

Proceedings for

2nd International Workshop on

Tool Support and Requirements Management
in Distributed Projects (REMIDI 2008)

Bangalore, India
August 17th, 2008

Edited by Chintan Amrit, Patrick Keil, Marco Kuhrmann

Title: *REMIDI 2008 Proceedings International Workshop on Tool Support and Requirements Management in Distributed Projects*

Authors: Chintan Amrit, Patrick Keil, Marco Kuhrmann (eds.)

Workshops Proceeding: WP0801

ISSN 1574-0846

Publisher: Center for Telematics and Information Technology (CTIT), Enschede, the Netherlands

As part of

International Conference on
Global Software Engineering, **ICGSE 2008**

Program Committee

Stefan Biffl,

Manfred Broy,

Pradeep Desai,

Vesna Mikulovic,

Neel Mullick,

Jürgen Münch,

Daniel J. Paulish,

Ita Richardson,

Bernhard Schätz,

Rini van Solingen,

Jos van Hillegersberg,

TU Wien

TU München

Tata Consultancy Services

Siemens AG Austria

IKaru Projects Pvt. Ltd.

Fraunhofer IESE

Siemens Corporate Research

University of Limerick

TU München

LogicaCMG and Drenthe University

University of Twente

Table of Contents

1. REMIDI CFP.....	1
2. Decentralized Software Process Coordination and Security using Electronic Contracts..... <i>Adailton Magalhães Lima, Rodrigo Quites Reis</i>	3
3. A Framework to Analyze Impact of Change in Component Based Software Engineering..... <i>Kuljit Kaur, Hardeep Singh, and Debasish Jana</i>	9
4. A suite of tools for the automation the management of the software process..... <i>Javier Berrocal, José Manuel García, Juan Manuel Murillo</i>	14
5. TESNA: A Tool for Detecting Coordination Problems..... <i>Chintan Amrit and Jos van Hillegersberg</i>	20



Proposal - Second International Workshop on Tool-Supported Development and Management in Distributed Projects (REMIDI'08)



Call for Papers

in conjunction with

IEEE International Conference on Global Software Engineering (ICGSE)

Bangalore, India
August 17-20, 2008

Today, distributed projects (often subsumed under terms like global software development (GSD), global collaboration, offshoring etc.) are common ways to overcome time and resource restrictions or lack of local expertise. Thus, today's projects take place in a global context. At the same time, tool integration and end-to-end tool chains are more and more getting on the agenda of researchers and industry to tackle the growing complexity of development projects.

Especially planning, coordinating and controlling software engineering in distributed settings are far more complex than in one-site projects. First, the process of analysis and design needs to be planned and organized differently. Second, the methods used to document, share and discuss design and architecture ideas need to take into account the fact that some project members involved in these tasks are spread over multiple sites and organisations and don't have contact to end-users. Third, as the development artifacts are wide spread, the development, integration and release of a high quality product is far more challenging.

As a conclusion, we need concepts and tools to support the specific needs, tasks and process requirements in distributed development projects. Experience shows that an appropriate tool chain increases efficiency and success of distributed projects since coordination and collaboration are far more complex than in on-site projects and need to be properly supported. Aspects like process assistance, knowledge management or project tracking ask for appropriate tools.

Therefore, the workshop will walk through the methods and concepts that are available and the tool chains that are used in global software development projects. After last year's successful edition (cf. www.ctit.utwente.nl/library/proceedings/proceedingsamrit.pdf and www4.in.tum.de/~kuhrmann/remidi07.shtml), this workshop will more explicitly focus on tools and infrastructures for GSD projects.

The participants will present and discuss project experiences, best practices and new approaches – based on academic research and / or on experiences from industry.

One of the objectives of this workshop is to structure the major research topics and to define a research agenda for further work in the area of "end-to-end" tool support in distributed system development. Besides that, there will be a demo session with presentations and live demonstrations of tools that are specifically dedicated to support distributed development projects.

In summary, the workshop will include different aspects of tool selection

Schedule:

- June 1** Deadline for paper submission to the workshop organizers
- July 1** Decision of acceptance to paper authors
- July 15** Final version of accepted papers due
- August 17-20** REMIDI Workshop

Paper submission:

Research and industry papers must be submitted in PDF format by email to the organizers. Your paper must conform to the IEEE proceedings publication format (8.5" x 11", Two-Column Format) described at IEEE/CPS and be no longer than 6 pages including all text, references, appendices, and figures. Tool papers must be no longer than 4 pages including all text, appendices, etc. and should contain screenshots and references to projects using the presented tools.

Submissions that exceed the page limit or do not comply with the proceedings format (cf. IEEE/CPS) will be desk rejected without review. The results described must not be under consideration for publication elsewhere. Accepted papers will be published in CTIT proceedings.

and orchestration in a distributed software development context, e.g.:

- *Administration and tracking of documents, concepts, code, etc.:* What are the consequences for the process and the design tools if (all or some of the) processes of requirements engineering, design, development etc. are distributed?
- *Collaboration and communication in software engineering:* How can teams be organized and coordinated when they are spread over two or more sites? How can projects achieve efficient collaboration and alignment? What are the lessons learned on tools and infrastructure for collaboration in different project phases? Which different requirements and characteristics do the different project phases have regarding tool support?
- *Process assistance and support:* What does an adequate process for distributed development look like and how should it be supported by tools and techniques? What tools or tool chains are adequate to assist different project roles?
- *Tool orchestration:* How should projects select their tools? How different are tool chains for different industries? What are the project characteristics that influence tool decisions most heavily? How different are the optimal tool chains for different levels of education and experience?
- *Economic aspects:* What is the Return on Investment for tools dedicated towards distributed development?
- *Project management:* Which tools can help to plan, control and track a project? Are risk management or workflow management tools different to those used in on-site projects?
-

Topics of the 1-day Workshop

The following is a non-exhaustive list of relevant topics:

- Models, tools and technologies for handling dynamics and complexity in the early phases of dispersed collaboration
- Models and tools for unifying processes respecting requirements engineering, software development, and operations and maintenance in global contexts
- Comparability and comparison with tools used in open source projects
- Process model design for distributed engineering and “mirroring” of these processes in SE tools
- Models and processes to define and predict usability, reliability, performance, quality and “adequacy” of development tools
- Impacts of tools on the cost efficiency of distributed development

These topics will be discussed based on presentations by participants. Based on these contributions, we will try to structure the problems and challenges and discuss a “research agenda for integrated tool infrastructures in GSD”.

An explicit tool track asks vendors and academic research teams to present their products or prototypes. Live demonstrations are welcome.

Addressees

The workshop targets practitioners as well as researchers interested or involved in geographically or organizationally distributed software development.

Organization Committee:

- Chintan Amrit, University of Twente, c.amrit@utwente.nl
- Patrick Keil, TU München, keilp@in.tum.de
- Marco Kuhrmann, TU München, kuhrmann@in.tum.de

Program Committee:

- Stefan Biffel, TU Wien
- Manfred Broy, TU München
- Pradeep Desai, Tata Consultancy Services
- Vesna Mikulovic, Siemens AG Austria
- Neel Mullick, IKaru Projects Pvt. Ltd.
- Jürgen Münch, Fraunhofer IESE
- Daniel J. Paulish, Siemens Corporate Research
- Ita Richardson, Lero, University of Limerick
- Bernhard Schätz, TU München
- Rini van Solingen, LogicaCMG and Drenthe University
- Jos van Hillegersberg, University of Twente

Decentralized Software Process Coordination and Security using Electronic Contracts

Adailton Magalhães Lima, Rodrigo Quitês Reis

*Software Engineering Laboratory - <http://www.labes.ufpa.br>
Federal University of Pará (UFPA)
Belém, Pará, Brazil
{adailton,quites}@webapsee.com*

Abstract

The Software Process Technology research area establishes methods and tools to support the good quality of software products. Process-centered Software Engineering Environments enable process modeling and enactment enforcement, and only a few of them supports decentralized coordination of process activities. Current projects in Global Software Development deal with many coordination and collaboration problems, such as share process context, information security and remote project monitoring. This paper describes a tool that provides a technological support to the cited problems on decentralized software process development. The objective of this tool is to provide a contract-based information filtering, and allows the security configuration when share process context among different organizations. We provide a description of the main expected results applying the proposed approach in different organizational contexts and a reference to future directions of this research.

1. Introduction

Software Engineering evolves as a theoretical and practical research area by proposing and applying methods, techniques and tools, which aims to increase software product quality. To develop and maintain software products it is necessary to cover a broad and coherent range of policies, organizational structures, technologies, methods and artifacts, which are all involved with software development process. According to [11], the software product quality depends strongly on the quality of the adopted software process.

The Software Process Technology research area evolves quickly to provide adequate automated support for the enactment of quality software processes. Briefly, it involves the development of tools and

environments to support the software process modeling and enactment. Software Process Technology promises to deliver automated facilities that can be useful to enable a software organization in order a high capability and maturity levels on reference models such as CMMI [3] and SPICE [25]. Integrated environments that enable the automation of software process modeling and enactment task are generically known as PSEEs (Process-centered Software Engineering Environments) [11]. The enactment of software process models is enabled by a software component known as process engine that enable guidance and/or enforcement of the process models with respect to the human enrollment in the context of a software project [6]. However, the decentralized coordination of process activities among different PSEEs is only supported by a few of them. Most of the PSEEs provide only a centralized solution based on the traditional client-server architecture.

Due to the involvement of different locations and organizations adequate support for the decentralized coordination of activities in software development projects implies on the absence of centralized controlling mechanism. According to [7] and [10], the provision of automated support for decentralized coordination of software projects is a challenge for the Software Engineering Community. More specific, there is the problem of the shared context among different remote sites in Insourcing and Outsourcing projects highlighted by [19]. Other problems, such as project monitoring and progress measurement, are also faced as challenging problems for current projects in Global Software Development [15].

In software development projects that use environments like PSEEs to control and coordinate their processes, tools like task agendas and process views are used to provide monitoring support for the software projects enacted by the organization [6]. Targeting specific problems of decentralized software development context, including the problems of

context sharing and project monitoring, this paper presents an extension to an existing PSEE to provide support for project monitoring and process integration in decentralized software projects.

The remainder of this paper is organized as follows. Section 2 presents the main concepts involving the coordination in decentralized software development. Section 3 makes an overview of the WebAPSEE environment, a PSEE used as implementation basis for this work. Section 4 illustrates the proposed extensions to provide decentralized software process enactment support in the decentralized software development context. The section 5 discusses the main expected benefits of this approach. Finally, section 6 presents the final remarks and future directions of this research.

2. Coordination in Decentralized Software Development

Decentralized Software Development is a field of study deeply influenced by the evolution of industry and globalization [10]. According to Sabherwal [21], project coordination focuses on the management of inter-dependencies among different activities in software development. In the context of decentralized software projects it is not feasible to have a unique, centralized entity to control process enactment.

The current state of the practice in this field is frequently based on shared repositories to monitor delivered and changed artifacts. However, this approach provides limited view to customers, since it does not show information about the internal processes models enacted by the supplier [8]. The lack of process monitoring is critical nowadays due to the increasing demand on quality models in acquisition projects.

To successfully orchestrate global software development, project managers need to share information about projects and communicate with other site managers to take advantages of this work model [14]. Ebert [7] pointed out the influence of insufficient contract management in decentralized software development and the low control over the quality and scheduling aspects on contracts with external partnerships. To manage the customer-supplier relationship a contract must describe a set of policies to control access information and monitor project milestones.

The contract management approach enables process enactment monitoring while provide awareness to customers and suppliers on current project status. In addition, it can be used to provide support for early notification for abnormal behavior on monitored projects. In this way, it provides the base to support the development of a number of additional facilities with

respect to the continuous monitoring of both product and process related information [15].

3. The WebAPSEE Environment

PSEEs are tools that can be used to minimize problems involving software development. Thus, PSEEs represent the use of software to help the management of the software development [12].

The WebAPSEE environment is a PSEE developed since 2004 at the Software Engineering Laboratory of Federal University of Pará, Brazil [18]. The current open source version of this PSEE can be found in <http://sourceforge.net/projects/webapsee>.

The main goal of WebAPSEE environment is to provide support for the definition and deployment of software process models using a graph-based formal semantic notation [12]. In a software process improvement initiative, the WebAPSEE environment represents a shared repository to define and monitor organization's processes. In this case, it will provide access to evidences of process definition and to the process improvement goals.

The majority of PSEEs adopts a centralized architecture, where they enact the processes on only one site, and client applications use remote procedure calls to access the processes information. As examples of this approach, we can cite the EPOS [4], Marvel [17] and WebAPSEE [18] environments.

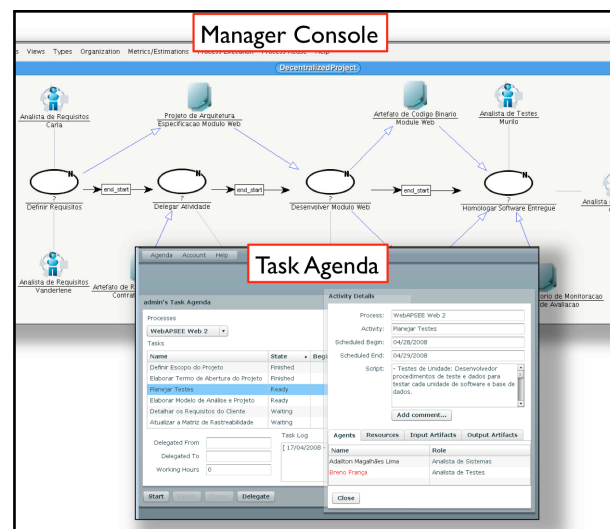


Figure 1 – Screenshot of the WebAPSEE Environment with the Manager Console and the Task Agenda.

Some clear problems of the centralized architecture approach are: a unique failure point; the system scalability is limited to the server machine processing power; the final response time is limited to server concurrency level. Beyond these technical problems,

the centralized architecture of PSEEs may represent an organization disadvantage with respect to the demand to share process information with external partners. Furthermore, a centralized server can open backdoors to strategic access process information from strange clients, what is a strategic organizational problem [24].

Even if the security policies among different users are used to support information filtering, the databases stay in an external organization domain (in some remote site service), and still remains the risk of unauthorized access of strategic data and the lack of trust among the remote sites [1].

The WebAPSEE Manager Console

The Manager Console Tool is integrated on the WebAPSEE environment, which uses a modeling formalism allowing the process definition using a graphical editor. Today, the WebAPSEE environment constitutes an integration framework for a number services related to process management, including modeling, enactment, visualization, instantiation and enactment events. Using the Manager Console, the software process manager can design a process model, manage the process enactment, visualize process reports and manage the organizational information (artifacts, agents, resources, and others).

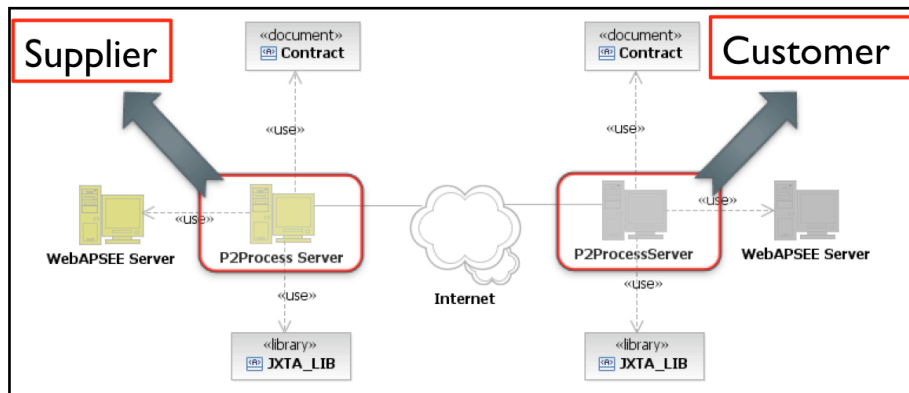


Figure 2 – General Deployment View

The WebAPSEE Task Agenda

Using the WebAPSEE Task Agenda Tool, the developer can visualize the software processes in which s/he is allocated to. Therefore, the developer interacts with the Task Agenda in order to provide feedback about the tasks performance status. Figure 1 contains the illustration of one screen of the Task Agenda Tool use, which represents the agent tasks at one specific process and allows accessing the task definition and to the produced and required artifacts

(associated to file upload and download functionalities).

4. A Decentralized Coordination Model to the WebAPSEE Environment

The decentralized process enactment approach uses distributed technologies to share process information among distributed organizations' sites. This approach contributes to a better process scalability avoiding the single failure point that exists at the centralized coordination model. As examples of PSEEs that follows this approach we can quote the environments Oz [2], Serendipity-II [13], Genesis [20] and SwinDeW [23].

In a general way, the proposal of this work to decentralize the processes enactment is summarized in the coordination of several activities among different PSEEs instances. The participating instances of PSEEs on the network must interact to maintain the consistency of the environment data and the shared processes.

As organizational advantages of this decentralized enactment model, we can quote: the opportunity to the organizations' systems to manage the cooperation with external partners; the opportunity to the process management systems balance the access to clients tools

(in this case, the access of Agendas and Manager Consoles will be balanced among the different decentralized sites); provide a model that can be reliable by customers and suppliers, because the organization's data is stored in its own domain.

Figure 2 presents in a UML deployment diagram the general view of this work that extends the current

WebAPSEE architecture defining the **P2Process** Figure 2 illustrates the customers and suppliers instances of their internal WebAPSEE environments. The **P2Process Server** component acts as a proxy among remote instances of a PSEE, integrating the process enactment through the exchange of messages among different decentralized remote sites.

Contract Management

Each of the involved environment instances possesses a copy of the component defined as Figure 2. The **Contract** component acts as a flexible information access control from the remote sites and

establish rules for data access in the processes decentralization. This approach allows the information filtering of confidential data among the involved sites.

The current tool prototype version already contains the electronic contract evaluator component, where the contract rules are evaluated to allow or to deny operations among involved decentralized remote sites. The distribution layer implements a peer-to-peer model using the JXTA peer-to-peer technology [9].

Process Integration

To delegate activities from the local process model to be performed by a decentralized remote site, the current prototype defines a delegation protocol. Depending of the current known suppliers, the customer chooses the destination of the activity's delegation process. The site that delegates an activity must send remote messages to the targeted supplier, and this delegation protocol has the following steps: a) request the operation to distribute a delegable activity; b) define the electronic contract to be applied to this delegation; c) send the additional data needed to enact the activity.

Process Enactment Events Synchronization

The enactment events synchronization among customers and suppliers provides the exchange of process enactment information among different instances of the PSEEs. Asynchronous messages define the enactment events, which are propagated to represent software process model events (as an activity state or an artifact state change). The internal implementation follows the publish/subscribe notification implementation approach to establish the communication among remote sites [5].

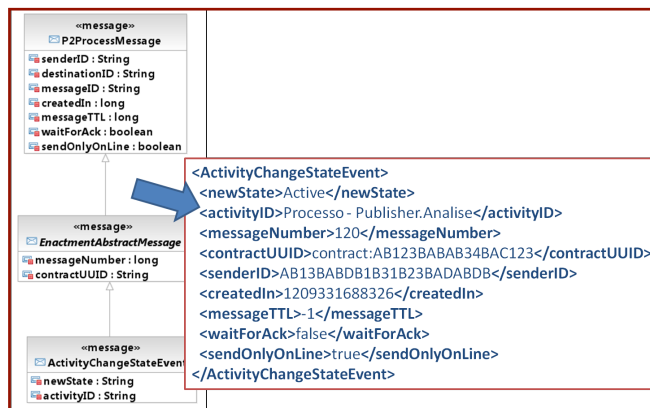


Figure 3 – Events Messages Serialization into XML

Figure 3 illustrates the serialization process that maps internal enactment events to XML messages

propagated through the peer-to-peer network. It provides an automatic and interoperable way to synchronize the different projects status in the decentralized software process context based on a PSEE approach.

5. Expected results from this work

This work focuses on decentralized software development, no matter if the remote sites are at different organizations (Outsourcing) or at different branches of the same organization (Insourcing). Depending on the organizational context of tool deployment, we expect a variation on the electronic contract rules. To the Outsourcing context, we expect more rigid rules instead of more openness rules for the Insourcing context. But in both cases, as the work described in [1], an explicitly contract definition creates a trustful environment in the customer-supplier relationship.

We expect that the decentralized enactment model using contracts to filter the remote access aggregate values in both directions of global software development: it aggregates competitive advantage to suppliers (which can offer a remote monitoring service) and management power to customers. Thus, the following sections present some expected results with the use of the proposed tool in real decentralized software development projects.

A Common Process Language

The authors Ebert [7] and Sengupta [22] report the need of a common language among the dispersed members of a remote relationship to have a better communication during the project. So, this work proposes that the benefits of a common process language in centralized PSEEs may be extended to decentralized software development projects. Therefore the visualization of information about decentralized projects using a process-oriented notation can help the project managers to realize the actual remote process enactment on their internal process model language.

Information Confidentiality in the Organization's Relationship

According to Sengupta [22] and Wells [24] the organization's information confidentiality is an indispensable aspect for a real deployment of software process decentralization. An important non-technical factor involved in the process decentralization is the trust among the involved parts. The methods and tools

proposed by this work must support alternatives to information confidentiality between customers and suppliers.

This work proposes the concept of an electronic contract, defining rules to enable or disable data access between customers and suppliers in a decentralized process context, as shown in Figure 4. Suppliers can configure the electronic contract, blocking the access of some information when sent to the customer. When customers request a supplier process view, it can visualize only the explicitly allowed components of the software process. Thus, a more openness access control can be defined to internal partners (the Insourcing case illustrated by the **Customer Process View A** in Figure 4) and external partners (the Outsourcing case illustrated by the **Customer Process View B** in Figure 4). These different visualizations allow the information confidentiality in the customer-supplier relationship.

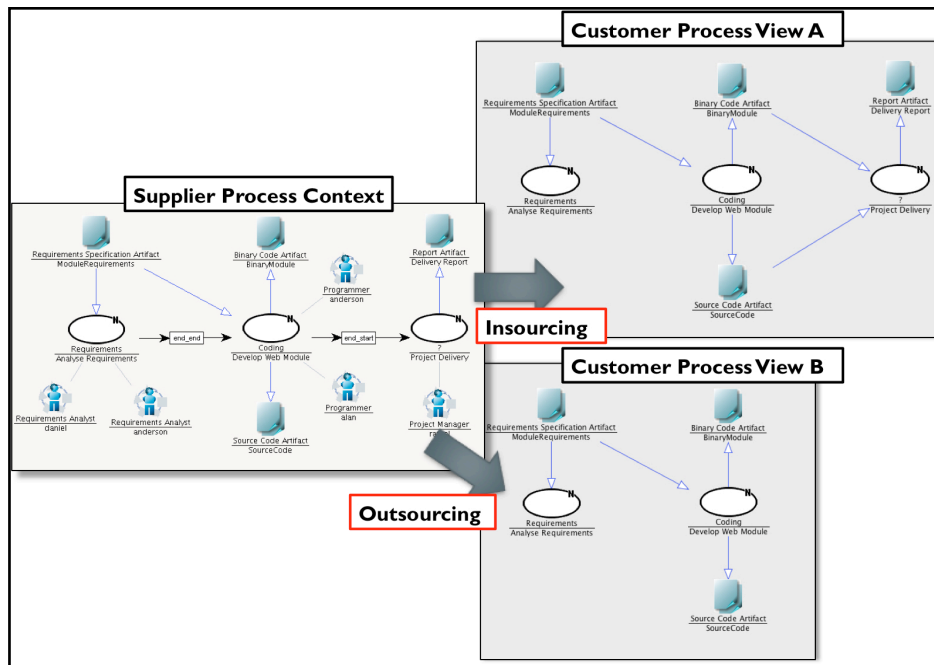


Figure 4 – Remote Process Context Visualization in Different Organizational Contexts

Adherence to the CMMI goals

The main rationale of this work deals with the demand of coordinated monitoring between customers and suppliers by process maturity models. The CMMI maturity model [3] and the eSCM model [16] demand to support the goal of work coordination between suppliers of software products. In the case of external suppliers, the monitoring of activities is an important aspect to verification of methods and products used in the software development.

The current proposal of decentralized process coordination can benefit the project monitoring process performed by the customer, by providing a higher automation degree on the remote information retrieval. As a consequence, this work proposal created an open channel of communication between customers and suppliers. In this way, the higher automation degree in suppliers' monitoring can help customers to better manage different suppliers simultaneously, since the retrieve of remote process status is effortless.

6. Conclusions and Future Work

The Genesis [20] and SwinDeW [23] tools define a homogeneous integration model to external tools. Instead of following this homogenous integration model, this work defines a XML way to exchange messages among different PSEEs, and open the opportunity to integrate different PSEEs or process-oriented tools on this coordination approach.

The need for consensus in Global Software Development tools [15] can be achieved by the contract definition in each one of the involved sites.

Applying the approach of this work in decentralized software development can lead to the customers the power to know the current supplier development status. It can make the risks management and decision-making process easier, because can lead to customers an early decision before the project

risks be higher than the accorded value with the supplier. To the suppliers, the process transparency can represent a market advantage.

The management and control over communication activities help minimize the management and communication problems. Integrating to process support tools, as PSEEs, we can bring a higher automation level to monitoring tasks in Global Software Development [15] and provides a better control over variable costs on software projects [21].

The functionality to retrieve remote process views (as shown in Figure 4) can help to minimize the context-sharing problem [19] by the process-oriented view of the remote project described in this work.

The current prototype of the tool is under beta version to evaluate its effectiveness in study cases with local industrial partners. In specific, it is expected to evaluate if the scalability of the proposed notification message propagation approach is able to handle the demand for Global Software Development projects [5].

As future work, one of the requirements cited by [8] is the difficulty to choose potential partners to support a development demand. Both the CMMI process maturity model [3] and the eSCM model [16] define that to choose IT suppliers it is necessary to prescribe the entire process and be based on objective criterion to choose the best supplier for each case. Thus, as a possible extension of this work we can provide automatic client-supplier relationships metric collection and support the choose of possible suppliers using the historical data from decentralized projects.

References

- [1] Babar, Ali; Verner, M.; et al. (2007) "Establishing and maintaining trust in software outsourcing relationships: An empirical investigation". In: *Journal of Systems and Software*, vol. 80, 9, September, 2007.
- [2] Ben-Shaul, Israel Z.; Kaiser, G.E. (1998) "Federating Process-Centered Environments: The OZ Experience". In: *Automated Software Engineering*, vol. 5, 1998.
- [3] CMMI. CMMI Web Site. <<http://www.sei.cmu.edu/cmmi/>>.
- [4] Conradi, Reidar; et al. (1994) "EPOS: Object Oriented Cooperative Process Modeling". In: *Software Process Modeling and Technology*, Research Studies Press LTD, 1994.
- [5] De Souza, C. R. B., Basaveswara, S. D., Redmiles, D. F. (2002) "Supporting Global Software Development with Event Notification Servers". In: *24th International Conference on Software Engineering, International Workshop on Global Software Development*, pp. 9-13, Orlando, Florida, EUA, 2002.
- [6] Derniame, Jean-Claude, et al. (2002) "A Comparative Review of Process-Centered Software Engineering Environments". In: *Annals of Software Engineering*, vol. 14, p. 311-340, 2002.
- [7] Ebert, Christopher "Global Software Engineering". IEEE Ready Note (e-Book), IEEE Computer Society, Los Alamitos, USA, 2006.
- [8] Ebert, Christopher (2007) "Optimizing Supplier Management in Global Software Engineering". In: *Second International Conference on Global Software Engineering, ICGSE 2007*, Munch, Germany.
- [9] Flenner, Robert; et al. (2003) "Java™ P2P Unleashed: With JXTA, Web Services, XML, Jini, JavaSpaces, and J2EE". Sams Publishing, EUA.
- [10] Fitzgerald, Brian; Ågerfalk, Pär J. (2006) "Flexible and distributed Software Process: Old Petunias in New Bowls?". In: *Communications of the ACM*, vol. 49, n. 10, October, 2006.
- [11] Fuggetta, Alfonso. (2000) "Software Process: A Roadmap". In: *Proceedings. of The Future of Software Engineering, ICSE'2000*, Limerick, Ireland, 2000.
- [12] Gruhn, Volker (2002) "Process-Centered Software Engineering Environments: A Brief History and Future Challenges". In: *Annals of Software Engineering*, vol. 14, 363–382. Kluwer Academic Publishers, 2002.
- [13] Grundy, J.C., et al. (1998) "An architecture and environment for decentralized, internet-wide software process modeling and enactment". In: *IEEE Internet Computing: Special Issue on Software Engineering via the Internet*, vol. 2, n. 5, IEEE CS Press, September/October, 1998.
- [14] Herbsleb, James D. (2007) "Global Software Engineering: The Future of Socio-technical Coordination". In: *International Conference on Software Engineering, Future of Software Engineering*, IEEE Computer Society, Washington, DC, USA, 2007.
- [15] Hillegersberg, Jos van; Herrera, Miles (2007) "Tool Support for Distributed Software Development: The past – present – and future of gaps between user requirements and tool functionalities". In: *Tools for Managing Globally Distributed Software Development (TOMAG 2007)*, Munch, Germany, 2007.
- [16] Hyder, E. B.; Heston, K. M.; Paulk, M. C. (2006) "The eSCM-SP v2.01: Model Overview, The eSourcing Capability Model for Service Providers (eSCM-SP) v2.01". Published in <http://itsqc.cmu.edu>.
- [17] Kaiser, G.E.; Barghouti, N.S.; Sokolsky, M.H. (1990) "Preliminary Experience with Process Modeling in the Marvel Software Development Environment Kernel". In: *Annual Hawaii International Conference On System Science*, Kona., p. 131-140
- [18] Lima, A. M.; Reis, R. Q.; Lima Reis, Carla A. "Gerência Flexível de Processos de Software com o Ambiente WebAPSEE". In: *XIX Brazilian Symposium in Software Engineering – Tools Session*, October, 2006.
- [19] Prikladnicki, R.; Audy, J. L. N.; Damian, D.; Oliveira, T. C. (2007) "Distributed Software Development: Practices and challenges in different business strategies of offshoring and onshoring". In: *Proceedings. International Conference on Global Software Engineering (ICGSE 2007)*, Munch, Germany, 2007.
- [20] Ritrovato, Pierluigi; Gaeta, Matteo "Generalised Environment for Process Management in Cooperative Software Engineering". In: *Proceedings. 26th Annual International Computer Software and Applications Conference*, 2002.
- [21] Sabherwal, R. (2003) "The evolution of coordination in outsourced software development projects: a comparison of client and vendor perspectives". In: *Information and Organization*, vol. 12, n. 3, pages 153-202.
- [22] Sengupta, Bikram; Chandra, Satish; Sinhá, Vibha. (2006) "A Research Agenda for Distributed Software Development". In: *Proceedings of International Conference on Software Engineering (ICSE'06)*, Shanghai, China, 2006.
- [23] Yan, Jun; Yang, Yun; Raikundalia, Gitesh K. (2003) "Decentralised Coordination for Software Process Enactment". In: *Lecture Notes in Computer Science*, vol. 2786, Heidelberg: Springer Berlin, 2003.
- [24] Wells, Thomas O.; Braunfeld, Roger. (2001) "Protecting Your Most Valuable Asset: Intellectual Property". In: *IEEE Special Issue in IT Professional*, vol. 3, issue 2, pages 11–17, March-April, 2001.
- [25] SPICE. ISO SPICE Web Site. <<http://www.isospice.com>>

A Framework to Analyze Impact of Change in Component Based Software Engineering

Kuljit Kaur¹, Hardeep Singh¹, and Debasish Jana²

¹Dept of Computer Science and Engineering, Guru Nanak Dev University, Amritsar, India

²Senior Member, IEEE, Simplex Infrastructures Ltd, Kolkata, India

kuljitchahal@yahoo.com, hardeep_gndu@rediffmail.com, djana@alumni.uwaterloo.ca

Abstract

Component Based Software Development is a reuse-based approach. New applications can be developed on the fly by integrating already existing software components. The components are either available in the in-house reuse library of the application development organization, or are procured from third parties. So in this process, development of an application is not confined to a single organization but distributed across the globe. One of the issues in this global software development scenario is that component users have limited control on the development and evolution of third party components. Component users or application developers have to keep track of new offerings produced by the component vendors and upgrade their products accordingly as vendors may not support the old versions of the components any more. In a component based software application, when a component is upgraded, added or removed, it can affect the components depended-by and dependent-upon this component in the application. In order to analyze the impact of change in one component on other components in the system, we have to identify and explicitly specify the inter-dependencies of the components. This paper studies dependency analysis, various types of dependencies and techniques used to manage the dependencies. We propose a matrix-based representation to record inter-dependencies of software components. Impact Analysis can be carried out using the proposed algorithms. An experimental study is presented at the end.

1. Introduction

In Component based software paradigm, developing an application involves the assembly or composition of preexisting, reusable and independent pieces of software components [8]. Components communicate and share information in order to provide

system functionalities. These components are self-contained units, which can interoperate with other such components in the system through well-defined interfaces. Dependencies in the software components should ideally be identified and controlled in the initial phases of the software development life cycle, so that the software product can be reconfigured easily without spending much effort, and quality of the product can also be maintained even after the revisions are done [9].

Object Oriented Development endorses compartmentalization of system into several self-contained reusable components. Reusable components provide ready-to-use structural as well as behavioral artifacts for use by other components. Less coupling and high cohesion are important criteria for a good component design. A software system is bound to change. Managing changes effectively and efficiently is prime concern of today to build a resilient architecture. In this context, our research work is targeted towards analyzing the impact of change in component-based software engineering. The paper is organized as follows. Section two emphasizes on the need and methodology of dependence analysis with types of dependencies. Related work in this area has been studied in section three. Our proposed approach towards our goal of this paper is presented in section four. Here we analyze the type of change of one component and its impact on the dependent components in the system. We also present the methodology of managing dependencies using dependence matrix and associated algorithms. Experimental study and results are shown in Section five. Finally, in section six, we conclude with vision for future work.

2. Dependence Analysis

Defining a Dependency - A dependency is a relationship involving two or more elements where a change of structure or behavior in one or more

elements leads to a potential for a change of structure/behavior in one or more other elements.

Instead of merely saying that two things are dependent on each other (either directly or indirectly through a transitivity chain), we are interested in the type of the dependency.

2.1. Types of Dependencies

A component can depend upon another component in either or both of two ways – Structure or Behavior [6]. Both structural and behavioral dependencies are important to capture and understand when analyzing architecture. A change in behavior that changes the publicly visible signature of a component affects the client. However, if the change is internal to the component, for example, instead of following FCFS manner, a queue works in a priority ordering, but the externally visible signatures for insertion and deletion remain same, then we mention that the client is behaviorally dependent but not structurally dependent. In case of structural dependencies, some categories as identified in [6] are –

- Includes – A component may be created from other subcomponents, using relationships such as aggregation or composition. For example, a Reservation Component for a Hotel Management System consists of CustomerManager Component, Billing Component and RoomAllocation component.
- Import/Export – The specification for a component may describe the information imported or exported between modules. Sometimes, the implementation of a function within a component may use some other component related through an association type of relationship. For example, in an order management system, an order component depends upon the Inventory component..
- Inheritance – The specification for a component may be created through inheritance from other components. The inherited components extends the basic functionality provided in the base component.

The behavioral dependencies (6) can be categorized as-

- Temporal – The behavior of one component precedes or follows the behavior of another component.
- Casual – the behavior of one component implies the behavior of another component.
- Input – A component requires information or stimulation from another component.

- Output - A component provides information or stimulation to other components.

In a nutshell, a component A can be dependent upon -

- The structure of another component B or
- The behavior of B
- Or both (i.e. the structure as well as behavior of B)

It is easy enough to develop a directed graph of the dependencies; however, such a graph is too simple to analyze complex software systems. We need a model of dependency relationships that goes beyond just knowing that two things are dependent upon each other.

3. Related Work

As inter module dependencies can have large impact on the present and future structure of a software product, the identification and exploitation of dependencies has been a subject of research. Most of the research work has been done to analyze dependencies at the implementation level i.e. in program code. Several tools are available that can automatically extract the dependencies in the program code such as JDepend, which identifies dependencies from programs written in Java language [5]. At the architectural level too, tools such as Aladdin are available, which can extract dependencies from specifications written in Architecture description language [6].

Li [3] has used dependency matrix based technique for managing dependencies. Li has identified eight types of dependencies in a component-based system : dependency related to data, control, interface, time, state, cause-effect, input/output and context. The value for a cell in the matrix is defined using Boolean OR operator over all eight types of identified dependencies. The cell takes a value 1, if any of the eight dependencies exists in a pair of software components and 0 otherwise. Li [3] has suggested using different matrices for eight different types of dependencies. Stafford et al [7] used dependence matrix to show interdependencies of interfaces of software components. Rows and columns of the matrix correspond to the in and out interfaces of software components. But, such a solution cannot be easily scaled up to large complex applications, because one component may support multiple interfaces in an application and number of operations per interface may also be very large. The work presented in this paper is an extension to the technique used by Li [3] for managing dependencies.

4. Proposed Approach

As per the previous discussion, a component can be dependent upon either the structure or the behavior of another component. The idea behind the proposed approach is that if a change is required in the structure of a component, then only those components need to be changed which depend upon its structure. As the behavior of a component changes, the components dependent upon its behavior should be considered for change. According to the proposed model, for every dependency in a pair of components, the type of dependency also needs to be recorded. This information can help in correct impact analysis, whenever a change is requested in the system.

4.1. Analysis of Type of Change and Its Impact

We will now analyze type of change and its impact through an example. Let us take an example of a component say *ElementSearch*, which can search for a given element in a given list and report the presence/absence of that element in the list. *ElementSearch* implements linear search algorithm. Another component say *DataManagement*, uses this component for searching purposes. *DataManagement* component depends only upon the behavior of *ElementSearch* component, not on its structure.

In table 1, we consider some situations in which a change in component *ElementSearch* can impact the component *DataManagement*.

We can observe in table 1, that a change in the structure of *ElementSearch* does not require a change in *DataManagement* component. But a change in behavior may result in change in the behavior of the dependent component. This later type of change may not always be necessary, because a component can support multiple interfaces. This approach is followed in Microsoft COM+ environment [4]. As in the last case discussed in table 1, *ElementSearch* can maintain a different version of the interface for its old clients (e.g. *DataManagement* in this case). But this kind of dependence analysis can help us to know about the overhead (cost of managing multiple versions of interfaces) caused by a change in the behavior of a component.

4.2. Managing Dependencies Using Dependence Matrix

Here we consider a dependence matrix as a 3-dimensional matrix. For a component-based system of N components, every cell of the $N \times N \times 2$ matrix contains information about type of dependence (i.e.

structural or behavioral) and strength of dependence (i.e. number of references) of the corresponding pair of

Table1: Impact of change in a component on a dependent component

Type of Change in <i>ElementSearch</i>	Description	Impact on <i>DataManagement</i>
Structure changed, Behavior unchanged	Static memory allocation is changed to dynamic memory allocation.	No change
Structure changed, Behavior unchanged	Linear search algorithm changed to binary search algorithm. Component uses a sub module to sort the unsorted list received from <i>DataManagement</i> component	No change
Structure changed, Behavior changed	Searching algorithm changed to binary search, but <i>ElementSearch</i> does not itself sort the data, <i>DataManagement</i> has to provide now a sorted list	Change required
Structure unchanged, Behavior changed	<i>ElementSearch</i> now provides the location of the searched item present in the list too	Change required (if the component wants to use this feature).

software components. Type of dependence is not stored but just indicated by the location of information. This has been done keeping in mind the memory savings. First node of the cell stores number of references for structural and second node stores the number of references meant for behavioral dependencies. For example in figure 1, number of structural dependencies is 6 and number of behavioral dependencies is 8.

6
8

Figure 1: Contents of a cell of a matrix.

The matrix for a hypothetical system design is given in figure 2. Let us consider a component-based software consisting of 3 components C1, C2, and C3.

Dependency Matrix for the system is given in figure 3. Rows show the components depended upon and columns show the dependent components. For example in fig. 2, C2 has a component C1 dependent upon it. C1 has two references to the structure of C2 and 6 references to the behavior of C2. So whenever a change to C2 takes place, C1 may also have to be modified. C3 has two dependents C1 and C2.

Table 2: A Dependency Matrix

Components → ↓	C1	C2	C3
C1			
C2	4		
	6		
C3	1	6	
	0	3	

4.2.1 Dependence Analysis Algorithms

Suppose that we have to see the impact of making a change to one component of a system. The steps to see the impact of change are as follows: -

1. Create a dependence matrix for the system.
2. Identify the type of change i.e. whether it is a change in the structure of the component or in the behavior of the component or both.
3. Proceed to
 1. Algorithm (1.A) if the change is in the structure of the component,
 2. Algorithm (1.B) if the change is in the behavior only,
 3. Algorithms (1.A), (1.B) in the sequence if component's structure as well as behavior is changed.

Algorithms (1.A) and (1.B) are explained in the following paragraphs-

Algorithm (1.A) – Algorithm for changes in the structure of a component C

- 1) Find the row of the dependency matrix corresponding to component C.
- 2) Select all those columns of the matrix in which the cells in this row have non-zero values for structural dependencies.
- 3) The components corresponding to this selected set of columns are the components that will be directly affected by this change.

For transitive dependencies of this change, repeat step 1-2 for the set of components identified in step 3.

Algorithm (1.B)- Algorithm for changes in the behavior of a component C

- 1) Find the row of the dependency matrix corresponding to component C.
- 2) Select all those columns of the matrix in which the cells in this row have non-zero values for behavioral dependencies.
- 3) The components corresponding to these columns are the components which will be affected by this change.

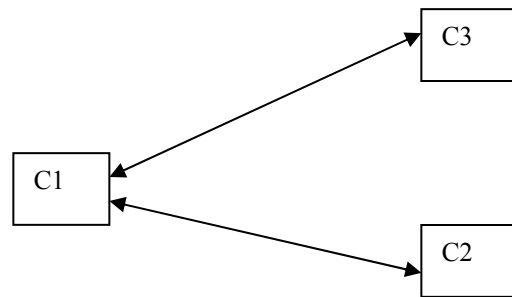


Figure 3: Interdependencies of components

Unlike the (1.A) algorithm we do not consider transitive dependencies in this algorithm, because one component can support multiple interfaces for different components. The components with which it directly interacts are the only ones that may be affected by this change. For example, in figure3, components C1, C2 and C3 communicate using two-way communication links. Now let us imagine that behavior of C1 with C2 changes.

This change will not affect component C3. C1 will provide to C3 same old interface, it will support a new updated interface for C2. So C1 will now be supporting two interfaces, one for C2 and another for C3.

5. Experimental Study and Results

Let us consider the component based software system for a *hotel reservation management system* as given in [1] .It consists of four components – *Reservation System, Billing, Hotel Mgt, Customer Mgt.* Dependency Matrix, given in figure 4 is constructed using the proposed model :

Components → ↓	Reservation System	Billing	Hotel Mgt	Customer Mgt
Reservation System				
Billing	0			
	1			
Hotel Mgt	0	0		
	3	0		
Customer Mgt	0	0	0	
	4	0	0	

Figure 4: Dependence Matrix for Hotel Reservation System

All the components are independent of each other as far as their definitions are concerned i.e. they do not have any structural relationships. Except the component *Reservation System*, other components are not directly connected with one another. So their behavior is also independent of others. Let us assume that a structural change is to be introduced in *billing* component, regarding the customer charging rules. Using the algorithm (1.A), we can see that no component is affected by this change, as in the matrix, row corresponding to billing component has got 0 structural dependencies. Comparatively, if Li [3]’s approach is applied in the same situation, then a structural change in billing component will reportedly affect reservation component, which is actually not the case. Because, the structural change (not any change in publicly exposed interface) in *billing* component is confined within the implementation of the *billing* component itself and should no way, affect the *reservation* component. As such, the proposed algorithm of our paper shows better results than Li [3]’s in this aspect. For a change in the behavior of *billing* component, as per (1.B) algorithm only component *reservation system* may be affected, so *billing* component has to provide an additional interface for one component only. In addition to this we can see that *reservation system* is more tightly coupled with *customer Mgt* component as compared to billing component.

6. Conclusion

In this paper, we have studied dependency analysis in component based software systems from impact analysis point of view. This can further help in estimation of maintenance cost of software i.e. higher is the impact of change in a part of the system; more is the cost of implementing that change. A dependency matrix based approach is proposed for recording the

interdependencies in software components. Algorithms have been formulated to know the effect of a change in a component of the system. It has been applied to a simple system too, and the results show that impact analysis is more correct than the existing approach used for the same purpose.

References

- [1] Cheesman, J., Daniels, J.: “UML components – A Simple Process for Specifying Components”; Component Software Series, Addison-Wesley.
- [2] Crnkovic, L., Larson, M.: “Building Reliable Component Based Software Systems”, Artech House.
- [3] Li, B.: “Managing Dependencies in Component Based Systems Based on Matrix Model”, Proceedings Of Net. Object. Days, 2003 - netobjectdays.org available at www.citeseer.ist.psu.edu/650086.html.
- [4] Microsoft COM+, <http://msdn.microsoft.com>.
- [5] Sanghal, N., Jordan, E., Sinha, V., and Jackson, D.: “Using Dependency Models to Manage Complex Software Architecture”, OOPSLA ’05, San Deigo, California, USA.
- [6] Stafford, J., Alexander, W.: “Architecture level Dependence Analysis in support of software Maintenance”, ISA W3, Orlando Florida, USA, ACM press.
- [7] Stafford, J., Richardson, D., Wolf, A.: “Chaining: A Software Architecture Dependence Analysis Technique”, Technical Report CU-CS-845-97, department of Computer Science, University of Colorado, USA
- [8] Szyperski, C.: “Component Software: Beyond Object-Oriented Programming”, Addison Wesley.
- [9] Vigdar, M., Kark, A.: “Maintaining COTS based Systems: Start with the Design”, IEEE Computer Society Press.

A suite of tools for the automation the management of the software process

Javier Berrocal, José Manuel García, Juan Manuel Murillo
Universidad de Extremadura
10071 Cáceres
{jberolm, jgaralo, juanmamu}@unex.es

Abstract

A proper management and the control of each element of the software development process have become very important to obtain high quality software. However, with the rise of globalized software development, these activities have become more complex and, at the same time, more important, not only to achieve high quality software, but also for the success or failure of a project; so new tools better adapted to these activities and environments are needed. This has led us to develop a suite of tools that facilitate and automate, as much as possible, the management and control of software development in distributed environments. Making the realization of tasks easier and, thus, increasing the productivity of each development team member, and building higher quality software.

1. Introduction

In the last few years all the software companies have greatly advanced in the development of better quality products, but in the shortest time and lowest cost possible. This evolution has been mostly directed towards two areas: to improve the software processes and to improve the business models.

In the past, a series of sequential processes were carried out which had many problems, such as overloading of some tasks, a higher price of the change of the requirements, etc. [1]. Nowadays, by improving the software processes, there are some iterative processes such as RUP, XP, Scrum, etc. that focus on a greater collaboration and communication between the development teams and with the customers [2] to obtain better products and a greater satisfaction of the customers and the development team.

Also, the business models have changed towards a higher industrialization and globalization of the software process, seeking new models to achieve

higher productivity but with lower costs. With respect to these models, the near-shore and off-shore factories have prevailed, in which developments are made in factories far from the customers, where the lower standard of living allows for lower development costs. However, the cultural differences, the time zone and some difficulties with the management, the coordination and the communication between the development team members or with the customers are the main handicaps for the success of the projects. Moreover, these problems especially arise when these models are combined with iterative processes, in which collaboration and communication are key practices [3]. A good example of this is reflected in a recent survey of the DR. Dobb's Journal [4], which shows that the offshore projects have a 42.7% success rate compared with 71.5% for the projects done with agile methodologies and on-site.

For these reasons, the software companies and the research groups have been encouraged to seek and build some kind of tool that reduces the problems associated with geographic scattering and supports the different tasks of the software process [5]. However, the use of different tools for each task, which are correlative or are very interrelated, entail the need of a greater effort in their coordination. So, even though they have begun to develop applications to coordinate the different tasks of the development [6], they are putting aside the integration with the tools for the management of the development, which is a crucial area to increase the quality of the products and the success of the projects. For these reasons, in this paper we propose a suite of tools which, besides supporting and coordinating most of the tasks of the software process, is also able to integrate and automate the activities for the management and execution of all kind of software processes in distributed environments, although for the moment we are focusing more on its adequacy for RUP. In this paper, besides presenting the architecture of that suite, we are going to show an

example explaining how the different tools work together.

With this suite of tools, a set of applications to manage and control the software process is provided, and with these the coordination and the management of the process are improved and the independence of location of each resource is achieved. In addition, we are going to show how by integrating two of these tools, one for the management and the other for the execution of the software process, the automation and agility of a large amount of the work of coordination and management is achieved, increasing the productivity and the quality of the development. These advantages were obtained with the introduction of the tools in the factory of Indra¹ in Cáceres, too.

The rest of the paper is structured in the following way: in section 2 we present some of the tools and methodologies that exist at present, as well as the motivation that leads us to present this work; in section 3 we describe the architecture of the suite of tools; in section 4 we present an evaluation of the tools in real cases; and, finally, in section 5 we present the conclusions and further works.

2. Background and motivation

In the last few years, software development has become a more industrialized practice, with developments similar to the production chains where the tasks are perfectly fixed and they are done by certain resources with fixed roles. In this environment, the management and coordination of each element of the process are essential activities to increase, or at least maintain, the quality of the software created. However, even though there are a lot of applications for the software process management, which even manage to contemplate the globalization of development (such as Jazz Team Concert [12]), they don't cover all the areas of the management, so, it is necessary to use different tools to control each area [6,13]. Therefore, the use of a lot of tools for the management, coupled with the lack of integration between them and with those that support the collaboration and the completion of each task of the software process, cause an increase in effort and a decrease in productivity.

In addition to the industrialization of the software process, companies are continuously evolving the software processes that govern them, making them more agile and efficient. But, as the software processes evolved, we could see the need to document each of the

different practices being carried out, in order to be properly transmitted to the entire development team, because, without this information, we could fall into some inconsistencies and lack of coordination when carrying out certain practices. Thus, for documenting and modeling these practices, some metamodels like SPEM [7] and tools like Eclipse Process Framework [8] have emerged. Nevertheless, while the software development has been scattered, the control and the management of the projects has appeared as a more and more complex activity, and the documentation of the process for the proper coordination and control of all of the tasks is not enough. Thus, it is necessary to develop methods and tools capable of automate the management and the control of the software process. In this way, in recent years many research groups have developed method for the modeling of software processes like workflows, using notations such as UML Activity Diagram, BPMN, etc.; and the execution of these workflows in some application, using similar technology by which Business Processes of an organisation are executed on a BPMS [9, 10, 11]. However, these studies were more oriented towards the documentation and the execution of the software process in order to automate some tasks of the process or a part of the development, so they obviated the utilization of this execution for automating and expediting the management and coordination of the development.

All of these problems have motivated us to build a suite of tools that is capable of: on the one hand, supporting (in an integrated fashion) all the work of the software process management, making the job of the control and the coordination of the activities of development easier, and is able to execute the software processes, which is used for automate much of the work of management, and on the other hand, building a subset of modules capable of supporting each of the tasks of the iterative processes, and which are perfectly coordinated among themselves and with the management and process execution modules.

3. Architecture of the suite of tools

In Figure 1 the architecture of the suite of implemented tools to support the management and execution of the software process is shown, as well as the set of modules used to do each task. As can be seen, the architecture is divided into three layers, each of which has a very clear set of responsibilities, and each layer is also divided into different modules in charge of carrying out specific responsibilities.

¹ Indra is a software enterprise with more than 23000 employees, and it has customers in more than 82 countries.

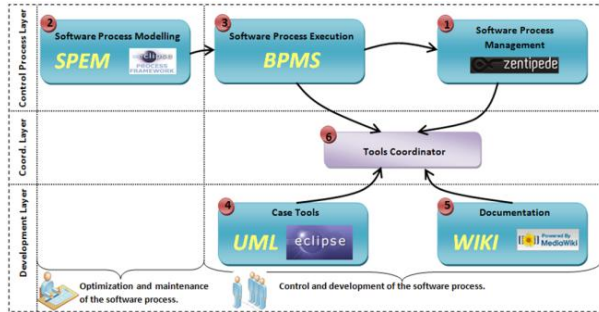


Figure 1. Architecture of the suite of tools

The first layer, *Control Process Layer*, is in charge of doing an exhaustive control of the software processes that govern the development company. To carry out this control, the layer has three essential modules to lead the software processes, which are:

- *Software Process Modeling*; this is a subsystem used for modeling and documentation of software processes. It is based on the SPEM meta-model and the EPF tool [7, 8]. But, SPEM only focuses on providing the needed structures to model software processes and it gives freedom to use any standard (UML2, BPMN, BPDM, etc.) to model its behavior. In this sense, EPF implements the entire SPEM specification but only allows the use of Activities Diagrams to model its behavior. For this reason, we have made an extension to model the behavior with BPMN in the EPF, thus facilitating its modeling like a business process and making its further execution easier. Also it facilitates the control of the existing variability in a process when it is applied in different projects, or when we want to focus more on a specific discipline.
- *Software Process Execution*; this module is for the execution of the software process models. We have based this execution on the BPMS technology but we have made some adaptations to fulfill the specific characteristics of the software process execution; it is in this BPMS where the software processes can be run, due to the previous model in BPMN of the behavior with the *Software Process Modeling* tool. With this execution a greater automation and agility in the management and in the coordination of each process's element is achieved.
- *Software Process Management*; this module is in charge of monitoring and managing each of the elements involved in the software development, making the tracing of the

projects and the resources easier and generating a large amount of statistics (such as the status of each task, the time of execution, the workload of each resource, etc.) which could be used to do an evaluation of the situation or for making future decisions.

The last layer, *Development Layer*, contains each of the necessary tools to carry out each of the tasks and activities for the development. Thus, this layer has two essential modules that are responsible for covering all the development areas, they are:

- *Case Tools*; a set of plug-ins based on the Eclipse IDE that support each of the activities of the software process, through which methods to automate them as much as possible are provided, facilitating their synchronization and consistency with the existing information and maintaining the traceability throughout the entire lifecycle of a project.
- *Documentation*; a tool based on a Wiki that has been specially adapted for the software development, with which any kind of artifact or document generated from the tasks of the development can be maintained and synchronized.

The middle layer, the *Coordinator Layer*, is in charge of abstracting and granting technological independence between the other two layers. This is a very important issue because the control of the software process and the realization of each task are in very close ambit with a lot of relationships among them, where a proper integration between both areas is crucial to expedite the development as much as possible; but at the same time, each tool in each area must be sufficiently independent to allow that changes in one area do not adversely affect others. To carry out this isolation, the module of this layer is:

- *Tool Coordinator*; this module grants the isolation and technological independence between the others layers, allowing it to evolve and modify the functions of a layer without having to make changes in the others. And, at the same time, it is in charge of coordinating and controlling all the modules of each layer to work together, making the tasks more agile and automating as many activities as possible.

To explain the different functionalities and operations of each of the above modules more clearly, in paragraph 3.1 we show an example of how to use all of them; nevertheless, we are going to focus on showing and explaining in more detail how the *Control Process Layer* modules works.

3.1. An example of use of the suite of tools

To see more clearly how these tools are coordinated to develop software, and specially how the *Control Process Layer* modules work together, Figure 2 shows a generic example for any process and project driven by use cases. If we would like to use another process we would only have to define the set of tasks and the artifacts generated during the development.

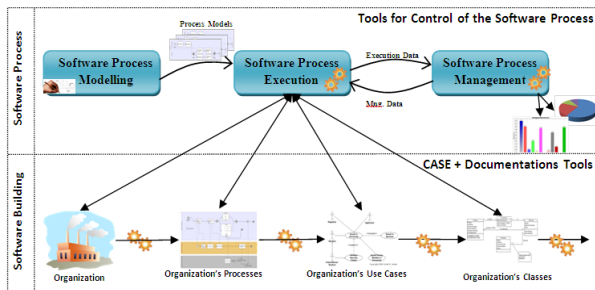


Figure 1. Example of software building with the set of tools.

For any software company, whether they are distributed or not, it is advisable document the software process and the practices that have to be followed for each project to ensure a better coordination between tasks and between resources and to provide a greater knowledge of the process to all members of the development team. In this way, before the development of a new project starts, an activity that should be done is to model and define the software process that will be followed during all the development. Because of this, the module *Software Process Modeling* is used, which is able to model the software process with SPEM and its behavior with BPMN. However, even if modeling the entire software process for each project could appear to be a complex and tedious task, the reality is that normally software companies have perfectly modeled and documented the software process that they follow, therefore this process can be used like a template that can be easily adapted to cover the variability of each project, in the event that it is necessary.

Afterwards, once the software process to follow is modeled or adapted, apart from for documentation, it can be executed in a BPMS, only if it is made with BPMN. Thus, once modeled, this process is deployed in the *Software Process Execution* to be run. Through this execution we can follow the sequentiality of each task inside the process and coordinate and control what task is being done in every moment, which has to be

the next, which is the status of each task or sub-process, etc. In addition, it makes the jobs related to the coordination, control and management of the process easier, because, through the execution, it is possible to automate a large number of activities, such as inserting tasks in the management module (because it knows when each task starts), updating the status of each task (because it knows its situation), allocating the task to the resource with the least workload (reading the workload of each resource from the management tool), etc. But the BPMS not only coordinates the work from the model made, but also the software process and its execution are adjusted while the project progress, thereby hugely increasing the flexibility of the process and the tool. To do this adaptation, the BPMS is able to read and evaluate the results and artifacts generated at the end of each task, and especially when these tasks are done using the tools built for that purpose (module 4 and 5); once the results are evaluated, the process is adjusted to be in keeping with the status of the project. Therefore, for example, if we are following a process driven by use cases, once the Use Case Diagram is made (which is the result of a task of the process), the BPMS reads that diagram and creates, for each use case, an instance of a sub-process which controls the status and directs its development until the end of the use case.

At the same time that the process is executed in the BPMS, an efficient administration and management of each project and each element of this project becomes necessary for the proper control of the development, because this information could be crucial for the success or failure of a project, mostly when its size is increased or the development is made in more complex environment like distributed environments. To carry out all the common administration and management activities, but in an integrated way and in the same environment, the module *Software Process Management* is used, which from the moment that the project starts, begins to administer and register all the important information for the development. But, this module is also able to obtain the information in a manual way, by the members of the development team, or in an automatic way, being extracted and caught from work carried out by the other modules, such as the BPMS. Moreover, this module has more functionalities which differentiate it from many of the other management applications, such as: facilitating statistics and functionalities oriented to software factory management (showing statistics about the manpower that has been estimated and invested, differences between estimation and real-time invested, etc), making the documentation of the management and the software

process activities easier (through the integration with the documentation module, which allows the auto-creation of documents and a better collaborative editing of these), a perfect traceability since a requirement is inserted until its source code is created (due to the implementation of a plug-in for Eclipse) and, through its integration with the *Software Process Execution* module, allowing the automation of the most activities and making the rest of activities for project management more agile.

Finally, to perform all tasks in a coherent and consistent way, the different tools that are used to carry them out have to be perfectly coordinated and working together, achieving: a greater efficiency to make each task and in turn a greater flexibility of the process, because the results are used by the *Software Process Execution* module to adapt the process. To obtain this coordination the module Tool Coordinator is used, which is in charge of, firstly, coordinating the different tools used to make each activity of the process and, secondly, isolating them in order to evolve or modify some functionalities without effecting the others.

4. Tools evaluations

In the last few sections we have shown the theoretical benefits of using this suite of tools, but what is really important is the validation of these benefits in real environments. To that end, this suite of tools has been deployed at the factory owned by Indra in Cáceres, providing a perfect environment to test all the features in situations of real development.

For the testing of the suite of tools, we began by checking its functionalities in the development of small projects that were carried out completely in the factory, where the interactions with other factories were only to know the state of development. In this situation, the members of these projects ratified the benefits listed above, including: the factory manager emphasized its usefulness to obtain deeper knowledge about the utilization and profitability of the factory, facilitating awareness of how to improve this; the project leaders highlighted its utilization to achieve a better coordination and management of the projects and a better traceability and control of each task; and the analyst detailed a better use of the manpower and an increase in the productivity of each user and their satisfaction.

Once the tools were tested in small projects and the profits were demonstrated, we began to incorporate larger projects whose developments involved several factories. In this situation, the profits were similar to smaller projects; however there were some problems

with the coordination between the different factories, which were mainly because the tools were only being used in the factory in Cáceres. Once we began to deploy the tools in the rest of factories most of the problems were solved.

Once that all the tests were finished, the results obtained by us, as well as those transmitted by the company to us, have been very positive, so now we have started to evaluate the use of these tools in a larger number of projects, as well as projects where the offshore model is applied.

5. Conclusions and further work

In the last few years the software companies have improved both the software processes that govern them and the business models that manage them with the purpose of developing higher quality products but in the shortest time and lowest cost possible. However, these evolutions have some handicaps that threaten the success of the projects, such as difficulties in communication, coordination and control of the software. In order to solve these problems, a tools suite has been shown, in this paper, which can cover everything from the documentation and modeling of the software process to the management and control of each task of the development. Finally, we have shown how the integration of these tools is the best method to automate most of the management activities and increase the flexibility supported by the execution.

Applying these tools to scatter environments, the most important benefits are: a better coordination of the different tasks and resources involved in the software process (even reaching the automation of some of the functionalities of this coordination) and an improvement of the coordination and collaboration between the development members, because they know the interactions between each task, and between tasks and resources better.

However, the work done in both tools is not finished yet; we still have to add new functionalities to cover more characteristics of distributed environments. Some of the planned improvements are:

1. Even if at the moment we can follow the traceability of each requirement of the product built, we have to include some methods that allow to us follow this traceability in a graphical form, facilitating the tracing of each element of the project and their changes.
2. Develop a plug-in to do transformations from software process modeled with EPF in UML Activity Diagrams to BPMN and BPEL. So

the software processes modeled with EPF will not require remodeling effort.

3. Since the BPMS is able to coordinate and monitor all users, it can also be a particularly useful way to improve communication between them. Therefore, one of the goals for the future is to adapt the BPMS to improve methods of communication between users.

6. Acknowledgments

This work has been financed by the projects PRI 2PR04B011 and CICyT TIN2005-09405-C02-02

7. References

- [1] C. Larman. *Agile & Iterative development: a manager's Guide*. Adison Wesley, December 2005. ISBN 0-13-111155-8
- [2] Agile Manifesto, <http://agilemanifesto.org/>
- [3] S. W. Ambler and K. Nizami. "Agile Strategies for Geographically Distributed Quality Management". December 2007. <http://www.agilejournal.com>
- [4] S. W. Ambler. "IT Projects Success Rates Survey". August 2007. <http://www.ambyssoft.com/surveys/success2007.html>
- [5] G. Booch and A. Brown. "Collaborative Development Environments". *Advances in Computers* Vol. 59, Academic Press, August 2003.
- [6] V. Sinha, B. Sengupta, S. Gosal. "An adaptative tool integration framework to enable coordination in distributed software development". *Proceeding of the International Conference on Global Software Engineering*. pp. 151-155. August 2007.
- [7] SPEM: Software Process Engineering Metamodel 2.0 Beta 2. <http://doc.omg.org/ptc/2007-11-01>
- [8] Eclipse Process Framework (EPF). <http://www.eclipse.org/epf/>
- [9] D. Riesco, D. Romero. "Un Workflow que Automatice los Procesos de Negocios del Proceso Unificado Rational". VI Workshop de Investigadores en Ciencias de la Computación. May 2004
- [10] D. Romero, M. Uva. "De los procesos de desarrollo a la definición de procesos de un workflow". Argentine Symposium on Information Systems. August 2005.
- [11] R. Bendraou, B. Combemale, X. Crégut, M.P. Gervais. "Definition of an Executable SPEM 2.0". *Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC'07) – Vol. 00*. pp 390-397. 2007
- [12] Jazz Team Concert. Beta 2. <http://jazz.net/pub/index.jsp>
- [13] J.J. Berrocal, J.M. García, J.M. Murillo. "Hacia una gestión del proceso software dirigida por procesos de negocio". I Taller sobre Procesos de Negocio e Ingeniería del Software. September 2007. ISSN 1988-3455
- [14] B. Sengupta, S. Chandra, V. Sinha. "A research agenda for distributed software development". ICSE, Shanghai, 2006, pp 731-740.
- [15] Best Practices. <http://www.w3.org/2001/sw/BestPractices/SE/>
- [16] F. Maurer, B. Dellen, F. Bendeck et al. "Merging Project planning and Web enabled dynamic workflow technologies". *Internet Computing, IEEE, Volume 4, Issue 3, May-June 2000*, pp 65-74
- [17] Parvathanathan, K., Chakrabarti, A., Patil, P.P., Sen, S., Sharma, N., and Johng, Y.B. (2007). *Global Development and Delivery in Practice: Experiences of the IBM Rational India Lab*
- [18] M.Cataldo, M. Bass, J.D. Herbsleb, L.Bass. "On coordination mechanisms in Global Software Development". *Proceeding of the International Conference on Global Software Engineering*. pp. 151-155. August 2007.
- [19] UML. <http://www.uml.org/>
- [20] Eclipse. <http://www.eclipse.org/>
- [21] BPMN. <http://www.bpmn.org/>
- [22] BPEL. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- [23] Apache ODE. <http://ode.apache.org/>
- [24] Intalio Tempo. <http://tempo.intalio.org/>
- [25] Zentipede. <http://www.zentipede.org>

TESNA: A Tool for Detecting Coordination Problems

Chintan Amrit, Jos van Hillegersberg
Department of IS&CM
University of Twente
{c.amrit, j.vanHillegersberg}@utwente.nl

Abstract

Detecting problems in coordination can prove to be very difficult. This is especially true in large globally distributed environments where the Software Development can quickly go out of the Project Manager's control. In this paper we outline a methodology to analyse the socio-technical coordination structures. We also show how this can be made easier with the help of a tool called TESNA that we have developed.

1. Introduction

Coordination in large scale software development is a very difficult [1, 2]. The coordination problem is further multiplied in a globally distributed scenario [3].

Some of these coordination problems can be solved by following best practices. An example of such a best practice is what has come to be known as Conway's Law [4]. Conway [5] states, organizations which design systems are constrained to produce designs which are copies of the communication structure of these organizations. Since Conway, researchers have invented various more detailed patterns which describe the preferred relationships between team communication structure (the social network) and technical software architecture [6]. We call such patterns Socio-Technical Patterns [7]. In addition to the classic law by Conway, various Socio-Technical patterns, including those from Coplien et al. [6] have been documented. However, these patterns have not been extensively validated empirically and can be hard to implement. The lack of empirical validation makes it complex for the project manager to decide on which Socio-Technical patterns to apply to his project. We provide the possibility to detect Socio-Technical

Coordination problems (that we call STSCs) and also validate such Socio-Technical Patterns with the help of the TESNA tool and method we have developed. In this paper we show how the method and tool of TESNA works in more detail.

2. TESNA Method

As defined in [7] an STSC or Socio-Technical Structure Clash occurs if and when a Socio-Technical Pattern exists that applies to the actual social network of the software development team and the technical dependencies within the software architecture under development. STSCs are indicative of socio-technical coordination problems in a software development organization. Some of these problems (or STSCs) concerning development activities have been collected and described by Coplien et al. [6] including a set of what they call Process Patterns to deal with some of these coordination problems. As the term process patterns is also used in business process management and workflow, we prefer to use the term Socio-Technical Patterns to refer to those patterns involving problems related to both the social and technical aspects of the software process.

We claim that continuous and early detection of STSCs can help project managers in monitoring the software development process and enable the managers to take actions whenever a STSC occurs.

Figure 1 represents the overview of the method behind TESNA. Our motivation is that when implementation and monitoring of patterns is enhanced, empirical validation of patterns will also become feasible. We provide a Method and Tool called TESNA that can improve the system development by regularly monitoring the software development project and alerting in case of a STSC.

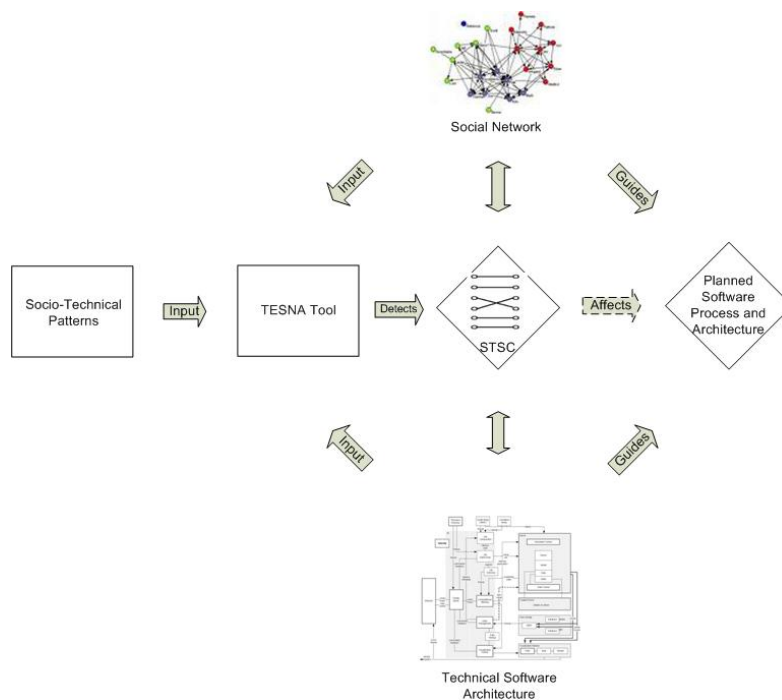


Figure1. Socio-Technical Structure Clashes and the planned Software Process

The Project Manager (who is implicitly present in Figure 1) decides which STSC to look for in Technical Software and Social Network diagrams that is shown with the help of TESNA. TESNA makes it easy for the Manager to detect STSCs and he/she can then decide if the problem is severe enough to warrant a change in the Planned Software Process and Architecture. For this tool and method to work, we need a data structure for storing the Technical dependencies as well as the social network connections. We accomplish this by using what are known as Dependency Structure Matrices.

3. Dependency Structure Matrix Overview

Since the concept of the Design Structure Matrix was first proposed by Steward [8, 9], Dependency Structure Matrices have been used in engineering literature to represent the dependency between tasks. A DSM highlights the inherent structure of a design by examining the dependencies between its component elements in a square matrix [8, 10].

Morelli and Eppinger (1995) describe a way comparing the predicted and actual communication in an organization [11]. Sosa Eppinger et.al.(2002)[12] describe factors that influence the frequency of communication and choice of media in a Sosa,

geographically distributed development organization. In a different study Eppinger and Rowles (2003) [13] compare the DSM formed through the interaction of system components with the DSM of the technical interactions among team members. Sosa, Eppinger and Rowles (2004) [14], highlight the factors that impact the misalignment of the product and the organizational structures.

Figure 1 shows an example of a simple DSM. The letters A-E, on both axis of the matrix, represent tasks. An 'x' in location (a,b) of the matrix means that the task of row a depends on the task in column b. Dependencies below the gray diagonal represent 'feed forward information', while tasks above the diagonal represent feedback, for example, task E gives feedback on task C. In this example, tasks A and B depend on each other.

	A	B	C	D	E
A		x			
B	x				
C		x			x
D	x	x			
E	x			x	

Figure 2. Example of a DSM

MacCormack et.al. (2006) [15] compare the DSMs of a commercial and a pure open source project and show how the structure of the code in the projects reflects the organizational structure that created it, much like what Conway said in his paper[5]. More recently Li et al. [16] use dependency matrices to analyze dependencies between components in a CBS. While Cataldo et.al. [17] show how DSMs can be used to predict coordination in a software development organization and then they compare the predicted coordination DSM with the actual communication DSM.

4. TESNA Tool Functionality

The tool TESNA consists of three modules namely the Social Structure, Technical Structure and the Socio-Technical Structure analysis modules. The tool uses the DSM data structure to analyse each of the different structures and

4.1 Social Structure Analysis

To analyse the *Social Structures* TESNA can construct and analyse metrics from logs of chat messages (from a chat server like Jabber). Moreover, TESNA displays the different metrics of the social network over a period of time. We have used this option to detect the Betweenness centrality match pattern [7] by calculating the betweenness centrality of the social networks over the period under study.

TESNA can mine bug tracking websites (like Mantis) to gather data on the social thread of responses for each bug posted. We have used this feature on a corporate case (eMaxx discussed below) that we are currently studying.

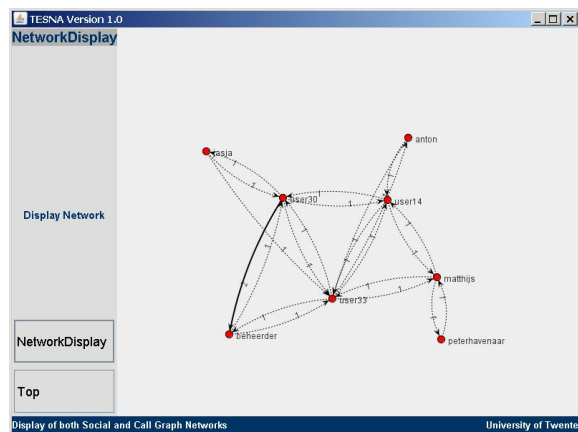


Figure 3 Social Network from the eMaxx case

In order to display the social network got through mining these repositories TESNA uses libraries from the Java Universal Network/Graph Framework (JUNG)[18], an open source library widely used by Network researchers. The display of the social network from the eMaxx case is shown in Fig 3. Here, each of the nodes represents a member of the social network (whose name is indicated by the label next to the node) and the thickness of the line between the nodes represents the number of messages exchanged between the people represented by the nodes. The more the number of messages are the thicker the line gets.

4.2 Technical Structure Analysis

To analyse the *Technical Structures* TESNA can read the source code file and construct the call graph of the software. At present, the tool supports reading java code files (jar files) to determine the technical dependencies between the different components or modules of the software. TESNA again uses JUNG[18] to display the call graph of the software as shown in Figure 4.

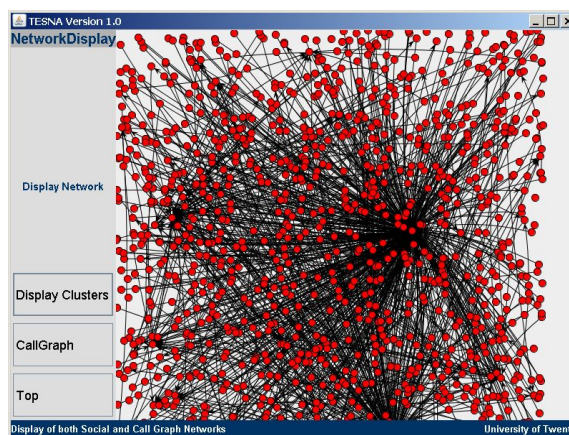


Figure 4: Call Graph of Jython

Figure 4 represents the Call Graph or the dependency graph of an open source project called Jython. Each red node represents one java class object of Jython. As this Call Graph is already quite complex, we don't display the names of the class objects and instead use the tool tip if the user hovers above interesting areas of the Call Graph. We will show later how we reduce this complexity further by clustering the Call Graph.

4.3 Socio-Technical Structure Analysis

TESNA can mine version control systems like CVS and SVN and find out the *Socio-Technical Dependencies* (the people working on the different parts of the software). TESNA uses JUNG to display the developer code network as shown in Figure 5.

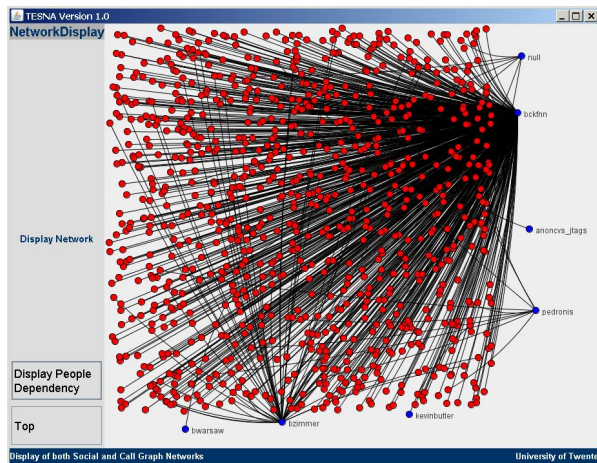


Figure 5: The developer code Socio-Technical Network of Jython

The red nodes in Figure 5 represent the software class objects that the developers, represented by the blue nodes have last modified. The names of the developers are displayed by the labels next the nodes. This graphic representation uses the normal bipartite graph functionality of JUNG. So, the links between the class objects are not displayed. Such a complex graph can provide us with limited information, for example, which developer modified how many files. Using the tool tips of the red class objects one can find out the names of the class objects and in-turn find out which developer modified which class object. As we will show later TESNA reduces the complexity of this graph by clustering the class objects and then displaying the developers working on the different clusters. We further combine the network of dependencies of the class objects with the network of the developers working on the different class objects (as described in the Chapter on the DSM approach). We thus come up with the graph of the developer dependencies. Figure 6 shows the developer dependencies of the Jython project. The red nodes represent the developers working on the different modules of Jython and the directed links represent the dependency, for example bedronis is dependent on bckfnn and vice versa

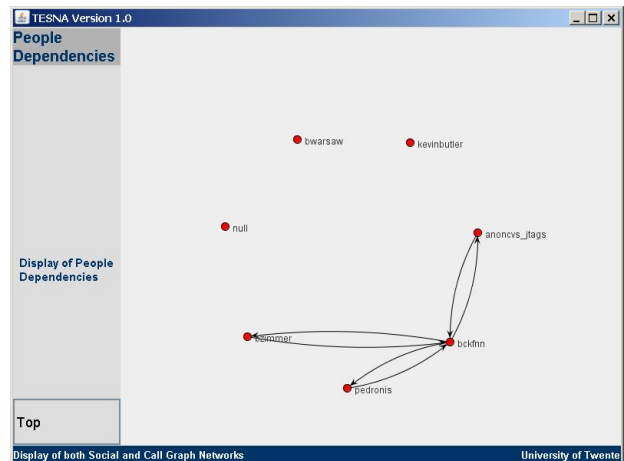


Figure 6: The developer dependency graph of Jython

TESNA displays the people dependencies (Figure 6) based on whether people are working on the same or dependent modules [17] and can compare this with the social network of the developers in order to detect Conway's Law STSC [7].

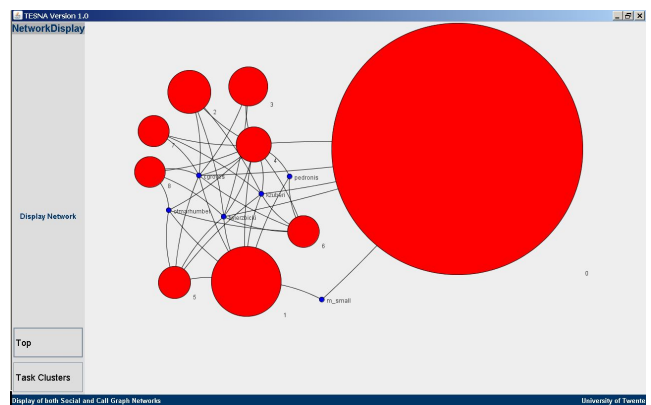


Figure 7: The Clustered Socio-Technical Network of Jython

5. The TESNA Visualization

Large graphs can cause problems of usability and discern-ability. Though, large graphs can give an indication of the overall structure or that of some location within it, in general the display of large graphs makes them difficult to comprehend. It follows that it is easier to comprehend and perform a detailed analysis of graph structures when the size of the graph is small [19]. In response to the need to make the graphs especially Figures 4 and 5 more understandable we cluster the graphs. For clustering

we use the algorithm by Fernandez[20] and used by MacCormack et al. [15]. We cluster the graphs into a fixed number of 9 clusters according to the golden 7 plus or minus 2 rule for human comprehension [21]. The clustering is done with the help of a DSM Clustering tool that we have developed [22]. Figure 7 is a representation of the Jython call graph clustered into 9 clusters along with the developers who have modified classes in different clusters. We also show the dependencies that exist among the clusters. Such a clustered display can enable a more easy detection of different STSCs [23].

6. Discussion

The visualizations created by TESNA help the Project Manager in identifying STSCs. Once the STSCs are identified the Project Manager can decide whether the current development process needs to be changed. We have tested this methodology in multiple case studies. Among the case studies, we have conducted two case studies in a corporate environment namely Mendix [7] and eMaxx (forthcoming publication). We have also conducted multiple case studies studying Open Source STSCs [23] and have got a few interesting insights into how STSCs in the open source environment differs from Corporate environment.

7. Other TOOLS

There are a few tools available to display the social network as well as the social call graph. Augur is a visualization tool that supports distributed software development process by creating visual representations of both the software artefacts and the software development activities [24]. de Souza et al. [25] are developing a tool called Ariadne that checks dependency relationships between software call graphs and developers. Also there is a tool under development for forecasting dependencies between developers in a Dynamic/iterative environment [17]. The limitation of these tools is that they check for only one particular STSC, namely the Conway's Law STSC.

REFERENCES

1. Kraut, R., E. and L. Streeter, A. , *Coordination in software development*, in *Commun. ACM*. 1995: New York, NY, USA. p. 69--81.
2. Curtis, B., H. Krasner, and N. Iscoe, *A Field-Study of the Software-Design Process for Large Systems*. *Communications of the Acm*, 1988. **31**(11): p. 1268-1287.
3. Herbsleb, J.D. and D. Moitra, *Global software development*. *Software, IEEE*, 2001. **18**(2): p. 16-20.
4. Herbsleb, J., D. and R. Grinter, E. , *Splitting the organization and integrating the code: Conway's law revisited*, in *ICSE '99: Proceedings of the 21st international conference on Software engineering*. 1999: Los Alamitos, CA, USA. p. 85--95.
5. Conway, M., *How do Committees Invent*, in *Datamation*. 1968. p. 28-31.
6. Coplien, J., O. and N. Harrison, B. , *Organizational Patterns of Agile Software Development*. 2004: Upper Saddle River, NJ, USA.
7. Amrit, C. and J. van Hillegersberg, *Detecting Coordination Problems in Collaborative Software Development Environments*. *Information Systems Management*, 2008. **25**(1): p. 57 - 70.
8. Steward, D., *The design structure system: a method for managing the design of complex systems*. *IEEE Transactions on Engineering Management*, 1981. **28**(3): p. 71-74.
9. Steward, D.V., *Partitioning and tearing systems of equations*. *SIAM J. Numer. Anal*, 1965. **2**(2): p. 345-365.
10. Eppinger, S.D., et al., *A model-based method for organizing tasks in product development*. *Research in Engineering Design*, 1994. **6**(1): p. 1-13.
11. Morelli, M.D., S.D. Eppinger, and R.K. Gulati, *Predicting technical communication in product development organizations*. *Engineering Management, IEEE Transactions on*, 1995. **42**(3): p. 215-222.
12. Sobol, M.G. and U. Apte, *Domestic and global outsourcing practices of America's most effective IS users*. *Journal of Information Technology*, 1995. **10**(4): p. 269-280.
13. Sosa, M.E., S.D. Eppinger, and C.M. Rowles, *Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions*. *Journal of Mechanical Design*, 2003. **125**: p. 240.
14. Sosa, M.E., S.D. Eppinger, and C.M. Rowles, *The Misalignment of Product Architecture*

- and Organizational Structure in Complex Product Development.* J Manage. Sci., 2004. **50**(12): p. 1674-1689.
15. MacCormack, A., J. Rusnak, and C.Y. Baldwin, *Exploring the structure of complex software designs: An empirical study of open source and proprietary code.* Management Science, 2006. **52**(7): p. 1015-1030.
 16. Li, B., et al., *Matrix-based component dependence representation and its applications in software quality assurance,* in *ACM SIGPLAN Notices.* 2005: New York, NY, USA. p. 29--36.
 17. Cataldo, M., et al., *Identification of coordination requirements: implications for the Design of collaboration and awareness tools,* in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work.* 2006, ACM Press: Banff, Alberta, Canada.
 18. Madadhain, J.O., et al., *The JUNG (Java Universal Network/Graph) Framework,* in *Technical Report UCI-ICS 03-17 2005,* University of California, Irvine.
 19. Herman, I., et al., *Graph visualization and navigation in information visualization: A survey*
Graph visualization and navigation in information visualization: A survey. Transactions on Visualization and Computer Graphics, 2000. **6**(1): p. 24-43.
 20. Fernandez, C.I.G., *Integration Analysis of Product Architecture to Support Effective Team Co-location.* ME thesis, MIT, Cambridge, MA, 1998.
 21. Miller, G.A., *THE MAGICAL NUMBER SEVEN, PLUS OR MINUS TWO: SOME LIMITS ON OUR CAPACITY FOR PROCESSING INFORMATION.* Psychological Review, 1956. **63**(2).
 22. Hegeman, J.H. *Towards a Comprehensible Representation of Software Development Tasks.* in *Twente Student Conference.* 2007. University of Twente.
 23. Amrit, C., J.H. Hegeman, and J. van Hillegersberg. *Exploring Coordination Structures in Open Source Software Development.* in *1st Workshop on Tools for Managing Globally Distributed Software Development.* 2007. Munich, ICGSE 2007: Centre for Telematics and Information Technology (CTIT).
 24. Froehlich, J. and P. Dourish, *Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams,* in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering.* 2004: Washington, DC, USA. p. 387--396.
 25. de Souza, C., R. B., et al., *Sometimes you need to see through walls: a field study of application programming interfaces,* in *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work.* 2004: New York, NY, USA. p. 63--71.