[0]) -> TR2[0]++;out1(PL2[1]) }
::atomic{inp1(PL2[1]) -> TR2[1]++;out1(PL2[2]) }
::atomic{inp1(PL2[2]) -> TR2[2]++;out2(PL2[3],I[2]) }
::atomic{inp2(PL2[3],I[3]) -> TR2[3]++;out1(PL2[4]) }
::atomic{inp1(PL2[4]) -> TR2[4]++;out1(PL2[5]) }
::atomic{inp1(PL2[5]) -> TR2[5]++;out2(PL2[6],I[1]) }
od }
proctype Airline ()
{PL3[0]=1;
do
::atomic{inp2(PL3[0],I[2]) -> TR3[0]++;out1(PL3[1]) }
::atomic{inp1(PL3[1]) -> TR3[1]++;out1(PL3[2]) }
::atomic{inp1(PL3[2]) -> TR3[2]++;out1(PL3[3]) }
::atomic{inp1(PL3[3]) -> TR3[3]++;out2(PL3[4],I[3]) }
::atomic{inp1(PL3[4]) -> TR3[4]++;out2(PL3[5],I[4]) }
od }

```

Note that in each process, the firing of a transition leads to add tokens not only in its own places but also in interface ones. The interface places are modelled by an array called I of n_i elements with n_i is the number of these places. The initial process, declared by *init*, initializes global variables, and instantiates processes via the *run* statement. This statement is used as a unary operator that takes the name of a process type. An executing process disappears again when it terminates (i.e., reaches the end of the body of its process type declaration), but not before all processes that it started have terminated. The sequence of composed processes has to be executed as one indivisible unit, non-interleaved with any other processes. Hence, we prefix the sequence of processes instantiations by *atomic*.

```

init
{
  atomic{
    run Traveler();
    run Agent();
    run Airline()
  }
}

```

To verify soundness of web services composition, three requirements need to be formulated as LTL formula. We introduce logical variables that take part from the LTL formula in the source code. Using the Promela source and the logical variable definitions, SPIN evaluates the LTL formula. The web service composition presented in fig 5 is sound because it satisfies the three formulae (termination, proper completion and deadlock-freedom).

For example, the evaluation of a formula $F (\langle \rangle p)$ of termination ($p: PL1[6] > 0 \ \&\& \ PL2[6] > 0 \ \&\& \ PL3[5] > 0$) have resulted in positive answers given that the number of errors returned by SPIN is 0. This positive result guarantees that the Promela model meets this requirement. All states have been explored by SPIN and the result generated shows that the number of states explored is 24.

6 Related Works

Many works such as [8, 16, 15] introduce formal method based on Petri nets for describing and verifying web service composition. In [15], semantics of Petri nets is defined by mapping BPEL process to a Petri net. A formal model of BPEL can be generated and allows the verification techniques developed for Petri nets to be exploited in the context of BPEL processes. Lohmann [16] focuses on the problem of analyzing the interaction between WS-BPEL processes. A BPEL process is seen as an open workflow net by an interface specifying the interactional behaviour of this process with its partners. The framework developed in this context allows the generation of compact Petri net models and gives a formal analysis of processes behaviour and a verification of controllability.

Some research efforts have already been proposed to use model-checking techniques for web service verification. Nakajima [12] describes a method based on model checking to verify web service flow description. The language adopted to describe web service composition is Web Services Flow Language (WSFL). The model checker used in their verification engine is SPIN. This paper presents means to translate WSFL primitives into PROMELA. In [13], Nakajima proposed to extract behaviour specification from BPEL processes to a variant of Extended Finite-state Automaton. Then, Automaton model is translated to PROMELA and is analyzed by SPIN model checker. Nevertheless, not all BPEL processes can be analyzed i.e. it does not deal with the semantics of handlers such as exception or compensation.

Bao [14] provides a model checking method to Verify BPEL4People processes that can detect deadlocks and validate constraints based on LTL. A tool is developed to translate BPEL4People process to promela automatically but it didn't support all feature activities.

7 Conclusions

In this paper, we have presented a specification and verification method of web services composition based on model checking. The interactions among the partners participating to a web services composition are modeled by Open WorkFlow nets (OWF-nets), where the communication is ensured by interface places. Hence, we begun with converting OWF-nets to Promela: the model specification language used to define the relevant aspects of the system needed to verify it.

The formal model presented in this paper captures and checks the control flow and the dynamic behaviour based on Promela specification that describes the behaviour of the web services composition. After this step, we have specified in Linear Temporal Logic (LTL) correctness properties that express requirements on the system's behaviour. Especially, we have defined the soundness properties of the web service composition in terms of LTL formulae.

Once the formal model and the correctness properties are defined, SPIN model checker verifies LTL formulae corresponding to properties of web service composition, by efficient exploration of the set of states generated from the Promela specification.

We propose to extend this work by defining additional verification criteria which are suitable for the composition of web services such as controllability.

References

1. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, vol. 8, (1998) 21-66
2. Barkaoui, K., Ben Ayed, R., Sbaï, Z.: Workflow Soundness Verification based on Structure Theory of Petri Nets. *International Journal of Computing and Information Sciences (IJCIS)*, Vol. 5(1), (2007) 51-61
3. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting WSBPEL processes using flexible model generation. *DKE* 64(1), (2008) 3854
4. Cimatti, A., Clarke, E. M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R. and Tacchella, A.: NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In *Proceeding of International Conference on Computer-Aided Verification*, (2002)
5. Henzinger, T. A., Jhala, R., Majumdar, R. and Sutre, G.: Software Verification with Blast. In *Proceedings of the 10th SPIN Workshop on Model Checking Software (SPIN)*, Lecture Notes in Computer Science 2648, Springer-Verlag, (2003) 235-239
6. Holzmann, G. J.: *The SPIN Model Checker, Primer and Reference Manual*. Addison-Wesley, (2003)
7. Holzmann, G. J.: *The Model Checker SPIN*. *IEEE Transactions on software engineering*, vol.23, no.5, (1997)
8. Martens, A.: Analyzing web service based business processes. In *Proceeding of International Conference on Fundamental Approaches to Software Engineering, Part of the European Joint Conferences on Theory and Practice of Software*, Lecture Notes in Computer Science vol. 3442, Springer-Verlag, (2005)
9. Leymann, F.: *Web Services Flow Language (WSFL 1.0)*. IBM Corporation, May 2001.
10. *Business Process Execution Language for Web Services (BPEL)*, Version 1.1, <http://www-128.ibm.com/developerworks/library/specification/ws-bpel>. (2002)
11. Thatte, S.: *XLANG: Web Services For Business Process Design*, Microsoft Corporation, (2001), (<http://www.gotdotnet.com/team/xml/wsspecs/xlang-c/default.htm>)

12. Nakajima, S.: Verification of Web service flows with model-checking techniques, presented at First International Symposium on Cyber Worlds, (2002)
13. Nakajima, S.: Model-Checking Behavioral Specification of BPEL Applications. *Electronic Notes in Theoretical Computer Science* 151 (2006) 89105
14. Bao, F., Zhang, L.: A Model Checking Method to Verify BPEL4People Processes. on The IEEE Symposium Advanced Management of Information for Globalized Enterprises. (2008)
15. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri nets. *Business Process Management, LNCS*, vol. 3649, (2005) 220235.
16. Lohmann, N., Massuthe, P., Stahl, C. and Weinberg, D.: Analyzing interacting WS-BPEL processes using flexible model generation, *Data and Knowledge Engineering* 64 (2008) 3854
17. Massuthe, P., Reisig, W., Schmidt, K.: An operating guideline approach to the SOA, *Annals of Mathematics, Computing and Teleinformatics* 1 (3) (2005) 3543
18. van der Aalst, V.M.P.: Structural characterization of sound workflow nets, *Computing Science Report 96/23*, Eindhoven University of Technology, (1996)

Semi-automatic Dependency Model Creation based on Process Descriptions and SLAs

Matthias Winkler¹, Thomas Springer², Edmundo David Trigos¹ and Alexander Schill²

¹ SAP Research CEC Dresden, SAP AG, Chemnitzer Str. 48, 01187 Dresden, Germany
{matthias.winkler, edmundo.david.trigos}@sap.com

² TU Dresden, Faculty of Computer Science, Institute for Systems Architecture
Computer Networks Group, Nthnitzer Str. 46, 01187 Dresden, Germany
{thomas.springer, alexander.schill}@tu-dresden.de

Abstract. In complex service-oriented business processes the composed services depend on other services to contribute to the common goal. These dependencies have to be considered when service compositions should be changed. Information about dependencies is only implicitly available from service level agreements and process descriptions. In this paper we present a semi-automatic approach to analyze service dependencies and capture information about them explicitly in a dependency model. Furthermore, we describe a system architecture which covers the whole process of dependency analysis, dependency model creation and provisioning. It has been implemented based on a healthcare scenario.

1 Introduction

According to the Internet of Services vision services traded via open marketplaces are composed to business processes. Services are provided fully automatically (credit card check) or involve manual steps (healthcare services). The composed services have to collaborate to achieve a common goal. Thus, the composition creates different types of dependencies between involved services, e.g. with respect to produced and consumed resources, timing, quality of service (QoS), and pricing. Dependencies occur between atomic services (horizontal dependency) or between atomic services and the composition (vertical dependency).

Explicit knowledge about dependencies is needed for the management of service level agreements (SLA) in service compositions. SLAs are negotiated between the service provider and consumer to regulate service provisioning. During the negotiation process it is necessary to ensure that all SLAs of the composition enable the proper collaboration between the different services and the fulfillment of all SLAs. Furthermore, dependency information is also needed for the handling of SLA violations or explicit SLA renegotiation requests by the different stakeholders. SLA violations as well as the renegotiation of SLAs may affect other services and lead to the violation of other SLAs. Thus, information about service dependencies is needed for SLA management by composite service providers. Required information about service dependencies is usually not explicitly available but is implicitly contained in SLAs and process descriptions. From these sources it has to be extracted to be available at runtime.

In previous work we presented an approach to managing dependencies in service compositions [1], where dependencies are analyzed at design time, dependency information is captured in a dependency model [2], and dependency information is used at runtime to evaluate effects of SLO violations and SLA renegotiation requests on other services. This paper extends our work by two major contributions. Firstly, we detail our work on dependency model creation by a process description. Secondly, we present an architecture for managing dependencies. The remainder of this paper is structured as follows: In Chapter 2 we describe the dependency model creation process followed by the presentation of the architecture of the approach in Chapter 3. We evaluate our work in Chapter 4 and discuss it with respect to related work in Chapter 5. Finally, we conclude this paper with a discussion and outlook on future work in Chapter 6.

2 Dependency Model Creation

In order to manage the dependencies between services throughout the lifecycle of a composite service, we developed an approach which captures dependencies in a dependency model at design time and which uses this information to evaluate effects of SLO violations as well as SLA renegotiation requests at runtime. We developed a lifecycle for managing and using dependency information. This lifecycle consists of four phases: creation and recalculation, validation, usage, and retirement. In this chapter we will detail the first lifecycle phase focusing on the model creation and its integration into the development process of the composite service. In Fig. 1 the dependency model creation process is depicted. This semi-automatic process is initiated by the composite service modeler. As pre-requisite for executing the process two aspects have to be fulfilled:

1. The composite service workflow has been modeled (e.g. BPMN process). It provides information on temporal relationships between services.
2. SLA offers for all services are available specifying information regarding execution time and location, handled resources, supported QoS, and service price. This information is needed for dependency model creation.

As a result a dependency model is created, which still needs to be validated with respect to the negotiated SLAs. This is necessary in order to avoid conflicts between the proposed SLAs (e.g. with respect to time and QoS attributes).

The creation process of a dependency model was realized as a semi-automatic approach consisting of automatic dependency discovery and the explicit modeling of dependencies. The discovery of dependencies automates part of the dependency model creation process. It also helps to reduce the chance of errors such as false or missing dependencies introduced by manual modeling. The manual extension and modification of the generated dependency model enables the expression of dependencies which cannot be discovered automatically. However, it also introduces the chance of errors being added to the dependency model. In the following sections the dependency discovery and dependency modeling are explained in more detail.

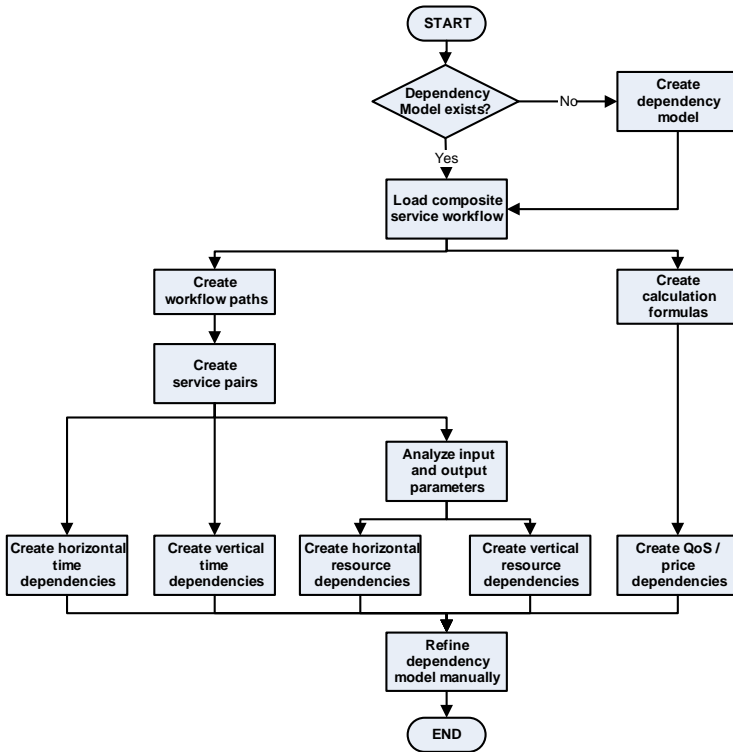


Fig. 1. Dependency model creation process.

2.1 Dependency Model Creation Process

As a first step in the process of creating a dependency model a new model instance is created if it does not already exist. After that two parallel tasks are started for the discovery of dependencies. This includes the creation of time and resource dependencies on the one hand and the creation of aggregation formulas for QoS attributes and price information on the other hand. For the creation of time and resource dependencies the first step is the creation of linear paths reaching from the start node to the end node of the composite service workflow. For each path pairs of services are created. The selection of the relevant services for pair creation is dependent on the type of dependency which is analyzed (see section 2.2). Based on the created pairs the analysis of dependencies is done. The creation of time dependencies is directly based on the different pairs. No further analysis is necessary, since time dependency creation is based on the process structure only, i.e. if a service S_2 follows service S_1 in the process, this implies that S_1 is executed before S_2 . For the creation of resource dependencies the input and output parameters of two services are compared. If a match is found, a resource dependency is created. The different dependencies are then added to the dependency model. The analysis for QoS and price dependencies is based on [3]. It starts with a reduction of the service workflow based on workflow patterns. Formulas for calculating composite QoS and price values for the respective workflow patterns are selected and an aggregation

formula is created. Finally, a dependency is created for each composite QoS and price value. Following the discovery of dependencies the created dependency model can be refined in a manual modeling step.

2.2 Dependency Discovery

The goal of our work is to provide composite service providers with information about service dependencies in a composition. This information should facilitate the management of SLAs. A SLA contains a list of parameters such as time, price, location, and quality of service provisioning. Dependencies occur with regard to these parameters: e.g. the composite service price depends on the atomic service prices; provisioning of the first atomic services in the composition can be started as soon as the provisioning of the composite service is initiated; provisioning of two atomic services needs to be started or finished at the same time. These brief examples show that different types of dependencies exist and that fine grained dependency information is required.

The discovery of dependencies is specific for each dependency type. We will now describe the discovery of time and resource dependencies in more detail including the selection of services for service pair creation as well as the type of dependency being created. Time dependencies are expressed based on time relations as used in project management [4] or as defined by Allen [5]. For resource dependencies the different resources are listed. An overview of the mappings for creating dependencies is presented in Table 1. The creation of aggregation formulas for composite QoS and price values is achieved as described by [3]. Due to space limitations we would like to point the reader to the respective work for further information.


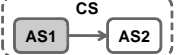

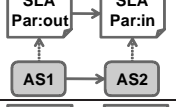
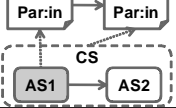
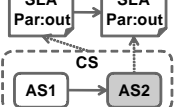
Horizontal Time Dependencies are created between each pair of services, where one service is directly connected to another service in a path. For each pair a *finish-to-start* time dependency is created between the earlier and the later service.

Vertical Time Dependencies are created between the composite service and the first and last atomic service within a path. Between the composite service and the first atomic service a *start-to-start* time dependency is created. Between the last atomic service and the composite service a *finish-to-finish* time dependency is created.

Horizontal Resource Dependencies are created between atomic services, which are directly or indirectly connected within a path. To check whether two services have a dependency the output of the preceding service is compared to the input of the succeeding service. If a match is found a resource dependency is created. Information on input and output of services is available from their SLAs.

Vertical Resource Dependencies are created between the composite service and an atomic service. For each path the composite service input and output is compared to the atomic services input and output. A resource dependency is created with all atomic services along a path, which have a matching input with the composite service input and which do not have a horizontal dependency regarding the matching resources. A further resource dependency is created with the last atomic service, which has a matching output with the composite service.

Table 1. Comparison of dependency model approaches.

Composite service construct	Description	Dependency model construct
	Two atomic services directly connected via control flow	Time dependency: finish-to-start
	Composite service and first atomic service in a path	Time dependency: start-to-start
	Last atomic service and composite service in a path	Time dependency: finish-to-finish
	Output of preceding atomic service matches input of succeeding service	Resource dependency: AS2.paramIn resourceDependent AS1.paramOut
	Input of composite service matches input of atomic service	Resource dependency: AS1.paramIn resourceDependent CS.paramIn
	Output of composite service matches output of atomic service	Resource dependency: CS.paramOut resourceDependent AS2.paramOut

2.3 Dependency Modeling

The dependency discovery algorithm produces a valid dependency model. However, there are several types of dependencies which cannot be discovered. This includes dependencies regarding the location for executing a service or QoS dependencies where no aggregation formula can be created automatically. Furthermore, time dependencies may exist between services which are not connected by the process flow. A concrete use case may have time constraints, which have to be modeled explicitly, i.e. the created dependency model is extended manually.

3 Architecture and Integration

In this chapter the architecture of the dependency management components as well as their integration into a service engineering toolchain are described. The components provide functionality for the creation, validation, and storage of dependency models (*Dependency Model Management*), the analysis and modeling of dependencies (*Dependency Analysis*), and the evaluation of the dependency model with respect to different events at runtime (*Runtime Dependency Evaluation*). An overview of the components is presented in Fig. 2. Details about their functionality are described below.

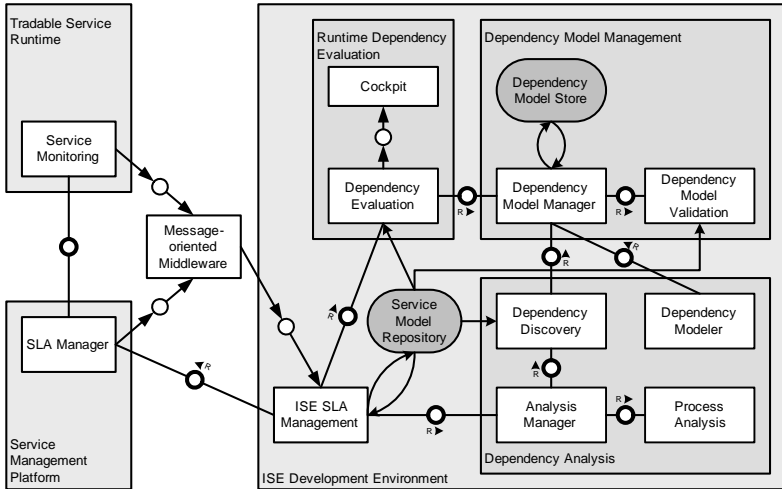


Fig. 2. Architecture Dependency Handling.

3.1 Dependency Model Management

The approach for managing service dependencies has at its core the dependency model, which is used to capture information about services and the dependencies that occur between them. The components, which are part of the dependency model management, are responsible for the creation, validation, and storage of dependency models and for making these models available to other components.

The *Dependency Model Manager* is the central component. It creates new dependency model instances for each new SLA negotiated for a composite service. It is also responsible for adding information to dependency models and making model information available to other components such as *Dependency Modeler* and *Dependency Evaluation*. The *Dependency Model Validation* component is responsible for validating the dependency model with respect to the defined constraints and the respective SLAs. An example is the validation of negotiated times which are discovered based on the workflow structure or modeled manually. Furthermore, validation regarding more general aspects is realized (e.g. each consumed resource needs to be provided by an entity). The final dependency model instances are stored in the *Dependency Model Store*.

3.2 Service Dependency Analysis

The analysis of dependencies is executed after creating the service composition and during the process of negotiating SLAs for the different services. It requires a process description and SLAs in the offer state (i.e. containing offered SLO values) as input. Our implementation is based on BPMN (Business Process Modeling Notation) process descriptions. BPMN represents a suitable means for modeling business processes from a business perspective at an abstract level. An alternative approach would be the usage of a BPEL (Business Process Execution Language) process notation. BPEL is, however, targeted at processes that are executed automatically and which are realized by web

services. Processes involving mainly human or machine tasks are typically not modeled using BPEL. The dependency analysis functionality is distributed between components supporting the automatic dependency discovery as well as dependency modeling. The analysis process and the involved components are presented in Fig. 3.

The *Analysis Manager* handles the process of dependency discovery. It is initiated by the composite service creator during the negotiation of SLAs with the consumer of the composite service as well as the atomic service providers. It retrieves the workflow description and SLA documents for the analysis and initiates the different steps of the discovery. The *Process Analysis* component is responsible for decomposing the process into linear paths leading from the start node all the way to the end node. These paths are used for the discovery of dependencies. The *Dependency Discovery* component realizes the different dependency discovery mechanisms. They include all horizontal and vertical dependency evaluation tasks as described in section 2.2. The implementation is based on the generated paths and SLA information. When it discovers a dependency it requests the *Dependency Model Manager* to add the respective information to the dependency model. Once the information has been added to the model, it is stored in the *Dependency Model Store*. From there it can be accessed for further handling.

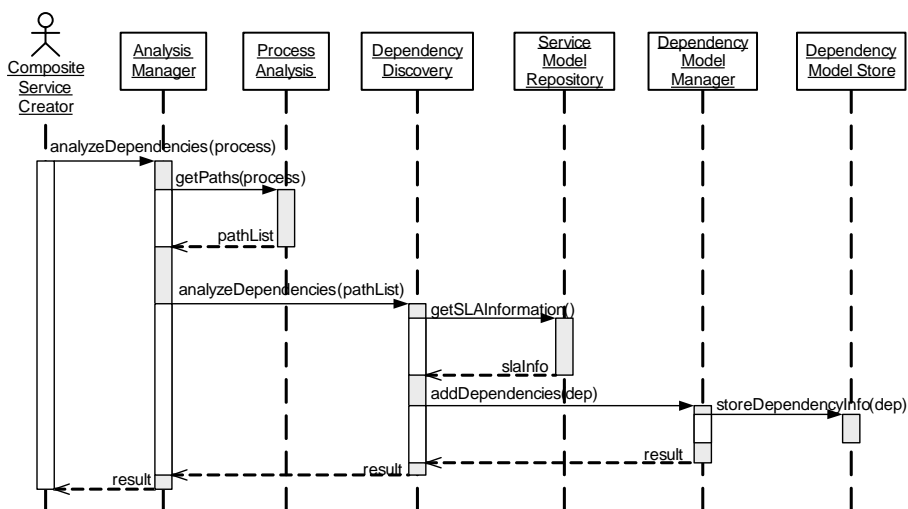


Fig. 3. Process and components for analysis of dependencies.

The *Dependency Modeler* provides dependency modeling functionality for the composite service creator. It enables the creation of new as well as the adaptation of existing dependency models. It was realized as a graphical model editor. The modeling process is initiated by the composite service creator. As a first step the *Dependency Modeler* requests a dependency model from the *Dependency Model Manager*. Once the model is available, the composite service creator uses the editing functionality to add, remove, or modify dependency and service information in the dependency model.

3.3 Runtime Dependency Evaluation

The *Runtime Dependency Evaluation* component is responsible for the evaluation of dependency information at runtime. The occurrence of SLO violation information requires the determination of effects of this violation on other services (atomic or composite service). Requests for renegotiating an SLA need to be evaluated with regard to effects on other services before accepting them.

The runtime evaluation of dependencies is initiated by the *ISE SLA Management* component calling the *Dependency Evaluation* component which executes the evaluation process. It requests the relevant dependency model from the *Dependency Model Manager* and evaluates it. Since the runtime dependency evaluation is not in the focus of this paper we do not present more details about this.

3.4 Integration with Service Engineering Toolchain

The different dependency management components are integrated into the ISE development environment, a tool created for the modeling of services. This enables proper handling of dependencies for composite service providers while modeling their services. Within the ISE development environment the dependency analysis components also have access to the necessary information for executing the analysis (i.e. the BPMN process description and SLA information). The *Dependency Modeler* tool is also integrated into the ISE development environment. The *Runtime Dependency Evaluation* component is integrated with the *ISE SLA Management* components, which handle the integration with the *Service Monitoring* on the *Tradable Service Runtime (TSR)* and the *SLA Manager* on the *Service Management Platform (SMP)* respectively. The TSR provides the service runtime infrastructure while the SMP offers service marketplace functionality.

4 Evaluation

In the first part of the evaluation we discuss the performance of the algorithms used for the automatic discovery of dependencies. In the second part we present a set of test cases to better illustrate the different steps of the dependency analysis process. The results of both parts are discussed in a third section.

The performance measurements and test case handling were executed using the workflow of a composite healthcare service (see Fig. 4). The scenario is based on a healthcare workflow presented in [6]. In this scenario a patient undergoes several examinations at a healthcare center. The different examinations are executed by different medical service providers. Further services include the analysis of blood samples, creation of documentation, and transport of the patient.

4.1 Performance Considerations

As a first step we measured the times taken for the different tasks of the dependency discovery approach for the healthcare service: path creation (5 ms), horizontal (7 ms)

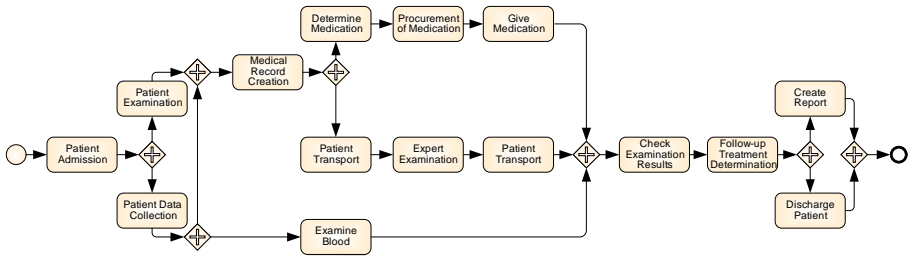


Fig. 4. Workflow of composite service - Stationary Patient Check-up.

and vertical (2 ms) time dependencies, horizontal (3 ms) and vertical (25 ms) resource dependencies. The results show that all tasks are executed within a few milliseconds.

Furthermore, a number of measurements were made to test the scalability of the approach. We modeled 5 business process workflows (P1..P5) of different complexity (number of nodes, number of splits and joins). We counted the number of artifacts created during dependency analysis and measured the times for creating relevant service pairs. The results presented in Table 2 show that with increasing workflow complexity the number of paths as well as duplicate time and resource pairs increases strongly. Thus it is necessary to ensure that the further analysis of pairs is only executed for pairs which have not been handled before. The results also show that the discovery of dependencies in relatively complex services is executed in less than a second.

Table 2. Measurement results.

	P1	P2	P3	P4	P5
Nodes	12	31	36	55	87
Split/join	1/5	13/12	28/9	17/32	94/44
Created paths	21	60	952	1933	908
Duplicate time pairs	76	102	7387	14554	5535
Non-duplicate time pairs	14	32	45	74	91
Duplicate resource pairs	222	139	33686	65569	20951
Non-duplicate resource pairs	57	103	422	679	916
Time to get time pairs (ms)	1.7	2.0	78.7	289.7	293.8
Time to get resource pairs (ms)	0.9	2.1	188.7	554.0	153.5

4.2 Test Case based Evaluation

A number of test cases serve as the base for validating the different steps of the dependency analysis process. Each test case is illustrated with a brief example. For the analysis of dependencies the service workflow (see Fig. 4) and SLA descriptions are needed. Due to space limitations only excerpts of SLAs of services relevant for the presented examples are listed in Table 3.

TC1 - Path Creation: The composite service workflow is decomposed into linear paths. *Results:* List of 10 paths reaching from the start to the end of the process. One

example path is the following: *Patient Admission - Patient Data Collection - Examine Blood - Check Examination Results - Follow-up Treatment Determination - Create Report*

Table 3. Sample SLA information.

Service	Input resources	Output resources
Patient Admission	-	patient ID
Examine Blood	patient ID, blood sample	laboratory test result
Create Report	medical record	examination report
Stationary Patient Check-up	-	examination report

TC2 - Horizontal Time Dependencies: Pairs of directly connected services are created along the paths. Duplicate pairs (e.g. pair *Follow-up Treatment Determination - Create Report* occurs in 5 paths) are removed. Time dependencies of type *finish-to-start* are created for each pair. *Results:* List of horizontal time dependencies. One horizontal time dependency between *Patient Data Collection* and *Examine Blood* is shown in Table 4.

TC3 - Horizontal Resource Dependencies: All pairs of different services along the paths are created. Duplicate pairs are removed. All pairs are analyzed with regard to matching input and output resources. Information about input and output resources is taken from the negotiated SLAs (see Table 3). Resource dependencies are created when matching resources are found. *Results:* List of horizontal resource dependencies. One resource dependency between the service *Patient Admission* and *Examine Blood* is shown in Table 4.

TC4 - Vertical Time Dependencies: Creation of vertical time dependencies between the composite service and the first (*start-to-start*) and last (*finish-to-finish*) atomic services in the paths. *Results:* List of vertical time dependencies. One vertical time dependency between *Stationary Patient Check-up* and *Patient Admission* is shown in Table 4.

TC5 - Vertical Resource Dependencies: All atomic services along the paths are analyzed with regard to matching input and output resources with the composite service. Dependencies are created for matching resources if no horizontal dependency exists regarding the matching resources. *Results:* List of vertical resource dependencies. One vertical resource dependency between *Create Report* and *Stationary Patient Check-up* is shown in Table 4.

TC6 - Dependency Model Extension: Manual creation of a location dependency between the *Patient Transport* service and the *Expert Examination* service. *Results:* One location dependency between *Patient Transport* and *Expert Examination* (see Table 4).

4.3 Discussion

In this chapter we demonstrated the general feasibility of the approach to create dependency models. We first presented an overview about performance measurements. One result of the measurements was that with increasing complexity of the workflows not

Table 4. Dependencies of healthcare process.

Antecedent - Dependant	Dependency	Description
Patient Data Collection - Examine Blood	time	endTime <i>finish-to-start</i> startTime
Patient Admission - Examine Blood	resource	patient ID
Stationary Patient Check-up - Patient Admission	time	startTime <i>start-to-start</i> startTime
Create Report - Stationary Patient Check-up	resource	examination report
Patient Transport - Expert Examination	location	endLocation <i>equals</i> startLocation

only the number of paths and relevant service pairs increased, but that a proportionally large amount of time would be necessary for handling duplicate services. Thus, they need to be removed. However, we also showed that the time needed to analyze relatively complex processes is still less than a second, which allows to use the approach at design time.

As a second part of the evaluation we applied different test cases which demonstrated the functioning of our approach. We showed a sample path created during the execution of the dependency discovery as well as a number of different dependencies. In total this scenario produces 40 time and resource dependencies, which are discovered automatically. Two more location dependencies can be modeled. A manual handling of all these dependencies would be very time consuming and error prone. In more complex processes the number of dependencies will be much higher, which renders a manual handling of dependencies even more difficult. Our semi-automatic process facilitates this.

5 Related Work

The handling of dependencies between services has been addressed for a variety of purposes including the automatic composition of services [7], the optimization of sequencing constraints of composite services ([8, 9]), root cause and impact analysis ([10, 11]), and SLA management ([3, 12]).

Wu et al. [8] present an approach for modeling and optimizing the synchronization dependencies of activities in business processes. A synchronization model, which contains dependency information, is used to support activity scheduling in business processes. In contrast to our approach automatic discovery of dependencies is not supported. In [9] the authors discuss control and data dependencies in business processes and argue that they form the base for sequencing constraints in business processes. They present an approach for deriving control dependencies from semantically annotated business activities by evaluating their pre-conditions and effects. Input and output parameters of business activities form the base for data dependencies. This approach differs from our approach in several ways: While our resource dependencies are similar to the data dependencies of their work, we also support dependencies regarding time, location, QoS, and pricing information. Furthermore, their approach is limited to depen-

dependencies between atomic services while our work also supports dependencies between atomic and composite services.

Ensel and Keller [10] introduce an approach to handle dependencies between managed resources (e.g. web application server, database, operating system) in a distributed system. The goal is to support root cause as well as impact analysis for service failure situations. Dependencies are represented in a distributed dependency model which captures the dependencies and attributes of these managed resources. However, no work is presented with regard to the discovery of service dependencies. The MoDe4SLA [11] approach supports the handling of response time and price dependencies of composite services on its atomic services. The goal of the system is to support root-cause analysis for problems caused by atomic services. The dependency information is captured by a modeling approach. The discovery of dependencies is not supported.

The COSMA approach [3] supports the providers of composite services to manage their SLAs. Dependencies between composite QoS values and atomic ones are expressed using aggregation formulas. The aggregation formulas for the different QoS values are automatically derived from the process description. Further constraints need to be added manually or from configuration files. In contrast to our work, COSMA focuses only on the relationship between composite services and atomic services, but dependencies between atomic services are not handled. Dependency types such as resource, location, and time are not covered. However, our approach to QoS and price dependency discovery is based on COSMA. Karnke et al. describe an agent-based approach to managing SLAs in value chains [12]. The focus is on SLA based resource management in hierarchies of service level agreements. As part of the agent-driven negotiation process, dependencies between services are considered. However, no work has been presented regarding the discovery of dependencies.

6 Conclusions

The approach presented in this paper enables management of service dependencies. We have shown that different types of dependencies can occur in parallel within complex service compositions representing business processes. It has been demonstrated that a significant subset of these dependency types can be automatically extracted from information provided by the process description and the SLAs negotiated between the involved service providers and consumers. Based on the different characteristics of service dependencies specific algorithms have been identified for automatic dependency discovery. These algorithms are embedded into the process of dependency management implemented by the presented architecture for dependency handling, particularly in the dependency analysis component. In the process the evaluation of service dependencies at runtime is foreseen. The automatically discovered dependencies are stored in a dependency model and are made available for runtime dependency evaluation. The presented performance measurements prove the applicability of our algorithms for dependency discovery at runtime, since the processing time is in the range of few seconds even for complex processes of up to 100 services. The test case based evaluation demonstrated the feasibility of our approach, illustrating the complexity of the dependency discovery and showing created artifacts.

In the future we will consider additional types of dependencies with respect to characteristics, detection algorithms and modeling. Further use case studies will be carried out to prove the applicability and practical relevance of our approach. Finally, the runtime dependency evaluation will be implemented.

Acknowledgements

The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference “01MQ07012”. The authors take the responsibility for the contents.

References

1. Winkler, M., Schill, A.: Towards dependency management in service compositions. In Filipe, J., Marca, D.A., Shishkov, B., van Sinderen, M., eds.: ICE-B 2009 - Proceedings of the International Conference on e-Business, Milan, Italy. (2009)
2. Sell, C., Winkler, M., Springer, T., Schill, A.: Two dependency modeling approaches for business process adaptation. In Karagiannis, D., Jin, Z., eds.: Knowledge Science, Engineering and Management, Springer (11 2009)
3. Ludwig, A., Franczyk, B.: Cosma—an approach for managing slas in composite services. In Bouguettaya, A., Krueger, I., Margaria, T., eds.: ICSOC 2008. (2008)
4. PMI: A Guide to the Project Management Body of Knowledge (PMBOK Guide). 4 edn. Project Management Institute (2008)
5. Allen, J.F.: Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11) (1983) 832–843
6. Reichert, M., Bauer, T., Fries, T., Dadam, P.: Realisierung flexibler, unternehmensweiter workflow-anwendungen mit adept. In Horster, P., ed.: Proc. Elektronische Geschäftsprozesse–Grundlagen, Sicherheitsaspekte, Realisierungen, Anwendungen. (2001) 217–228
7. Zhou, J., Pakkala, D., Perala, J., Niemela, E.: Dependency-aware service oriented architecture and service composition. In: IEEE International Conference on Web Services. (2007) 1146–1149
8. Wu, Q., Pu, C., Sahai, A., Barga, R.: Categorization and optimization of synchronization dependencies in business processes. In: Proceedings of IEEE 23rd International Conference on Data Engineering (ICDE’07). (2007) 306–315
9. Zhou, Z., Bhiri, S., Hauswirth, M.: Control and Data Dependencies in Business Processes Based on Semantic Business Activities. In: Proceedings of iiWAS2008, ACM (2008)
10. Ensel, C., Keller, A.: An approach for managing service dependencies with xml and the resource description framework. *Journal of Network and Systems Management* 10 (2002) 147–170
11. Bodenstaff, L., Wombacher, A., Reichert, M., Jaeger, M.C.: Monitoring Dependencies for SLAs: The MoDe4SLA Approach. In: IEEE SCC (1). (2008) 21–29
12. Karaenke, P., Micsik, A., Kirn, S.: Adaptive sla management along value chains for service individualization. In: Proceedings First International Symposium on Services Science (ISSS’2009). (2009)

**SOA APPLICATIONS - HOMECARE
AND EMERGENCY SUPPORT**

Service Tailoring: Towards Personalized Homecare Services

Mohammad Zarifi Eslami, Alireza Zarghami, Brahmananda Sapkota
and Marten van Sinderen

Information Systems Group, University of Twente, Enschede, The Netherlands
{m.zarifi, a.zarghami, b.sapkota,
m.j.vansinderen}@ewi.utwente.nl

Abstract. Health monitoring and healthcare provisioning for the elderly at home have received increasingly attention. Since each elderly person is unique, with a unique lifestyle, living environment and health condition, personalization is an essential feature of homecare software services. Service tailoring, which is creating a new service to meet individual requirements may be achieved in a cost-effective and time-efficient manner if new services can be configured and composed from already existing services. In this paper, we propose an effective service tailoring process and architecture to personalize homecare services according to the individual care-receiver's needs. In addition, we present a scenario to highlight the need for service tailoring and to demonstrate the feasibility of the proposed approach.

1 Introduction

As computer networks are becoming widespread, the number of distributed services available over networks grows rapidly [1]. However, these services are usually designed for a general purpose, user or situation. In reality, different people have different requirements, therefore they prefer personalized services. The requirements for personalization reflect what the user wants, needs, and likes, all of which may depend on the context at hand (for example, location, physical characteristics of the environment, available resources, people nearby). Personalization also has an evolutionary aspect as requirements –demands, needs and preferences– of users change over time. Services have to operate in a constantly evolving environment of people, content, electronic devices, and legacy systems [2].

Thus, application functionality provided to users as services should (1) be aligned with the uniqueness of each user's requirements, (2) evolve with changes in these requirements, and (3) take the dynamic context of the user into account. Ideally this would call for tailor-made services; however developing such services from scratch would be economically and technically infeasible. A better approach would be to reuse existing services, configure and compose them to satisfy the unique requirements of each individual user. *Service tailoring* is a way of creating a new service to satisfy the specific requirements of an individual user. Although service tailoring is an essential feature in any application domain, we focus on homecare application services. Tailorability has been studied extensively in the context of

specific technologies and applications [3-7], but those approaches are not suitable for homecare domain as homecare services have their own specifications such as high dynamicity of user's requirements and low level of user's technical skills [8].

To support well-being, health monitoring and independent living, there is an increasing tendency to provide homecare services for the elderly, especially in developed countries [9-11]. Several technological challenges concerning homecare applications have been previously studied, but our focus is to address the uniqueness of each elderly person by applying the service tailoring approach. Current homecare systems are generally stand-alone for treating specific diseases, assuming a 'standard' patient and in a static situation [8]. However, in reality each user is unique in the way he or she experiences or is affected by a disease or disability. This is not only because of each individual's mental and physical condition, but also because of the social and physical environment. Current automated homecare systems are often technology-driven. They can be difficult to use by non-technical users and difficult to change or adapt when new requirements arise. The key question therefore is: how can services be tailored to the requirements of the users in this domain, namely, care-receivers and caregivers?

In this paper, we propose a service tailoring approach for the homecare domain. The proposed service tailoring approach allows creating a new homecare service through composing and configuring existing services, based on the user's specific requirements. Also, earlier tailoring results can be incrementally changed through subsequent applications of tailoring to match evolving user's requirements.

In Section 2, we define a scenario to motivate service tailoring for homecare. We then define a generic service tailoring process in Section 3. This process is 'generic' in the sense that it is independent of the knowledge and skills of the person who does the tailoring. A tailoring architecture for implementing the tailoring process is proposed in Section 4. This architecture identifies the components of a possible service tailoring platform. In Section 5, by using a tailoring example applicable to our scenario, we show how the proposed service tailoring process and architecture aid in creating a personalized homecare service. Finally, we conclude our paper and discuss possible future work in Section 6.

2 Example Scenario

To highlight the need for tailored homecare services, we use the following scenario:

“Jan and Linda are 74 and 67 years old respectively. Despite their age and having various medical conditions, they prefer to remain living in their own home. They need to take certain medicines at certain times. However, they suffer from Alzheimer's disease and may not remember when to take which medicines and how much. In this situation, a reminder service may help them remember the prescribed time and a dispenser service may help to take the correct dosage of the right medicines. There may be situations in which the reminder and dispenser services may not be sufficient to ensure that Jan and Linda actually take their medicines. Jan, for example, sometimes ignores the reminder and does not take his medicine because he is too disoriented. In such situations, assistance or help from other (voluntary or professional) people is necessary. An alarm service may help detect such situations

and trigger a request for external help. Moreover, Jan has a hearing problem and uses a hearing aid, and Linda cannot see well and so she must use glasses.

During the morning, they have their breakfast and Jan reads the newspaper on the screen while Linda listens to the radio through the multimedia system. (There are two multimedia interactive systems, one in the kitchen and one in the living room. These systems include TV, radio, reminder texts and news, which users select via a touchscreen). Then, they visit a park close to their home and meet their friends. They usually have their lunch in a nearby restaurant. They spend most of their evenings at home; Jan watching TV and Linda reading books. During the weekend, their children usually come to visit them and Jan and Linda have other things to do and so they have a different schedule than weekdays one. ”

As this example scenario shows, Jan and Linda have individual requirements and preferences, and even the same person has different requirements and preferences depending on the current situation and time. For example, for the medicine reminder service, during the morning they prefer to have the reminder on the screen in the kitchen or from the radio. After breakfast, if they go out as usual, they prefer to receive the reminder on their mobile phone, and in the evening, Jan prefers to receive his reminder on TV while Linda wants to receive it through her wheelchair vibrator. Therefore, the desired services have to deliver their functionalities in various ways in response to changing requirements, preferences and circumstances.

3 Tailoring Process

For doing tailoring, we assume three distinct types of users depending on the individual knowledge and skill sets they possess: care-receiver (elderly people), caregiver (family doctor, nurse or relative), and service developer (someone proficient with the service-tailoring facility and the underlying technologies). A care-receiver has contextual knowledge about his or her own needs and physical environment, but probably possesses little domain or technical knowledge. A caregiver has domain knowledge about healthcare practices and procedures, but probably possesses little contextual and technical knowledge. Finally, a service developer has technical knowledge about service modeling and technology, but probably possesses little contextual and domain knowledge. Therefore, although the proposed tailoring process is common among the different types of users, they may need different interfaces (for tailoring) and may have various authoritative roles (levels of tailoring). However, actors may have distinct roles when they are interacting with the system: Jan, for example, is a care-receiver, but because of his knowledge in IT may also take on the role of service developer.

The tailoring process consists of six different steps. These steps are illustrated using BPMN notation in Figure 1, with each step having a corresponding activity. We do not show a data flow in the figure because we present the process focusing on tailoring platform view, regardless of its interaction with the user.

Of these steps, Steps 2, 3 and 4 are further refined into multiple activities. These six steps and their constituent sub-activities are explained below. To show the feasibility of the process and to make it easier to follow, each step is exemplified in Section 5.

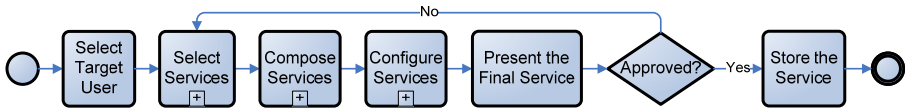


Fig. 1. Service tailoring process in BPMN notation.

Step 1: Selecting the Target User

Our service tailoring process starts with specifying for who the service is being created. The user data, such as age, their specific functional requirements and health status, is stored in a user profile. When a target user is selected in this step, the data stored in the corresponding user profile is used for tailoring the service in the other steps to suggest the candidate services, with proper configuration, to the user. For example, if Jan is chosen as a target user (care-receiver), the system knows that he has a hearing problem and will not offer him any services that use sound.

Step 2: Selecting the Services

Originally, the web was designed for human-human communication, but later it also became a machine understandable information space [12]. This is also true for web services. The goal of designing web services was not only to interact with humans, but also with other applications [13]. This means that data exchanged among services carry semantics. So the semantics has been used to provide machine-understandable information and enable services or frameworks to exploit machine-understandable information to facilitate interoperability. The semantics help systems to automate various aspects such as discovery, invocation and composition. Moreover, utilizing semantic similarities among components and services improves adaptability in selection. For adding semantics to our tailoring process, we exploit ontology. The ontology aids the tailoring framework to understand user requirements and thereby discover and suggest services which suit the best to these requirements.

Creating a new integrated service by a nontechnical user is a difficult task: the user needs to know which existing individual services do what. To make the job easier for the user, a goal-based method [14, 15] can be used. Goal-based methods have been used in different areas of computer science to identify stakeholder's objectives, discover requirements for software systems and guide the system's behavior. In service-oriented computing goals can be used to express users' requirements.

The second step includes five activities as shown in Figure 2:

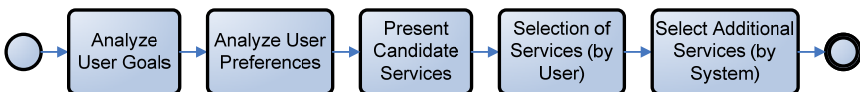


Fig. 2. Step 2: Selection of services.

In this step, first user specifies care-receiver's goals (requirements and preferences). The tailoring platform utilizes the goal ontology for homecare domain to *analyze user goals and preferences*. The aim of using a goal ontology is to minimize possible terminological heterogeneities due to autonomously created goals (user requirements) and tasks (existing services). Besides a goal ontology, a task ontology is needed to associate existing services with their support functions (tasks). Once a user asks for a service (determine his or her requirements), the tailoring platform matches the user goal (requirement) with the proper task existed in the task ontology. Then it tries to find and select the services associated with that task and suggests it to the user. Therefore, a goal based method can be used to help users in service requests. In this manner the users have a way of expressing what they want at a higher abstraction level.

As shown in Figure 3, the user goals are more objective and nonfunctional, which is easier to be understood by a non-technical user. Whereas tasks are functional but not concrete, and existing services are more concrete.

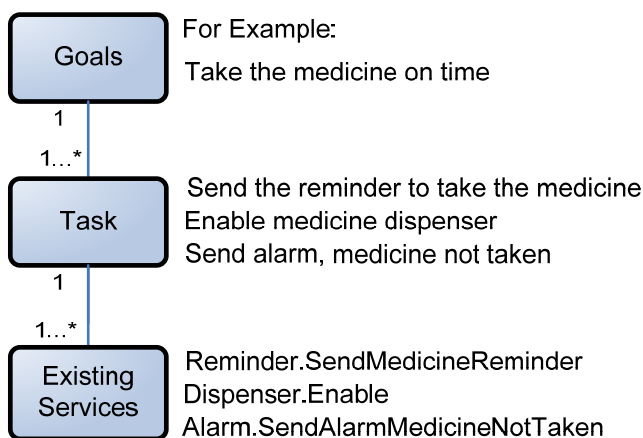


Fig. 3. How user goals wind up to the system services.

After identifying the *user goals and preferences*, for *presenting the candidate services*, the service tailoring platform finds and suggests several suitable services. To do so, the service tailoring platform utilizes a repository of services and a task ontology which depicts functionalities provided by these services. Each goal in the goal ontology is associated with a set of tasks in the task ontology. Therefore, after identifying the goal, the tailoring platform finds the most suitable tasks and then suggests relevant services to the user.

The tailoring platform also uses the recommendations besides the goal based method to suggest certain services to the user. The recommendations, using other care-receivers' information to suggest certain existing services which are used by them, in a similar context [16]. As an example, assume that Ben is another elderly person who lives at his own home with similar requirements as Jan. They both use medicine dispenser service. If Ben uses an alarm service with the medicine dispenser, the tailoring platform of Ben recommends the use of alarm service to Jan's tailoring

platform through the network of homecare applications. Then the alarm service will be recommended in the service selection step. However, services in homecare must be selected carefully; any recommendation must be confirmed by an authorized user as will be explained in Step 5.

Later, for *selecting services*, the caregiver selects his desired services from the set of services suggested in the previous step by the tailoring platform. In this way, the user selects explicitly desired services. However, the tailoring platform may also *select additional services* because of service dependencies. Some services may need or recommend other services' functionalities to work, as a precondition. This means that –based on user-selected services– the system adds or removes certain other relevant services. For the second part, we use the service bundling techniques [17, 18]. In a service bundle, dependencies between services are utilized to help a user to select and discover services automatically and without the user intervention.

Step 3: Composing the Services

The third step includes five activities as shown in Figure 4:



Fig. 4. Step 3: Composing of services.

In this step, the tailoring platform composes the chosen services to satisfy the user requested requirements. This composition is done based on the given *user goals and stored preferences* in the user profile. Then this *composite service(s) will be presented* to the user through our tailoring platform. Predefined service compositions, stored in the Composition repository by the service developer. To suggest the proper composition, we can exploit the Event-Condition-Action (ECA) rules. An ECA rule has the general syntax on event, if condition, do actions. The event part specifies when the rule should be triggered, which, in our case, can be the user goals. The condition part specifies the conditions of this event, which, in our case, can be again the user goals or preferences (configuration of services). The action part states the actions to be performed automatically if the condition holds. In our case, this action can be the suggested composition to the user. Therefore, ECA rules can be expressed as conditions for decision making and can be used for configuration of services or composition [1, 19].

After presenting the suggested compositions, the *user should select one* of them. The user can also *edit the desired composition*, for example, changing the candidate services in the composition or their order in the suggested composition. To allow user to edit the composition, we use a mashup-like technique [20, 21]. Unlike developer-centric composition techniques, mashup provides a user-friendly graphical approach to compose services and applications.

Step 4: Configuring the Services

In general, to configure the services two different methods can be used: rule based

and learning user behavior. The rule based method is necessary because it may not be possible to derive rules for new users or applications from user behavior. On the other hand, learning user behavior by learning user preferences (in different contexts) from history information is also necessary. It might be difficult to define a generic rule that is applicable to every user. Therefore, learning the user behavior can be used to update the predefined rules. We use both methods in our process.

The fourth step includes four activities as shown in Figure 5:

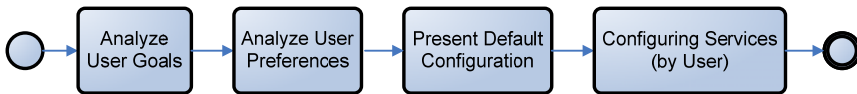


Fig. 5. Step 4: Configuring the services.

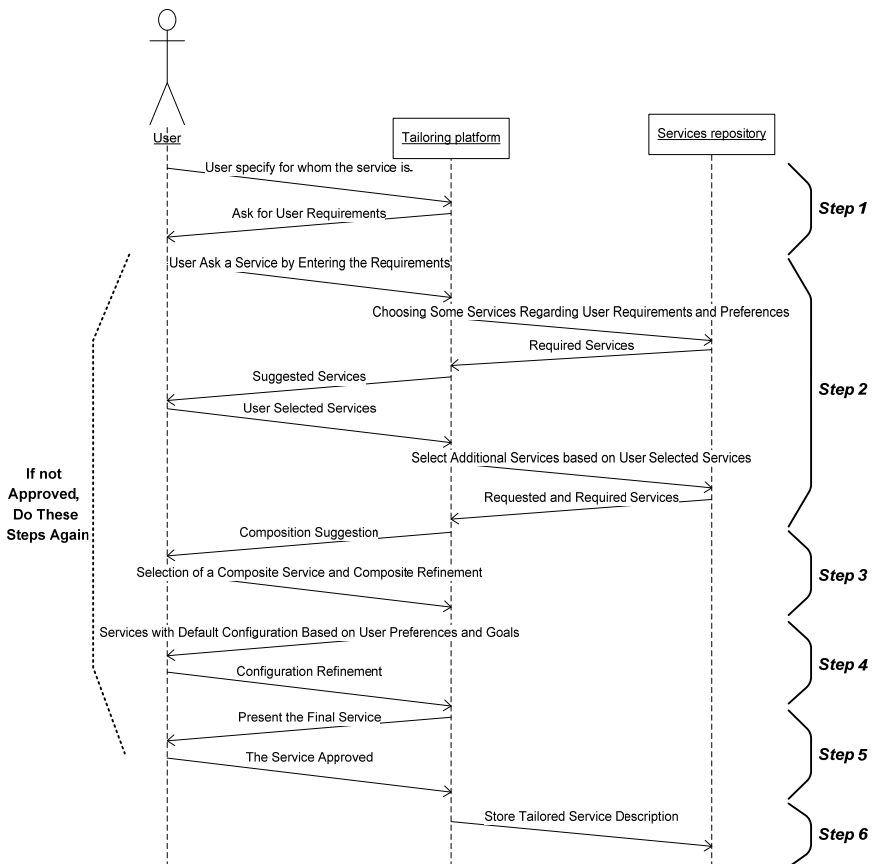


Fig. 6. Sequence diagram of the service tailoring process.

The services selected in the previous step are presented to the user based on the *default configuration* derived from *user preferences* as indicated in the service profile and *user goals*. However, the user is also allowed to (*re*)configure the basic services

to specify his or her current preferences and needs. To configure the services, user can use a configuration interface.

Step 5: Presenting the Final Service

In this step, the final newly created service will be presented to the user. After this step, the user confirms whether the service meets his or her requirements. If the user does not approve any of the suggested compositions, the process returns to Step 2.

Step 6: Storing the Service

Finally, once the user has approved the suggested service, the description of this newly created service will be stored for later use. Also, user preferences in the user profile may change because of configuration of the services by the user.

The sequence diagram shown in Figure 6 summarizes our discussion about the service tailoring process. It shows the interaction between the ‘user’ and the ‘tailoring platform’. Although the ‘Service repository’ is a part of tailoring platform, because of its importance we presented it separately from the tailoring platform in Figure 6.

4 Tailoring Architecture

To support the tailoring process described in Section 3, we identify the required components and propose a tailoring architecture for homecare services. The tailoring architecture, as presented in Figure 7, has three categories of components: *TailoringManager*, *Repositories* and *Modifiers*. These components and their relationships are described separately in the remainder of this section.

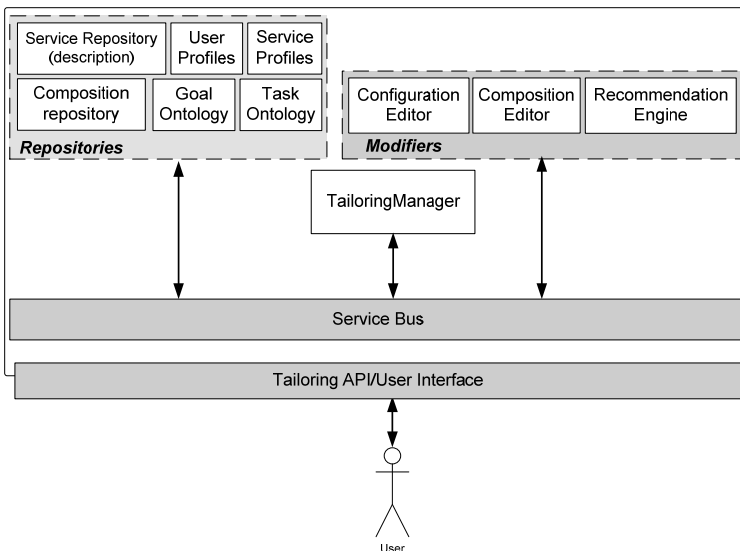


Fig. 7. Service tailoring architecture.

- **TailoringManager.** This component is responsible for receiving and resolving the user requests by utilizing goal and task ontology, suggesting existing services to the user by utilizing service repository, making initial configuration of services by utilizing user profile and service profiles, suggesting composition by utilizing composition repository, and storing the final services in the service repository.
- **Repositories:**
 - **Service Repository.** The service repository is a repository of existing services which stores detailed information (for example, in the form of WSDL) about several patient-neutral homecare services.
 - **User Profile.** This component stores user information. It has two parts: static part which keeps the static information of user such as name and gender, and dynamic part which keeps the evolving information of the user such as user specific requirements.
 - **Service Profile.** This component stores the default configuration of the services.
 - **Composition Repository.** This is the repository of processes which keeps different composition of services.
 - **Goal Ontology.** The Goal ontology is used for helping users to specify their desired goals.
 - **Task Ontology.** The tasks are supported functionalities of existing services which are defined in the Task ontology. In the Task ontology, each functionality offered by of a service is associated to one or more tasks (e.g., reminding, alarming, etc.). The service developer defines these goals and tasks ontology.
- **Modifiers:**
 - **Configuration Editor.** This component supports the user (caregiver or service developer) in defining and alerting the service configurations. These configurations can be stored in the service profiles repository.
 - **Composition Editor.** This component support user (caregiver or service developer) to define processes. These processes are stored in the composition repository.
 - **Recommendation Engine.** This component recommends services to the users (caregiver or service developer) in service selection and composition steps. To do so, we assume all the instances of our homecare systems are registered in a central server and they can make inquiry about the services which are used by each of these instances anonymously. Assuming the central server, enables all of these recommendation engines to maintain a list of similar anonymous users and their utilized services, by exploiting the information preserved in the user and service profiles.

5 Run through Example

In this section, we show the support provided by the proposed tailoring process and architecture for meeting the requirements derived from the example scenario

presented in Section 2.

“Assume that Jan is prescribed to take certain medicines at certain times. Nancy, his caregiver, wants to create a service to assist Jan in taking his medicine at right times.”

We follow the service tailoring process step by step to see how Nancy creates the desired service for Jan:

Step 1: Nancy, by choosing Jan’s profile, specifies through the tailoring interface that a new service to be created is for Jan. By receiving this information, the tailoring platform looks at Jan’s profile and finds his specific requirements and preferences, for example, Jan has a hearing problem and will not be able to use any services or devices which utilize sound as a means to convey messages to him.

Step 2: As shown in Figure 8, Nancy sends a request to the tailoring platform for a service which can help Jan to take his medicine at right times. For this purpose, she chooses a goal among the ones listed by the system and presented in the screen. Next, she chooses sub-goals to refine or finally define her goal. For example, from the goals list, she chooses *reminder*, and then from the new sub-goals’ list she chooses *reminder for medicine*. The tailoring platform, after receiving and analyzing this request and considering the fact that Jan can not use sound related services, suggests a list of services to Nancy. She chooses a *reminder service* which will send a reminder to Jan such that he will not forget the prescribed time to take a medicine. Further, she selects a *dispenser service* which will release right medicines such that Jan can take the right amount of the right medicines. After selecting the reminder and dispenser services by Nancy, the tailoring platform adds alarm service as an additional service to the user selected services. As reminder and dispenser services are bundled with the *alarm service*, which helps to detect hazard situations (in this case ignoring the reminder and not taking the medicines) and to trigger a request for help.

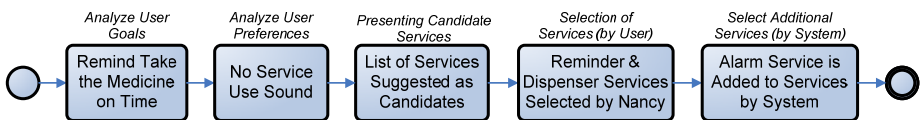


Fig. 8. Step2: Selection of services for the example.

Step 3: As shown in Figure 9, the tailoring platform returns a set of possible compositions, based on user goals and preferences. For example, because Jan prefers to have reminder three times, the composition will change in such a way that the reminder service sends the reminder three times. The tailoring platform proposes different compositions such as P1 and P2, because all these compositions can satisfy user’s requirements and preferences. This means that the composite service P1 first sends a reminder message to Jan and then enables the dispenser to let him take the medicine, whereas the composite service P2 first enables the dispenser and then sends the reminder message to Jan. Then, Nancy chooses P2 as a one of the possible compositions. After choosing P2, she may want to modify it, for example, because the medicine is pain killer and ignoring of taking it by Jan is not a hazard situation, so

Nancy removes the alarm service from the P2.

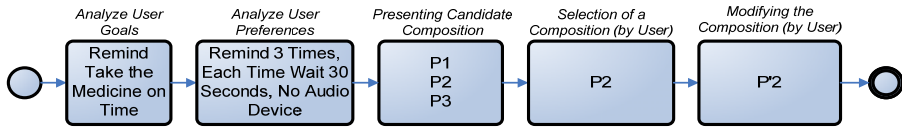


Fig. 9. Step 3: Composing the services for the example.

Step 4: As shown in Figure 10, the tailoring platform returns three selected services with default configuration to Nancy. These default configurations are done based on the user goals and preferences. For example, Jan prefers a reminder to him be repeated three times, allow 5 minutes for him to react, deliver this reminder as text and deliver it on the TV. So the default configuration for the reminder will be to send the output to the URI of TV and repeat each 5 minutes if the user does not respond. Then Nancy adds other configurations, such as reminding Jan to take the medicines from 10th May to 30th June 2010, everyday at 10 AM & 8 PM. She can also edit the default configuration. She changes, for example, the repeating time of the reminder from 5 to 10 minutes.

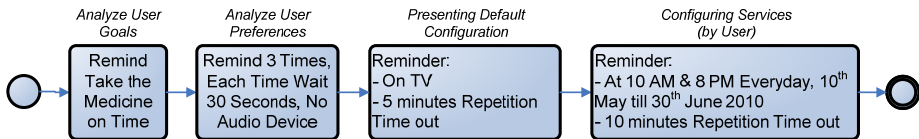


Fig. 10. Step 4: Configuration of services for the example.

Step 5 and 6: The tailoring platform presents the final tailored service to Nancy, and if she approves it the description of the service will be stored. The default configuration of the reminder service also will be changed, from 5 to 10 minutes repetition time out. If Nancy does not approve the final service from any of suggested compositions, the process returns to the Step 2 till Nancy approves the tailored service.

6 Conclusions and Outlook

Since every care-receiver is unique, personalization is one of the main concerns of homecare services. To achieve this, a service tailoring approach for homecare, with a corresponding process and supporting architecture, is proposed. Our service tailoring approach assumes the availability of basic care-related services which can be used as building blocks in the tailoring process.

The service tailoring process defines the flow of actions involving the user of the service tailoring platform to define a personalized homecare service for a care-receiver. Although we show the feasibility of the proposed service tailoring process, it is not claimed to be the only or most ideal solution for achieving service tailoring. In this paper, we describe a process and architecture for achieving personalized homecare services but further investigation is required to:

- **Developing a ‘generic’ Service Tailoring Process.** The service tailoring process is 'generic' in the sense that it is independent of the knowledge and skills of the user of the service tailoring platform. We foresee that the service tailoring platform should offer different interfaces to optimally support different types of users. Moreover, the service tailoring process should be tested to identify whether the service tailoring process has the right 'genericity', and can be extended and specialized for different user types.
- **Decreasing Privacy Risks of Recommendations.** Our service tailoring process may use recommendations from care-receivers. However, one of the main drawbacks of recommendations is privacy risks. To protect the privacy of care-receivers, distributed recommendation methods such as exploiting peer-to-peer networks would be interesting as our future work [22].

Acknowledgements

This work is part of the IOP GenCom U-Care project (<http://ucare.ewi.utwente.nl>) which is sponsored by the Dutch Ministry of Economic Affairs under contract IGC0816.

References

1. Fujii, K. and T. Suda, Semantics-based context-aware dynamic service composition. *ACM Trans. Autonom. Adapt. Syst.*, 2009. 4(2): 1-31.
2. Di Nitto, E., et al., A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, 2008. 15(3-4): 313-341.
3. Gehlert, A., et al., Towards Goal-Driven Self Optimisation of Service Based Applications, in *1st European Conf. on Towards a Service-Based Internet*. 2008, Springer-Verlag. pp. 13-24.
4. Hielscher, J., et al., A Framework for Proactive Self-adaptation of Service-Based Applications Based on Online Testing, in *1st European Conf. on Towards a Service-Based Internet*. 2008, Springer-Verlag. pp. 122-133.
5. Choi, O. and S. Han, Personalization of Rule-based Web Services. *Sensors*, 2008. 8(4): 2424-2435.
6. Kumanayaka, O. and N. Ranasinghe, Ontology based Web Service Personalization, in *Intl. Conf. on Information and Automation, ICIA*. . 2006. pp. 69-74.
7. Kazhamiakin, R., et al., Having Services "YourWay!": Towards User-Centric Composition of Mobile Services, in *First Future Internet Symp., FIS*. 2009, Springer-Verlag. pp. 94-106.
8. Zarifi Eslami, M. and M. Van Sinderen. Flexible home care automation: adapting to the personal and evolving needs and situations of the patient. in *3rd Intl. Conf. on Pervasive Computing Technologies for Healthcare, PervasiveHealth*. . 2009. London, UK. pp. 1-2.
9. Korhonen, I., J. Parkka, and M. Van Gils, Health monitoring in the home of the future. *Engineering in Medicine and Biology Magazine, IEEE*, 2003. 22(3): 66-73.
10. White, C.C., et al. Improving Healthcare Quality through Distributed Diagnosis and Home Healthcare. in *1st Transdisciplinary Conf. on Distributed Diagnosis and Home Healthcare, D2H2*. . 2006. pp. 168-172.

11. Kim, Y.-J., et al., Hallym Jikimi 3rd system: web-based monitoring for u-health care service, in 4th Intl. Conf. on Persuasive Technology. 2009, ACM. pp. 1-5.
12. Berners-Lee, T., Semantic Web Roadmap. 1998. Available at: <http://www.w3.org/DesignIssues/Semantic.html>.
13. Papazoglou, M.P. and D. Georgakopoulos, Introduction. Communications of the ACM, Special section: Service-oriented computing 2003. 46(10): 24-28.
14. Kangkang, Z., L. Qingzhong, and S. Qi. A Goal-driven Approach of Service Composition for Pervasive Computing. in 1st Intl. Symp. on Pervasive Computing and Applications. 2006. pp. 593-598.
15. da Silva Santos, L.O.B., et al. Towards a Goal-Based Service Framework for Dynamic Service Discovery and Composition. in Information Technology: Sixth Intl. Conf. on New Generations ITNG '09. 2009. pp. 302-307.
16. Manikrao, U.S. and T.V. Prabhakar, Dynamic Selection of Web Services with Recommendation System, in Intl. Conf. on Next Generation Web Services Practices. 2005, IEEE Computer Society. 5 pp.
17. Gordijn, J., S. de Kinderen, and R. Wieringa. Value-driven Service Matching. in 16th IEEE Intl.Conf. on Requirements Engineering, RE '08.. 2008. pp. 67-70.
18. Jun, H., et al., Personalized Active Service Spaces for End-User Service Composition, in IEEE Intl. Conf. on Services Computing, SCC '06. 2006. pp. 198-205.
19. Wang, F. and K.J. Turner, Towards personalised home care systems, in 1st intl. conf. on Pervasive Technologies Related to Assistive Environments. 2008, ACM. pp. 1-7.
20. Xuanzhe, L., et al., Towards Service Composition Based on Mashup, in IEEE Congress on Services. 2007. pp. 332-339.
21. Nan, Z. and M.B. Rosson, What's in a mashup? And why? Studying the perceptions of web-active end users, in IEEE Symp. on Visual Languages and Human-Centric Computing, VL/HCC 2008. pp. 31-38.
22. Schafer, J.B., et al., Collaborative filtering recommender systems, in The adaptive web: methods and strategies of web personalization. 2007, Springer-Verlag. pp. 291-324.

Enabling Publish / Subscribe with Cots Web Services across Heterogeneous Networks

Espen Skjervold, Trude Hafsoe, Frank T. Johnsen and Ketil Lund

Norwegian Defence Research Establishment, Instituttveien 20, 2007 Kjeller, Norway
{Espen.Skjervold, Trude.Hafsoe, Frank-Trethan.Johnsen, Ketil.Lund}@Ffi.no

Abstract. In scenarios such as search-and-rescue operations, it may be required to transmit information across multiple, heterogeneous networks, often experiencing unreliable connections and limited bandwidths. Typically, there will be traffic within and across radio networks, as well as back to a central infrastructure (e.g., a police command post) when a reach-back link is available. This implies that using Publish/Subscribe is advantageous in order to reduce network traffic, and that store-and-forward capabilities are required to handle the instability of radio networks. At the same time, it is desirable to use commercial software based on standards as far as possible, in order to reduce cost and development time, and to ease interconnection of systems from different organizations. We therefore propose using SOA based on Web services in such scenarios. Indeed, Web services are targeted at stable, high-speed networks, but our work shows that such usage is feasible. In this paper, we add Publish/Subscribe functionality to standard, unmodified Web services through the use of our prototype middleware solution called the Delay and Disruption Tolerant SOAP Proxy (DSProxy). In addition to the ability to make Web services delay and disruption tolerant, the DSProxy enables SOAs in scenarios as described above. The DSProxy has been tested in field trials, with promising results.

1 Introduction

Commercial off-the-shelf (COTS) Web services are generally based on Request/Response (client-server) mechanisms [5]. However, many systems, environments, and situations could benefit from using the Publish/Subscribe paradigm instead, which is characterized by scalability, decoupled communication peers and asynchronous communication. In particular, Publish/Subscribe is essential to support mobile ad-hoc networks (MANETs), i.e., dynamic collections of nodes with rapidly changing multi-hop topologies that are composed of wireless links [7].

In search-and-rescue operations, it may be required to transmit information between many or all participants, and this can require traversing multiple heterogeneous networks. In order to enable different organizations running systems developed by different vendors to interoperate, it is crucial to base such communications on open and widely accepted standards. Considering the ubiquity of standard Web services and the usefulness of Publish/Subscribe, bringing the two together would provide

important benefits, as organizations and enterprises can leverage the power of Publish/Subscribe mechanisms with their existing Web services and clients.

Web services technology is mostly associated with the traditional client-server paradigm. As pointed out by (Vinoski, 2004), this scheme is generally much less efficient than push-based communication such as Publish/subscribe, and with OASIS' WS-Notification and W3C's WS-Eventing, the Publish/Subscribe paradigm has entered the Web service arena.

WS-Notification is a standard organized in a group of specifications that enable Publish/Subscribe-based communication between Web services. It comprises WS-BaseNotification, WS-BrokeredNotification and WS-Topics. While WS-BaseNotification defines which interfaces consumers (clients) and producers (servers) should expose, WS-BrokeredNotification introduces the concept of a message broker, an intermediary node which decouples consumers and publishers, and relieves producers from several tasks associated with Publish/Subscribe. WS-Eventing basically defines functionality similar to WS-BaseNotification, but with the addition of the Subscription Manager role, which enables subscription-related tasks to be handled by other nodes than the producer.

While WS-Notification and WS-Eventing offers standards-based Publish/Subscribe using Web services, they require the introduction of supporting frameworks, and new Web services and Web service clients that adhere to the standards must be developed. Standard Web service clients normally create outbound connections to Web services using HTTP and TCP, and since they typically do not run inside application servers, they have no way of listening for incoming connections. Similarly, ordinary Web services typically do not initiate outbound connections, and must be replaced by Publish/Subscribe-capable Web services. Furthermore, while we expect industry support for WS-Notification and WS-Eventing to mature in the future, it currently seems not to be a large selection of supported commercialized implementations available. Finally, it should also be noted that when services based on WS-Notification or WS-Eventing send notifications to subscribers, they do so by setting up an individual point-to-point connection to each subscriber. This means that these mechanisms have an untapped potential for efficiency improvement with respect to network traffic.

We have addressed these challenges using a middleware approach. Our primary goals have been 1) to enable Publish/Subscribe for existing standard Web services and clients without having to rewrite any software; and 2) to enable such Publish/Subscribe Web services to traverse multiple heterogeneous networks.

The result is a lightweight, prototype middleware system, called the Delay and Disruption Tolerant SOAP Proxy (DSProxy), which is able to meet the challenges described above. The DSProxys form an overlay network, which hides network heterogeneity and instability from the Web services and the clients. Between the DSProxys, optimizations such as data compression are used to reduce overhead, and several different transport-protocols are available, for handling different types of networks.

Externally, the DSProxy middleware solution is compatible with standard Web services by employing communication based on SOAP over HTTP/TCP. This means that existing Web services can be used together with the DSProxy overlay network

entirely without modification, while clients only need to replace the URL that addresses the services.

This solution also means that the DSProxy overlay network can be deployed only where needed; it is not an all-or-nothing solution that must be deployed everywhere. Typically, a DSProxy overlay network will be deployed within a MANET and between the MANET and a fixed network, while it is not needed internally in the fixed network. Clients in the fixed network will still be able to invoke services in the MANET and vice versa.

The remainder of this paper is structured as follows: In Section 2 we present related work, before describing the design and principles of the DSProxy in Section 3. We then present the configuration and results from a large field trial where the DSProxy was tested in Section 4, before concluding in Section 5.

2 Related Work

Publish/Subscribe systems have been around for a long time, and one of the earliest publicly described Publish/Subscribe systems was reportedly the “news” system of the Isis Toolkit, described at the 1987 ACM Symposium on Operating Systems Principles conference [2]. There exist many systems that support Publish/Subscribe-based information dissemination, ranging from open-source projects to commercial Enterprise Service Buses (ESBs). Some of the most well known middleware systems are Apache ActiveMQ, OMG Corba Event Service and Notification Service, IBM WebSphere MQ, and Real-Time Innovations (RTI). In ActiveMQ, the Publish/Subscribe functionality is built on top of Java Messaging Queues (JMSs). RTI is based on DDS, which is an open middleware specification for enabling Publish/Subscribe communications in real-time and embedded systems. While JMS-based solutions are message-centric, DDS and RTI differ by being data-centric, and in addition to topics offer keys which uniquely identifies objects. While offering Publish/Subscribe, these solutions all require developers to create custom applications utilizing the provided APIs.

The work by [1] points out that a scalable and efficient approach for achieving event dissemination in Publish/Subscribe systems is to employ an overlay network of brokers. In their work, they attempt to reorganize the overlay to reduce overhead associated with event dissemination. They point out that an alternative way to achieve efficient event dissemination is to use smart dissemination algorithms that avoid flooding events on the overlay, but they do not implement this. Also, their solution, named SIENA, is based on a proprietary, experimental Publish/Subscribe system. In our work, we do not want frequent overlay reconfiguration if we can avoid it, due to the overhead associated with management traffic. Thus, we use the complementary approach of smart dissemination algorithms. Further, we base our implementation on open Web services standards rather than a proprietary API, making our solution usable for a broader range of client applications and services. Another distinction from the work of [1] is that while SIENA is a content based Publish/Subscribe system, Web service Publish/Subscribe specifications adhere to a topic based scheme.

[4] have created a distributed event notification system (DENS) for MANETs. They developed a fuzzy logic based subscription language allowing expressive subscriptions and sophisticated event filtering. DENS is delay tolerant, an important feature in dynamic environments such as MANETs, and it builds an overlay, which performs store-and-forward of event messages. This solution is implemented and evaluated in a network emulator. The solution is shown to function well, proving that building a store-and-forward overlay for event notification in a MANET is both feasible and efficient. In our work we leverage this knowledge, by making the DSProxy system create an overlay network for event notification in MANETs. However, we also introduce the capability of configuring static routes for some nodes in our overlay, thus allowing dynamic MANETs to connect to WANs through reach-back links. Again, it is important to notice that while DENS is a proprietary research protocol, we leverage the open Web services standards, maintaining compatibility with existing clients and services.

[7] argue that the Publish/Subscribe paradigm can be used effectively to facilitate coordination of mobile users, for example, in a disaster recovery application: Rescuers equipped with networked devices (e.g., PDAs) can publish information, and other members can selectively subscribe to the information they need to perform their tasks. The authors argue that scalability is an essential condition of the Publish/Subscribe system, and propose a Publish/Subscribe system suitable for large MANETs. They combine document flooding and content-based routing techniques in a hierarchical manner, and evaluate their solution in a simulator. Our work is similar to this, in that we address and solve the same problem of creating a scalable Publish/Subscribe solution for MANETs. However, in order to stay interoperable with existing software we have based our solution on topic-based routing and open Web service standards.

[5] argue that Web services and Publish/Subscribe-based schemes up until now mostly have been considered separately, and that it is not clear that a possible unification will adhere to any overarching, pre-planned approach. To address this situation they developed a theoretical and conceptual framework extending current Web service programming models and describing the necessary underlying middleware. While the implementation of such middleware was beyond the scope of their paper, an infrastructure based on collaborating brokers was outlined. In this respect, our work is similar to this, in that we employ an overlay network where DSProxy instances take on broker responsibilities. [5] emphasizes the importance of exerting as small an impact as possible on the existing Web services programming models. However, as our middleware solution aims to leverage the power of Publish/Subscribe schemes with existing Web services and clients, it requires no extensions to existing Web service programming models.

PUSMAN [3] is a middleware system for topic-based Publish/Subscribe in MANETs. It uses a collection of brokers to forward advertisements and subscription information. By detecting mobility through monitoring, PUSMAN will reconfigure its overlay to ensure a high delivery success ratio. The work done here is orthogonal to our own; we aim to ensure delivery through the use of a store-and-forward mechanism, which could potentially be improved by combining it with the reconfiguration approach used in PUSMAN.

3 The DSProxy

The DSProxy system [6] is our prototype middleware solution developed in Java. It is a novel, lightweight and cross-platform system with pluggable components. The core DSProxy features include providing store-and-forward capability to SOAP messages, utilizing compression of SOAP and XML and facilitating the traversal of multiple heterogeneous networks. At the same time, it remains compliant with unmodified COTS Web services and clients. By placing one or more DSProxys between a Web service and a Web service consumer, store-and-forward functionality is introduced into the network, which provides increased robustness in dynamic, heterogeneous networks and MANETs. The DSProxy instances self-organize into an overlay network using an internal service discovery mechanism based on UDP multicast (or it can be statically configured where UDP multicast is unavailable). For more details on how the overlay network is built and organized, we refer to [6].

Once organized into an overlay network, DSProxys exchange information about advertised services and the number of hops required to get there. The overlay network allows for smart routing of Web service requests at the application level, but utilizes the underlying routing protocol for IP routing.

3.1 DSProxy Core Features

Figure 1 displays a simple network layout comprising two separate networks and three physical nodes; the client machine, a gateway machine, and the server. The gateway node is equipped with two network adapters providing a physical data link to each network. A standard Web service client and a Web service run on the client machine and the server respectively, and a DSProxy instance is running on the gateway machine, effectively bridging the two different networks together. This enables communication between these two networks even if no IP-level routing is available. Additionally, DSProxy instances run on both the client machine and on the server. When the client wishes to invoke the Web service residing in the other network, instead of initiating an end-to-end connection directly to the server, it sends the SOAP invocation request to its local DSProxy instance, which relays the request to the gateway DSProxy, and so on.



Figure 1: A simple network layout showing two discrete networks bridged with a DSProxy, adding store-and-forward capability to a standard Web service.

The only difference between a direct invocation of a Web service method and an invocation of the same Web service method through the DSProxy overlay network is

the URL used to address the service. The original URL is replaced with a URL in the following form:

```
http://127.0.0.1:7000/?uniqueServiceName=weatherService
```

The 127.0.0.1-address indicates that the Web service client is relaying the invocation request through a DSProxy instance running on the same physical machine, and the uniqueServiceName-parameter instructs the DSProxy overlay network to route the invocation request to the DSProxy instance responsible for invoking the Web service (typically the DSProxy running on the server hosting the service). The TCP-connection between the Web service client and the first DSProxy is kept open until the DSProxys return the response data.

All three DSProxys are part of the same overlay network, and by monitoring their environments, all DSProxys know their neighboring DSProxys, and where to route a request in order to invoke a particular service. Upon receiving a service invocation request, the gateway DSProxy then relays it to the server DSProxy. Finally, the server DSProxy, knowing it is within reach of the actual Web service, invokes the service, and returns the response data using source routing (back-tracking the invocation route).

While based on ordinary Request/Response-principles, this deployment offers two important benefits: First, store-and-forward capabilities are introduced into the network. This means that if any of the data links become unavailable, the DSProxy closest to the broken link will store the invocation message and retry the transmission at regular intervals. It can also choose another route to the destination if available. If, for instance, the link from the client (e.g., a search-and-rescue agent, part of a MANET) into the network breaks down, the DSProxy instance running locally on the client machine will cache the request, and thereby provide store-and-forward capability from the very first hop and on into the network. Note that running a DSProxy local to the client is not required, but it is usually advantageous, as described above.

Second, the DSProxys eliminate the requirement of initiating an end-to-end connection between the client and the server. In order to stay interoperable and standards-compliant with COTS Web services, HTTP and TCP are utilized between the client and the first DSProxy and between the last DSProxy and the Web service, while any transport protocol may be used between DSProxy instances. The latter can be of great importance when operating in highly dynamic MANETs, where changing topologies and unreliable data links may require other protocols than connection-oriented TCP.

As long as DSProxy instances are running locally on both the client machine and the server, TCP-connections can always be used for the first and last hop (intra-machine), no matter what kind of networks and data links that connect them.

3.2 DSProxy and Publish/Subscribe

The first development iteration of the DSProxy focused on enabling heterogeneous network traversal and bringing robustness to Web services through store-and-forward capabilities. In the second iteration we focus on bringing together standard Web services and the Publish/Subscribe paradigm.

Figure 2 illustrates how, by utilizing two DSProxys, one can enable Publish/Subscribe operation with COTS Web service clients and services. By deploying a DSProxy instance on the same physical machine as the Web service, the DSProxy can perform frequent, continuous polling of the Web service, without adding any traffic load to the network. This intra-machine polling enables the DSProxy

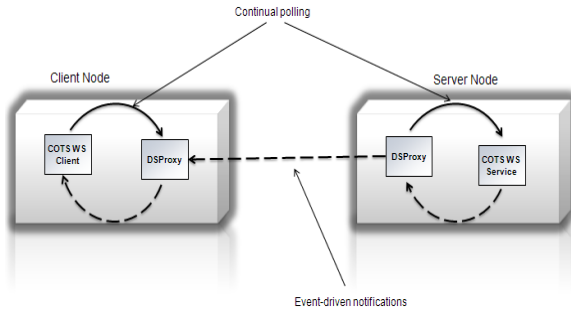


Figure 2: Two DSProxy instances enabling a COTS Web service and client to perform in a Publish/Subscribe fashion.

to discover updated information almost instantly, and the DSProxy may then notify anyone interested in the information, in this case, the DSProxy instance running on the client machine. The server DSProxy will create a hash over the Web service response data, and compare it to the previous response data hash, to determine whether the response is identical (and thereby already been pushed to subscribers) or not.

On the client machine, the same approach is used; the Web service client is instructed to poll the DSProxy instance (sending normal Web service requests) running on the same physical machine continuously. This circumvents the inability of Web service clients to accept incoming connections, and enables the client to receive information almost instantly by polling the DSProxy frequently. By utilizing this polling mechanism on both the client machine and on the server, only the machines' internal buses are burdened, and no additional traffic enters the network. Only when the server DSProxy discovers new, updated information is the network utilized, and the information is pushed from the server DSProxy to the client DSProxy, with the former initiating a connection to the latter. These mechanisms enable regular Web services and clients developed with COTS Web service software to communicate in a Publish/Subscribe-based fashion, and will be referred to as the DSProxy native Publish/Subscribe scheme. The practical minimum Publish/Subscribe-enabling setup requires only two instances, preferably deployed on the client and server machines to achieve intra-machine polling. However, it will often be beneficial to deploy multiple DSProxy instances into a network, as this will increase robustness through multiple store-and-forward points and alternative routes between client and service. As with all inter-DSProxy communication, any transport protocol may be used for sending the notifications, depending on the underlying network class and characteristics.

The server DSProxy can be configured to poll different Web services at different intervals, depending on the nature of the service, the latency requirements, and the resource constraints. For an instant messaging service, one would typically require

low latencies and frequent polling, and the DSProxy might poll the Web service every second, in order to facilitate fast message exchange. For a weather forecasting service on the other hand, the latency requirements may be substantially lower, and a polling frequency of 5 minutes might suffice.

3.3 DSProxy and Subscriptions

In order to subscribe to a Request/Response-based Web service, the client needs to inform the overlay network about this. If we compare the subscription request to a regular invocation of the same Web service, the subscription is set up simply by doing a small modification of the URL used for the regular service invocation.

Using the example from Section 3.1, assume that the Web service client wants to subscribe to the weather service, i.e., receive new forecasts as they become available, instead of having to poll the service at regular intervals using Request/Response. Although the service itself is unaware of the Publish/Subscribe concept, the DSProxys allow the client to set up a subscription by just slightly modifying the URL described in Section 3.1:

```
http://127.0.0.1:7000/?uniqueServiceName=weatherService&pubSub=true
```

Here, the extra parameter `pubSub=true` instructs the DSProxy overlay network to handle this as a DSProxy native Publish/Subscribe subscription. The first DSProxy will then determine whether the requesting client is already subscribing to the Web service method. If not, it will create a subscription and store it locally. Next, it will forward the request to the next DSProxy instance in the network (one hop closer to the service), and this DSProxy will perform the same check. If this DSProxy is the one responsible for invoking the actual service, it will do so, and start the server DSProxy polling cycle explained earlier. This polling will continue as long as one or more subscribers are active, meaning they have subscribed to, and not yet unsubscribed from, the service.

When a Web service client wishes to unsubscribe from the service, it simply replaces the `pubSub=true`-parameter with the `pubSub=unsubscribe`-parameter, which causes all involved DSProxys to delete the subscription. On the server DSProxy, if the unsubscribe-request results in no subscribers any longer being active, the polling cycle for the specified Web service method ends.

The DSProxy native Publish/Subscribe mechanism enables Publish/Subscribe to be used with standard, COTS Web services and clients. However, as implementations of WS-Notification and WS-Eventing continue to mature, and become more plentiful and widespread, we anticipate benefits of being able to interoperate with such clients and services as well. Therefore, when using a WS-Eventing client together with the DSProxy overlay network, it becomes part of the Publish/Subscribe-tree by sending a WS-Eventing compliant subscription message using a URL on the following form:

```
http://127.0.0.1/?uniqueServiceName=weatherService&pubSub=wseSubscribe
```

By giving the `pubSub`-parameter the value `wseSubscribe`, the DSProxy overlay network is instructed to interpret the subscription message accordingly. This will create one subscription that covers all requests for the same combination of service, filter dialect and filter expression (i.e., a new request for the same combination will not be forwarded, since the subscription is already established). Before forwarding the re-

request to the next DSProxy in the network, the sending proxy will modify the WS-Eventing subscription message. Specifically, it alters fields such as `NotifyTo`, to ensure the actual WS-Eventing service sends its notification through the DSProxy overlay network (addressing the last DSProxy instance in the chain).

While Figure 2 presents a simple, conceptual Publish/Subscribe network layout, larger, more complex Publish/Subscribe-trees can be created. Figure 3 shows such a tree, presenting a hierarchical Publish/Subscribe structure. As shown, subscribers to a DSProxy can be both Web service clients and other DSProxys. Such a tree optimizes the flow of information, and can reduce bandwidth requirements in situations where many clients are interested in the same information. When DSProxy C retrieves updated information from the Web service (by polling), it will actively notify DSProxy A and B, and make the same information available for Client 4, allowing it to retrieve the information during its next polling cycle. Subsequently, DSProxys A and B will make the information available for their clients, ensuring that all subscribing clients eventually retrieve the new information. DSProxy C will not delete its subscription to the Web service and end the polling cycle until DSProxy A, DSProxy B and Client 4 have all unsubscribed to the service.

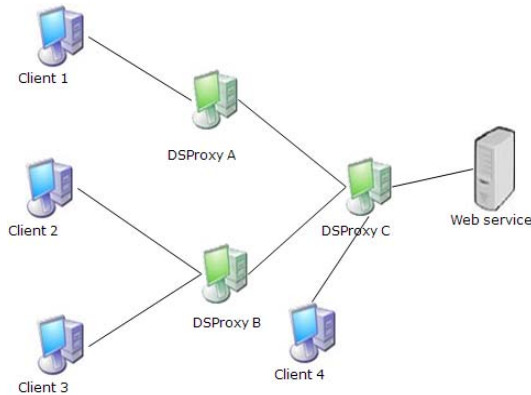


Figure 3: A Publish/Subscribe-tree comprising three DSProxys in a dynamic hierarchical structure.

In highly dynamic MANETs with frequently changing topologies, such a Publish/Subscribe-tree is susceptible to failures, as nodes re-arrange, become unavailable, and disappear without warning. However, the DSProxy Publish/Subscribe-mechanisms are built on top of the overlay network, which is self-organizing, and able to adjust to changing environments. When a DSProxy notices that it has not received any notifications within a configurable period of time, it will resend the original subscription-request to the DSProxy now reporting to be the one closest to the Web service. This may or may not be the same DSProxy that it originally subscribed to. If it is not, the aforementioned chain of events will start again, ending in an active subscription and an initiated polling cycle at the DSProxy now closest to the actual Web service. If it is still the same DSProxy, it may simply be the case that no new notifications have been produced, and it will merely ensure that the subscription is still active.

4 Results and Discussion

In order to verify the DSProxy functionality and capabilities, the solution was tested in a series of field trials, with promising results. The DSProxy middleware solution was used for providing store-and-forward capability within a MANET, as well as for bridging it with a separate, static network. As shown in Figure 4, the MANET comprised 3 mobile nodes (deployed in vehicles) and a stationary gateway node, all running IP-based radios. The gateway ran two radios of different types back-to-back, offering physical data links to each network (one radio link effectively functioning as a reach-back link to the static network). Running DSProxys, the static network operators were able to subscribe to services offered by the mobile nodes located in the MANET, such as imaging services and position services. The static network operators would continuously have updated GPS-positions pushed to them from the vehicles, allowing them to track and visualize the positions of the vehicles on a map as they moved.

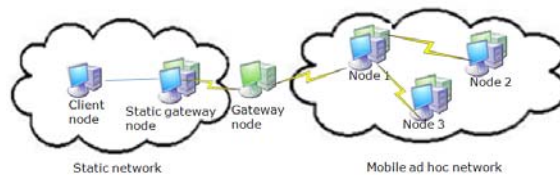


Figure 4: The field trial setup, showing a gateway node running a DSProxy, bridging the MANET and the static network.

When the vehicle operators spotted interesting events, they would take pictures of the events and publish these onto the network. The operators on the static network would then be notified and receive the pictures almost instantly. Given this configuration, with the information having to traverse two bridged, heterogeneous networks and 4 nodes, we experienced typical latencies of 3-5 seconds, from the time that the information was published to it was displayed on the client side. Due to frequent disruption of the radio links, the store-and-forward capability of the DSProxys was demonstrated: As DSProxys running in the MANET attempted to push images to the DSProxy running in the static network, a disruption of a radio link would cause the DSProxys to cache the notifications locally. When the link became available again, the notifications would be re-sent, successfully delivering the pictures to the static network and hiding the erroneous events from the clients and the end users.

This Publish/Subscribe-based interoperation was made possible by utilizing the DSProxy system, and enabled the static network nodes to have information pushed from our regular, non-Publish/Subscribe Web services. It should be noted that such interoperation using the DSProxy native Publish/Subscribe mechanisms does require small modifications to be made to the otherwise regular Web service clients. As with all communication going through the DSProxy overlay network, the URL to the end Web service must be modified, as described in Section 3.1 and 3.3. This information is usually embedded in human readable configuration files (e.g., in locally cached WSDL-files), and do not require recompiling the client application.

In addition, if DSProxy native Publish/Subscribe is to be used, the client must be modified to engage in the polling cycle described earlier. This typically involves wrapping the request-statement within the code in a loop structure. Finally, clients must handle null-responses, which occur when the client-side DSProxy being polled do not offer any new notifications. The two latter modifications are usually quick and easy to implement and do not include touching the actual business logic, but they do require the client to be recompiled.

The Web services on the other hand, require no modifications. Still, considerations should be made when selecting which Web services to Publish/Subscribe-enable through the DSProxy overlay network. As a DSProxy determines whether or not the information is updated based on the hash produced, constantly changing information will produce ever changing hashes, thus, also a constant flow of notifications. For instance, responses that contain fine-grained time-stamps would always produce new notifications, even though the information may otherwise be unchanged.

Although the intra-machine polling mechanism utilized by DSProxys to retrieve updated information from COTS Web services does not generate any network traffic, it does require server resources such as CPU and memory. While a typical Web service invocation only requires a small amount of resources, it may ultimately limit the number of Web service methods and the polling frequency that can be used. In order to establish this limit, an experiment was conducted using a setup similar to the one presented in Figure 1, with one client machine and one server machine, both running one DSProxy instance. Simple “Hello world”-like services were developed using C# and ASP.NET 3.5, which returned strings consisting of 100 random characters for each invocation. This made the services produce new responses for every invocation, which in turn lead to notifications being produced for every poll, in order to produce a “worst-case” scenario.

The services were deployed inside a Microsoft Internet Information Services (IIS) 5.1 Web Server instance running on the server. The server machine was equipped with an Intel Pentium 4 dual core 2.6 GHz CPU, 1 GB RAM running MS Windows XP Pro SP3. A standard Web service client application capable of invoking the services in the normal Request/Response-fashion were developed using .NET 3.5.

The client application ran on the client machine, and by modifying the Web service invocation URL, the invocation requests were relayed through the two DSProxy instances and delivered to the Web service. The URL was also modified to instruct the DSProxy overlay network to initiate Publish/Subscribe, and the server DSProxy would start one polling-cycle for each service subscribed to (running as parallel threads). Because the client application ran on a separate machine, all CPU load on the server was associated with the server DSProxy repeatedly polling the services and notifying the client DSProxy (in addition to some load associated with the actual Web services and the IIS).

The server DSProxy was configured to poll each service once every second, and the number of services subscribed to could be controlled from the client application.

The CPU utilizations were measured using MS PerfMon, and averages were monitored for 60 seconds during each run. Figure 5 shows the varying average CPU loads when subscribing to 5, 10, 15 and 20 Web services. As seen from the graph, the performance scaled close to linearly, ranging from 0.755 % to 4.469 % CPU

utilization. Even when 20 different services were subscribed to and polled every second, creating and sending 20 new notifications every second, less than 5% of the available CPU-cycles were utilized on average.

An extrapolation of this data indicates that the current setup could theoretically handle subscribing to nearly 450 services, polling each of them and producing

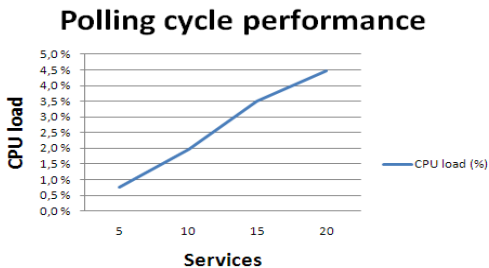


Figure 5: The graph shows CPU utilization for a given number of services being polled every second.

notifications once every second. However, the Web Server Software places limitations on the number of possible concurrent Web service invocations. Also, more complex services requiring CPU-intensive calculations or IO-operations would reduce the performance, dividing the available CPU cycles between tasks associated with Publish/Subscribe and the actual work being performed by the services. On the other hand, most production servers would greatly surpass our test-server performance-wise. In addition, for many services, a considerably lower polling frequency will suffice.

It is important to note that, because the server DSProxy does the actual invocation of the service, the polling frequency is constant, regardless of the number of subscribers per service. This means that in a scenario with many subscribers to a service, the polling frequency may be considerably lower than if each individual client were to poll the service using Request/Response. In fully distributed environments, participants may function as both clients and servers by exposing their own services. The hardware hosting such services may be limited devices such as PDAs, and this should be taken into account when configuring polling cycle frequencies.

While standards-based interoperation with COTS Web services using DSProxy native Publish/Subscribe requires minimal modifications to the Web service clients, using WS-Eventing-based Publish/Subscribe in the DSProxy requires no modifications to any software. By placing DSProxy instances between WS-Eventing-based clients and services in dynamic MANETs, added robustness is achieved through store-and-forward capabilities. Additionally, network performance gains are achieved when multiple WS-Eventing clients are interested in the same information, in other words subscribing to the same services using the same filter dialects and expressions. Instead of the WS-Eventing service and the DSProxys having to send the same information to each of the clients directly, as is the case for regular WS-Eventing and WS-Notification, the information is sent to one or a few subscribing DSProxys. Since the overlay network effectively constitutes a multicast tree, less traffic is relayed through central parts of the network. Client-specific data, such as the

ReferenceProperties-field, are stored and added to the notifications at the DSProxy instances closest to each client.

5 Conclusions and Future Work

Through a series of field trials, the functionality and capabilities of the DSProxy were tested, demonstrating its usefulness in heterogeneous and error-prone networks and showing potential for typical search-and-rescue scenarios. By utilizing the lightweight DSProxy system in both MANETs and static networks, regular Web services can leverage the power offered by the Publish/Subscribe paradigm, requiring only minor modifications to be made to the Web service clients. The practical minimum setup for achieving this would only require two DSProxy instances to be deployed into the network, preferably as close to the client and the service as possible.

Experiments have shown that the DSProxy polling cycles consume relatively low amounts of resources, and together with the fact that the invocation frequency is independent of the number of clients, this means that a server can potentially handle a large number of services and clients. As we expect implementations of WS-Notification and WS-Eventing to mature and become more plentiful and widespread in the future, the DSProxy system supports WS-Eventing, with support for WS-Notification currently under development. The DSProxy WS-Eventing-based Publish/Subscribe mechanisms allow bandwidth-optimized routing of information, requiring no modifications to clients or services. Both modes benefit from the store-and-forward capabilities provided by the DSProxys, facilitating Web service based Publish/Subscribe in MANETs and other unreliable networks. Additionally, both modes of Publish/Subscribe can be used with Web services across multiple heterogeneous networks.

Future work includes compliancy with the WS-Notification standard and additional schemes for optimizing maintenance of Publish/Subscribe-trees. Also, a UDP multicast-based notification capability is under development, which is expected to further reduce traffic loads in radio networks where multiple clients and DSProxys subscribe to the same information.

References

1. Baldoni, R., Beraldi, R., Querzoni, L., and Virgillito, A., 2007, Efficient Publish/Subscribe through a Self-Organizing Broker Overlay and its Application to SIENA, *The Computer Journal*, volume 50, num. 4, Oxford University Press, pp 444—459.
2. Birman, K., and Joseph, T., 1987, Exploiting virtual synchrony in distributed systems, *SIGOPS Oper. Syst. Rev.* 21, 5, pp 123-138.
3. Denko, M. K., 2006. Pusman: Publish-subscribe middleware for ad hoc networks, *IEEE CCECE/CCGEI*, Ottawa.
4. Lekova, A., Skjelsvik, K.S., Plagemann, T., and Goebel, V., 2007, Fuzzy Logic-Based Approximate Event Notification in Sparse MANETs, *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 02*, pp 296-301.

5. Silva-Lepe, I., Ward, M. J., and Curbera, F., 2006, Integrating Web services and Messaging, IEEE International Conference on Web services, Chicago, USA, pp 111-118.
6. Skjervold, E., Hafsøe, T., Johnsen, F.T., and Lund, K., 2009. Delay and Disruption Tolerant Web services for Heterogeneous networks. IEEE MILCOM, Boston, MA, USA.
7. Vinoski, S., 2004, Web services Notifications. *IEEE Internet Computing*, vol. 8, no. 2, pp 86-90.
8. Yooa, S., Sonb, J.H., and Kima, M.H., 2009, A scalable Publish/Subscribe system for large mobile ad hoc networks, *Journal of Systems and Software*, Volume 82, Issue 7, pp 1152-1162.

Author Index

Astudillo, H.	52
Barkaoui, K.	80
Becerra, C.	52
Eckert, J.	42
Eslami, M.	109
Franch, X.	52
Hafsøe, T.	122
Hofman, W.	3
Johnsen, F.	122
Khadka, R.	67
Lampe, U.	42
Li, Y.	16
Lund, K.	122
Missaoui, A.	80
Riccobene, E.	29
Sapkota, B.	67, 109
Sbaï, Z.	80
Scandurra, P.	29
Schill, A.	93
Schuller, D.	42
Shiu, Y.	16
Sinderen, M.	109
Skjervold, E.	122
Springer, T.	93
Steinmetz, R.	42
Trigos, E.	93
Winkler, M.	93
Zarghami, A.	109

Proceedings of ACT4SOC 2010
4th International Workshop on
Architectures, Concepts and Technologies for Service Oriented Computing
ISBN: 978-989-8425-20-1
<http://www.icsoft.org>