Lecture Notes in Computer Science 4470

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Qing Wang Dietmar Pfahl
David M. Raffo (Eds.)

Software Process Dynamics and Agility

International Conference on Software Process, ICSP 2007
Minneapolis, MN, USA, May 19-20, 2007
Proceedings

Springer

Volume Editors

Qing Wang
Chinese Academy of Science, Institute of Software
No. 4 South Fourth Street, Zhong Guan Cun, Beijing 100080,China
E-mail: wq@itechs.iscas.ac.cn

Dietmar Pfahl
University of Calgary, Schulich School of Engineering
Department of Computer Science & Electrical Engineering
2500 University Drive N.W., Calgary, Alberta T2N 1N4, Canada
E-mail: dpfahl@ucalgary.ca

David M. Raffo
Portland State University, School of Business Administration
P.O. Box 8491, Portland, OR 97207, USA
E-mail: raffod@pdx.edu

Library of Congress Control Number: 2007925873

CR Subject Classification (1998): D.2, K.6.3, K.6, K.4.3, J.1

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-72425-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-72425-4 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12062041 06/3180 5 4 3 2 1 0

Preface

This volume contains papers presented at the International Conference on Software Process (ICSP 2007) held in Minneapolis, USA, May 19-20, 2007. ICSP 2007 comprised two successful series of process-related workshops, the International Workshop on Software Process Simulation and Modeling (ProSim) and the Software Process Workshop (SPW).

The theme of ICSP 2007 was "Coping with Software Process Dynamics and Agility." Software developers work in a dynamic context of frequently changing technologies and limited resources. Globally distributed development teams are under ever-increasing pressure to deliver their products more quickly and with higher levels of quality. At the same time, global competition is forcing software development organizations to cut costs by rationalizing processes, outsourcing part or all of their activities, reusing existing software in new or modified applications and evolving existing systems to meet new needs, while still minimizing the risk of projects failing to deliver. To address these difficulties, new or modified processes are emerging, including agile methods and plan-based product line development. Open source, COTS and community-developed software are becoming more popular. Outsourcing coupled with 24/7 development demand well-defined processes to support the coordination of organizationally and geographically separated teams.

The increasing challenges faced by the software industry combine to increase demands on software processes.

ICSP 2007 was a continuation of two successful series of process-related workshops, ProSim (Software Process Simulation and Modeling Workshop) and SPW (Software Process Workshop). SPW and ProSim were conducted jointly for the first time in 2006 as a co-located event to ICSE 2006. ICSP 2007 continued a long tradition of software process research, positioning itself as the new leading-edge event for systems and software process research.

In response to the call for papers, 98 submissions were received from 14 different countries and regions: Australia, Brazil, Canada, China, France, Germany, Japan, Korea, The Netherlands, Pakistan, Spain, UK, USA, and Turkey. Every paper was rigorously reviewed and held to very high quality standards, and finally 28 papers were accepted as regular papers for presentation at the conference.

The papers were clustered around topics and presented in five regular sessions, each consisting of two threads. Topics included *Process Content, Process Tools and Metrics, Process Management, Process Representation, Analysis and Modeling, Experience Report,* and *Simulation Modeling.*

Highlights of the ICSP2007 program were two keynote speeches, delivered by Larry E. Druffel (President and CEO, SCRA, USA) and Merwan Mehta (Department of Technology Systems, East Carolina University, USA).

A conference such as this can only succeed as a team effort. All of this work would not have been possible without the dedication and professional work of many colleagues. We wish to express our gratitude to all contributors for submitting papers. Their work formed the basis for the success of the conference. We would also like to

thank the Program Committee members and reviewers because their work guaranteed the high quality of the workshop. Particular thanks also go to the keynote speakers for giving their excellent presentations at the conference. Finally, we would also like to thank the members of the Steering Committee, Barry Boehm, Mingshu Li, Leon Osterweil and Wilhelm Schäfer, for their advice, encouragement and support.

We wish to express our thanks to the organizers for their hard work. The conference was sponsored by the International Software Process Association (ISPA) and the Institute of Software, the Chinese Academy of Sciences (ISCAS) and the ISCAS Laboratory for Internet Software Technologies. We also wish to thank the 29th International Conference on Software Engineering (ICSE 2007) for sponsoring this meeting as an ICSE Co-located Event. Finally, we acknowledge the editorial support from Springer for the publication of this volume.

For further information, please visit our Web site at http://www.icsp-conferences.org/icsp2007.

March 2007

David M. Raffo
Qing Wang
Dietmar Pfahl

International Conference on Software Process 2007

Minneapolis, USA
May 19–20, 2007

General Chair

David M. Raffo, Portland State University, USA

Steering Committee

Barry Boehm, University of Southern California, USA
Mingshu Li, Institute of Software, Chinese Academy of Sciences, China
Leon J. Osterweil, University of Massachusetts, USA
Wihelm Schäfer, University of Paderborn, Germany

Program Co-chairs

Dietmar Pfahl, University of Calgary, Canada
Qing Wang, Institute of Software, Chinese Academy of Sciences, China

Program Committee Members

Stefan Biffl	Technische Universität Wien, Austria
Thomas Birkhölzer	University of Applied Science, Konstanz, Germany
Keith Chan	Hong Kong Polytechnic University, Hong Kong, China
Sorana Cimpan	University of Savoie at Annecy, France
Jacky Estublier	French National Research Center in Grenoble, France
Anthony Finkelstein	University College London, UK
Dennis Goldenson	Carnegie Mellon University, USA
Volker Gruhn	University of Leipzig, Germany
Paul Grünbacher	Johannes Kepler University Linz, Austria
Dan Houston	Honeywell, USA
LiGuo Huang	University of Southern California, USA
Hajimu Iida	Nara Institute of Science and Technology, Japan
Katsuro Inoue	Osaka University, Japan
Ross Jeffery	University of New South Wales, Australia
Natalia Juristo	Universidad Politécnica de Madrid, Spain
Rick Kazman	University of Hawaii, USA
Jyrki Kontio	Helsinki University of Technology, Finland

Jian Lv	Nanjing University, China
Ray Madachy	University of Southern California, USA
Frank Maurer	University of Calgary, Canada
Hong Mei	Peking University, China
Jürgen Münch	University of Kaiserslautern, Germany
Flavio Oquendo	University of South Brittany, France
Dewayne E. Perry	University of Texas at Austin, USA
Dietmar Pfahl	University of Calgary, Canada
Dan Port	University of Hawaii, USA
Antony Powell	Science Applications International Corporation, USA
David M. Raffo	Portland State University, USA
Juan F. Ramil	The Open University, UK
Andreas Rausch	Technische Universität Kaiserslautern, Germany
Günther Ruhe	University of Calgary, Canada
Mercedes Ruiz	University of Cádiz, Spain
Ioana Rus	Fraunhofer Center, USA
Kevin Ryan	University of Limerick, Ireland
Walt Scacchi	University of California, Irvine, USA
Barbara Staudt Lerner	Mt. Holyoke College, USA
Stan Sutton	IBM T. J. Watson Research Center, USA
Colin Tully	Middlesex University, UK
Qing Wang	Chinese Academy of Sciences, China
Yongji Wang	Chinese Academy of Sciences, China
Brian Warboys	University of Manchester, UK
Paul Wernick	University of Hertfordshire, UK
Laurie Williams	North Carolina State University, USA
Ye Yang	University of Southern California, USA
Yun Yang	Swinburne University of Technology, Australia

External Reviewers

Ahmed Al-Emran	University of Calgary, Canada
Marta Lopez	Universidad Complutense de Madrid, Spain
Alicia Mon	Universidad Nacional de la Matanza, Argentina
Ricardo Imbert	Universidad Politecnica de Madrid, Spain
Anna Cecilia Griman	Universidad Simón Bolivar, Venezuela
Oscar Dieste	Universidad Politecnica de Madrid, Spain
Carmen Zannier	University of Calgary, Canada

Table of Contents

Process Content

Process Tools and Metrics

Process Management

Process Representation, Analysis and Modeling

Experience Report

Simulation Modeling

Extending Microsoft Team Foundation Server Architecture to Support Collaborative Product Patterns

Fuensanta Medina-Domínguez, Maria-Isabel Sanchez-Segura, Antonio Amescua, and Javier García

Computer Science Department, Carlos III Technical University of Madrid
Avda. Universidad, 30, Leganes 28911, Madrid, Spain
{fmedina, misanche, amescua, jgarciag}@inf.uc3m.es

Abstract. This paper provides a practical solution, based on process reuse and knowledge management techniques, to make software engineering theories more accessible, easier, and cheaper for software development organizations to implement. It shows how the PIBOK-PB architecture (Process improvement based on knowledge-pattern based) and the extensions of a commercial product, Microsoft solution Visual Studio Team System, are used to achieve this.

Keywords: Software Engineering, Process Management, Reuse, Patterns.

1 Introduction

Software engineering provides enough formalisms to guarantee the execution of a software project. However, if we look at the data on software projects, we will observe that, in 1995, on average, only around 20% of software projects were completed on time and within the budget [1]. What happened to the remaining 80%? These percentages have changed little since, and many projects still fail to comply with the triple constraints of scope, time and cost [2]. Poor project management and insufficient use of software engineering techniques are some of the reasons for this non-compliance.

This data can help us to understand that although the theories in the software engineering field are sufficiently matured and are widely known, it is the implementation of software engineering best practices that helps organizations improve their productivity, software quality, and reduce costs [3]. But it is very difficult to implement them because these organizations must first know the theory, and how to implement them successfully [4] [5] [6], which is not gathered in the literature. Evidence also reveals that the implementation of software processes in software development organizations is a complex and expensive process, mainly for small and medium enterprises [7]. We have focused on bridging the gap between theory and practice in software development processes and best practices to make them accessible, and thus beneficial, for small and medium enterprises.

The experience of the American Federal Aviation Administration (FAA) indicates that knowledge management combines positively with process improvement, benefits the organization and the process improvement programmes [8]. Some papers, like the one published in [9], suggest that technology, structure, culture, knowledge process

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 1–11, 2007.

architecture of acquisition, conversion, application, and protection are essential organizational capabilities or "preconditions" for effective knowledge management.

We believe that knowledge management can be applied to software engineering to transform software engineering data and information, described as process models, standards, methodologies, etc., to knowledge and innovation. This is possible once the knowledge of experts on process model, standards, methodologies, etc., is elicited and translated into a computable model. Therefore, a software system can make use of this knowledge in order to reduce the cost of process definitions and hasten the maturity of the processes.

Data and information must be encapsulated to allow their subsequent recovery and reuse, and their evolution into knowledge. The artefact to encapsulate this knowledge is the pattern concept. Patterns are an established and well-known format to capture engineering knowledge [10], but the real power of patterns to enable knowledge transfer for practical use is still under development due to the limited number of approaches that endow patterns with practical implementation. The authors proposed the *product pattern* concept as well as a model called PIBOK-PB to support process improvement based on patterns [11]. In this paper, we describe the architecture that supports the PIBOK-PB model as well as the Microsoft VSTS extensions to provide a collaborative practical solution based on process reuse and knowledge management techniques.

We have no evidence of research groups working on combining software engineering, patterns, and knowledge management techniques supported by collaborative working environments. The US Federal Aviation Administration (FAA) and the SINTEF group (Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology) are currently working on process improvement based on knowledge management, but their solutions are not supported by collaborative working environments.

The remainder of this paper is structured as follows: section 2 describes PIBOK-PB model architecture; section 3 explains the extensions done to Microsoft VSTS to satisfy the PIBOK-PB architecture requirements; section 4 summarises related work and finally section 5 presents the conclusions and future trends.

2 PIBOK-PB Architecture Description

The PIBOK-PB model (Process Improvement Based On Knowledge – Pattern Based Model) [11] is a knowledge-based software process improvement model that would allow the use of *product patterns* as the artefact to encapsulate the knowledge for use in the development of activities and tasks of the process model chosen.

PIBOK-PB model [11] architecture is made up of four layers, see Fig. 1. These layers are described below:

- Organization layer: this layer processes the data and characteristics of the organization and their business processes. It is responsible for obtaining the data of the organization, for example, the kind and size of the organization, field of their business processes. In summary, this layer is responsible for gathering data, provided by the project manager, related to the forces of the organization.

This layer supports the appropriate selection of process models, methodologies, etc, for each organization and project, depending on their features.

- Project instantiation layer: this layer is responsible for obtaining information related to the project to be developed, for example, the kind of the project, the paradigm chosen to implement the project, the number of employees and their roles in this specific project. These data establish the context and forces of the project to be developed.

  This layer allows the instantiation of process models, methodologies, etc., in a specific project, so it can be customized.
- Patterns instantiation layer: this layer is responsible for the instantiation of the product patterns that best fit the activities included in the process model template selected. The product pattern is an artefact which contains the expert‘s knowledge to obtain a specific software product. The product pattern concept is described in more detail later.

  This layer allows the recovery of knowledge to be reused in the specific project under development.
- Collaborative layer: this layer provides a collaborative platform which contributes to the functionalities and advantages of collaborative environments. It also provides collaborative functionalities when they are demanded by the roles involved in the execution of a product pattern or when collaboration among patterns is required.

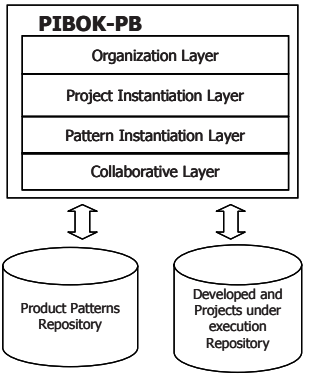


Fig. 1. PIBOK-PB model architecture

The four layers interact with two repositories:

- A repository of product patterns.
- A repository with the information of previously developed projects and projects in progress.

The interactions among layers are described below:

- The Organization and Project instantiation layers provide the context and forces of the organization and project the patterns instantiation layer. Both

layers store these data in the repository that contains the information on developed and current projects. In this way, the patterns instantiation layer will have enough information to execute the next rule, and find the potential product patterns to be applied in the activity and/or tasks. Only the problem field is necessary and this is obtained from the task or activity to be developed in the process model. When all the data are ready, the patterns instantiation layer can execute a rule similar to the following:

If you find yourself in this **context**
(and) with this **problem**
(and) entailing these **forces**
then
map a product pattern in your project
(and) look for more product patterns

- The Collaborative layer interacts with the patterns instantiation layer to provide the collaborative functionalities in order to develop the tasks and/or activities through product patterns. The collaborative layer also provides the patterns instantiation layer with the existing collaboration among product patterns.

In order to understand the above rule, the product pattern concept and its fields are described in detail. Product patterns come from the Alexandrian patterns [12] and are artefacts that gather the knowledge of software engineering experts to obtain a specific software product. Product is defined as anything produced during the whole software development process (for example, Effort and Duration Estimation Using COCOMO II. Available at: http://sel.inf.uc3m.es/C3/Journals/ProductPattern/ProductPattern.pdf. This concept was presented in [11]. The fields of product pattern have since been extended as follows:

- **Name:** the name can be a word or short phrase related to the product pattern.
- **Related Pattern:** static or dynamic relationships between this pattern and others.
- **Initial Context:** the pre-conditions under which the pattern is applicable - a description of the initial state before the pattern is applied.
- **Result Context:** state or configuration of the system after the pattern has been applied. Describing the positive and negative effects.
- **Problem:** description of the problem to be solved.
- **Forces:** restrictions that are classified as follows:
    - Kind of organization
    - Kind of system to be developed
    - Kind of client
    - Rationale
- **Solution:** static relationships and dynamic rules to describe how to achieve the objective. The solution consists of:
    - Process
    - Development time
    - Activities diagram
    - Adjustment degree (very low, low, normal, high very high)

- **Roles:** people or participants involved.
- **Entries:** previously obtained products necessary to develop this pattern:
    - Name (text)
    - Kind of information (.doc, .xml…)
    - Is software configuration management going to be applied? Yes/No
- **Lessons Learned:** documented experiences gathered while using this pattern:
    - Name (text)
    - Kind of information (.doc, .xml…)
    - Hyperlink
- **Templates:** templates that can be used to obtain the exit of this pattern:
    - Name (text)
    - Kind of information (.doc, .xml…)
- **Examples:** one or more sample applications of the pattern
- **Exit:** product obtained when the pattern is used:
    - Name (text)
    - Kind of information (.doc, .xml…)
    - Is software configuration management going to be applied? Yes/No
- **Collaboration:** collaboration among product patterns or among roles during the development of a product pattern.
- **Capability Level:** maturity level. This is useful when the product pattern represents a product included in some software improvement approach:
    - Name (text)
    - Level (text)
- **Information Resources:** references and documentation (for example, books, papers) used to develop this product.

3 Extending Microsoft VSTS to Support PIBOK-PB Architecture

In this section we explain the Microsoft VSTS extensions to satisfy the PIBOK-PB architecture requirements.

The Visual Studio 2005 Team System (VSTS) is Microsoft's proposal to maximize the information technology work teams. VSTS provides a set of extensible, productive and integrated tools, which facilitates communication and collaboration among teams and individuals in a software organization during the project execution.

VSTS is based on Visual Studio 2005 Professional and is made up of Visual Studio Team Edition and Visual Studio Team Foundation Server (TFS). Visual Studio 2005 Team Edition is the client and comprises Visual Studio Team Architect Edition, Visual Studio Team Developer Edition, and Visual Studio Team Test Edition. Visual Studio Team Foundation (TFS) corresponds to the server side of the application and offers a set of collaborative capabilities that allow the project manager to coordinate appropriately the project tasks. (VSTS architecture is shown in Fig. 2.)

VSTS architecture offers a development environment supported by collaborative services. There are other IDEs (integrated development environments) that offer similar features, but the authors of this paper selected VSTS because of its philosophy on the use of development methodologies (by default**;** MSF for CMMI, MSF Agile). This feature emphasizes the reuse of software process models and some of the

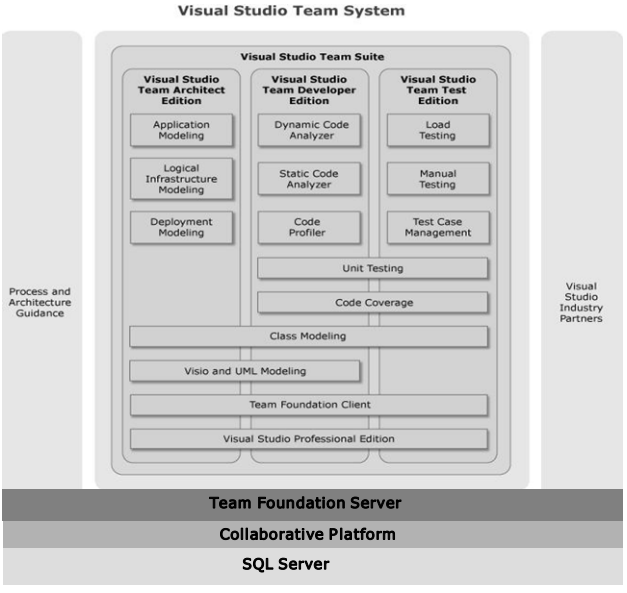


Fig. 2. Visual Studio Team System architecture (source [13])

techniques the proposed process models recommended through the integrated development environment.

VSTS philosophy will allow the extensions required, which will be part of what we call PIBOK-PB tool, to satisfy PIBOK-PB architecture requirements. In order to do so, we had to implement the organizational projects instantiation and pattern instantiation layers which are not supported by the VSTS. These layers are shaded in Fig. 3.

Next, we enumerate the main requirements of the PIBOK-PB architecture, identifying how VSTS supports each requirement and the extension developed. These extensions were developed in Visual Studio 2005 platform and the programming language was C#.

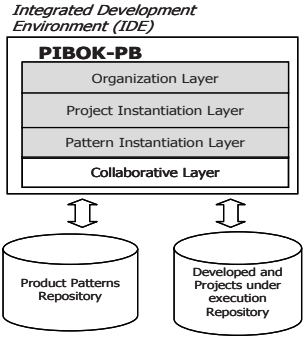


Fig. 3. Shaded layers added to the VSTS Architecture

PIBOK-PB architecture requirements

1. *to select the process model that best suits each project under development according to the organization's features.*
    a. *VSTS support:* the tool provides a wizard that allows the selection of just two process models templates (Agile, and CMMI).
    b. *VSTS extension (completed):* we developed a wizard that answers the questions related to the organization that is going to develop the project, and the project itself. The system then selects the process model template that best fits the project under development. Therefore, templates are selected according to criteria instead of randomly as VSTS currently does.

2. *to generate an activities tree indicating the precedence among activities once the process model has been selected, so that the project manager can decide whether to delete and/or include new activities to be developed.*
    - *VSTS support:* the tool provides a static view of the process model template structure selected; no operation is allowed in this view.
    - *VSTS extension (completed): w*e extended the tool to provide a structure that represents the process model template showing the precedence among activities. With this tree, these activities can be executed. Insertion, modification and deletion of activities are allowed.

3. *to choose the product patterns that best fit the project activities for each process model selected.*
    - *VSTS support:* no information is provided to execute the activities proposed in the selected process model template to date.
    - *VSTS extension (completed):* for each activity included in the activities tree, the system provides the project manager with the existing product patterns. The completed extension will allow the project manager to select the product pattern that best fits each activity. The tool is also endowed with the capability to do an automatic matching among product patterns and activities. In order to associate product patterns with activities, process model templates and product patterns must be compatible.

4. *to create an Active Electronic Process Guide for each Project, once the process model, and the product patterns to be implemented, are selected. This guide provides the information needed for each process under development. The process is executed collaboratively when required.*
    - *VSTS support:* the tool provides an Electronic process guide that is only the web representation of the information included in the process model template. The information is static; no operation is allowed.

- *VSTS extension (completed): w*e developed an Active electronic process guide which provides all the information the developer needs to execute each activity on the spot. This information is obtained from the product patterns instantiated for each activity.

4 Related Works

Currently, some organizations are customizing and extending VSTS [14]. We have these describe these extensions below, identifying the enterprise involved as well as the specific extensions they are focusing on.

Table 1 summarizes the main extensions under development in the field of test.

Table 1. Extensions under development in the field of test

Enterprise	Extensions under development
AutomatedQA	This enterprise has extended its TestComplete tool with VSTS. The extension provides developers with a complete and well-integrated testing solution.
Compuware	The integration of VSTS with Compuware Testpartner provides development access to the same testing assets as testers, allowing them to resolve errors more quickly, improve communication and collaboration, and improve application quality in a cost-effective way.
Mercury Interactive Corportaion	This enterprise plans to integrate with VSTS by sharing testing assets such as unit tests and functional and load tests in both developments. It will also integrate with regard to collaboration on the diagnosis and resolution of application defects, performance bottlenecks, and scalability problems across the entire application life cycle.

Table 2 summarizes the main extension under development, focusing on version control.

Table 2. Extensions under development in the field of version control

Enterprise	Extensions under development
SourceGear	This enterprise is developing a tool called Allerton to access the Team Foundation Server from outside Visual Studio environments. This tool allows version control and work item tracking features of VSTS from other platforms, including Mac Os or Linux.

Table 3 summarizes the main extensions under development, focusing on requirements process.

Table 3. Extensions under development in the field of requirements process

Enterprise	Extensions under development
Borland	This enterprise has a tool called CaliberRM, which provides requirements management. The extension of this tool would allow interacting with VSTS, linking life-cycle artefacts with requirements throughout the application life cycle, and providing end-to-end, requirements-to-test traceability.
Serena	The integration would allow business users to rapidly visualize their application requirements while collaborating more effectively with IT architects, developers, and testers.

Table 4 summarizes the main extensions under development in the field of process templates.

Table 4. Extensions under development in the field of process templates

Enterprise	Extensions under development
Cochango	This enterprise has Developer, a Scrum methodology template for VSTS.
Osellus	The IRIS tool generates process templates for VSTS as well as templates for Microsoft Project from the tailored processes. The IRIS visual modelling environment can be used to model software development processes, irrespective of the methodology chosen. The resulting process models are fully compliant with VSTS and can be enacted across multiple VSTS

Table 5 summarizes the main extension under development in the field of application maintenance.

Table 5. Extensions under development in the field of test

Enterprise	Extensions under development
AVIcode	The tool called Intercept Studio integrated with VSTS is going to cut down on application maintenance and support costs associated with application maintenance and support by quickly identifying the root cause of operational problems.

There is an approach which focuses on workflow capabilities; you can see a summary in Table 6.

Table 6. Extensions under development in the field of workflow

Enterprise	Extensions under development
Identify	The tool called AppSight with VSTS automates and accelerates the tasks of application problem resolution. It also adds embedded user interfaces and new workflow, for all the members of the application life cycle.

There are many enterprises currently extending VSTS; this shows the importance and possibilities of VSTS extensions. After studying these tools, we can say that proposed solutions focus on specific processes like testing, version control, requirements. Our solution is wider in that it covers the whole lifecycle. The use of patterns in the PIBOK-PB architecture also promotes specific features like reuse, and emphasizes improving collaborative capabilities to maximize productivity.

5 Conclusions and Future Trends

In this paper, we have described the PIBOK-PB architecture and the extensions to Microsoft solution Visual Studio Team System in order to provide a solution based on process reuse and knowledge management techniques. The extensions allow:

- the definition of new process models templates to be selected in order to develop a software project.
- the selection of process models templates according to a set of rules and criteria.
- the execution of process models activities following the order determined by the project manager.
- the recovery of data, information and knowledge from a repository where product patterns and process models are stored.
- the execution of product patterns associated with project activities, reusing existing knowledge, and allowing stakeholders access to the information and knowledge of the product to be obtained on the spot, thus improving the efficiency of use.

We are currently working on the extension of the catalogue of product patterns available and the catalogue of process models templates. The product patterns and process model templates are being stored in a repository XML compatible.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Technology through the TIC2004-7083 project.

References

1. Standish Group, “CHAOS Report” http://www.standishgroup.com/sample_research/chaos_1994_1.php
2. Nienaber, R., Cloete, E. A software agent framework for the support of software project management. (2003). Proceedings of the SAICSIT 2003. Pp 16-23.
3. Capell Peter, PhD. “Benefits of Improvement Efforts”. Special Report CMU/SEI-2004-SR-010. September, 2004.
4. Iversen Jakob, Ngwenyama Ojelanki. Problems in measuring effectiveness in software process improvement: A longitudinal study of organizational change at Danske Data. International Journal of Information Management 26 (2006) 30-43

5. Niazi, Mahmood, Wilson, David, Zowghi Didar. A maturity model for the implementation of software process improvement: an empirical study. The journal of System and Software 74 (2005) 155-172
6. Arent Jesper, Norbjerg Jacob. Software Process Improvement as Organizational Knowledge Creation: A Multiple Case Analysis. (2000).
7. Pinto R. and Shoemaker D. "The Cost of CMM in a Conventional IT Organization: A Field Study", 2002.
8. Burke, D., Howard, W. Knowledge Management and Process Improvement: A Union of Two Disciplines. The journal of defense software engineering. Jun 2005. Available at: http://www.stsc.hill.af.mil/crosstalk/2005/06/0506Burke.html
9. Gold, A., Malhotra, A., and Segars, A. Knowledge Management: An Organizational Capabilities Perspective. Journal of Management Information Systems. (2001). Vol. 18 No. 1, pp. 185 – 214.
10. Haggen, L. Lappe, K. Sharing requirements engineering experience using patterns. IEEE software. January-February 2005. Pp 24-31.
11. Amescua, A., García, J., Sánchez-Segura, M., Medina-Domínguez, F. A pattern-Based Solution to Bridge the gap between theory and practice in Using process models. Lecture Notes in Computer Science. ISSN0302-9743 Vol. 3966 pp. 97-104. Book Software Process Change. 2006
12. Alexander, C. "The Timeless Way of Building". Oxford University Press. 1979.
13. MSDN. http://msdn2.microsoft.com/en-us/teamsystem/default.aspx
14. Hundhausen, R. Working with Microsoft® Visual Studio® 2005 Team System. 2006.

The REMIS Approach for Rationale-Driven Process Model Evolution

Alexis Ocampo and Jürgen Münch

Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1,
67663 Kaiserslautern, Germany
{ocampo, muench}@iese.fraunhofer.de

Abstract. In dynamic and constantly changing business environments, the need to rapidly modify and extend the software process arises as an important issue. Reasons include redistribution of tasks, technology changes, or required adherence to new standards. Changing processes ad-hoc without considering the underlying rationales of the process design can lead to various risks. Therefore, software organizations need suitable mechanisms for storing and visualizing the rationale behind process model design decisions in order to optimally introduce future changes into their processes. This paper presents REMIS (Rationale-driven Evolution and Management Information System), a prototype tool we have developed for providing support to process engineers during the task of collecting the reasons for process changes, introducing the changes, and storing them together in a process model evolution repository. Additionally, we present lessons learned with REMIS during the evolution of a reference process model for developing service-oriented applications.

Keywords: Process evolution, rationale, process management, prototype tool, resource description framework.

1 Introduction

Process models can be used to guide developers, automate and improve processes, support management and execution, and store experience [8]. Changing these models in organizations is typically a complex and expensive task [25]. Process engineers are faced mainly with the following challenges: a) to rapidly update the process model so that the organization can keep up with its business environment; b) to introduce changes that are realizable and acceptable for practitioners; c) to introduce changes that are consistent or do not affect the process model consistency. Achieving a compromise that satisfies such challenges usually depends on the information available for rapidly judging if a change is consistent and can be easily adopted by practitioners. Having information about the rationales of the process design at hand can be of great help to process engineers for overcoming the previously mentioned challenges. Currently, the common situation is that there is a lack of support for systematically evolving process models. Combined with other facts such as budget and time pres-sure, process engineers often take shortcuts and therefore introduce unsuitable or inconsistent changes or go through a long, painful update process.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 12–24, 2007.

It has been shown that systematically describing the relationships between an existing process and its previous version(s) is very helpful for efficient software process model evolution [2]. Such relationships should denote differences between versions due to distinguishable modifications. One can distinguish the purpose of such modifications if one can understand the rationale behind them.

In the product domain, rationale has been defined as the justification for a decision by software product designers, who have done extensive research on capturing, organizing, and analyzing design rationales [9]. By making rationale information explicit, decision elements such as criteria, priorities, and arguments can improve the quality of software development decisions. Additionally, once new functionality is added to a system, the rationale models enable developers to track those decisions that should be revisited and those alternatives that have already been evaluated.

The situation is not much different in the process modeling domain, where this topic seems to be less developed, or not yet considered relevant by software process engineers. We are currently working on transferring rationale concepts into the process modeling domain. We do this based on the assumption that the rationale for process changes can be used for understanding the history of software process changes, for comprehensive learning, and for supporting the systematic evolution of software processes. The research roadmap we are following consists of the following steps: a) identification of a taxonomy of reasons for process change (documented in [26]); b) definition of a structured conceptual model of rationale; c) definition of a method that provides guidance on how to perform systematic process evolution supported by rationale; d) implementation of a prototype; e) validation of the concepts and the approach in process evolution projects. The steps are performed iteratively so that the experience acquired in the evolution projects can be used for fine tuning the concepts and the approach.

The research work described in this article consists of the definition of concepts and the implementation of the REMIS prototype that can be used for collecting information about the rationale underlying process changes and as tool support for systematically evolving a software process model. This is one of very few attempts per-formed so far whose goal is to connect the reasons for changes to the actual history of a process model. Section 2 presents a retrospective of work performed on rationale concepts and methods as well as on tools suitable for collecting the rationale of processes and products. Section 3 presents a characterization of the rationale support tools. Section 4 presents the current conceptual model. Section 5 describes the REMIS prototype and its most relevant features. Section 6 presents the experience and lessons learned from a practical application in industry where we used REMIS. Section 6 presents a summary and future research work.

2 A Retrospective of Rationale-Driven Approaches and Tools

The design rationale research community has invested much effort into developing concepts, methods, and techniques for capturing, retrieving, and analyzing the reasoning behind design decisions. The initiators of the field were Kunz and Rittel [15], who developed the IBIS method based on the principle that the design process for complex problems is basically a conversation among stakeholders (e.g., designers,

customers, implementers). They started looking at the conversational process of arguing about complex problems during the design of buildings and cities. Each stakeholder contributed with his experience to the resolution of design issues. This approach was oriented to promote debate as a mechanism to provide a means for understanding the other's view and, in consequence, to obtain a more comprehensive view of the complex problem. IBIS led the way for approaches such as the Design Space Analysis proposed by McLean et al. [20] and better known as QOC, the Procedural Hierarchy of Issues (PHI) approach [22], and the Decision Representation Language (DRL) [17].

These approaches are called argumentation-based approaches because they focused on the activity of reasoning about a design problem and its solution [9]. Afterwards, these argumentation-based approaches were transferred to the mechanical engineering and software engineering fields and applied to design problems. Tools were considered an important factor for successfully managing rationale information. Most tools supporting argumentation-based approaches are hypertext-based systems that connect all pieces of information through hyperlinks, e.g., gIBIS [7], SYBIL [18], and the recently developed Compendium [6].

2.1 Software Engineering and Rationale

Dutoit et al. [9] introduce the term Software Engineering Rationale, claiming that this term is more useful for discussing rationale management in software engineering. They emphasize that the software development life cycle contains several activities where important decisions are taken, and where rationale plays an important role. In software engineering, most approaches have contributed to the rationale domain by providing new ideas and mechanisms to reduce the risk associated with rationale capture. Such approaches were conceived having in mind the goal of providing short-term incentives for those stakeholders who create and use the rationale. For example, SCRAM [34], an approach for requirements elicitation, integrates rationale into fictitious scenarios that are presented to users or customers so that they understand the reason for them and provide extra information. They can immediately see the use and benefit of rationale. Something similar happens in the inquiry cycle [27], which is an iterative process whose goal is to allow stakeholders and developers to work together towards a comprehensive set of requirements.

Most of the approaches developed for Software Engineering Rationale offer tool support provided as either adaptations or extensions of specific requirements and development tools, e.g., SEURAT [5], Sysiphus [10], DRIMER [29], or the Win-Win Negotiation Tool [38]. SEURAT integrates into a development environment a sort of plug-in for rationale capturing especially enhanced with an ontology of rationale terms and a rationale checking mechanism that guides developers in efficiently collecting rationale information and showing them at once the benefits of it. A similar short-term incentive strategy is adopted by Sysiphus, but in a collaborative modeling environment. DRIMER [29] is a software development process and tool for applying design patterns. It provides storage and retrieval of patterns application examples and their rationale. Designers who are looking for a pattern can better understand how to use it by looking at the rationale. Finally, the Win-Win negotiation tool, which

supports the corresponding model, is an example of rationale as a driver of a software development project [4]. The set of requirements to be implemented in each iteration is decided by following the Win-Win model, where issues, i.e., disagreements between parties, with different win conditions are discussed. Options are proposed and an agreement is taken. Win conditions are prioritized and scheduled to iterations based on risks. Other examples of tools are REMAP [31] and C-ReCS [13].

2.2 Process Modeling and Rationale

Little work has been done in other areas apart from design and requirements. One of them is the process modeling area. Here, the need and value have been identified, and a couple of research initiatives have been followed with the goal of generating rationale information from project-specific process models. One approach developed by Dellen et al. [11] is Como-Kit. Como-Kit allows automatically deducing causal dependencies from specified process models. Such dependencies could be used for assessing process model changes. Additionally, Como-Kit provides a mechanism for adding justifications to a change. The Como-Kit system consists of a modeling component and a process engine. Como-kit was later integrated with the MVP approach [3]. The MVP approach consists of the MVP-L language and the MVP-E system, which supports the modeling and enactment of software processes. The result of such an integration effort was the Minimally Invasive Long-Term Organizational Support platform (MILOS) [37], [21]. MILOS enables the modeling of both algorithmic and creative processes, the collection of data for the purpose of process guidance, and experience management. Sauer presented a procedure for extracting information from the MILOS project log and for justifying project development decisions [33]. According to Sauer, rationale information could be semi-automatically generated. However, the approach does not capture information about alternatives that were taken into account for a decision.

Weber et al. [39] introduce an agile process mining framework that supports the whole process life cycle as well as continuous adaptation to change. The framework combines three different domains, namely process mining [36], adaptive process management (PM) [32], and conversational case-based reasoning (CCBR) [39]. Changes are registered in change logs during project execution. Changes can be referenced to cases in a case-base. A case represents a concrete ad-hoc modification of one or more process instances. A case consists of a textual problem description, a set of question-answer pairs, and the solution. The process engineer can provide information on the case so that future analysis for understanding the context of and the reasons for discrepancies between process models and related instances are possible.

3 Characterization of Rationale Tool Support

Table 1 provides an overview of the diversity of tools implemented for providing rationale support to a given approach.

Table 1. Tool Support

Approach	Tool/Prototype Support
Category 1	
IBIS [15]	gIBIS[7], Compendium [6]
Design Space Analysis (QOC) [20]	Compendium [6]
The Decision Representation Language (DRL) [17]	SYBIL [18]
Inquiry Cycle Potts et al.- [27]	Active HyperText Prototype [28]
Category 2	
Contribution Structures- Gotel and Finkelstein - [16]	Contribution Manager Prototype [16]
Como-Kit [11]	Como-Kit System [11]
Agile Process Mining - Weber et al. - [39]	ADEPT [32] + CBRFlow [45]
Category 3	
Hierarchy of Issues (PHI) [22]	JANUS [12], PHIDIAS [23]
REMAP - Ramesh and Dhar- [31]	REMAP System [31]
C-ReCS - Klein [13]	C-ReCS System [13]
SEURAT- Burge and Brown - [5]	SEURAT System [5]
Sysiphus- Dutoit and Paech - [10]	Sysiphus [10]
WinWin - Boehm et al.- [4]	WinWin Negotiation Tool [38]
DRIMER - Pena-Mora and Vadhavkar - [29]	SHARED-DRIMS [29]

These tools can be classified into three major categories: tools that support debate/argumentation; tools that support editing work and rationale documentation; and tools that support integrated editing work and debate/argumentation.

Category 1 - Support for debate/argumentation: The main feature of the tools in this group is to support the collaborative debate of complex problems. Rationale capture, management, and visualization are important functionalities of these tools. Visualization is implemented with graphical browsers that connect each rationale piece of information as hypertext. Usually, these tools provide a linking mechanism to reference the external artifact being discussed. Examples are: gIBIS [7], SYBIL [18], Compendium [6], and the Active Hypertext Prototype [28].

Category 2 - Support for editing work and rationale documentation: This group consists of tools that incorporate rationale as important additional information, but whose main feature is to provide support for users on the task they are performing. The front end in these tools is the specialized task editor. Possibilities for capturing, generating, visualizing, or retrieving rationale information for a given task or task element are provided. Examples are: Contribution Manager Prototype [16], Como-Kit System [11], and the integration of ADEPT [32] and CBRFlow [39].

Category 3 - Support for integrated editing work and debate/argumentation: The main rationale behind these tools is to avoid the criticism of the costs involved with capturing rationale and its intrusiveness by seamlessly integrating debate/argumentation into the collaborative work. Usually, these tools provide mechanisms to easily switch from the task editor to the rationale editor and to visualize both the task and its rationale in one place. The set of tasks and its rationale is conceived as a whole and therefore, changes to each task propagate to all users.

Some task editors are specialized according to the activity as in the case of requirements or design, while some others have attempted to provide a more "generic" task editor. Examples of such tools are: JANUS [12], PHIDIAS [23], REMAP [31], C-ReCS [13], SEURAT [5], Sysiphus [10], WinWin Negotiation Tool [38], and SHARED-DRIMS [29].

4 Process Rationale Concepts and Prototype

The following is a conceptual model that can be considered a second version of our attempt to understand the information needs for capturing the rationale behind process changes (see Figure 2). The results of our first attempt have been documented in [26]. We decided to start with a small set of concepts that will be refined in time. The reason for keeping the model as simple as possible comes from the criticism regarding the high costs of capturing rationale information. We wanted to avoid these high costs and find those appropriate concepts needed to describe the rationale for process changes.

4.1 Concepts

We decided to take the basic concepts of the argumentation-based approaches and connect them to three entities that were relevant for us, i.e., event, changes, and process element (the non-shadowed classes in Fig 1).

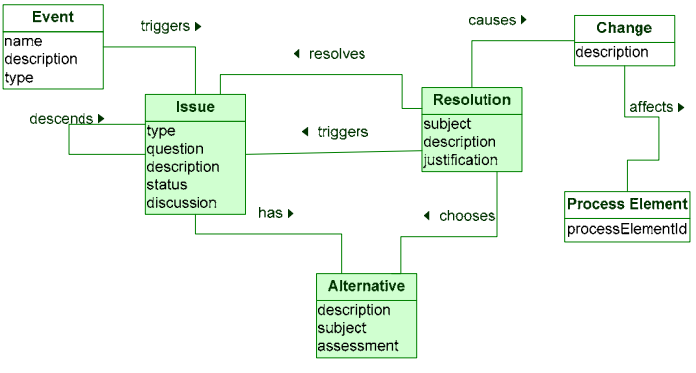


Fig. 1. Rationale Model (UML static structure diagram)

An *event* is considered to be the trigger of issues. Events can happen inside (internal) or outside (external) a given organization. Examples of different types of internal events are: new/updated process engineering technology (e.g., a new process modeling technique); new/updated regulatory constraints; Examples of different types of external events are: responses to failures to pass internal or external appraisals, assessments or reviews (e.g., changes needed to address a failure in passing an FDA audit); new/updated best practices emerging from "lessons learned" in just-completed projects (e.g., a new "best practice" approach to handling design reviews).

Issues are problems that are related to a (part of a) process and that need to be solved. Issues are stated usually as questions in product-oriented approaches. In this work, the question has the purpose of forcing process engineers to reason about the situation they are facing. Additionally, an issue also contains a long description, a status (open, closed) and a discussion. The discussion is intended for capturing the emails, memos, letters, etc. where the issue was treated by process engineers. Additionally, an issue can be categorized by a type. This type can be selected from a classification of issues that needs to be developed or customized for an organization [26]. The classification can be used as a basis, which should be refined continuously based on experience gained from process evolution projects.

Alternatives are proposals for resolving the issue. Alternatives can be captured with subject (short description) or long descriptions. Alternatives are evaluated and assessed regarding their impact and viability by process engineers.

Finally, a *resolution* chooses an alternative whose implementation causes changes to the process models. At the same time, one resolution could lead to opening more issues. Note that a resolution has a subject (short description), a long description, and a justification. The justification is intended for capturing a summary of the analysis of the different alternatives, the final decision, and the pro-posed changes. *Changes* are the result of implementing the decision captured in the resolution. They are performed on process elements. Some examples of changes performed to process elements are: *activity x has been inserted; artifact y has been deleted; activity x has been moved to be a sub-activity of activity z.*

4.2 Prototype and Technical Infrastructure

This section presents the current state of the REMIS prototype, which we developed with the goal of supporting our work concerning the systematic evolution of a process model. It is important to mention that ideas behind REMIS were taken from previous research work, where we developed an approach for evolving a text-based process description within the aerospace domain [2]. Our solution relies on the fact that modern word processing programs increasingly support the Extensible Markup Language (XML) as a document format [24]. As an open format, XML can be processed using a variety of widely available tools, including high-level libraries that can be invoked from most modern programming languages. Using the interpreted, object-oriented Python programming language [19], we developed a parser that is able to navigate through the XML-specific version of the ASG process model description, identifying the section headings and rationale information tables, and moving information to and from the process evolution repository as necessary. This functionality allowed us to update a database (i.e., the process evolution repository) automatically after a set of changes, and to check the data for consistency before doing any further editing.

We have used the Resource Description Framework (RDF) as a basis for representing both process and rationale information in the process evolution repository. In brief, RDF was originally designed for representing metadata about Web resources, such as the title, author, modification date of a Web page, and copyright. However, it is possible to generalize the concept of “Web Resource” and say that RDF can be used to represent “things” that are identifiable. We see the

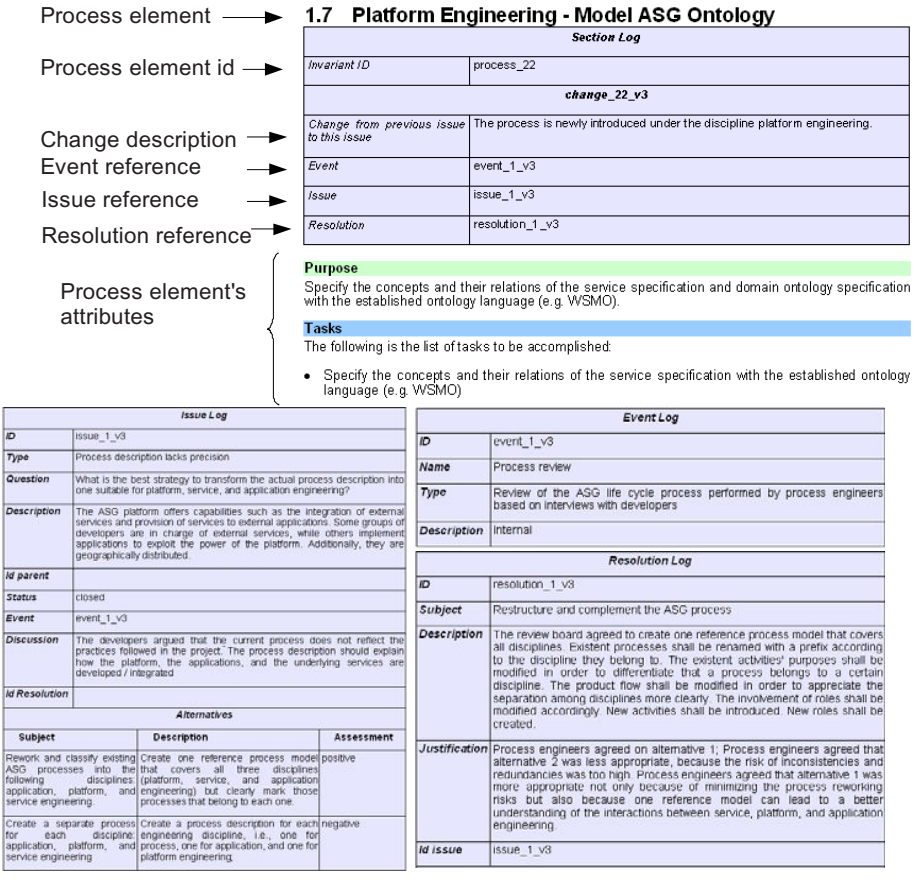


Fig. 2. Rationale Information Integrated into the Process Model

rationale as metadata about processes. Fig. 2 shows an excerpt of the ASG process model description as seen by the process engineer. It can be observed that the document contains a section for rationale information references and a section for the actual process description of attributes. The actual rationale information can be documented in special tables at the end of the document. The process engineer can then introduce the rationale information, perform the changes in the respective parts of the document, and then establish a reference to the corresponding rationale.

Such metadata can be queried for describing the evolution of processes. RDF's conceptual model allows describing 'things' by using statements and models such as nodes and arcs in a graph. We use the RDF/XML syntax [14] for storing and querying RDF graphs in the database.

Fig. 3 presents the REMIS' user interface and the main functionality offered. This functionality supports the method we have designed so far for systematically changing a process model. Let us assume that the process engineer(s) or person(s)

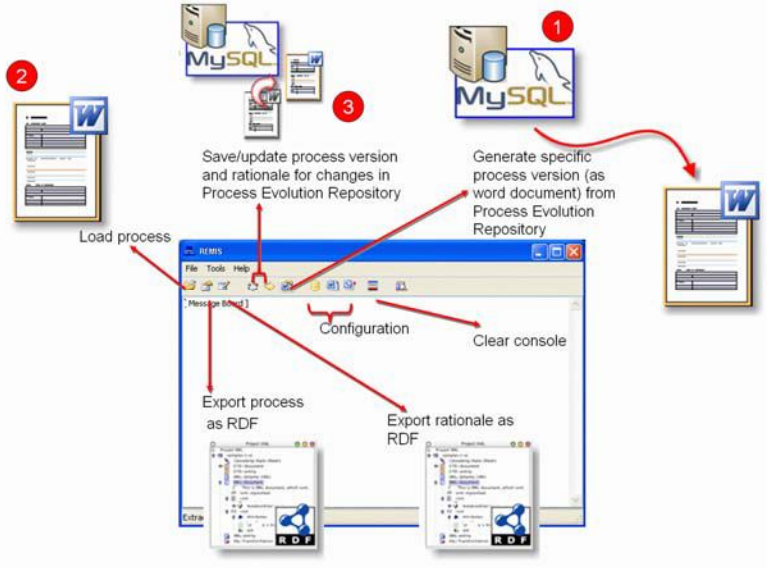


Fig. 3. Prototype Tool Support

responsible have identified an issue. Let us assume that the process engineer(s) or person(s) responsible have identified an issue. First, he/they should extract the current version from the process evolution repository (step 1 in Fig. 3).

This can be done using the REMIS function, which, given a parameter with the desired version, generates a word document from the process evolution repository.

The process engineer can document the event, the identified issue, the alternatives proposed to solve the issue, and the resolution taken (see Fig. 2). Once this is done, the process engineer can change the process model, and reference the changes to the just documented rationale. After having performed all changes, he can use the load functionality from REMIS (step 2 in Fig. 3). Once this is done, the process engineer can update the process in the process evolution repository with the new changes. He can do this by using the REMIS functionality that updates the process information and the corresponding rationale (step 3 in Fig. 3). This functionality is implemented in such a way that REMIS compares the current version in the repository with the just loaded and changed version, identifies the changes, and stores the new version with its corresponding rationale information in the process evolution repository. We have included as additional functionality the possibility to export the process model or the rationale information to RDF models. This function was implemented for giving the user the possibility to visualize the information in RDF-capable tools such as Protegé [30] or for making queries to the RDF models with languages such as SPARQL [35].

5 Experience and Lessons Learned

The environment where we applied our conceptual model and tool corresponds to the Adaptive Services Grid (ASG) project [1]. The ASG project was intended to develop

a software infrastructure that enables design, implementation, and use of applications based on adaptive services, namely the ASG platform. We were in charge of defining, establishing, evaluating, and systematically evolving the development process applied in the project to develop the platform. Development activities were performed, for instance, within the ASG project by several teams from different companies, universities, and research institutes. Development teams ranged from two-person teams consisting of a PhD student and a master student to ten professional programmers. Development teams were not collocated and team members spoke different native languages.

The software process was described in terms of activities, artifacts, roles, and tools, which are concepts that correspond to process element shown previously in Fig. 1. The resulting process model includes both textual descriptions and diagrams that illustrate the relationships between the entities of the model in a graphical way (e.g., workflows and role-specific views).

The ASG reference process model was developed mainly in 5 iterations. This means that there are 5 versions of the model. At certain points in time, we interviewed developers about the current process model version. Such interviews were taken as a basis for performing changes to the process model. We discussed the interviews, decided on the changes, and documented their rationale. Then we proceeded to perform the approved changes. These activities were supported by the REMIS prototype.

The final version contained 26 processes, 31 artifacts, 10 roles, and 11 tools. 353 changes to the model were performed in total from version 1 to version 5. These changes correspond to 15 issues identified from the interviews. The interviews were designed to elicit from developers important aspects such as the current problems and improvement suggestions, which could be directly mapped to the issues and alternatives. One major concern we had was the difficulty involved in first discussing and then performing the changes to the model. At the beginning, it was hard for us, acting as process engineers, to get accustomed to this way of work. However, after having discussed a couple of issues, we felt more comfortable and saw the advantages of it. REMIS assured that all relationships established in the Word document between process changes and rationale were kept and stored in the process evolution repository. REMIS also assured the consistent storage of different versions of the process model. The information stored in the process evolution repository allowed us to answer questions such as: Which process elements were affected by a change? Which process element was affected by the highest number of changes? Which issue had the largest impact on a process? Which are still unresolved issues? Which type of issues demand the highest number of changes?

Concerning the visualization, it was important for us that before changing a process model, we could see previous changes, where they were introduced, and why. This way, we could better justify our new changes. REMIS supported us by generating a given version of the process together with its rationale information as a Word document. Another mechanism to visualize the information was querying the RDF models exported from the tool, or the RDF models stored in the process evolution repository. We used internally developed tools for that purpose. We observed that the amount of information in one visualization graph or table can become overwhelming and difficult to read. We are currently investigating how to minimize this problem.

6 Summary and Outlook

This article presented the current results of our research work towards a systematic mechanism for rationale supported process evolution. In particular, we described the REMIS prototype developed for proving our conceptual model.

Despite certain difficulties arising from applying our concepts and prototype for the first time to a practical, real world project, we consider our results quite satisfactory. REMIS proved to be suitable for the set of problems at hand, and showed the potential for being applicable to future similar problems. REMIS helped process engineers in connecting the reasons for changes to the process model and in consistently storing both in one place.

We think that our technology choice played a central role for the success of this initial prototype trial. First of all, being able to produce easily processable XML documents directly from a standard Word processing application was instrumental to many of the tasks we performed in the project. On the one hand, using a standard word processor (as opposed to a specialized process modeling application) not only made our work easier and more comfortable, but also allowed us to interact with other stakeholders in a straightforward way. On the other hand, having such documents readily available in XML form provided us with a wide choice of technologies to analyze and process the data.

During our work, we identified several open research questions. One of them deals with the visualization of the large amount of information stored in the process evolution repository. We are currently investigating mechanisms that facilitate such visualization, e.g., we are trying to identify a set of "most wanted queries" based on the special interests of organizations interested in managing process evolution. Such queries can be deduced from the goals of the organization and reduce the scope of the information to be analyzed.

Acknowledgments. We would like to thank all ASG project members who participated in the interviews as well as Sonnhild Namingha from Fraunhofer IESE for preparing the English editing of this paper. This work was partially supported by the German Federal Ministry of Education and Research (V-Bench Project, No.01| SE 11 A).

References

1. ASG: Adaptive Services Grid. Integrated Project Supported By the European Commision. Available at: http://asg-platform.org/cgi-bin/twiki/view/Public
2. Armbrust O, Ocampo A, Soto M. Tracing Process Model Evolution: A Semi-Formal Process Modeling Approach. In: Oldevik, Jon (Ed.) u.a.: ECMDA Traceability Workshop (ECMDA-TW) 2005 - Proceedings. Trondheim, 2005, 57-66.
3. Bröckers A, Lott, C.M, Rombach H.D, Verlage M. MVP-L Language Report Version 2. Technical Report 265/95, Department of Computer Science, University of Kaiserslautern, Germany, 1995.
4. Bohem B, Egyed A. Kwan J, Port D, Shah A, Madachy R. Using the WinWin Spiral Model: A Case Study. IEEE Computer, vol 31, no 7, pp 33-44, 1998.

5. Burge J, Brown D.C. An Integrated Approach for Software Design Checking Using Rationale. In: Gero, J (ed) Design Computing and Cognition '04. Kluwer Academic Publishers, Netherlands, pp. 557-576, 2004.
6. Compendium Institute. http://www.compendiuminstitute.org/.
7. Conklin J, Begeman M.L. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. ACM Transactions on Office Information Systems, vol. 6, no. 4, pp. 303-331. 1988.
8. Curtis, B., Kellner, M. I., Over, J. 1992. Process modeling. Commun. ACM 35, 9 (Sep. 1992), 75-90.
9. Dutoit A. (Ed), McCall R. (Ed.), Mistrík I.(Ed.), Paech B. (Ed.). Rationale Management in Software Engineering. Berlin: Springer-Verlag, 2006.
10. Dutoit A, Paech B. Rationale-Based Use Case Specification. Requirements Engineering Journal. vol. 7, no 1, pp. 3-19, 2002.
11. Dellen B, Kohler K, Maurer F. Integrating Software Process Models and Design Rationales. In. Proceedings of 11th Knowledge-Based Software Engineering Conference (KBSE '96), Syracuse, NY, pp. 84-93, 1996.
12. Fischer G, Lemke A, McCall R, Morch A. Making Argumentation Serve Design. In Moran TP, Carroll JM (eds) Design Rationale, Concepts, Techniques and Use, Lawrence Erlbaum Associates, Mahwah, NJ, 267-294.
13. Klein M. An Exception Handling Approach to Enhancing Consistency, Completeness, and Correctness in Collabora-tive Requirements Capture. Concurrent Engineering Research and Applications, vol 5, no 1, pp. 37-46. 1997.
14. Klyne, G., Carroll., J eds.: Resource Description Framework (RDF): Concepts and Abstract Syntax W3C Recom-mendation 10 February 2004. Available at: http://www.w3.org/TR/rdf-concepts/
15. Kunz W, Rittel H. Issues as Elements of Information Systems. Working Paper No. 131, Institut für Grundlagen der Plannung, Universität Stuttgart, Germany, 1970.
16. Gotel O, Finkelstein A. Contribution Structures. In: Proceedings International Symposium on Requirements Engi-neering, IEEE, York, pp. 100-107, 1995.
17. Lee, J. A Qualitative Decision Management System. In P.H. Winston & S. Shellard (eds.) Artificial Intelligence at MIT: Expanding Frontiers, Vol.1, pp. 104-133, MIT Press, Cambridge, MA, 1990.
18. Lee, J. SIBYL: a Tool for Managing Group Design Rationale. In Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work (Los Angeles, California, United States, October 07 - 10, 1990). CSCW '90. ACM Press, New York, NY, 79-92.
19. Lutz, M.: Programming Python (2nd Edition). O'Reilly & Associates, Sebastopol, California (2001)
20. MacLean A, Young R.M., Belloti V, Moran T. Questions, Options, and Criteria: Elements of Design Space Analy-sis. Human-Computer Interaction, Vol. 6, pp. 201-250, 1991.
21. Maurer F, Dellen B, Bendeck F, Goldmann S, Holz H, Kötting B, Schaaf, M. Merging Project Planning and Web-Enabled Dynamic Workflow Technologies. IEEE Internet Computing (2000), Vol. 4(3), pp.65-74.
22. McCall, R. PHIBIS: Procedural Hierarchical Issue-Based Information Systems. Proceedings of the Conference on Architecture at the International Congress on Planning and Design Theory, American Society of Mechanical Engi-neers, NY, pp. 17-22, 1987.
23. McCall R, Bennett P, D'Oronzio P, Oswald J, Shipman F.M, Wallace N. PHIDIAS: Integrating CAD Graphics into Dynamic Hypertext. In Streitz N, Rizk A, André J (eds), Hypertext: Concepts, Systems and Applications, Cam-bridge University Press, N.Y, pp. 152-165.

24. Merz, D.: XML for Word Processors. IBM Developer Works: 25 February 2004. Available at: http://www-128.ibm.com/developerworks/library/x-matters33/
25. Nejmeh B.A, Riddle W.E. The PERFECT Approach to Experience-based Process Evolution. Advances in Com-puters, M. Zelkowitz (Ed.), Academic Press, 2006
26. Ocampo A, Münch J.: Process Evolution Supported by Rationale: An Empirical Investigation of Process Changes. In: SPW/ProSim 2006 - Proceedings. Berlin Springer-Verlag Lecture Notes in Computer Science 3966, pp.,334-34, 2006.
27. Potts C, Bruns G. Recording the Reasons for Design Decisions. In Proceedings of the 10th International Conference on Software Engineering (ICSE'10). Los Alamitos, CA, pp. 418-427, 1988.
28. Potts, C. and Takahashi, K. An Active Hypertext Model for System Requirements. In Proceedings of the 7th interna-tional Workshop on Software Specification and Design (Redondo Beach, California, December 06 - 07, 1993). IEEE Computer Society Press, Los Alamitos, CA, pp. 62-68, 1993.
29. Pena-Mora F, Vadhavkar S. Augmenting Design Patterns with Design Rationale. Artificial Intelligence for Engi-neering Desgin, Analysis, and Manufacturing, vol 11, pp. 93-108, 1996.
30. Protegé: Ontology Editor. 06 January 2006. Available at: http://protege.stanford.edu/
31. Ramesh B, Dhar V. Supporting Systems Development by Capturing Deliberations During Requirements Engineer-ing. IEEE Transactions on Software Engineering, vol 18, no 6, pp. 498-510, 1992.
32. Reichert M, Dadam P. ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. Journal of Intelligent Information Systems 10, 93- 29, 1998.
33. Sauer T. Project History and Decision Dependencies. Diploma Thesis. Univer-sity of Kaiserslautern 2002.
34. Sutcliffe A, Ryan M. Experience with SCRAM, a Scenario Requirements Analysis Method. In Proceedings of the 3rd International Conference on Requirements Engineering, Colorado Springs, CO, pp. 164-173, 1998.
35. SPARQL: Query Language for RDF. 06 January 2006. Available at: http://www.w3.org/ TR/ rdf-sparql-query/
36. v.d. Aalst W, van Dongen B, Herbst J, Maruster L, Schimm G, Weijters A: Workflow mining: A Survey of Issues and Approaches. Data and Knowledge Engineering 27, 237-267 2003.
37. Verlage M, Dellen B, Maurer F, Münch J, A Synthesis of Two Process Support Approaches, Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering (SEKE'96), pp. 59-68, Lake Tahoe, Nevada, USA, June 10-12, 1996.
38. WinWin. The Win Win Spiral Model. Center for Software Engineering University of Southern California http://sunset.usc.edu/research/WINWIN/winwinspiral.html
39. Weber B, Rinderle S, Wild W, Reichert, M. CCBR–Driven Business Process Evolution. Int'l Conf. on Case-Based Reasoning (ICCBR'05), pp. 610-624, Chicago, August 2005.

On the Measurement of Agility in Software Process*

Beijun Shen[1] and Dehua Ju[2]

[1] Dept. of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China
bjshen@sjtu.edu.cn
[2] Application Solutions & Technologies Inc., Shanghai 200230, China
dh.ju@computer.org

Abstract. Agile software process may become one of the most rational development patterns in global economic environment to assist software enterprise to make rapid response to the market. This paper proposes a method to measure agility in software process using goal-driven techniques and balanced scorecard. Using this method, we design a set of representative agility metrics for measuring agility in software process. We also perform one case study for the proposed agility measurement.

Keywords: Agility, Software Process, Measurement.

1 Introduction and Motivation

As market globalization raises competitive pressures worldwide, one essential requirement for enterprises' survival is their inherent ability to meet customer needs and demands. To meet this challenge, new software development paradigm – agile software process – was born in time since 1996 [1]. In late 2001, Agile Alliance was formed and stated their famous manifesto [2]. Until now, agile processes have been applied successfully at some software enterprises, and all practices indicate they are effective in some specific domains.

However, we find that there are two myths in the agile process practices in some Chinese companies:

1) There seems to be a general feeling in the agile community that if you follow all the practices associated with your chosen method then you are agile in name. While this may be true for original agile methods, which have defined a set of practices with emergent properties such that the team becomes more agile as a result of the process, it is still possible to apply XP or Scrum without gaining much in terms of agility.

2) "Light" is one form of agile. Under the pressure of budget and schedule, in fact, some organizations have put quite "light" development into practices. They spend a little efforts and times on estimation, analysis, design, testing and documents, which looks like implementing agile process imperfectly. On the pretext of various objective factors, they become weak in promoting communication, simplicity and feedback. At

* This research is supported by the National Natural Science Foundation of China (No. 60373074).

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 25–36, 2007.

the same time, they have the courage without discipline to take risks of chaotic management. It is obviously inconsistent with the basic principles of agile process.

How to determine that a software process is really agile? "One size does not fit all", so it is impossible that only one agile process template can be suitable for all projects. Applying in different contexts, scopes or project types, it could result in total difference in agility of the same software process. Even at this moment there is no enough data available to prove that agile methods are better than traditional methods. Although anecdotal evidence suggests that agile methods are more effective at delivering working solutions, the only practical way of determining whether agile methods actually make a difference to your software development process is to measure your own process and to see whether agile methods are actually effective or not.

The question is how to design such measurements and for what we should measure. In this paper, we propose a goal-driven approach to measure agility in software process. It combines GQM and balanced scorecard methodologies, and guides software organizations develop their own set of agility measures in context of their business goals. After applying the approach to four typical software companies and further investigation on more organizations, a set of representative agility metrics are recommended and formalized. These metrics are focused on the main aspects of customer, financial, internal process, and innovation. Preliminary experiments show that proposed approach and metrics can help assess organization's health and performance systematically, and find existed problems and possible improvement opportunities.

The rest of the paper is organized as follows: Section 2 illustrates related works. In section 3, we present a new measurement approach based on GQM and balanced scorecard. In section 4, a set of representative agility metrics are proposed after practices and investigation. In section 5, preliminary experiments to evaluate capability of proposed approach are shown. Finally section 6 presents the conclusions and future work.

2 Related Works

There have been some attempts at measuring and proving the efficacy of agile software development methods versus traditional methods. Some of them focus to demonstrate the benefits of agile methods using traditional software metrics [3][4]. These studies have shown that agile methods are at least as good as traditional methods. Apparently it is more difficult to gather and analyze metrics in agile projects, so some people try to propose the lightweight measurement approach [8][9]. However, all this kind of studies adopts traditional software metrics, without mentioning the agility metrics.

At the same time, agility metric attracts attention from industry and academy. Williams et al propose an interesting set of agile metrics [5], and Ron Jeffries defines one agile metric – RTF [6]; but these proposed metrics are not formalized. John Favaro designs a ROI metric for agile methods from the economic aspect; however he also suggests using this metric with others [7]. The common disadvantage of these studies is that they only pay attention to one aspect or one metric of agility, not from a comprehensive and systematic view.

There are some of other valuable works we should mention here: Barry Boehm et al describe a risk-based approach to balance the agility and discipline to develop software successfully [10]; D. Hartmann and R. Dymond define which agile metrics are appropriate [18]. Although they do not provide agility metrics, their thoughtful analyses help us find right way.

Our work differs in that it seeks to define a systematic approach to measure software process agility at the level of business from the main aspects of customer, financial, internal process, and innovation. Each software organization could develop its own set of agility measures in context of its business goals. Using this approach, a set of representative agility metrics are formalized, which are appropriate and valuable according to [18] and preliminary experiments.

3 Measurement Approach Based on GQM and Balanced Scorecard

Many software organizations define measures to reflect the relative maturity and health of their software processes [11]. These measures help guide an organization's overall performance and process improvement effort. Rather than just gather metrics directly like lines of code, code complexity or quality metrics which have clearly meanings for the measurement of software systems, agility is very difficult to measure [18]. It is not usually assessed from technical aspect, but from business performance goals such as frequent delivery of software. Thus, we try to develop measures and associated indicators for measuring a software process's agility based on the synergistic application of the balanced scorecard and goal-driven measurement methodologies. Through this iterative approach, a software process's agility and its subgoals are mapped to the balanced scorecard and refined. The goal-question-measurement methodology is then applied to identify indicators and measures for each scorecard dimension.

3.1 GQM and Balanced Scorecard

Two methodologies often employed to develop origination and process measures are the balanced scorecard [12] and goal-driven measurement [13] methodologies. Both methodologies are well known, but usually applied separately. It is suggested to combine the techniques, taking advantage of the best of each. [14]

1) Balanced scorecard

The balanced scorecard is an industry-recognized best practice for measuring the health of an organization. It can be used as a management tool for translating an organization's mission and strategic goals into a comprehensive set of performance measures that provide the framework for an enterprise measurement and management system [12]. It is based on four perspectives of an organization's performance—customer, financial, internal process, and learning and growth.

Using the balanced scorecard framework, an organization can systematically set enterprise strategic goals for each perspective and develop a set of indicators and measurements for the desired outcomes and performance drivers that will enable the achievement of the enterprise outcomes. The result is a set of interconnected goals

and measurements with defined cause-and-effect relationships. As a template, the balanced scorecard can be applied to most businesses.

2) GQM

One of the most effective high level models for the application of metrics to the development process is the Goal-Question-Metric (GQM) paradigm, developed by Victor Basili for NASA [13]. GQM is nominally a three stage process as the name implies. Starting from a goal, the method user develops a set of questions which, if answered, would indicate whether or not the goal had been achieved. Expressing the questions in a quantifiable form leads to choice of metrics.

The goal-driven measurement process aligns measures and indicators with goals, thus ensuring that the measures and indicators selected will be used to show success in achieving these goals.

3.2 New Measurement Approach

Our suggested approach for developing process agility metrics is typically iterative and contains the following three steps:

Step1: set agility as the strategic goal based on organization mission and vision

Agility in software process is more formally defined as the ability of an enterprise to respond to continuous and unpredictable changes. It tries to achieve higher performance through just enough modeling and documentation, without sacrificing quality. Facing fierce competition, agility has been the most one of strategic goals in software enterprises, from which the enterprise can derive more detailed subgoals and activities for achieving its vision.

Step 2: derive subgoals from each quadrant of the balanced scorecard

Agility is purposely at a high level of abstraction, it is necessary to derive subgoals using the balanced scorecard methodology. The subgoal should have quantitative expression of achievement, for example, “Increase market share by 15% in the next fiscal year”. There may be many subgoals derived from each quadrant of the balanced scorecard. It is beneficial to prioritize these subgoals to develop a meaningful and efficient strategy for achieving them, rather than trying to achieve every subgoal. The four quadrants provide the full picture of enterprise, which inherently linked together.

Step 3: apply GQM to pose relevant questions and determine requisite metrics

GQM method is used to pose relevant questions for each subgoal in each quadrant of the balanced scorecard. For example, “How do customers evaluate our timeliness?” from the Customer Perspective. Phrase these questions in a manner that elicits a quantitative response. Using the answers to these questions, some metrics can be derived.

Corresponding agility measurement model based on new approach is shown in Fig.1. This goal-driven approach combines the GQM and Balanced Scorecard techniques, taking advantage of the best of each.

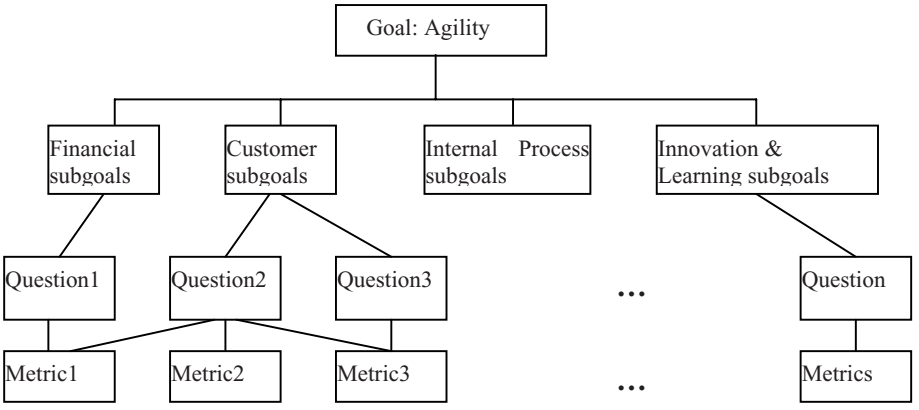


Fig. 1. Agility Measurement Model

4 Representative Agility Metrics of Software Process

Agility metrics are difficult to define, mainly due to the multidimensionality and vagueness of the concept of agility. Applying the above measurement approach, this section tries to propose a set of representative agility metrics for a "typical" software organization. Here, typical is defined as an aggregate of several organizations with similar characteristics.

4.1 Identification of Representative Agility Metrics

We choose four "typical" software organizations in China since 2004 to apply the proposed measurement approach [15], which are:

- ASTI Co., an independent software development and service provider, located in Maryland, United States, founded in 1990. ASTI has two foreign wholly-owned subsidiary companies in Shanghai and Beijing, and one development center in Xi'an, China. ASTI Shanghai and Beijing with about 120 staff is involved in this practice.
- Wanshen Co., a software development and system integration company, founded in 1993. Its software development department with about 30 staff is involved in this practice.
- Baosteel online Co., an online iron and steel e-commerce service provider in Shanghai, founded in 2000 by Shanghai Baoshan Iron and Steel Co. Its software R&D department with about 20 staff is involved in this practice.
- Quality Software Shanghai Co., a Japan software outsourcing company in Shanghai, founded in 2001. The whole company with about 40 staff is involved in this practice.

These four companies all consider the agility as one of the most important strategy goal, and are improving their capability maturity based on ISO9001 or CMMI. The

proposed method won't compare one company with others; rather it is a means to measure relative performance in the same company for the purpose of assessing process improvement. Based on their own business goal and features, these companies established their own measurement models at the end of 2004. Although the derived metrics are not same, there are about eleven common or similar agility metrics. These eleven metrics are selected and unified, and further questionnaire is designed to obtain information from more software companies. Based on the feedback, five representative agility metrics are identified, shown in Table 1. All of these metrics are considered valuable in more than 65% companies, according to the questionnaire data.

Table 1. Representative Agility Metrics

Metric Name	Quadrant of BSC	Brief Description
ROI	financial	Return on investment
Productivity	internal process	Relationship between production of an outputs and the resource inputs used in software development
Quality	customer	Quality of product and service
Adaptability	customer	Ability of adaptation to changes
Innovation	learning and growth	Ability to innovate, improve, and learn

Although these agility metrics are not complete and also not requisite, they can help organizations to measure relative performance, and find possible improvement chances. Also these metrics are lightweight and thus easy to collect.

4.2 Formal Definition of Representative Agility Metrics

Based on a relational system of software development process, the representative agility metrics are defined in this section.

4.2.1 Measurement Theory Base

From the view of throughput accounting, a software development process, P, can be conceptualized as a relational system consisting of object-elements, empirical relations and binary operations that can be performed on the object-elements [16]. By starting with these definitions, the mathematical role of metrics as a transformation can be formally outlined. Notationally:

$$\mathbf{P} = (I, O, Ds, Ts, Is)$$

Where

I (Investment) is the value of process input (i.e. idea).
O (Throughput) is the value of process output (i.e. product).
Ds (Development) are “Do” operations of process, which consist of requirement, analysis, design, coding, and other operations, defined as D1, D2 …Dn.
Ts (Test) are “Compare” operations of process, which consist of unit test, integration test, and acceptance test, defined as T1, T2 …Tn.
Is (Iteration) are loops of process.

Fig. 2 shows a basic system mapping for the software development process. It simply describes how a single idea is transformed into a product. In the figure, investment is the sum of money invested in the system of software production plus the sum spent to obtain the idea; throughput is rate of cash generated through delivery of product into production; and operational expense is the sum of money spent in the system to build from idea to product. In particular, "software inventory" (V) is introduced in this system, which was suggested by Beck in 2002 [17]. The Lean Software Development paradigm is also closely associated with the analogy to inventory management [2].

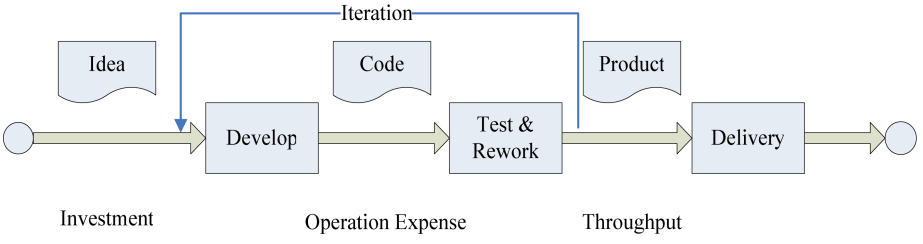


Fig. 2. Software development process

In this relational system, following formulas can be satisfied:

- Investment = ValueInput
- Throughput = ValueOutput = Sales Price - Direct Costs, where Direct Costs are marginal costs directly associated with the sale of working code. These could include packaging and delivery, but more likely cover installation, training, and support.
- Value Added = ValueOutput - ValueInput

4.2.2 Definitions

In the context of the relational system of software process, five representative agility metrics are defined here.

ROI. The change of focus from cost to value is inherent in the agile paradigm shift. Here, value is defined as software put into production that can return an investment over time. The more rapidly high-value software can be rolled out, the quicker value is realized. Return On Investment (ROI) consists simply of the ratio of profit to the amount invested:

$$\text{ROI} = \text{Profit/Investment} = (\text{Throughput} - \text{OperationExpense})/\text{Investment}$$

Productivity. It is a measure of efficiency, which records the relationship between production of an output and one, some, or all of the resource inputs used in accomplishing the assigned task. It is usually measured as a ratio of output per unit of input over time:

$$\begin{aligned}\text{Productivity} &= \text{Value}_{\text{Added}}/\text{Effort} = (\text{Value}_{\text{Output}} - \text{Value}_{\text{Input}})/\text{Effort} \\ &= (\text{Throughput} - \text{Investment})/\text{Effort}\end{aligned}$$

Quality. Pre-release quality is a surrogate measure of quality. We use both pre-release defect density and defect removal efficiency as indicators of quality:

Pre-release defect density = Pre-release test defects / KLOC

Defect removal efficiency = Pre-release test defects / (pre-release defects + post-release defects)

Here, pre-release defects are found before delivery usually through reviews and testing, not including unit testing.

Adaptability. It is measured as how long it takes a unit of V to pass through the system from input to output. The unit of V is either a new requirement or a changed requirement. Apparently, "Adaptability" is not same as the cycle time of project. It indicates how fast a new request/idea can be turned into working code and delivered to a customer.

Innovation. It focuses on an organization's ability to innovate, improve, and learn. Moreover, it is not just having new ideas, but also bringing them to products. It is measured here as a percent of current year sales due to new products released in the past three year:

$$\text{Innovation} = \text{Throughput}_{\text{newProduct}} / \text{Throughput} * 100\%$$

5 Preliminary Experiments

This session summarizes results and findings from the preliminary experiments in ASTI Co., which is the earliest one of four companies to apply the proposed measurement approach.

5.1 Planning of the Experiments

As mentioned in section 4.1, ASTI is an independent software vendor, which provides software solutions and services for oversea and domestic enterprises, including software tools, enterprise information system development, and software localization. Like most other companies in the IT business, ASTI faces new challenges that require changes in the way they develop, deploy and maintain software applications. One of these challenges is that the pace of business change is increasing. Therefore ASTI must improve agility to be able to respond to new and changing requirements. In

Table 2. Staged Measurement Approach

No of Phase	Duration	Number of participants	Number of projects	Activity description
Phase 1	Oct. 2004 ~ Dec. 2004	6 (experts and SEPG)	0	Establish agility measurement model
Phase 2	Jan. 2005 ~ Dec. 2005	48 (ASTI shanghai)	12	Collect data and measure in ASTI Shanghai
Phase 3	Jan. 2006 ~ Dec. 2006	120 (ASTI global)	28	Collect data and measure in ASTI Global

order to facilitate this agile shift, ASTI begins to introduce SE practices from agile methods, including XP and SCRUM. At the same time, we use a staged measurement approach to implement agility metrics, as shown in table 2.

5.2 Results of the Experiments

There are 40 projects participating in this practice during the latest two years, where software tool development projects account for 20%, application projects for 55%, and software localization projects for 25%. Average size of these projects is 19KLOC, and average duration is 176 days. When the process is enacted, quantitative data is gathered by project reports as well as by automated data retrieved by the development environment. Based on the data repository, the agility metrics are calculated, as shown in table 3 ~ table 8. Here, numerical value in phase 1 stands for performance in 2004.

Table 3. ROI Measurement Result in ASTI

Type of Projects	Phase1	Phase2	Phase3
Software tool development	7%	15%	20%
Application development	8%	10%	12%
Software localization	12%	10%	11%
Overall	9%	11%	13.4%

Due to fierier competition, ROI of software localization business in china decreased more than 30% on average in 2005. It is observed that ASTI performed better than its competitors.

Table 4. Productivity Measurement Result in ASTI

Type of Projects	Phase1	Phase2	Phase3
Software tool development	unavailable	0.9KLOC/PM	1.0KLOC/PM
Application development	unavailable	1.6KLOC/PM	1.8KLOC/PM
Software localization	34KWORD/PM	36KWORD/PM	35KWORD/PM

Table 5. Quality (Pre-release defect density) Measurement Result in ASTI

Type of Projects	Phase1	Phase2	Phase3
Software tool development	8.8/KLOC	8.5/KLOC	8.1/KLOC
Application development	7.5/KLOC	7.3/KLOC	7.3/KLOC
Software localization	1.1/KWORD	1.4/KWORD	1.1/KWORD

Table 6. Quality (Defect removal efficiency) Measurement Result in ASTI

Type of Projects	Phase1	Phase2	Phase3
Software tool development	unavailable	78%	80%
Application development	unavailable	85%	88%
Software localization	90.5%	90%	91%
Overall	unavailable	85%	87%

Table 7. Adaptability Measurement Result in ASTI

Type of Projects	Phase1	Phase2	Phase3
Software tool development	unavailable	18 days	17 days
Application development	unavailable	12 days	10 days
Software localization	unavailable	5 days	6 days
Overall	unavailable	10 days	9.5 days

Table 8. Innovation Measurement Result in ASTI

	Phase1	Phase2	Phase3
Innovation	22.3%	22.5%	30.2%

5.3 Findings and Discussion

Although this practice is preliminary, the study has revealed the following findings:

- Measurement itself will inspirit staff to work harder.
- Productivity and quality is lower in software tool development than in application development, because of complexity.
- It is most difficult to improve agility in software localization, because of regular works without much innovation.
- Some practices from XP, RUP and other agile methods, such as peer review and test driven development, reduce the defect density effectively, therefore improve the productivity.
- Through the agility measurements, possible improvement chances are found in time, such as lack of architecture design training, inadequate design review and code walkthrough. And another outcome is that project teams put more attention on business value, not only technologies.

It is encouraging to observe that after two-year practices "managers are very surprised to see something running", experienced engineers emphasize "the real feedback they get every iteration, the ease of combining inexperienced people in the project, and the way they are aware of problems almost immediately when they occur", younger engineers are satisfied from the direct communication and connection with the customer and from the process itself. Based on the feedback from customer questionnaires, the conclusion is that ASTI is moving in the right direction.

We do not compare ASTI with other company, rather measure its relative performance, and find existed problems and possible improvement chances.

6 Conclusion and Future Work

The agility measurement can be used to make investment decisions and process improvement. This paper proposes a goal driven method to measure agility in

software process using GQM and balanced scorecard. Using this method, a set of representative agility metrics has been identified and defined. We also perform one case study for the proposed agility measurement.

Currently, we are analyzing other three case studies conducted at Wanshen Co., Baosteel online Co., and Quality Software Shanghai Co. The results of this family of case studies and that of other researchers will build an empirical body of results concerning agility in various contexts in various organizations. It is also noticed excitedly that a semipublic data repository has been establishing for software benchmarking, supported by Shanghai government. Based on this repository, we can do more case studies, and get more valuable observations. Moreover, we want to automate our measurement approach to save effort.

References

1. Robert Cecil Martin: Agile Software Development, Principles, Patterns, and Practices. Prentice Hall (2002)
2. Agile Alliance, http://www.agilealliance.com/
3. John Noll and Darren C. Atkinson: Comparing Extreme Programming to Traditional Development for Student Projects: A Case Study. In Proceedings of the 4th International Conference of Extreme Programming and Agile Processes in Software Engineering (2003)
4. Francisco Macias, Mike Holcombe, Marian Gheorghe: A Formal Experiment Comparing Extreme Programming with Traditional Software Construction. In Proceedings of the Fourth Mexican International Conference on Computer Science September (2003)
5. Williams, L., Succi, G., Stefanovic, M., Marchesi, M.: A Metric Suite for Evaluating the Effectiveness of an Agile Methodology. In Extreme Programming Perspectives. Addison Wesley (2003)
6. Ron Jeffries: A Metric Leading to Agility, http://www.xprogramming.com/xpmag/jatRtsMetric.htm (2004)
7. John Favaro: Value-based Management and Agile Methods, Proceedings of 4th International Conference on XP and Agile Methods (2003)
8. Julias Shaw, Jeff Patton: Software Metrics for Agile Projects, Agile International Conference (2005)
9. Yael Dubinsky, etc.: Agile Metrics at the Israeli Air Force. Proceedings of the Agile Development Conference (2005)
10. Barry Boehm, Richard Turner: Balancing Agility and Discipline, A Guide for the Perplexed, Addison Wesley (2004)
11. CMU SEI: Capability Maturity Model® Integration (CMMISM), Version 1.1. CMU/SEI-2002-TR-003 &CMU/SEI-2002-TR-004 (2002)
12. Kaplan, R. S., Norton, D. P.: The Strategy-Focused Organization, How Balanced Scorecard Companies Thrive in the New Business Environment. Boston, MA: Harvard Business School Press (2001)
13. Park, R., Goethert, W., Florac, W.: Goal-Driven Software Measurement—A Guidebook. CMU/SEI-96-HB-002 (1996)
14. Wolfhart Goethert, Matt Fisher: Deriving Enterprise-Based Measures Using the Balanced Scorecard and Goal-Driven Measurement Techniques. CMU/SEI-2003-TN-024 (2003)

15. Beijun Shen, Dehua Ju: Goal Driven Measurement of Agility in Software Process. In Proceedings of the 8th International Symposium on Future Software Technology, Japan (2004)
16. David J. Anderson: Agile Management for software Engineering, Pearson Education (2004)
17. Beck, K.: Software-in-Process: A New/Old Project Metric, http://groups.yahoo.com/group/softwareinprocess (2002)
18. Deborah Hartmann, Robin Dymond: Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value, Agile International Conference (2005)

Coping with the Cone of Uncertainty: An Empirical Study of the SAIV Process Model*

Da Yang[1,3], Barry Boehm[2], Ye Yang[2], Qing Wang[1], and Mingshu Li[1]

[1] Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
{yangda,wq,mingshu}@itechs.iscas.ac.cn
[2] University of Southern California, 941 w. 37th Place Los Angeles, CA 90089-0781
{boehm, yangy}@sunset.usc.edu
[3] Graduate University of Chinese Academy of Sciences, Beijing 100039, China

Abstract. There is large uncertainty with the software cost in the early stages of software development due to requirement volatility, incomplete understanding of product domain, reuse opportunities, market change, etc. This makes it an increasingly challenging issue to deliver software on time, within budget, and with satisfactory quality in the IT field. In this paper, we introduce the Schedule as Independent Variable (SAIV) approach, and present the empirical study of how it is used to cope with the uncertainty of cost, and deliver customer satisfactory products in 8 USC (University of Southern California) projects. We also investigate the success factors and best practices in managing the uncertainty of cost.

Keywords: process model, SAIV, cost estimation, cone of uncertainty.

1 Introduction

Cost estimation is the basis for bidding, budgeting, and planning. It may come from expert intuition, analogy with historical projects, formal parametric cost estimation models, etc [1]. However, because of the incomplete information about the system scale or cost drivers, the learning process of the project stakeholders, the requirement volatility, the market change, etc [2, 3, 4], it is difficult to get accurate cost estimation at the early stages of a software project. And the uncertainty of cost is a threat to ensuring the on-time and within-budget delivery of software.

It is illustrated in [5] that the uncertainty ranges of cost estimations present a decreasing trend as the software development lifecycle proceeds. This phenomenon is named as the cone of uncertainty [6, 4, 7].

Kitchenham states in [8] that the uncertainty is an inherent character of cost estimation, the managers do not understand how to use estimates correctly and, in particular, they usually do not handle properly the uncertainty and risks inherent in estimates.

* This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082 and 60473060; the National Hi-Tech Research and Development Plan of China under Grant No. 2006AA01Z185; the National Key Technologies R&D Program under Grant No. 2005BA113A01.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 37–48, 2007.

Cantor [9] also proposes that the variances in the estimate of schedule and budget are quite high, and that the reason many projects fail to meet stakeholders' needs is that they are managed as if these variances do not exist.

Despite the awareness that coping with the uncertainty of cost is important, there is a lack of empirical study in the current literature. Here we studied several instrumented e-services software projects performed at USC (University of Southern California) to address how the practitioners can effectively make cost estimation and handle the uncertainty of cost.

As Brooks states in [10], there is no silver bullet to the success of software project. We think it is the same with the issue of coping with the cone of uncertainty, and cost estimation techniques alone can't solve the problem. Empirical studies on how practitioners handle the uncertainty of cost can give us insights on resolving this problem.

Since 1996, there are about 10 real-client projects every year accomplished by the students enrolled in the software engineering class at the University of Southern California. These projects span across broad areas like digital library, e-business, credit card theft monitoring, etc. The main challenges for these projects are that the development teams are required to deliver customer satisfactory products within 24 weeks. The Schedule as Independent Variable (SAIV) process model [11], an architecture-based extension of timeboxing, is adopted by these teams, and guides them to consistently deliver on-time, within-budget, and stakeholder satisfactory software products.

In this paper, we will discuss how the project teams make cost estimates, assess the uncertainty of cost, make project plan, allocate resources and ensure the delivery of stakeholder-satisfactory products. We use the empirical data to analyze the uncertainty of cost estimations and their influence over the projects. We'll also discuss the critical success factors and best practices of these projects.

2 Related Work

Lederer [2] found that requirement changes, users' lack of understanding of their requirements, lack of adequate estimation methodology or guidelines, lack of historical data, etc. can all contribute to the inaccuracy or uncertainty of estimates. Todd Little presented in [4] that according to the Landmark Inc. data, the uncertainty of cost estimation remains high until late in project development. He observed a pipe rather than a cone of uncertainty. As a reply to Little, Gryphon proposed that the Cone of Uncertainty doesn't reduce itself, and it may be reduced by the improved estimation methods that become available as the project progresses [7].

Jørgensen asserted that reflecting the underlying uncertainty of cost estimation would improve the budgeting and planning process. He also proposed several guidelines for the assessment of software development cost uncertainty [3]. The COCOMO II [5] cost estimation model can make the optimistic and pessimistic estimations, which form the interval of cost and schedule with 90% confidence. Other models such as SLIM [12], SEER[13], and Knowledge PLAN [14] provide similar capabilities. In recent years, several probabilistic cost estimation methods also try to assess the uncertainty of cost estimation and use probability distributions to reflect the uncertainty [15, 16, 17].

Though many researchers have addressed the issue of software development cost uncertainty, there is still lack of empirical research on how practitioners properly handle the uncertainty of cost.

In this paper, we will investigate 8 USC software projects, analyze how the uncertainty of cost is handled, and identify the critical success factors. We use the same uncertainty terminology as described in [3]. The uncertainty is defined in terms of probability, i.e., the degree of uncertainty of an event is described by the probability that the event will happen.

3 Backgrounds

This empirical study is based on 8 USC real-client projects, which began in Fall 2005 and completed at the end of Spring 2006 semester. These projects have real world clients from business companies, governmental organizations, and academic organizations. The software products include: Online Reading Assessment, PubMed Data Mining, Football Recruiting Database, Code Generator, XML Editing Tool, EBay Notification System, Rule-based Editor, and Code Count. These projects followed the SAIV process model, and used the MBASE approach and the Lean-MBASE development guideline [19].

The MBASE Model-Based Architecting and Software Engineering [18, 19] is a process framework and also a process model generator. It uses a set of common anchor point milestones [20, 21]: key life-cycle decision points at which a project verifies that it has feasible objectives (LCO: Life Cycle Objectives); a feasible life-cycle architecture and plan (LCA: Life Cycle Architecture); and a product ready for operational use (IOC: Initial Operating Capability).

In the USC real-client projects, the top constraint of the success model is that the teams have to develop the LCA packages in 12 weeks during the Fall semester, and to develop and transition the IOC packages in 12 weeks during the Spring semester. As a result of this success model constraint, the SAIV [18] process model is generated from the MBASE. In SAIV, the schedule becomes the independent variable, and the lower-priority features become the dependent variable. The SAIV is defined by explicitly enacting the following six process elements [11]:

- Shared vision and expectations management
- Feature prioritization
- Cost estimation
- Architecture and core capabilities determination
- Incremental development
- Change and progress monitoring and control

4 Empirical Analysis of the SAIV Development Process

In the 8 projects under investigation, the students made cost estimations and detailed development plans at the LCA milestone. We find the uncertainties of the cost estimates are high. The students, however, didn't go on making more accurate cost

estimations during the later construction phase as suggested in [7]. They have used the SAIV process to cope with the uncertainties.

We identified, in this empirical study, four success critical factors of coping with the cone of uncertainty as: estimate cost and its uncertainty, create the opportunities to handle uncertainty, enable flexible process to cope with uncertainty, and risk driven strategies for uncertainty mitigation. Fig. 1 shows the four success factors (in rectangle) and the six related SAIV process elements (in ovals).

In this section, we will discuss the four success factors in subsections 4.1-4.4, and evaluate the project performance in subsection 4.5. In each subsection we will present the related SAIV process elements.

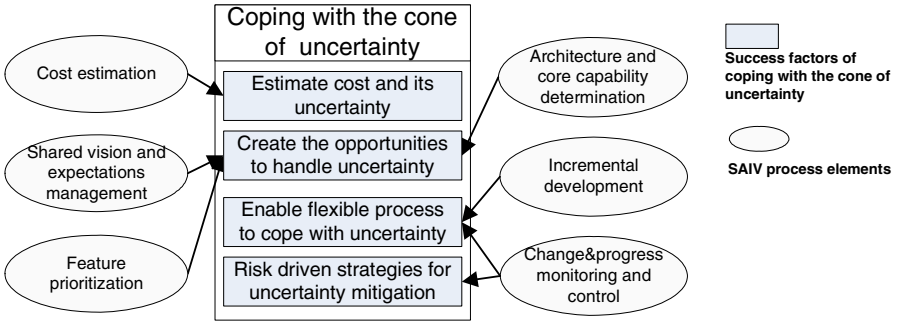


Fig. 1. SAIV elements & Coping with the cone of uncertainty

4.1 Estimate Cost and Its Uncertainty

4.1.1 Cost Estimation

All the 8 projects used COCOMO II for the cost and schedule estimation. As the students' projects are smaller and have a shorter schedule than the projects used in COCOMO II calibration, the students are provided with a new usage guideline based on past development experiences of USC projects.

The students disregarded COCOMO II schedule estimates and used COCOMO II effort estimates to determine how large a team is needed for a 12-week fixed schedule. The estimations are based on the following assumptions:

- Assume 12 hours/week of dedicated effort per person
- Assume 10 of the 12 weeks fill COCOMO II Construction phase (72% of total effort estimate); the final 2 weeks are for product transition into operations.
- Assume 100 hours/person-month for COCOMO estimates

According to the above assumptions, we can derive the following results for the construction phase of the students' projects:

- The estimated effort is "COCOMO II person months"*100*0.72.
- The assumed available construction effort is "number of team members"*12*10.
- Number of team members be larger than "COCOMOII person months"/1.67

Table 1 shows the three COCOMO II effort estimations for the construction phase (Optimistic, Most Likely, and Pessimistic), the number of developers of each team, and the assumed available construction effort (persons * 12 hoursPerWeek * 10 weeks). The column "Most Likely Effort vs. Available Effort" in Table 1 measures how much the estimated most likely effort deviates from the assumed available effort:

Most Likely Effort vs. Available Effort = (Most Likely–Available) /Available (1)

Table 1. Cost Estimation

Team	COCOMO II effort Estimations			Persons	Available Effort	Most Likely Effort vs. Available Effort	Actual Effort	Relative Error (RE)
	Optimistic	Most Likely	Pessimistic					
1	475.2	590.4	741.6	5	600	-0.02	1131.50	-0.478
2	540	669.6	842.4	6	720	-0.07	998.42	-0.329
3	403.2	504	633.6	5	600	-0.16	960.83	-0.475
4	576	712.8	892.8	6	720	-0.01	669.67	0.064
5	597.6	741.6	928.8	5	600	0.24	739.17	0.003
6	518.4	640.8	806.4	5	600	0.07	661.67	-0.032
7	554.4	691.2	864	5	600	0.15	467.08	0.480
8	532.8	662.4	835.2	5	600	0.10	607.67	0.090

We can see that the "Most Likely Effort vs. Available Effort" values are small and below 24%, which means that, according to the COCOMO II cost estimation, in most cases the students can finish the construction phase on time with around 12 hours/week dedicated effort per person. The stakeholders thus considered the current system architecture feasible with respect to the schedule and requirements, and committed to project development.

The cost estimation provides a useful basis to form the shared stakeholder vision on how many features can be delivered within schedule. The expectation management, feature prioritization, and cost estimation can be concurrently conducted.

4.1.2 The Uncertainty of Cost Estimation

The actual effort for construction phase in Table 1 is collected from the students' daily effort report. The accuracy of cost estimation is measured with relative error:

RE(effortEst) = (Most Likely Effort - Actual Effort) / Actual Effort (2)

While the accuracy of an individual cost estimate can be assessed by comparing it to actual effort, the individual cost uncertainty assessment has no obvious corresponding actual values. To assess the uncertainty of a series of estimates, however, we can compare the percentage of confidence level to the proportion of correct assessments ("Hit rate") [3]. The following definition of "Hit rate" is based on uncertainty assessments on the cost prediction interval format, e.g., that it is believed to be "90 percent probable that the actual cost is in the interval [Optimistic cost; Pessimistic cost]".

$$HitRate = \frac{1}{n}\sum_i h_i \ , \ h_i = \begin{cases} 1, Optimistic_i \le Act_i \le Pessimistic_i \\ 0, Act_i > Pessimitic_i \vee Act_i < Optimistic_i \end{cases} \quad (3)$$

We find the actual effort is within the optimistic-pessimistic estimation interval in 4 out of the 8 projects and the HitRate is 50%, lower than the COCOMO II 90%

confidence level. That means the actual uncertainty of cost estimation is even higher than the assessed. Causes for the high uncertainty can be the lack of experience in cost estimation, uncertainties about COTS or open-source component capabilities, the learning process of the students, etc. We will discuss in the following sections how these projects effectively handle the uncertainties.

4.2 Create the Opportunities to Handle Uncertainty

In the SAIV process, the success factor of creating the opportunities to handle uncertainty is related to the process elements: "shared vision and expectations management", "feature prioritization", and "architecture and core capability determination".

4.2.1 Shared Vision and Expectations Management

Expectation management holds the key to providing win-win solutions to the stakeholder negotiation [22]. As described in [23], many software projects lose the opportunity to assure on-time delivery by inflating client expectations and over promising on delivered capabilities. The first element in the SAIV process model is to avoid this by obtaining stakeholder agreement that delivering the system's Initial Operational Capability (IOC) is the most critical objective, and that the other objectives such as the IOC feature content can be variable. The expectation management also provides a basis for effective system feature prioritization.

4.2.2 Feature Prioritization

For each project, the stakeholders used the USC/GroupSystem.com EasyWinWin requirements negotiation tool to converge on a mutual satisfactory set of project requirements. In the negotiation results, there are four categories of requirement priority as "**Must** have", "**Should** have", "**Could** have", and "**Want** to have".

Table 2. Feature prioritization and Core capaility

Team	**Capability Requirements (CR)**					**Percentage of**	**Core Capabilities (CC)**					**CC Total**
	Must	Should	Could	Want	**Total**	**top priority**	Must	Should	Could	Want	**Total**	**/CR Total**
1	20	14	4	6	44	0.45	20	0	0	0	20	0.45
2	9	3	4	1	17	0.53	8	2	1	1	12	0.71
3	6	2	2	2	12	0.50	5	1	0	0	6	0.50
4	6	2	1	0	9	0.67	4	0	0	0	4	0.44
5	5	2	2	0	9	0.56	3	0	0	0	3	0.33
6	12	0	0	1	13	0.92	12	0	0	0	12	0.92
7	13	0	1	3	17	0.76	13	0	0	0	13	0.76
8	5	5	0	2	12	0.42	5	5	0	0	10	0.83

Table 2 shows the distribution of the capability/functional requirements among four priority levels. Column "Percentage of top priority" measures the percentage of the top priority features marked with "Must", and the average percentage is 60%.

The feature prioritization is vital to be able to establish the core capabilities, which should be delivered on time even under pessimistic cases.

4.2.3 Architecture and Core Capability Determination

The core capability requirements must be selected so that its features add up to a coherent and workable end-to-end operational capability. The core capability should

have at least 90% assurance of being completed in 24 weeks, which means even under pessimistic COCOMO II estimation the core capability can be completed. The architecture must also take into account the remaining lower-priority requirements, and make it easy to add or drop borderline features.

Table 2 shows the number of capability requirements (CR), the core capabilities (CC), and the percentage of core capability requirements (CC Total / CR Total). We can see that the core capability usually includes most of the capability requirements marked with "Must" and some of the capability requirements marked with "Should".

4.3 Enable Flexible Process to Cope with Uncertainty

This success factor of enabling flexible process is related to the process elements: "incremental development" and "change and progress monitoring and control".

4.3.1 Incremental Development

The project teams are required to prepare an incremental development plan at the LCA milestone. In their project incremental development plan, the construction is to be completed with two or more iterations. The first iteration will implement the core capability and the remaining iterations will add the lower-priority features. After the first iteration there will be a client-operated Core Capability Drive-Through (the core capability completion milestone).

Since the core capability has 90 percent assurance of being completed in 24 weeks, this means that under the pessimistic case of COCOMO II estimation, the core capability can still be completed within schedule, sometimes with some extra effort.

Table 3. Percentage of duration for iteration one

Team	Planned	Pessimistic	Actual	RE
1	0.72	0.91	0.71	0.01
2	0.43	0.54	0.54	-0.20
3	0.54	0.68	0.57	-0.06
4	0.64	0.80	0.76	-0.16
5	0.51	0.64	0.75	-0.33
6	0.70	0.88	0.75	-0.07
7	0.68	0.85	0.62	0.09
8	0.59	0.74	0.58	0.01

We compare the duration of the first iteration with that duration of the construction phase, and calculate the percentage as showed in Table 3. The planned first iteration will take 43%-72% of the construction-phase duration. To assess the first iteration duration under the pessimistic case, we use the rate of under-estimate to adjust the planned duration:

pessimistic duration = planned duration * (pessimistic effort / most likely effort) (4)

The pessimistic percentage of the duration for core capability implementation is between 54% and 91%, that means even under pessimistic case the core capability can be achieved with 9%-46% construction phase time remaining.

In the most likely (planned) case, however, the project will achieve its core capability with about 28-57% of the schedule remaining as planned.

Table 3 also shows the actual duration of the first iteration. The relative error (RE) measures the uncertainty of planned duration for the core capability implementation:

RE(scheduleIter1) = (Planned Duration – Actual Duration) / Actual Duration (5)

4.3.2 Change and Progress Monitoring and Control

There are several major sources of change that may require re-evaluation of the projects' plans, such as requirements change, technical difficulties, underestimate of effort, staffing difficulties, COTS changes, customer or supplier delay, etc. The core capability completion milestone is a client-operated Core Capability Drive-Through, which often leads the client to reprioritize the remaining planned features.

The project teams may take many options to accommodate to these challenges. They may drop or defer lower-priority features, dedicate more time each day in construction, reuse existing software, or add expert personnel. In some cases, the changes can be accommodated within the existing plans. If not, there is a need to rapidly renegotiate and restructure the plans.

4.4 Risk Driven Strategies for Uncertainty Mitigation

MBASE is a risk-driven process framework, and the SAIV is also a risk driven process model [18]. The projects' monitoring and control activities include:

- Development of a top-N project risk item list that is reviewed and updated weekly to track progress in managing risks (N is usually between 5 and 10)
- Inclusion of the top-N risk item list in the project's weekly status report

When the uncertainty is high, the risk management can help the students determine what to do next and how much is enough, e.g., prototyping, COTS evaluation, architecting, testing, and business case analysis. The risk management strategies include Buying Information, Risk Avoidance, Risk Transfers, Risk Reduction, and Risk Acceptance [24]. Take Team 1 as an example. The team members explained in their problem report "The lack of GUI prototypes may lead to significant rework at the end of the project in order to accommodate the clients GUI requirement changes". The students were suggested to adopt the Buying Information strategy and construct more GUI prototypes to mitigate the uncertainty of GUI requirements.

4.5 The Performance of SAIV Process

4.5.1 The Execution of Iteration Plan

Though the total project duration is an independent variable in the SAIV process, the duration of the first iteration can change to accommodate to the uncertainty of cost or other changes. When there is under-estimate of effort, the teams can extend the duration of the core capability development, delay some capabilities to future iterations, or drop more low-priority features.

The core capability features should be completed by the first iteration according to the iteration plan. We present in Table 4 the total number of core capabilities (CC Total), and how many of the core capabilities have been completed in iteration one as planned. Comparing the total number of completed capabilities and the core capabilities, we get the Completion Rate (Completed Total/CC Total) and Relative Error:

$$\text{RE(capabilityIter1)} = (\text{CC Total} - \text{Completed Total}) / \text{Completed Total} \quad (6)$$

Table 4. Core capability requirements completed in iteration one

Team	CC Total	Completed in Iteration One					Completion Rate	RE
		Must	Should	Could	Want	Total	(Completed Total / CC Total)	
1	20	19	0	0	0	19	0.95	0.053
2	12	8	2	0	1	11	0.92	0.091
3	6	4	1	0	0	5	0.83	0.200
4	4	4	0	0	0	4	1.00	0.000
5	3	4	0	0	0	4	1.33	-0.250
6	12	7	0	0	0	7	0.58	0.714
7	13	13	0	0	0	13	1.00	0.000
8	10	4	5	0	0	9	0.90	0.111

Fig. 2 shows the uncertainties using boxplot [25], which simultaneously displays the median, the inter-quartile range, and the smallest and largest values for each group. We find the magnitude of relative error of iteration duration or core capability implementation is much smaller than that of effort estimation.

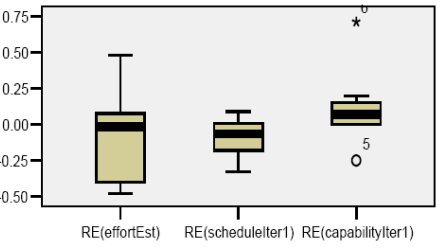


Fig. 2. The uncertainty of cost estimation, iteration1 duration, and iteration1 capability implementation

We use the Pearson's correlation analysis [25] to reflect the correlation between the inaccuracy of cost estimation and the deviations of iteration plan execution. Cohen and Holliday [26] suggest the following rule of thumb to interpret the Pearson's coefficient: 0.19 and below is very low correlation; 0.20 to 0.39 is low correlation; 0.40 to 0.69 is modest correlation; 0.70 to 0.89 is high correlation; and 0.90 to 1 is very high correlation.

Table 5. Correlations among the relative errors

Pearson's Coef.	RE (effortEst)	RE (scheduleIter1)	RE (capabilityIter1)
RE(effortEst)	1	.208	-.137
RE(scheduleIter1)	.208	1	.328
RE(capabilityIter1)	-.137	.328	1

Table 5. shows that there is no significant correlation among the inaccuracy of effort estimation and the deviations of iteration plan execution (change in duration or capabilities implemented).

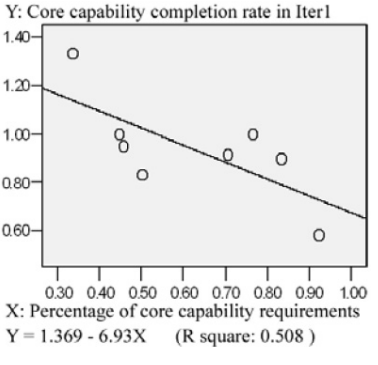


Fig. 3. Linear Regresssion

We find that the completion rate of core capability in the first iteration has significant negative correlation with the percentage of core capability requirements.

The linear regression in Fig. 3 graphically shows the correlation between the core capability completion rate in the first iteration and the percentage of core capability requirements. The relation shows, when high percentage of capability

requirements are assigned as core capability, it may become more difficulty to complete the core capability in the first iteration as planned.

An extremely high percentage of core capabilities also indicates that there may be some problems with the stakeholder expectations management or requirement prioritization, e.g., the high percentage of 92% in the case of Team 6. This team only completed 58% of the core capability requirements in the first iteration, which was the lowest completion rate among the 8 teams. The team members explained in the iteration review report that the clients required all the requirements to be implemented as core capability and it resulted in a lot of confusion amongst the team members. The students also proposed "This problem would not have arisen, if all the team members were more in touch with the distant client's need".

By analyzing how well the 8 projects executed their iteration plan, we find that:

- Even though the uncertainties of cost are high, the project plans have very small magnitude of uncertainty.
- There is no significant correlation between the inaccuracy of cost estimation and the error in iteration plan execution.
- The completion rate of core capability in the first iteration is well correlated to the percentage of core capabilities.
- The stakeholder expectations management and requirement prioritization are important for establishing a feasible project plan.

4.5.2 Clients' Evaluation of Projects

At the end of the project, the clients evaluated the delivered product with regards to five categories of criteria, which are documentation, team interaction, system preparation and testing, implementation, and overall value. The full grade is 20, and Table 6 shows the clients are satisfied with the development process and the delivered products. The product deliveries received high evaluations as every team finally implemented all the core capabilities and the planned lower-priority features.

Table 6. Client evaluation

Team	Evaluations
1	20
2	20
3	18
4	20
5	20
6	18
7	20
8	20

Table 7. Correlation analysis for client evaluation

Pearson's Coef.	Client Evaluation	RE of effort estimation	Actual Construction effort
Client Evaluation	1	.319	-.086
RE of effort estimation	.319	1	-.955
Actual Construction effort	-.086	-.955	1

The correlation analysis in Table. 7 shows that the stakeholder satisfaction correlates neither to the uncertainty of cost nor the dedicated effort by the team members. The customers' concern is not an accurate cost estimation or the dedicated development effort, but receiving their desired system capabilities within schedule.

5 Conclusions

We find in this empirical study that even though the uncertainty of cost is high, which may be due to the limited experience of cost estimation, the steep learning curve, reuse uncertainties, etc., the students can successfully deliver product on time with satisfactory quality. The project teams can accommodate to changes and complete the core capability iteration with 24%- 46% construction-phase time remaining. In addition, all the core capability features and planned lower-priority features are completed. The clients are satisfied with the development process and product delivery, and their satisfaction doesn't correlate to the uncertainty of cost or the dedicated development effort.

The SAIV process plays a critical role in coping with the cone of uncertainty by: estimating the cost and its uncertainty, creating opportunities to handle the uncertainty, enabling flexible process, and providing risk driven uncertainty mitigation strategies.

The critical practices for the 8 projects are:

- Win-Win stakeholders negotiation and effective expectation management
- Getting clients to develop and maintain prioritized requirements
- Establishing the core capability and architecting the system for ease of adding and dropping features
- Planning development increments and ensuring the on-time delivery of core capability even under pessimistic cases
- Risk driven progress monitoring and corrective action

To cope with uncertainty, agile methods also offer useful practices, e.g., embracing change, fast cycle / frequent delivery, simple design, and refactoring, and the plan-driven methods offer practices like requirements management, quantitative process management, project tracking and oversight [27]. The SAIV process is a balance of agility and discipline. Its usage on USC projects over the last 10 years and other research works [18, 28] indicate that the key practices introduced in this case study are applicable to a wide spectrum of software projects. Practitioners should choose appropriate practices to cope with the cone of uncertainty according to their development environment, and they can use risk, spiral model anchor point, Results Chain, etc. to balance the agility and discipline [27].

Managing the uncertainty of cost is an on going research, and our future work includes:

- Compare the current practices of coping with the cone of uncertainty, and provide more general guidelines.
- Investigate the sources of cost uncertainty and improve the current cost uncertainty assessment method.
- Provide tools to analyze the information of cost uncertainty, make feasibility analysis with given constraints or dependencies, and facilitate the stakeholder win-win negotiation and project planning.

References

1. Boehm B., Abts C., and Chulani S., Software Development Cost Estimation Approaches—A Survey, Annals of Software Engineering, Vol. 10, (2000) 177-205
2. Lederer A.L., and Prasad J., Nine Management Guidelines for Better Cost Estimating, Communications of the ACM, Vol. 35(2), Feb. (1992) 51 - 59
3. Jørgensen M., Evidence-Based Guidelines for Assessment of Software Development Cost Uncertainty, IEEE Transactions on Software Engineering, Vol. 31(11) (2005)
4. Little T., and Graphics L., Schedule Estimation and Uncertainty Surrounding the Cone of Uncertainty, IEEE Software, May/June (2006)
5. Boehm B., et al, Software Cost Estimation with COCOMO II, Prentice Hall (2000)
6. McConnell, S., Rapid Development: Taming Wild Software Schedules, Microsoft Press, (1996)
7. Gryphon S., Kruchten P., McConnell S., and Little T., Letters: The Cone of Uncertainty, IEEE Software, Vol. 23 (5), September/October (2006) 8-10
8. Kitchenham B., Linkman S., "Estimates, Uncertainty, and Risk", Software, May (1997)
9. Cantor M., Estimation Variance and Governance, http://www-128.ibm.com/developerworks/rational/library/mar06/cantor/
10. Brooks F.P., The Mythical Man-Month, Addison-Wesley Publishing Co., (1995)
11. Boehm, B., and Brown W.. "Mastering Rapid Delivery and Change with the SAIV Process Model," Proceedings, ESCOM2001, Apr. 2001.
12. Putnam, L., Software Life Cycle Model (SLIM), QSM, (2001) http://www.qsm.com
13. Galorath, D., SEER-SEM, Galorath, Inc., (2001) http://www.galorath.com
14. Jones, C., Knowledge PLAN, Artemis/SPR, (2001), http://www.spr.com
15. Briand L.C., Emam K., and Bomarius F., "COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment", Proceedings of the 20th international conference on Software engineering, IEEE CS Press, Washington DC, (1998) 390-399
16. Pendharkar P.C., Subramanian G.H., and Rodger J.A., "A Probabilistic Model for predicting software development Effort", IEEE Transactions on Software Engineering, IEEE Press, Piscataway, NJ, July 2005, pp. 615-624.
17. Yang D., et al, "COCOMO-U: An Extension of COCOMO II for Cost Estimation with Uncertainty", Proceedings of International Software Process Workshop and International Workshop on Software Process Simulation and Modeling, 2006, pp.132-141.
18. Boehm B., Port D., Huang L.G., Brown W., Using the Spiral Model and MBASE to Generate New Acquisition Process Models, SAIV, CAIV, and SCQAIV, Cross Talk, Jan. (2002)
19. Boehm W., and et al, Guidelines for Lean Model-Based Architecting and Software Engineering (Lean MBASE), http://greenbay.usc.edu/csci577/spring2006/site/guidelines/LeanMBASE_Guidelines_V1.5.pdf
20. Boehm, B., Anchoring the Software Process, IEEE Software, Jul. (1996) 73-82
21. Royce W.E., Software Project Management: A Unified Framework, Addison-Wesley, (1998)
22. Boehm B., The Art of Expectations Management, Computer, Jan. (2000)
23. Yourdon E., Death March, Prentice Hall, (1997)
24. Boehm, B., Software Risk Management, IEEE Computer Society Press, 1989
25. Bryman A., and Cramer D., Quantitative Data Analysis with SPSS, Routledge, (2005)
26. Cohen L., and Holliday M., Statistics for Social Scientists, London:Harper & Row, (1982)
27. Boehm, B. and Turner, R., Balancing Agility and Discipline: A Guide for the Perplexed, Addison-Wesley Professional, (2003)
28. Boehm, B. Port, D. and Jain, A., Achieving CMMI Level 5 Improvements with MBASE and the CeBASE Method, Cross Talk, May. (2002)

Effects of Architecture and Technical Development Process on Micro-process

Liming Zhu, Ross Jeffery, Mark Staples, Ming Huo, and Tu Tak Tran

NICTA, Australian Technology Park, Eveleigh, NSW, Australia
School of Computer Science and Engineering, University of New South Wales, Australia
{Liming.Zhu, Ross.Jeffery, Mark.Staples, Ming.Huo, TuTak.Tran}@nicta.com.au

Abstract. Current software development methodologies (such as agile and RUP) are largely management-centred, macro-process life-cycle models. While they may include some fine-grained micro-process development practices, they usually provide little concrete guidance on appropriate micro-process level day-to-day development activities. The major factors that affect such micro-process activities are not well understood. We propose that software architecture and technical development processes are two major factors. We describe how these two factors affect micro-process activities. We validate our claim by mining micro-processes from two commercial projects and investigating relationships with software architecture and technical development processes.

Keywords: micro-process, macro-process, architecture, process mining.

1 Introduction

The increasing demands imposed on software-intensive systems require more rigorous engineering and management of integration among the products being developed, the technology being used and software development processes [8, 16].

In process engineering, a macro-process describes the high-level behaviours of process while micro-process describes the fine-grained internal workings of processes [15]. Current software development methodologies (such as Agile and RUP) are largely project management-centred, macro-process life cycle models. While they may include some fine-grained micro-process development practices, they usually provide little concrete guidance on appropriate micro-process level day-to-day development activities. This has hindered the wide adoption of rigorous development processes by developers because they do not usually find macro-processes useful for their immediate needs at a micro-level. This gap between macro-processes and micro-processes has been recognized previously [15], [13]. In this paper, we suggest that software architecture and technical development are two major factors that affect fine-grained micro-processes:

1) Software architecture is important for decomposing a system into functional modules. This can be used to support task allocation when planning development.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 49–60, 2007.

However, architecture has other influences on development. We observe that module dependency, degree of such dependency, and architectural refactoring play major roles for micro-processes.

2) A technical development process is a development process for a particular technology, such as XML, service orientation, object orientation or a programming language. Technical development processes are composed of technical steps, best practices, and checklists for different types of technology-specific components at different stages. However, these are not necessary aligned with the normal phases of software development or project iterations. Companies consider technical development processes an important addition to their macro-process, bringing competitive advantage [10]. Some technologies, such as programming languages and object orientation, have not been considered to have a major impact on the normal flow of a development process, but have rather been associated with development efficiency and product quality. We observe that some new technologies, such as XML and service orientation, do have major effects on the development process. This is largely because such technologies are not confined to the design and implementation phases. XML has been used to directly define business level requirements and communication standards. Service governance has become a normal business activity, since it gives direct control over service development beyond the phases of initial development.

We have conducted micro-level process mining in two commercial projects. After excluding the "normal" micro-process activities, such as detailed designing, implementing, testing a single module, integration and system testing of a particular sub-system, we find most of the micro-processes and activities that are significant in terms of effort and recurring patterns are indeed affected by software architecture and technical development processes. These effects are reflected in micro-process activities themselves, cost of the micro-process activities and recurring process patterns. We use effort and recurring patterns as indicators of importance among hundreds of activities. As later revealed in structured interviews, explicitly considering these factors may increase the efficiency of the process and the quality of the project and can give more concrete guidance for developers at a micro-level.

The main contribution of this work is to provide a better understanding of the major factors that can influence micro-processes. It will be valuable in offering further assistance to actively planning micro-processes and bridging the gap between the macro-process and micro-process.

2 Related Work

Connections between macro-processes and micro-processes are usually created through organization- or project-specific process tailoring, which can be either top down or bottom up [4, 7, 14, 18].

In the top down approach, a macro-process is instantiated by considering both project characteristics and organizational factors. It has been recognized that the main problem here is the low level of reuse during tailoring; additional context information used in the tailoring is not systematically reused [7]. Moreover, the resulting process is not fine-grained enough to provide concrete guidance to developers. The tailoring

process is usually about making or selecting optional process elements, or modifying variant process elements [1]. Since such process elements usually do not include technology- and product-specific information, the resulting process is not fine-grained and "micro" enough. This paper addresses these challenges by aiming to develop a new understanding of major factors in fine-grained tailoring, such as product architecture and technology-specific development processes.

In the bottom-up approach, an organization usually has a large number of process assets. However, the composibility of these assets is often poor and the suitability of a composed process hard to verify [7]. The research in this paper does not address this issue directly; however, we believe that explicitly considering the additional factors has the potential to significantly improve the composibility of these assets and to provide further criteria for assessing the suitability of the composed process.

Both approaches are used to produce development processes for a particular type of project or technology, such as embedded systems [10] and COTS [12]. However, both the top-down and bottom-up approaches are not "micro" enough and the resulting processes are hard to verify.

Traditionally, software architecture affects the software development process in two different ways. First, architecture design and evaluation methods constitute part of the whole life-cycle macro-process. Currently there is active research into streamlining these methods by using additional phases [11]. Such streamlining does consider the architecture itself, but only the method for producing it. Secondly, architecture is used to decompose systems into functional modules, which can then be used in task allocation. This reflects some finer-grained processes, but is confined to the early planning phase and is fairly simplistic, so that the full potential of the architecture is not exploited. In this study, we look into taking more advantage of architecture in the development process.

We believe the missing link between micro-process and macro-process can be further bridged by considering the architecture and technical development processes.

3 Effects of Architecture and Technical Development Process on Micro-process

3.1 A Framework of Factors That Affect Micro-process

We first propose a framework that includes all the factors that may have influence on micro-process, as shown in Figure 1. Some of them, such as macro-process methodologies, are understood at least in terms of the existence of their influence, although the exact nature and size of their influence is not completely understood. Our recent work has explored the existence of other factors, such as the effect of business processes and business data on software development processes [9].

1. **Macro-process methodologies:** As mentioned earlier, the generic software development process is the foundation for any fine-grained micro process through top-down or bottom-up process tailoring.
2. **Functional Requirements:** Functional requirements, especially functional scoping in iteration planning define what needs to be developed at a fine-grained level. However, they provide information only on 'what' to do but not 'how' to do it.

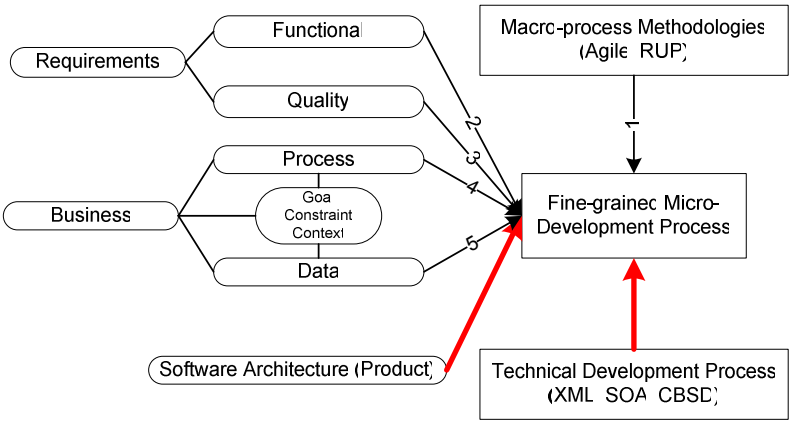


Fig. 1. Factors that affect fine-grained micro-process

3. **Quality requirements:** Quality requirements (or non-functional requirements) guide additional quality-related processes. For example, a high-reliability system may require more testing to be done, or the use of a particular testing technique.
4. **Business Process:** The need for mechanisms to support the analysis and tracing of relationships between the business process and the software process is discussed in [9]. It is critical for instantiating elements of that business process in software.
5. **Business Data:** Industries have developed electronic business data "standards" to improve business efficiency and business-to-business interoperability. Such standards inevitably have to map to technology infrastructures such as service-oriented architecture. The design decisions embodied in the business data standards often affect the development process at a fine-grained level.

In this paper, we propose two additional factors: software architecture and the technical development process.

3.2 Software Architecture Factor

Architectures provide a wide range of information that can benefit micro-process planning and monitoring. Architectural models of inter-module dependency are particularly relevant for micro-process. We propose that architectural dependency models will influence micro-processes in ways including but not limited to the following:

Claim 1: A micro-process activity usually concerns multiple inter-dependent architecture modules at the same time.
For example:

- All modules involved in a single micro-process activity when designing, implementing and testing a module with high dependency.

This is important because coupling information between modules has not been explicitly used in fine-grained process planning. Tightly coupled modules may be only suitable for a close team to develop while loosely coupled modules can be developed in a distributed and parallel manner.

Claim 2: The cost of a micro-process activity on a module will be affected by the architectural dependency characteristics of the module.
Examples of this include:

- Cost of code understanding may be high when the module has high dependency.
- Cost of integration testing will be higher between highly cross-dependent modules

This is important because cost estimation models usually only concern with sizes of features and function points. Dependencies between functions are not recognised as a cost factor. Some generic complexity metrics have been used in cost estimation but are not useful for fine-grained activity costing.
Claim 3: Micro-process patterns can often be better explained by the influence of the various development stages of different inter-dependent modules rather than by macro-process phases.
For example:

- During stub creation for unit and integration testing when one unit relies on the existence of another yet-to-be developed module.

This is important because increasingly, large-scale software is developed in a concurrent manner. The nature of interactions between these parallel development processes is important, and architecture is an important factor.

We realize that process planning usually starts at an early stage, sometimes even before a contract is awarded and the architecture is known. However, we should have a plan about how fine-grained processes will respond to architecture definitions.

3.3 Technical Development Process Factor

We propose that a technical development process will affect micro-processes by imposing technology-specific activities, sequences and best practices through process interactions with macro-processes:

Claim 1: A micro-process activity usually "is" an activity described by a technical development process rather than an instantiated or tailored macro-process activity. For example:

- XML schema development processes are technical processes and are also micro-processes.

This is important because limitations exist in instantiating or parameterizing macro-processes. A complete top-down approach will never be able to cover the richness of technical development process activities.
Claim 2: The cost of a micro-process activity will be affected by its technical characteristics rather than macro-process activities. For example:

- Designing, implementing and testing unique types of technology modules. In the case of XML development, data update module, up translate/down translate and cross-translate modules are developed very differently with different cost implications.

This is important because the technical characteristics of an activity have not been used in cost estimation models. However, they may have definitive cost implications.

Claim 3: Micro-process patterns are often determined directly by a technical development process or interactions between a technical development process and a macro-process rather than a tailored macro-process. For example:

- Designing schema in XML development has a unique sequence of activities. It little resembles traditional macro-processes.
- A technical development process requires compliance with certain best practices. A micro-process pattern emerges from interactions between the compliance processes and the normal development process.

This is important because most process patterns are currently related to macro-process methodology. Factors affecting micro-process patterns need to be investigated and eventually used in process planning.

4 Case Study

The primary goal of this case study is to validate the existence of strong influence of architecture and technical development process in micro-process, through process mining, architecture reconstruction and structured interviews. Using these factors and measuring their effects is beyond the scope of this work.

4.1 Project Selection

The details of the projects will be described in section 4.3 and 4.4, along with the analysis. The general reasons for selecting these two projects are as follows:

- These are two typical and representative projects within the company, not pilot projects for trialling a new technology, nor instrumented with any particular process measurement techniques other than what the company is already doing.
- Time sheets for fine-grained process measurement (in addition to billing) are recorded for all projects. They directly record individual micro-level activities, and also enable us to mine process patterns from recurring sequences of activities.
- The company has explicitly used software or system architecture in their process planning. However, only functional module decomposition is used. In one project, we helped them reconstruct additional architectural views to investigate the influence of architecture on micro-processes. The influence of technical development processes (concerning Java) is also evident in this project.
- The company considers that their major process competitive advantage is their technical development process, in this case related to XML technologies. These processes are actively used, but are not systematically integrated with their macro-process definition. In the XML project, we mainly investigate the technical development processes, although the architecture factor also has some influence.

To build their competitive advantage, the company has focussed on their micro-processes in XML development. Historically, these are materialized in technical checklists, best practices, metrics and governance, loosely grouped around macro processes in an EPG system. However, technical development processes are not used in their process planning and monitoring.

4.2 Data Source and Techniques

The evidence for this case study is collected from multiple sources to avoid any single-source bias. The data includes source code (for architecture reconstruction), documentation, time sheets (for micro-process mining) and interviews.

Statistical Mining
To understand how a micro-process is affected by certain factors, we need information about activities at the micro-level. The time sheets recorded at the company directly provide us with this information. For example:

	A	B	C	D	E	F	G	H
1	Project	TaskNo	Employee	Hours	Comments	module	epgType	activityID
2	2307	41	56	7	testing and integration of the profile manager	screeningTool	DD/CODE	253
3	2307	41	56	1.5	design and coding of the User and UserHome classes	screeningTool	DD/CODE	254
4	2307	41	56	4	Modify code accordingly to design changes, wrote inpu	screeningTool	DD/CODE	255

From this data, we can extract recurring sequences of activities as micro-process patterns. Our previous work has successfully performed such mining [5]. Essentially, we reconstruct a recurring sequence of activities as a micro-process pattern.

Out of all the activities and recurring sequences of activities that we mined, we excluded the "normal" micro-process activities, such as detailed designing, implementing, testing a single module, integration and system testing of a particular sub-system. We then selected the most effort-wise significant micro-processes and activities and analysed their relationship with architecture and the technical development process. For the mining of the two projects, please refer to [6]. This paper only includes a subset of all the mined activities and micro-process patterns that are relevant to architecture and technical development process.

Reverse Architecting from Source Code
To understand architectural influence on micro-process, we need relevant architecture views of a system. We associate these views with the mined micro-process. Although very high-level architecture views exist for both projects, it was still necessary to reconstruct views that reflected dependency and degree of dependency between modules.

We used two tools, JDepend and Structural Analysis for Java (SA4J), to conduct an architecture reconstruction. The aim was to reconstruct the dependency views between components and see if the degree of dependency influenced micro-process activities. The following is a brief summary of the each tool's capabilities related to dependency:

JDepend - JDepend is an open source tool which provides design metrics beyond traditional class-level OO metrics by looking at cross-module quantitative inter-dependency. Among the many metrics supported, the following proved to be particularly relevant:

- **Afferent/Efferent Couplings of Modules:** They indicate the outgoing and incoming dependency degree for a particular module.
- **Instability of Modules:** This is an indicator of module's relative *resilience to change.*

Structural Analysis For Java (SA4J) - SA4J is a tool from IBM for analysing Java dependencies. The uniqueness of this tool is its transitive impact analysis and skeleton diagrams for indirect dependency. They are different from the standard coupling measurement and appear to be more useful in micro-process. The following metrics are the most relevant ones:

- **Global Butterfly:** If the module is changed, it may affect many other components.
- **Global Breakable:** The module is often affected if anything in the system is changed.
- **Global Hub:** The module is both a global butterfly and a global breakable.
- **Skeleton:** This layered view of the system is constructed by putting modules with no dependencies on the bottom layer. Modules that are dependent on the lowest layer appear in the above layer, and so on. In this view, a stable system should have a pyramid shape. An unstable system may look like an upside down pyramid.

Structured Interviews
Structured interviews were used to validate our process mining findings, to avoid any single source bias, and to get further insights. We presented micro-process patterns that were not aligned with macro-processes documented and used through an electronic process guidance system in the company. In the interview we elicited causes of these deviations.

4.3 Project A: The Finance Project

Project Description
This project demonstrates the influence of architecture on micro-process. The code base is an integral part of a series of financial products that are written in Java. The system generates financial data which can be accessed by subscribers to the service. The financial data produced by the application needs to be generated in an extremely flexible manner so that it can be easily tailored to each subscriber's needs. The content delivery mechanism exploits XML formats and XSLT transformations to render tailored views. The code has been refactored on several occasions to progressively improve the design quality and functionality. It has 50k lines of code with 296 classes/interfaces in 27 packages. It took 1600 man hours to produce.

Reconstructed Architecture
Figure 2 shows the reconstructed module dependency view and skeleton view. Drilling down the dependency line can reveal the degree of dependency. The skeleton view also reflects indirect dependency. The skeleton diagram shown here particularly depicts the dependencies of the *util* module. The grey squares represent classes and interfaces in the whole system. The red (black) [1] squares represent the classes/ interfaces in the *util* package. The orange (light grey) squares represent the classes/interfaces that depend on the *util* (red) package. For details of the reconstruction, please refer to [3].

[1] Colors in braces are for black and white prints of this document.

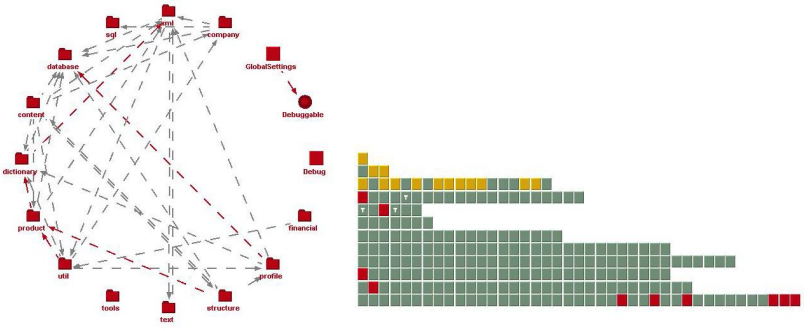


Fig. 2. Module Dependency Diagram and Skeleton Diagram

The tool returned a ranked list of global breakables, global butterflies and global hubs. The top 3 modules for each of them are:

- Global Breakable: Content.ViewPortfolio, Content.ViewPortfolioProfile
- Global Butterfly: XMLUtils, Debuggable, XMLSerializable
- Global Hub: HomeFactory, LicenceHome, HomeManufactruable

These highly ranked modules are used in cross-checking with modules involved in unusual micro-process activities and cost.

Data Analysis

According to our criteria for mining, we selected the most effect-wise significant activities. Among the most costly activities (more than 30 man-hours to complete) and recurring process patterns, the following nine have a direct relationship with architecture. Each of them supports claims about architecture influence on micro-processes in section 3.2. We confirmed these findings in follow-on interviews.

- Conducting integration testing between two highly dependent modules [claim 2]
- Designing a specific module when understanding of a highly dependent module is needed [claim 2]
- Understanding and analysing existing modules. These modules are global hubs. [claim 1]
- Refactoring at architectural level, including creating new modules that were not previously used for task allocation [claim 1]
- Cleaning up code for integration of two highly dependent modules [claim 1]
- Making certain part of the UI "smart" (dynamic rather than hand-coded) [claim 1]
- Working on a logical group of dependent modules together [claim 1]
- Producing cross-phase process patterns between design and implementation when detailed design is not available [claim 3]
- Extending existing open-source modules when there is a high dependency between modules [claim 3]

4.4 Project B: The XML Project

Project Description
This project demonstrates the influence of technical development processes. The project developed a Java tool with XML processing capabilities. The project took 2900 man-hours to complete. We did not have access to the source code, and so the architecture recovery tools could not be used, but a high-level system architecture with functional decomposition was made available, and was matched against the modules recorded in the time sheet records.

XML Technical Development Process
The following is a high-level view of the XML technical development process the company has used. For each of the activities and sub-activities there are associated best practices, activities and checklists. The process is largely derived from previous work on XML process models [17].

- Develop (Analyze/Design/Implement/Test) data capture module
  - Create Data
  - Update Data
- Develop data query modules
  - Server-side query module
  - Client-side query module
- Develop data transformation module
  - Up-translate module: transform non-XML documents to XML documents
  - Down-translate module: transform XML documents to non-XML documents
  - Cross-translate module: transform XML documents to XML documents
- Develop intermediary schemas.

Data Analysis
The criteria for the following activities or sequence of activities are the same as before. Among the most costly activities (more than 30 man-hours to complete) and micro-process patterns, the following seven have a direct relationship with the technical development process. Each of them supports claims about technical development processes in section 3.3. We confirmed these findings in follow-on interviews.

- Create and code intermediary XSD [claim 1 and 2]
- Technology investigation and learning (even for the developers, who are experienced XML developers, understanding certain new trends and best practices took a significant amount of time) [claim 1 and 2]
- Set up the technology environment
- Develop a data update module [claim 1 and 2]
- Develop a up-translate module [claim 1 and 2]
- Develop a client side query module [claim 1 and 2]
- Schema re-design during implementation [claim 3]
- Requirement negotiation during development [claim 3]

For example, the cost effects had a distinctive pattern: developing the cross-translate component takes the least amount of time, while developing the up-translate component takes the most amount of time.

5 Discussion

A number of cross macro-process phase activities were found during the mining. They do not follow the macro-process, even considering iterations. This strongly validates the observations on programming rework [2] which sees cross-phase rework are much more complicated than simple redo and iteration. However, we have not yet investigated the factors affecting rework.

In our previous work [6], we thought discrepancies were due to enactment problems or that the macro-process needed to be changed. However, after further study, we have found that it is due to the nature of differences between macro-process and micro-process. Iterations on a macro-process level involve finishing one phase then going to the next one, usually over a period of days, if not weeks. At a more detailed level, we have observed that developers switch between phases in a unusual order for many reasons, including:

- Highly-coupled modules at different development stages
- Requirements maturity for a particular module
- Unique features of some technical development process
- Interactions among different processes (such as generic processes, technical development processes, quality assurance and compliant processes) at different abstraction levels

This is especially evident in the Project B, due to the nature of XML development, which is more about understanding the requirements and making tradeoffs in schema development (sometimes this is even done by the customer), and incorporating technical best practices and strong quality assurance to the normal development in a parallel development fashion.

We realize that there are a number of limitations of this study:

1) There are actually a large number of factors, as identified in Figure 1. We only considered two factors in this paper due to limits to the scope of our investigation. We realize that certain factors may play a more dominant role than others.

2) Using these factors actively in process planning may be different to after-event observation. We are planning a case study on actively investigating the use of these factors in process planning.

6 Conclusion and Future Work

Across the industry, more sophisticated process engineering is needed. This requires increased understanding of fine-grained micro-process and filling the gap between macro- and micro-processes. In this paper, we investigated two major factors: software architecture and the technical development process. We are currently planning a new full-scale case study on using these factors in continuous micro-process planning and monitoring. We are also developing a technical

governance framework to be used by both management and developers for communicating effectively.

References

[1] J. Bhuta, B. Boehm, and S. Meyers, *Process Elements: Components of Software Process Architectures*, 2005.

[2] A. Cass and L. Osterweil, "Programming Rework in Software Processes," Department of Computer Science, University of Massachusetts UM-CS-2002-025, 2002.

[3] I. Gorton and L. Zhu, "Tool Support for Just-in-Time Architecture Reconstruction and Evaluation: An Experience Report," in *27th International Conference on Software Engineering (ICSE)*, 2005, pp. 514 - 523.

[4] G. K. Hanssen, H. Westerheim, and F. O. Bjornson, "Tailoring RUP to a Defined Project Type: A Case Study," in *Product Focused Software Process Improvement (PROFES)*, 2005, pp. 314-327.

[5] M. Huo, H. Zhang, and R. Jeffery, "An exploratory study of process enactment as input to software process improvement," in *International Workshop on Software Quality at International Conference on Software Engineering (ICSE)*, Shanghai, 2006.

[6] M. Huo, H. Zhang, and R. Jeffery, "A Systematic Approach to Process Enactment Analysis as Input to Software Process Improvement or Tailoring," in *Asia-Pacific Software Engineering Conference (APSEC)*, 2006.

[7] O. Jaufman and J. Munch, "Acquisition of a Project-Specific Process," in *Product Focused Software Process Improvement (PROFES)*, 2005, pp. 328-342.

[8] R. Jeffery, "Achieving Software Development Performance Improvement Through Process Change," in *Software Process Workshop (SPW)*, Beijing, China, 2005, pp. 43-53.

[9] R. Jeffery, "Exploring the Business Process-Software Process Relationship," in *Software Process Workshop/Workshop on Software Process Simulation and Modeling (SPW/ProSim)*, 2006.

[10] E. Johansson, J. Nedstam, F. Wartenberg, and M. Host, *A Qualitative Methodology for Tailoring SPE Activities in Embedded Platform Development*, 2005.

[11] R. Kazman, H. P. In, and H.-M. Chen, "From requirements negotiation to software architecture decisions," *Information and Software Technology,* vol. 47, pp. 511-520, 2005.

[12] M. Morisio, C. B. Seaman, V. R. Basili, A. T. Parra, S. E. Kraft, and S. E. Condon, "COTS-based software development: Processes and open issues," *Journal of Systems and Software,* vol. 61, pp. 189-199, 2002.

[13] J. Münch, "Transformation-based Creation of Custom-tailored Software Process Models," in *International Workshop on Software Process Simulation and Modeling (ProSim)*, Scotland, UK, 2004.

[14] A. Ocampo, F. Bella, and J. Münch, "Software Process Commonality Analysis," in *International Workshop on Software Process Simulation and Modeling (ProSim)*, Scotland, UK, 2004.

[15] L. Osterweil, "Unifying Microprocess and Macroprocess Research," in *Software Process Workshop (SPW)*, 2005, pp. 68-74.

[16] D. Rombach, "Integrated Software Process and Product Lines," in *International Software Process Workshop (SPW)*, 2005, pp. 83-90.

[17] T. T. Tran, "An Interim Report of XML Process Models," School of Computer Science and Engineering, University of New South Wales UNSW-CSE-TR-0613, 2006.

[18] H. Washizaki, "Building Software Process Line Architectures from Bottom Up," in *Product-Focused Software Process Improvement (PROFES)*, 2006, pp. 415-421.

Comparative Experiences with Electronic Process Guide Generator Tools

Monvarath Phongpaibul, Supannika Koolmanojwong, Alexander Lam, and Barry Boehm

Center for Systems and Software Engineering
University of Southern California
Los Angeles, CA 90089-0781
{phongpai,koolmano,alexankl,boehm}@usc.edu

Abstract. The primary objective of all software engineering courses is to help students learn how to develop successful software systems with good software engineering practices. Various tools and guidelines are used to assist students to gain the knowledge as much as possible. USC's Center for Systems and Software Engineering (CSSE) has found that the keystone course in learning software engineering is a year-long real-client team project course. Over the last ten years, CSSE has evolved a set of guidelines for the course, and has experimented with early tests for creating electronic process guides for MBASE (Model-Based (Systems) Architecting and Software Engineering) Guidelines using Spearmint/EPG. Currently, CSSE has been developing and experimenting with Eclipse Process Framework's (EPF) to situate the LeanMBASE Guidelines. This paper reports our comparative experiences of using the earlier and current tools to generate the electronic process guidelines. In our analysis, we used the objectives defined by Humphrey and Kellner[17] to compare the process tools. The evaluation identifies some research challenges and areas for future research work.

Keywords: Process modeling tools, Electronic process guide generator tools.

1 Introduction

In the keystone two-semester team project graduate software engineering course sequence CS577ab [28] at USC, students learn through experience how to use good software engineering practices to develop software systems from the Inception Phase to the Transition Phase, all within a 24-week schedule. Students in the class form 6-person teams to develop real-client software system projects. From 1998 – 2005, students used Model-Based (Systems) Architecting and Software Engineering (MBASE) [23] method and MBASE Guidelines as the methodology to develop their systems. However, in 2005, the LeanMBASE replaced the MBASE in order to reduce the documentation workload and to fit with small-sized and limited scheduled projects by focusing on high-value activities. We are now preparing to transit the paper-based LeanMBASE guidelines into an electronic modeling framework.

In most project course situations, students use a set of paper-based guidelines outlining what they need to describe and do to build a system. But paper-based

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 61–72, 2007.

guidelines have their limitations as they cannot be used effectively to show how a software process works. Students just learn what they are suppose to document, but won't learn how a process should be applied and tailored to a project. Other benefits of electronic-based guidelines include easy access for all users, making updated information immediately available to all users, providing links to templates and examples, and easier navigation of the guidelines [2],[5],[19],[24]. Thus, we have explored using process modeling tools to teach students what a software process is. The end result of these programs is a website that students can view to learn about the software process. The benefit of using such tools is that it shows what the relationship is between actors, tasks, guidance tools, and work products of a software process. In addition, the tools allow for easier tailoring of software processes.

In this paper, we compare our experiences in using two electronic process guide generator tools: Spearmint/Electronic Process Guide (EPG) [4] and Eclipse Process Framework (EPF) using the criteria defined by Humphrey and Kellner [17]. The Spearmint tool was used to model the earlier MBASE Guidelines and was used in the software engineering classes between 2000 and 2003. Currently, we are investigating how to use EPF to model the LeanMBASE Guidelines.

This paper is organized into 6 sections. Following this introduction, section 2.1 introduces MBASE, LeanMBASE and OpenUP including their similarities and differences. In section 2.2, we present the overview of two process modeling tools that are Spearmint/EPG and EPF Composer. Section 3 describes the experience comparison between modeling MBASE Guidelines into Spearmint/EPG and modeling LeanMBASE Guidelines into EPF Composer. The future challenges are discussed in section 4 followed by conclusion in section 5 and references in section 6.

2 Background and Overview

2.1 Process Guideline Overview

2.1.1 MBASE and MBASE Guidelines

Model-Based (System) Architecting and Software Engineering (MBASE) is a set of guidelines that explain software engineering principles and techniques that is used for developing software projects[23]. MBASE shares many aspects with the Rational Unified Process (RUP) [22], including the use of the Unified Modeling Language (UML) [27] and the spiral model anchor point milestones [6].

From Fall 1998 to Spring 2005, MBASE had been used as guidelines for students in the software engineering courses to develop real-client service projects. The MBASE guidelines mainly specify the content, format and templates for 19 types of project artifacts in various phases. [11]. The MBASE Guidelines also cover several software tools that the students should use for various activities, such as the Easy WinWin negotiation tool [9], effort reporting tool, risk identification tool, and USC COCOMO II [8] and COCOTS [1] software cost modelling tools. To fit with the class's nature, stakeholders can be primarily categorized as client, maintainer, end user, teaching staff, and development team. The development team is composed of 5-6 on-campus students and 2 Independent Verification and Validation (IV&V) students. The IV&V people are off-campus students who act as independent peer reviewer and quality assurance agents.

A major learning and grading artefact in the course is a reflective critique of the student's project experience. From the critiques of using the MBASE Guidelines in our software engineering course, we found that with the limited schedule and the small project size, the development teams spent increasingly too much time in documenting the project artifacts. As a result, we reduced the project document size by getting rid of unnecessary and duplicated information for light-weight projects. In Fall 2005, the 260 pages of MBASE Guidelines were replaced with 90 pages of LeanMBASE Guidelines.

2.1.2 LeanMBASE Guidelines

LeanMBASE is the light-weight version of MBASE, which inherits all core approaches from MBASE such as milestones, iterative refinement, using the risk-driven, Win–Win Spiral approach [6], all critical activities such as requirement negotiation, risk identification and mitigation, project planning, business case analysis, use-case driven process, risk-driven prototyping [11], and same set but leaner version of project artifacts.

MBASE and LeanMBASE contain similar set of artifacts, but in order to enhance traceability, LeanMBASE avoids all duplication and makes the LeanMBASE more customizable based on project needs. LeanMBASE introduces a new artifact, which acts as an artifact package header that contains status of the package, glossary, traceability matrix and document tailoring information.

LeanMBASE has been used in software engineering classes at USC since Fall 2005. The result of effort report and document analysis has shown that LeanMBASE remarkably reduced document size and time spent in their project development.

2.1.3 Open Unified Process (OpenUP)

"OpenUP" (Open Unified Process) is a revision of the iterative Rational Unified Process for software development process that is minimal, complete, and extensible. The process is minimal in that only fundamental content is included. The process is complete in that it can be manifested as an entire process to build a system. The process is extensible in that it can be used as a foundation on which process content can be added or tailored as needed [26].

OpenUP is similar to LeanMBASE in the sense that it is leaning toward agile approach. As with LeanMBASE, OpenUP is a combination of best practices from both plan-driven and agile methodologies. OpenUP not only has the essential characteristics of a Unified Process, which includes iterative development, use cases and scenarios driving development, risk management, and an architecture-centric approach [22], but it also contains the agile concepts such as customer collaboration, test-first design, continuous integration, time-boxed iteration, scrum meeting and refactoring. By combining the agile approaches into its process, OpenUP considers itself as an agile process rather than a lightweight process [26].

2.2 Electronic Process Guide (EPG) Generator Tool Overview

Many software processes are complex. It is hard for both process engineers to capture all the process guidelines and process performers to follow these guidelines. In order to make the software process easier to follow, process engineers can use the formal

language called process definition languages (PDL) to specify the process that have to be done or they can use tools to help generate the electronic process guide (EPG).

In attempt to model MBASE/LeanMBASE, we chose EPG generator tools over PDLs because of the limited capabilities of PDLs to represent a non-sequential process like MBASE/LeanMBASE. For PDLs, the pre-condition, post-condition, and the sequence of tasks have to be specified in advance. In MBASE/LeanMBASE, there is no required sequence of tasks users have to follow. MBASE is risk-driven and in some cases, it would be appropriate based on risks to perform the tasks in a different order as suggested by the guidelines.

2.2.1 Spearmint/EPG (Electronic Process Guide)

Spearmint is an integrated environment for modeling, analyzing, and measuring process [3]. The objective of Spearmint is to improve understanding and communicating of software process. It is the tool for process engineers to model their process and convert the process to an electronic version called Electronic Process Guide (EPG) [25].

Spearmint provides four different views of a process model for process engineer: product flow view, properties view, decomposition view and textual view [4]. Each view is designed to model the different perspective of a process. For example, the product flow view is the graphical view, which contains the relationship between artifacts, activities, roles and tools. The properties view is for capturing the detail of a process model element such as agent/role, activity, artifact and tool. All of the process model elements are kept as objects in the database and to be generated to EPG by the EPG generator. EPG generator generates a set of html files that can be accessed through web browser [2],[3].

EPG is composed of a project main page (Figure 1) and an individual page for each element. The project main page (Figure 1) provides lists of all the activities, artifacts, agents (roles) and tools. All the process model elements are displayed as hyperlink to its individual page [4].

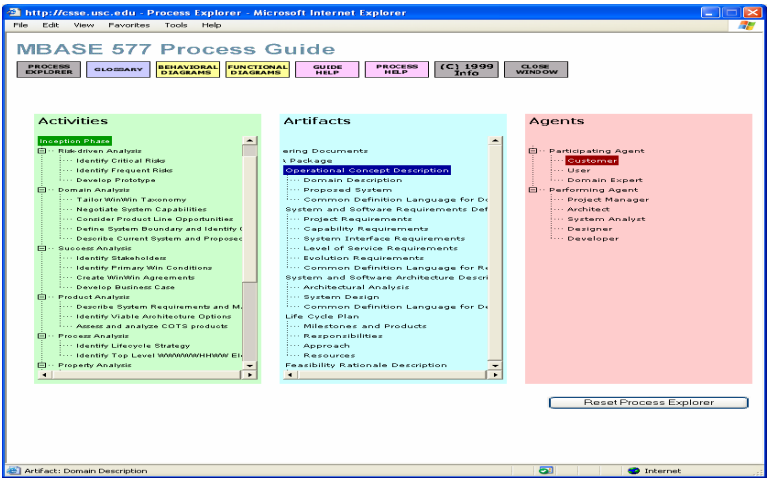


Fig. 1. A project main page of MBASE 577 process guide

2.2.2 EPF Composer (EPFC)

EPF Composer (EPFC) is a process-management tool platform [16]. It also provides the extensible process framework (called OpenUP as described in 2.1.3) for authoring and tailoring. There are two main objectives of EPF. The first objective is to provide a central knowledge base to the process performers. All of the process elements are stored in the objects called method content. Method content is where the method elements (roles, tasks, artifacts and guidance) are defined regardless of how they will be used in the process [15].

The second objective is to provide a tool for process engineers to select, tailor and assemble their process from the method content. Since EPF stores the process content separately from the process, the process engineer can create a new process by configuring the pre-defined content in the method content area. As a result, process engineers can create different processes for different types of project using the pre-defined content in the method content library [15].

The same as Spearmint, EPF Composer automatically converts the process content into electronic process (html files). The process performers can access the electronic process via the internet or intranet.

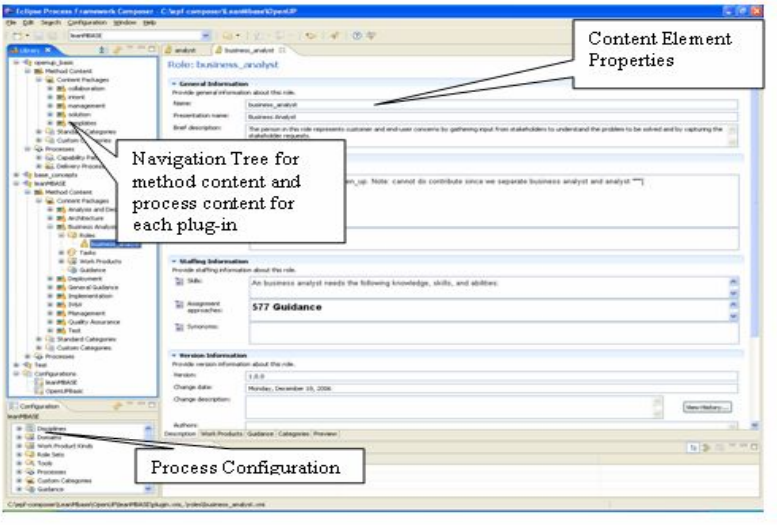


Fig. 2. An example of EPF Composer Interface

3 EPG Generator Tool Comparison

In our analysis, we used the following objectives to compare the process tools as defined by Humphrey and Kellner. The first objective is that it should effectively communicate the process to the end-users. The second objective is that is should allow for easy reuse of an existing process since process development can be time-consuming. The third objective is that the tool should allow the process to easily

evolve. Finally, the fourth objective is that the tool should help with process management allowing users to measure project status against the process. [17]

For this comparison, we evaluated the Communicate Process objective in several ways. We evaluated the representation of the process elements and the relationship between the elements each tool used. Furthermore, we examined how a software process was represented and communicated to process users. For the Process Reuse objective, we examined how each tool supported reusable process elements and content. For the Process Evolution objective, we analyzed how easy it was to tailor an existing process and how changes to the process were stored for reference. In addition, we looked at how easy it was evolve a process for each tool. For the Process Management objective, we did not look at this, as both tools did not allow for project tracking. However, it is possible in EPF to export a Work Breakdown Structure to a project-tracking tool such as Microsoft Project. In addition to these 4 objectives, we analyzed ease of use for each tool from a process engineer point of view.

3.1 Representation of Process Elements (Roles, Tasks, Artifacts, and Tools/Guidance)

Both EPG and EPFC have the functionality of modeling process elements such as roles/agents, tasks/activities, artifacts, tools, templates and properties. However, EPFC offers more representation for guidance such as checklist, example, guideline, template, and tool mentor. These extra representations give the process engineer more alternatives to give guidance and give the process performers multiple types of help. For example in our LeanMBASE plug-in, we can include the example of an architecture model or example of a benefit chain model. We can attach the risk identification checklist into the "Identify Risk" task, which is one task in the LeanMBASE plug-in for the students to make sure all the major risks are identified. We also provide the checklist for the completion of the milestone, which the students can easily access to check that they have accomplished the goals for the current milestone and are ready to begin the next one.

Besides offering the extra representations, EPFC also provides more properties for each process element than EPG. For example, for the role element, EPG only describes the artifacts and activities that the role is responsible for. EPFC provides the attribute for skills of the role and the assignment approaches to guide who should be assigned to this role. For the task elements, EPFC has an attribute by which the process engineers can enter the detail steps to perform a task. The process engineers can configure these steps to indicate which iteration to perform.

3.2 Representation of Relationship Between the Process Elements

One main purpose of modeling process is to connect the relationship between the process elements. In EPG, the relationship of an element is only described textually. There is no visual representation to illustrate this relationship. The process performers need to browse to an individual page to search for the relationship. On the other hand, EPFC provides the visual diagram to present the relationship between role, tasks and work product. Figure 3 shows an example of a visual diagram illustrating the project manager and his/her responsibilities.

Fig. 3. Example of a visual diagram representing the project manager and his/her responsibility

3.3 Representation of a Process

The main advantage of EPFC over EPG is that EPF provides the process representation via work breakdown structure (WBS) and activity diagram [27]. EPG only generates the behavior and functional diagram for the process. EPFC uses the concept of nested activities to define the process. The activity in WBS can breakdown into sub-activities. Processes that are defined in EPFC are composed of a set of activities, which in turn can be composed of another set of activities. For example, in Figure 4, the Inception phase in LeanMBASE process lifecycle is the set of initial project, manage iteration, manage requirements and determine architectural feasibility activities. In the manage requirements, we can define "requirement negation" activity as a nested activity.

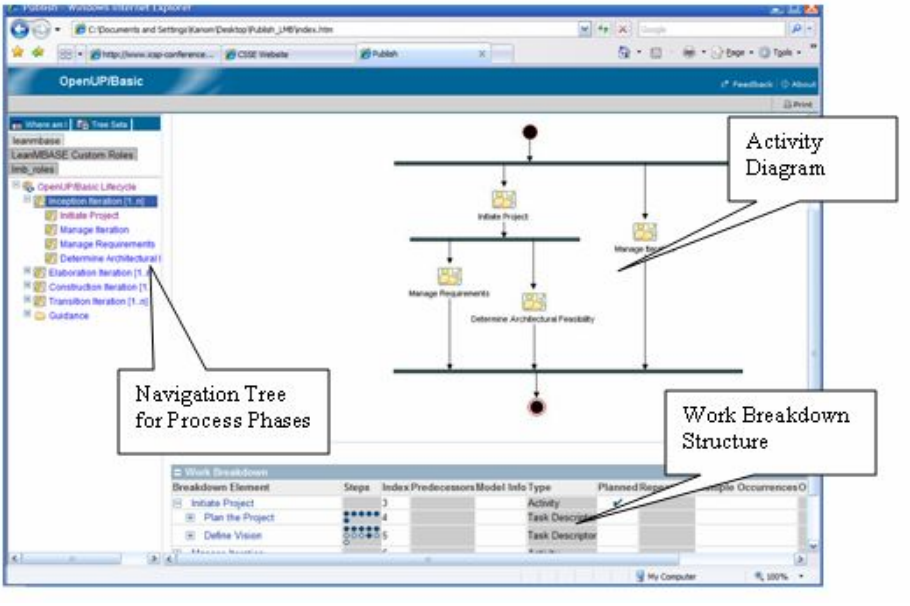


Fig. 4. An example of activity diagram and WBS provided by EPFC

Figure 4 shows the activity diagram and WBS of the Inception phase. It also provides the navigation tree to all the phases on the left side. The process performers can see that there are 4 main phases for LeanMBASE: Inception, Elaboration, Construction and Transition. They can click on the Inception phase to see the decomposition of the activities and tasks.

The WBS and activity diagram give an overview to process performers to see how the process life cycle should look like, which iteration to perform which activities and in which order. For example, the process performers can see that in the Inception phase, the "determine architectural feasibility" activity and "manage requirement" activity are performed concurrently and "manage iteration" is performed from the beginning of iteration to the end of iteration.

However, in order to model LeanMBASE, EPFC still has limitations. In our class, the students use the LeanMBASE with spiral model [13]. In the spiral model, there are tasks and steps, which need to be performed more than once per iteration. EPFC cannot model loop tasks at the activity level and cannot model the concurrent steps and multiple loop steps at the task level (if there is any). For example, if an architect determines that the software's architecture is not feasible, the architect may want to go back in the iteration to have the project team renegotiate requirements and/or re-plan the iteration.

Furthermore, when performing the spiral process, there may be (sub)-spirals inside of a spiral. For example, in the satellite-experiment software [7], there is the uncertainty about whether the fault-tolerant features are going to cause an unacceptable degradation in real-time performance. The paper suggested that the best ways to reduce this source of risk is to buy information about the actual situation by investing in a prototype to better understand the performance effects of the various fault-tolerance features. As a result, the development of the prototype project should follow its own spiral, which is nested within the project's spiral.

3.4 Support Reusable of Content in the Process

EPFC supports reusability of content in the process. EPFC's approach is to separate the method content and process content so the process engineer can make changes in the method content without changing the process content. Thus, method content and process content are independent. For example, when the process engineer wants to create a new process, the engineer can quickly assemble a new process by reusing existing method elements (such as task or role) from the method content library. In addition, EPFC allows the process engineers to copy whole or parts of existing processes from the library.

However, there are some limitations. The process engineer cannot change or edit the content or relationships of the content element, which inherits from the existing method contents.

3.5 Dynamic Process Configuration

By dynamic process configuration, we define as the project teams have ability to tailor a process guideline in real-time. Currently, neither EPG nor EPFC provide this capability. In our software engineering courses, the LeanMBASE process does not fit

all projects. Some teams will have to tailor the process in real-time to meet their project needs. For instance, the LeanMBASE Guidelines recommends team do a technology-independent architecture model, followed by a technology-specific architecture model. However for teams who are constrained to use specific technologies, a technology-independent architecture model will be of no use to them. Thus, the process will need to be tailored down to remove the technology-independent architecture model task. In the EPFC case, the team will have to learn how to use EPFC at its process composition level and modify the process elements as needed.

3.6 Integration to the Other Software Engineering Tools

EPFC provides the integration to two software engineering tools. First EPFC allows you to export a WBS to Microsoft Project or integrate with Rational Portfolio [18] for project planning and tracking. The managers can thus use the WBS as an initial set of project plan elements to help them plan their project. Second, a process engineer also can integrate EPFC with configuration management tools such as Concurrent Versions System (CVS). This aids process engineers in keeping track of the evolution of the process.

3.7 Comparative Usage Statistics

Development of the MBASE EPG using Spearmint took a PhD student roughly 2 person months and produced a Guide including roughly 8 roles, 11 artifacts, and 45 activities. Students found it quite helpful for exploring the basic relations between agents, artifacts, and activities and for accessing artifact templates. However, they found it too limited with respect to tailoring options, and too difficult to quickly learn how to tailor the guidelines at the Spearmint level. Also, later PhD students found Spearmint extremely difficult to use in updating the MBASE EPG, resulting in discrepancies between the evolving MBASE Guidelines and the EPG.

Development of the LeanMBASE EPF capability involved two PhD students and a total of roughly 100 person-hours, or roughly two-thirds person-month at 152 work hours/month. The result included approximately 12 roles, 15 artifacts, and 20 tasks, along with 50 guidances (which includes guidelines, examples templates, tool mentors, and checklists). Evolving the resulting EPF capability has been significantly easier than with Spearmint, subject to the desired additional capabilities discussed above.

4 Future Challenges for Software Process Modeling Tools

The increasing diversity of software projects, with various combinations of COTS, open source, legacy, and custom software and rapidly evolving products, methods, and tools create a number of challenges for software process modeling tools. Not only will they need high degree of flexibility and tailorability, but also they will need strong change propagation and version control capabilities to ensure consistent model

changes and the ability to evolve versions for some users without upsetting support of other users.

These are nontrivial challenges, but the kind of support that advances process modeling tools will provide will be absolutely essential to future software teams attempting to succeed on the complex, globally distributed software projects of the future.

5 Conclusion

In this paper, we compared two process generator tools: Eclipse Process Framework and Spearmint/EPG. In our evaluation, we used four objectives to compare the process tools as defined by Humphrey and Kellner [17]: communicates process, process reuse, process evolution, and process management. Table 1 is the summary how Spearmint/EPG and EPF fared when compared to the above objectives.

From our experience, Spearmint, which is one of the early process generator tools, has some limitations on usability, modifiability and extendibility. EPF, which is a more recent tool, provides more capabilities to model process and is easier to use, but EPF still has its own limitations of dynamic process configuration. Finally, for the future, projects will be more diverse and thus will need high degrees of flexibility and tailorability when modeling the project's software processes. In addition, the tools will need strong change propagation and version control capabilities to ensure consistent model changes and the ability to evolve versions for some users without upsetting support of other users.

Table 1. Comparing Process Tools

	EPG	EPF
Communicates Process	Uses text and some pre-drawn diagrams to communicate process. Provides semi-detailed description of each activity.	Uses activity diagrams, work-breakdown structures, and text to communicate process. Provides step-by-step instructions on how to perform each task.
Process Reuse	Not available - Have to start over with each new process.	Process elements meant to be reusable by other processes
Process Evolution	Difficult – Need to re-specify the content to tailor the new process.	Allow the extension and tailoring of existing process.
Process Management	Does not allow tracking of process	Does not allow tracking of process

References

1. Abts, C., Boehm, B. and Clark B., "COCOTS: a COTS software integration cost model," Proceedings ESCOM-SCOPE 2000 Conference.
2. Becker, U., Hamann, D., Münch, J., and Verlage, M., "MVP-E: A Process Modeling Environment". IEEE TCSE Software Process Newsletter, (10):10–15, Fall 1997

3. Becker, U., Hamann, D., and , “Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance”. Matthias Jarke, Andreas Oberweis(Eds.): Advanced Information Systems Engineering, Proceedings of the 11th International Conference CAiSE'99, Lecture Notes in Computer Science, Vol. 1626, pp. 119-133. Springer, 1999
4. Becker, U. and Verlage, M., “The V-Modell Guide: Experience with a web-based approach for process support” Proceedings of Software Technology and Engineering Practice 99, 1999
5. Becker, U., Scott, L. and Zettel, J., "Process engineering with Spearmint(/EPG". Proceedings of the 22nd International Conference on Software Engineering, pp. 791-792, 2000
6. Boehm, B., “Anchoring the Software Process”. IEEE Software, pp. 73-82, July 1996
7. Boehm, B., “Software Risk Management: Principles and Practices”. IEEE Software, pp. 32-41, January 1991
8. Boehm, B., Horowitz E., Madachy R., Reifer D., Clark B., Steece, B. Brown AW., Chulani, S. Abts, C “Software Cost Estimation with Cocomo II” Prentice Hall, July 2000
9. Boehm B., Grünbacher P., Briggs B., “Developing Groupware for Requirements Negotiation: Lessons Learned”, IEEE Software, May/June 2001, pp. 46-55
10. Boehm, B. and Port, D., “Escaping the Software Tar Pit: Model Clashes and How to Avoid Them”. ACM Software Engineering Notes, January 1999, pp. 36-48
11. Boehm, B., Port, D., Abi-Antoun, M., and Egyed, A., “Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) deliverables for Model-Based Architecting and Software Engineering (MBASE)”. USC Technical Report 1998
12. Boehm, B., Port, D., Egyed, A., Abi-Antoun, M. “The MBASE Life Cycle Architecture Milestone Package: No Architecture is An Island”. In First Working IFIP Conference on Software Architecture (WICSA'1), 1998
13. Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., and Madachy, R., "Using the WinWin Spiral Model: A Case Study", IEEE Computer, July 1998, pp. 33-44
14. Booch G., Rumbaugh, J., and Jacobson, I., “The Unified Modeling Language User Guide”, Addison Wesley, 1999
15. Haumer, P., “Eclipse Process Framework Composer Part 1 and 2”. User Documentation, http://www.eclipse.org/epf/general/getting_started.php
16. Haumer, P. “Increasing Development Knowledge with EPFC”. Eclipse Review, Vol. 1, no. 2, pp. 26-33, Spring 2006
17. Humphrey, W. and Kellner, M. “Software Process Modeling: Principles of Entity Process Models.” Proceedings of the 11th International Conference on Software Engineering, PA, USA, 1989. pp. 331 – 342.
18. IBM Rationale Portfolio Manager, http://www-306.ibm.com/software/awdtools/portfolio/index.html
19. Kellner, M. “Software process modeling support for management planning and control”. In Mark Dowson, editor, Proceedings of the First International Conference on the Software Process, pages 8–28. IEEE Computer Society Press, August 1991
20. Kellner, M. Becker, U., Riddle, W., Tomal, J., and Verlage, M., “Process guides: Effective guidance for process participants”. Proceedings of the Fifth International Conference on the Software Process, pages 11–25, Chicago, IL, USA, June 1998 ISPA Press
21. Kroll, P. and Sand, P. “A Development Library at Your Fingertips”. Eclipse Review, Vol 1, no. 3, pp. 258-28, Summer 2006
22. Kruchten, P., “The Rational Unified Process (2nd ed.)”. Addison Wesley, 2001
23. “MBASE Website” http://sunset.usc.edu/csse/research/mbase/

24. “OpenUP/Basic – A Process for Small and Agile Projects”. User Documentation, http://www.eclipse.org/epf/general/getting_started.php
25. Scott, L., Carvalho, L., Jeffery, R. and D’Ambra, J., “An Evaluation of the Spearmint Approach to Software Process Modelling”. Proceeding of the European Workshop on Software Process Technology 2001 (EWSPT 2001), page 77-89, 2001
26. Scott, L., Jeffery, R., and Becker-Kornstaedt, U., “Preliminary Results of an Industrial EPG Evaluation” 4th ICSE Workshop on Software Engineering over the Internet, IEEE Computer Society, California, USA, 2001, pp. 55 – 58
27. UML Resource Page, http://www.uml.org/
28. USC Software Engineering Class I Website, http://greenbay.usc.edu/csci577/fall2006/site/index.html

Jasmine: A PSP Supporting Tool

Hyunil Shin, Ho-Jin Choi, and Jongmoon Baik

Information and Communications University, School of Engineering,
119 Munjiro, Yuseong-gu, Daejeon, 305-732, Korea
{linugee,hjchoi,jbaik}@icu.ac.kr

Abstract. The PSP (Personal Software Process) was developed to help developers make high-quality products through improving their personal software development processes. With consistent measurement and analysis activities that the PSP suggests, developers can identify process deficiencies and make a reliable estimate on effort and quality. However, due to the high-overhead and context-switching problem of manual data recording, developers have difficulties to collect reliable data, which can lead to wrong analysis results. Also, it is very inconvenient to use the paper-based process guide of the PSP in navigating its process information and difficult to attach additional process-related information to the process guide. In this paper, we describe a PSP supporting tool that we have developed to deal with these problems. The tool provides automated data collection and analysis to help acquire reliable data and identify process deficiencies. It also provides an EPG (Electronic Process Guide) in order to provide easy access and navigation of the PSP process information, which is integrated with an ER (Experience Repository) to allow developers to store development experiences.

Keywords: Personal Software Process, Electronic Process Guide, Automated Data Collection, Experience Repository.

1 Introduction

Continuous process improvement has been regarded as a solid solution to make high-quality products at the team and personal level as well as at the organization and project level. The PSP [1] was developed to help individual developers make high-quality products through improving their personal software development processes. The PSP provides a set of methods and practices to assist individual software developers to improve product and process quality such as defined and measurable process, size and effort estimation based on historical data, code and design review, precise designs, process quality measures, detailed plan, and earned value tracking. While the PSP has been proved as an effective way to improve the accuracy of effort estimation and to reduce defects in case studies [13, 14, 15], its manual data recording and paper-based process guide act as barriers in following the PSP process.

Among those methods and practices, the measurement and analysis is a central and core practice in identifying process deficiencies and providing a focus on process improvements. Sets of historical project data are used to make a reliable estimate on effort and quality. However, due to the high-overhead and context-switching problem

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 73–83, 2007.

of manual data recording, developers have difficulties to acquire reliable data, which can lead to wrong analysis results [2, 3]. The problem can be overcome through an automated tool for collecting the PSP data and analyzing the collected data. However, since an automated tool can not collect all necessary data, manual data recoding should be supported as well. Manual data recording can be still a problem, but data errors can be decreased because it is reduced to a few items. To help developers collect reliable data and all necessary data, it is therefore required to develop a tool for supporting both automated and manual data collection.

The PSP provides a set of increasingly evolved processes to help developers learn the methods and practices. To guide developers in following the processes, materials such as scripts, templates, and checklists are presented in a paper form, which can be seen as a paper process guide. A paper process guide generally has problems in its usability and maintenance because it is very inconvenient for developers to search and navigate process information and difficult to add process-related information or to modify existing information [8]. To solve these problems caused by a paper process guide, an EPG using the web technology is proposed allowing easy access to all process-related information [5, 8]. To allow easy navigation of the PSP process information and to enable storing additional information, it is necessary to develop an EPG which enhances the contents and usability of the paper-based PSP guide.

In this paper, we describe a PSP supporting tool, named Jasmine, which have been developed to address the issues above. Aiming at supporting personal process and quality management, the Jasmine provides capabilities to collect reliable data automatically and analyze the collected data. It also provides an EPG for the PSP guide for easy access, modification and addition of information.

The rest of this paper is organized as follows. The next section gives a short overview of sensor-based automated data collection, an EPG and an ER. This is followed by the description of the Jasmine's architecture and salient features. Section 4 presents a comparison with existing PSP supporting tools, and section 5 concludes the paper and describes future works.

2 Background

2.1 Sensor-Based Automated Data Collection

To reduce the high overhead and context-switching in manual data collection, tools like Hackystat [2, 9], PROM [12] have been developed. They collect automatically the PSP data and provide various analyses on the collected data. These tools do not require any efforts of developers in data collection, except in installation and configuration of sensors. Sensors, which are attached to development-related tools such as Eclipse, Microsoft Office, and JBuilder, are central components for automatic data collection. A sensor collects unobtrusively low-level data (e.g., information on files that developers are editing, results of unit test executions) by monitoring application-generated events of a development-related tool. Then, it sends the low-level data to a server where the data are stored and analyzed.

Although Hackystat and PROM collect the PSP data automatically, all necessary data can not be collected automatically and the collected data do not have all

necessary information. For example, the time data collected automatically are associated with modification activity of software artifacts such as source files and design documents. In this way time spent on implementation or design activity can be automatically collected, but time spent on other activities (e.g., meeting, design review) can not be collected because not all important developer activities involve modification of software artifacts. Also, it is hard to identify which phase automatically collected time data are spent on. Defect data are automatically collected by sensors attached to unit testing mechanism such as JUnit or to bug reporting systems such as Bugzilla. However, there is no way to automatically collect defects in design/design review/code review phases where developers manually find defects, and automatically collected defect data do not have all information such as the time spent on finding and fixing the defect, the phase when it was injected, and its defect type.

2.2 EPG and ER

A process guide is a reference document to help process participants understand and execute a given process, providing guidance of the process and other useful information [8]. Basic information of process guides are details regarding activities, artifacts, roles, and relationships between them. Process guides are necessary for software process improvements where process knowledge transfer is crucial. Process guides traditionally were offered in a paper form, but it is said that they are not useful in their contents and layouts [8]. It is hard to navigate and search easily process information and to put related information together (e.g., an activity and its input and output artifacts) in paper-based process guides, because its layout is linear and static. Also, it is difficult to modify existing information or add new process information because it requires publishing a new edition of its process handbook.

These problems of paper-based process guides can be mitigated by an EPG which provides a process guide using the web technology [5, 8]. However, simply providing a process guide in forms such as PDF, Microsoft Word, or other electronic formats or converting the contents of a process guide into HTML is not treated as an EPG. In [8], a set of basic requirements are proposed which an EPG should meet.

- An EPG should provide all information contained in a good paper process guide.
- It is recommended that each web page contain so small manageable unit that process participants can easily understand and digest.
- An EPG should provide hyper-links, a graphical overview, and hierarchical activity decompositions for flexible navigation and easy access. Also, related information such as an activity and its associated artifacts should be linked together using hyper-links.
- All web pages should have the same basic structure in order to facilitate the usage.

Beyond the basic requirements above, an EPG can contain additional process-related information such as examples of a document, personal annotation, or discussion, which leads to more general knowledge and experience management. As a result it is recommended to integrate an EPG with an ER [7]. An ER is a system which is used to collect, structure, and reuse key management and development experience, and to make it quickly and easily accessible to users [6]. An ER plays a crucial role in

knowledge and experience management where past knowledge and experience is seen as resources to solve today's problems.

Some works have been done to integrate an EPG with an ER. In [4, 7], a successful implementation of coupling an EPG with an ER in a small organization is presented. In the combined tool, an experience entity is attached to its related process element for easy access to a large number of collected experience data. The idea to structure experience data to related process elements is also supported by [6], which proposes that a good experience repository should be organized to its related process.

3 High-Level Architecture and Main Features of Jasmine

In this section, we describe the high-level architecture and main features of the Jasmine, which consists of two sub-systems, PPMT (Personal Process Management Tool) and PSPG/ER (PSP Guide/Experience Repository) as shown in Fig. 1. PPMT supports project planning, earned value tracking, and quality management by facilitating data collection and analyses. It automates large parts of data collection to reduce the high overhead and context switching. It also provides various data analyses in forms of charts, graphs, or tables. In PSPG/ER, the EPG provides the PSP process guide in the web and the ER is used to store and share development experience which can be linked to the EPG contents.

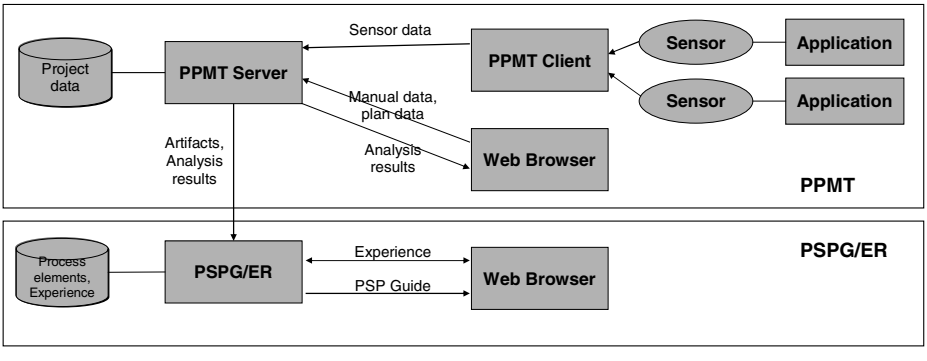


Fig. 1. Jasmine Architecture

3.1 PPMT

PPMT is designed using a client-server architecture, as illustrated in Fig. 1, in which the client consists of sensors developed for automated data collection. The server provides all functionalities except automated data collection. It was implemented as a web application which interacts with users through a web browser. The main components of PPMT are as follows.

- Sensor: It is attached to a development-related application. It collects automatically data by monitoring the application and then sends the data to the PPMT Client.

- PPMT Client: The main functionality of the PPMT Client is to receive sensor data from the sensors and to send them to the PPMT Server. It plays a temporary storage for collected sensor data when it is not connected to the server, and sends them to the server when the connection to the server is re-established. If necessary, it can preprocess sensor data before sending them to the PPMT Server.
- PPMT Server: It provides most of functionalities for PPMT: manual data recording, data storage, data analyses, earned value calculation, and users/projects administration. The implementation is based on Java technologies (such as Java Servlet, JSP, Java Beans, and JDBC), and on Apache Tomcat to execute Java Servlets and JSP.
- Database: It stores the collected PSP data from sensors and manual recording such as time and defect logs, task and schedule plan data, and information on users/projects. MySQL is used for the database implementation.

XML is used to send and receive sensor data among sensors, the PPMT Client, and the server. Its language-independent characteristic simplifies sensor data transmission because sensors are implemented using various programming languages. The main features of PPMT are presented below.

Sensor-based automated data collection. To facilitate recording time, defects, and software size, PPMT provides a sensor-based automated data collection mechanism like Hackystat, PROM. Time and defect data collected automatically are recorded in the time and defect log respectively, which allows modification and insertion of the data when necessary.

By monitoring software artifacts or tools, time spent on design, coding, review, and testing can be collected automatically. The current version of the Jasmine collects automatically time spent on: source code modification by monitoring continuously source files' size; manual testing of windows applications and web applications by monitoring mouse or key events occurred in the target application. An Eclipse sensor tracks Java source code modification and manual testing of a windows application executed in Eclipse. Testing web applications using Internet Explorer is tracked by an IE sensor. A set of consecutive time data is stored as an item in the time log.

Failed unit tests, bugs, compile errors and so on can be automatically collected as defects. The Jasmine collects automatically failed unit tests, compile errors, and runtime errors, each of which is stored as a defect in the defect log as shown in Fig. 2.

Table 1. Defect information of failed unit tests, compile errors, and runtime errors

	Failed unit tests	**Compile errors**	**Runtime errors**
Remove phase	"Test"	"Compile"	"Test"
Description	The stack trace of the exception	The description of the syntax error	The stack track of the exception
Defect type	The exception type	"Syntax"	The exception type
Found date	(automatic)	(automatic)	(automatic)
Inject phase	(manual)	(manual)	(manual)
Fix time	(manual)	(manual)	(manual)

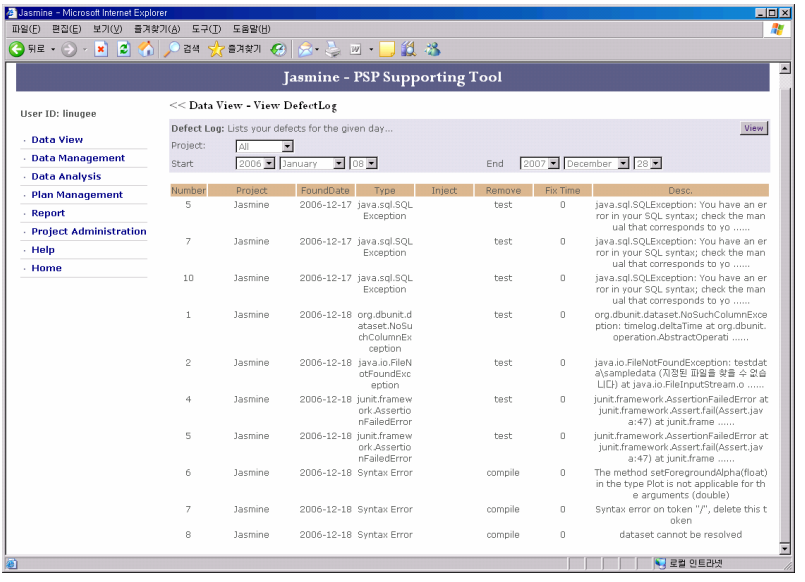


Fig. 2. Defect log

The Eclipse sensor collects the results of unit tests executed by JUnit, Java compile errors, and Java exceptions. As described in Table 1, information on removal phase, description, defect type, and found date are automatically recorded.

Software size can be automatically collected as lines of code (LOC) measured by a line counting tool. The current implementation collects LOC measured by LOCC [16].

Support for planning and earned value tracking. Developers should make a detailed plan in the planning phase and track the progress with the earned value. In order to assist the project planning and tracking, PPMT provides forms to prepare the standard task and schedule planning templates, and automatically calculates the earned value of all planned tasks using planned data that a developer enters and actual data calculated from the recorded time log.

Data analyses and report generation. PPMT provides various analyses over the collected data in forms of charts or tables. It reports a summary of analyses results. Available analyses include trend charts which show the trend of data over time and an earned value chart which displays the planed value, the earned value, and the predicted earned value over time. It also provides Pareto charts for defect analysis and quality measures such as process yield, A/FR (Appraisal to Failure Ratio), and phase ratio. Also, it can generate a weekly report which summarizes project data during a given week and a project report which summarizes project data during the whole period.

3.2 PSPG/ER

The main elements provided by PSPG are the PSP activities (e.g., planning, design, and design review), artifacts (e.g., task and schedule plan, project plan summary), and

the PSP processes (e.g., PSP0, PSP0.1). The PSPG/ER homepage provides a single point access to the PSP processes. A number of activity and artifact pages provide the guides of the PSP activities and artifacts, respectively. Every activity page consists of three frames as shown in Fig. 3: a navigation bar, a diagrammatic process flow, and a description section. The navigation bar consistently maintained in all of pages displays the current position. The diagrammatic process flow shows a flow of activities highlighting the selected activity and supports fast navigation to other activities. The description section contains the description of the selected activity, links to its related artifact pages, and links to experience data associated to it. Each artifact page consists of three frames as shown in Fig. 4: a navigation bar, a list of artifacts, and a description section. The list in the left frame contains a list of all the artifacts which must be produced in the selected PSP process. The description section includes the description of the selected artifact, its templates, and links to experience data related to it.

The ER enables developers to collect development experiences gained from previous projects by following the PSP process and to share them among team members. To provide easy access to a number of collected experiences, they are structured according to relevant process elements. That is, developers should insert an experience data to its related activity or artifact page. For example, a document example should be linked to its related artifact page. Experience data are categorized into example (only available in artifact pages), generic experience, and discussion. In the example category, examples of an artifact are provided in forms of files such as PDF, Microsoft Word, or other file formats which can be downloable, or HTML pages which are generated in PPMT. The generic experience category can include any helpful information such as lessons learned, code fragments, and links to useful web pages. The discussion category allows developers to discuss process elements with other developers.

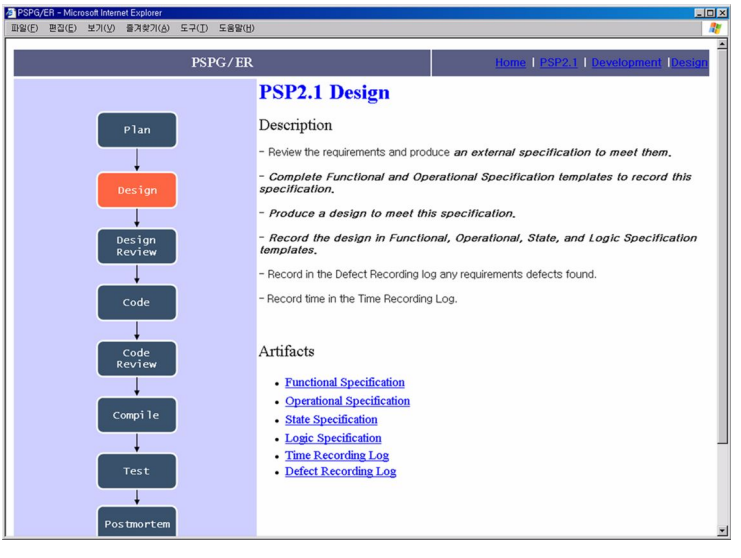


Fig. 3. An example of an activity page

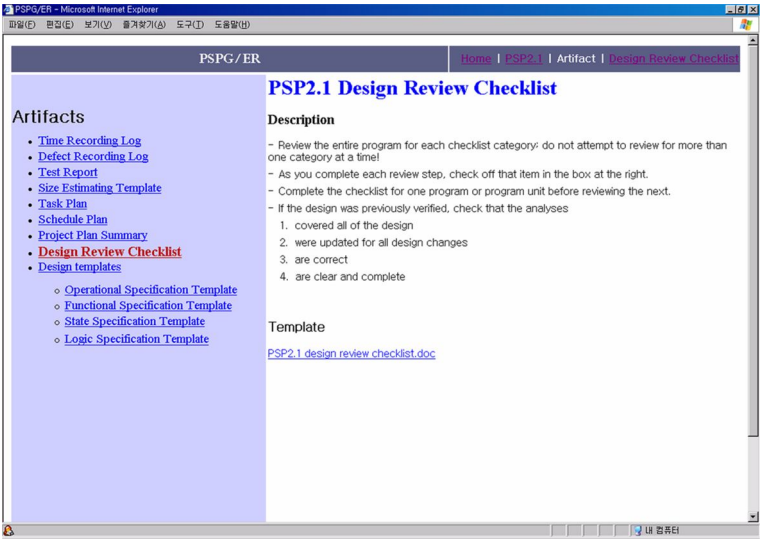


Fig. 4. An example of an artifact page

3.3 Interaction Between PPMT and PSPG/ER

One of main features in PSPG/ER is to store examples of artifacts such as time logs, defect logs, and task/schedule plan. Examples can be stored in a HTML format which is produced in PPMT. Developers can store their artifacts such as time/defect logs and task/schedule plan in an example category of a relevant artifact and their analyses results such as charts, tables, and reports in any experience category. This feature would make it easy to store development experience. Another way of interaction is to provide links to relevant pages. For example, the time log artifact page has a link to the time recording form of PPMT, and in reverse the form contains a link to the artifact page of PSPG/ER. This feature would allow developers to access easily relevant process information.

4 Comparative Analysis of Related Tools

Several PSP support tools have been developed such as Process Dashboard [11], Hackystat [2, 9], and PSPA [10] to help automatic data collection and analyses. Among those tools, Hackystat provides the most similar functionalities to the Jasmine in that both tools provide sensor-based automated data collection. The primary difference between the Jasmine and Hackystat lies in the goal that each aims for. Hackystat is a tool for data collection and analyses rather than a PSP supporting tool since it focuses on only automated data collection and analyses. Therefore, Hackystat does not support the other PSP activities such as planning, plan tracking, and estimation. It also provides the limited data analysis capabilities. This insufficiency of Hackystat is caused by not supporting manual data recording and not collecting automatically all necessary information of the PSP data.

Table 2. Comparison of sensor-based automated data collection

Data		Jasmine	Hackystat
Time	Time spent on source modification	Eclipse	Eclipse, Visual Studio, JBuilder, IntelliJ Idea
	Time spent on modification of other documents	X	Microsoft Office, OpenOffice, Emacs
	Time spent on code review	X	Jupiter
	Time spent on manual testing	Internet Explorer (for web application testing), Eclipse (for windows application testing)	X
Defect	Failed unit tests	JUnit	JUnit, CPPUnit
	Compile errors	Eclipse	X
	Runtime errors	Eclipse	X
	Post-release bugs	X	Bugzilla, Jira

On the other hand, the Jasmine aims for supporting the whole PSP activities. In the Jasmine, the automatically collected time and defect data are recorded in the time and defect log respectively in order to allow developers to modify the data or insert necessary information to the data, which enables more various data analyses compared to Hackystat. Also, it provides an EPG for the PSP guide incorporating with an ER.

In a comparison of sensor-based automated data collection, while currently the Jasmine does not support as many development-related tools as Hackystat does, it collects automatically time spent on manual testing, compile and runtime errors which Hackystat does not collect, as shown in Table 2. The Jasmine would be easily extended to support various tools by reusing the Hackystat sensors, since Hackystat has been developed as an open source.

5 Conclusion and Future Work

This paper has described the Jasmine developed to help developers perform the PSP. The Jasmine not only automates large parts of data collection to mitigate the problems of manual data recording, but also supports planning and plan tracking. It also provides various kinds of data analyses. These features help developers identify process deficiencies, make a process improvement plan to remove the identified deficiencies, and make a reliable estimate on effort and quality for more effective and efficient process management. Moreover, the Jasmine includes an EPG to allow easy navigation of the PSP process elements and an ER to allow storing and sharing additional process-related information. This integrated EPG and ER would help developers understand and perform the PSP more effectively. A number of collected experiences would be used as resources to solve problems that they can run up against in the PSP process.

This work has been done as a first step of the project that aims to develop a TSP/PSP supporting tool. TSP (Team Software Process) support is planned as one of future works, which includes providing automated data collection and analyses for team data and supporting team planning process and plan tracking, in order to help developers as well as team managers follow the TSP. Also, the sensor-based automated data collection and analyses will be extended continuously with more features. New sensors for various development-related tools (e.g., Visual Studio, Microsoft Office) and new sensor data types (e.g., test coverage, post-release bugs, and code quality metrics) will be developed. We will also provide diverse data analyses to facilitate identification of process deficiencies and product's quality problems. Further, Six Sigma analysis techniques such as control charts, regression analyses will be integrated to help systematic process control. Finally, to improve and extend the tool based on real usage, we will apply this tool to student projects in a class or industrial projects.

Acknowledgement. This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2006-(C1090-0603-0032)).

References

1. W. S. Humphrey. PSP(sm): A Self-Improvement Process for Software Engineers, SEI Series in Software Engineering, Addison-Wesley Professional, 2005
2. P. M. Johnson, H. B. Kou, J. M. Agustin, C. Chan, C. A. Moore, J. Miglani, S. Zhen, and W. E. Doane. Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In Proceedings of the 2003 International Conference on Software Engineering, Portland, Oregon, May 2003.
3. Disney, A. & Johnson, P. Investigating Data Quality Problems in the PSP, Sixth International Symposium on the Foundations of Software Engineering (SIGSOFT'98), Orlando, FL., November, 1998.
4. Felicia Kurniawati, Ross Jeffery. The Long-term Effects of an EPG/ER in a Small Software Organisation, 2004 Australian Software Engineering Conference.
5. L. Scott, L. Carvalho, R. Jeffery, J. D'Ambra and U. Becker-Kornstaedt, "Understanding the use of an Electronic. Process Guide," Information and Software Technology 44. (10), 2002, pp. 601-616.
6. Kurt Schneider, Jan-Peter von Hunnius, "Effective Experience Repositories for Software Engineering," icse, p. 534, 25th International Conference on Software Engineering (ICSE'03), 2003.
7. Louise Scott, Lucila Carvalho, Ross Jeffery, "A Process-Centred Experience Repository for a Small Software Organisation," apsec, p. 603, Ninth Asia-Pacific Software Engineering Conference (APSEC'02), 2002.
8. M. Kellner, U. Becker-Kornstaedt, W. Riddle, J. Tomal, M. Verlage, "Process guides: effective guidance for process participants," in: Proc. of the Fifth International Conference on the Software Process, Chicago, IL, USA, June 1998, ISPA Press, 1998, pp. 11-25.

9. Johnson, P.M.; Hongbing Kou; Agustin, J.M.; Qin Zhang; Kagawa, A.; Yamashita, T., "Practical automated process and product metric collection and analysis in a classroom setting: lessons learned from Hackystat-UH," International Symposium on Empirical Software Engineering, 2004.
10. Raymund Sison, David Diaz, Eliska Lam, Dennis Navarro, Jessica Navarro, "Personal Software Process (PSP) Assistant," apsec, pp. 687-696, 12th Asia-Pacific Software Engineering Conference (APSEC'05), 2005.
11. Process Dashboard, http://processdash.sourceforge.net/
12. Sillitti, A.; Janes, A.; Succi, G.; Vernazza, T., "Collecting, integrating and analyzing software metrics and personal software process data," Euromicro Conference, 2003. Proceedings. 29th , vol., no.pp. 336- 342, 1-6 Sept. 2003.
13. Pekka Abrahamsson, Karlheinz Kautz, "The Personal Software Process: Experiences from Denmark", EUROMICRO 2002: 367-375.
14. Lutz Prechelt, Barbara Unger, "An Experiment Measuring the Effects of Personal Software Process (PSP) Training," IEEE Transactions on Software Engineering, vol. 27, no. 5, pp. 465-472, May, 2001.
15. Hayes W., and J. Over. The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers, Technical Report SEI-97-TR-001, December 1997.
16. LOCC, http://csdl.ics.hawaii.edu/Tools/LOCC/

A Tool to Create Process-Agents for OEC-SPM from Historical Project Data

Lei Zhang [1,2], Qing Wang[1], Junchao Xiao[1,2], Li Ruan[1,2], Lizi Xie[1,2], and Mingshu Li[1,3]

[1] Laboratory for Internet Software Technologies, Institute of Software,
The Chinese Academy of Sciences, Beijing 100080, China
{zhanglei, wq, xiaojunchao, ruanli, xielizi,
mingshu}@itechs.iscas.ac.cn
http://www.cnsqa.com
[2] Graduate University of Chinese Academy of Sciences, Beijing 100039, China
[3] Key Laboratory for Computer Science, The Chinese Academy of Sciences
Beijing 100080, China

Abstract. Software processes are highly people-dependent and they relay on the capabilities of a group of developers and their creative works. Therefore an Organization-Entity capability based software process modeling method OEC-SPM was proposed to modeling the software process by adopting Process-Agent (*PA*) as key element. Since the OEC-SPM needs a mass of PAs to perform precise modeling but then the current process of creating PAs is inefficient and people-dependent, this paper presents a tool to create the PAs for OEC-SPM automatically from the Historical Project Data (*HPD*). The paper makes an overview of the PA's structure in OEC-SPM then gives the definition of HPD. After that the paper introduces to the process of creating PAs in the tool and illustrates the application of the tool with an example on a software quality management system SoftPM. Finally the paper illustrates the tool's result and then presents the future works.

Keywords: Process-Agent, OEC-SPM, SoftPM, Knowledge Extraction.

1 Introduction

Software processes are highly people-dependent and they rely on the capabilities of a group of developers and their creative works[1-3]. In a software organization, the executers of the process are the *Organization-Entities (OE)* who has the needed capabilities. These entities generally display dynamic, autonomous and active behaviors so that the precise definition of them would seem to be a requisite in the software process modeling[4]. Regarding that, in an *O*rganization-*E*ntity *C*apability based *S*oftware *P*rocess *M*odeling method *OEC-SPM*[5, 6] proposed by Institute of Software, Chinese Academy of Sciences(ISCAS), we defines the *Organization-Entity* that contains definite resource capabilities (goals, skills, knowledge, productivities, experiences, historical data records and equipments…etc.) as *Process-Agent(PA)*[7], with the *PA*s, related software process elements would be dynamically assembled into project software processes (project plans) via the self-adaptive reasoning mechanism of the *PA*. More about *OEC-SPM* is given in the appendix of the paper.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 84–95, 2007.

OEC-SPM regards *PA* as the key and fundamental element of the whole method, which implicates a demand for creating a mass of *PA*s accurately and efficiently. However, the current process of creating PAs in [4-7] is not only deeply relies on modelers' capabilities and experiences, which is detrimental to the precision and stability of the *PA*s but also makes people have to conduct complex analyses and processing on an extremely large amount of various information of the organization manually to take into account the complex nature of the software process (since "software processes are software too" [8, 9], the complexity of software process is not second to the software), which leads to a poor efficiency.

In view of that, we implement a tool to create the *PA*s for *OEC-SPM*. The tool creates *PA*s from the historical project data that is traditionally used for organizational historical tracking[10] regarding to its reflecting of the organization's real state and capability. With the aid of the tool we can fully eliminate the affects of modelers' personality over creating *PA*s, and meanwhile can significantly improve the efficiency owing to the tool's automation.

This paper is organized as follows. Section 2 makes an overview of the process of *OEC-SPM* and then introduces to the structure of *PA*. Section 3 introduces the definition of *HPD*. Section 4 introduces to the process of creating PAs in the tool, while section 5 illustrates the application of the tool with an example on a software quality management system *SoftPM* [11]. Section 6 illustrates the tool's result. Finally, section 7 presents the conclusion and future work.

2 The Process of OEC-SPM and the Structure of PA

In *OEC-SPM*, the *PA*s *Percept* environment actively and react automatically to the *Environment Knowledge*, *Goals, Changed Requirements and Constraints* on the basis of particular environment states, so that the they can establish the *Software Processes* self-adaptively through a negotiation-based *Cooperation* taking advantages of their *Intelligent Behaviors* to achieve the *Goals*. Moreover, the capabilities of the *PA*s would be ceaselessly *improved* and o*ptimized* based on the feedbacks from the *Process Data* that is produced by the *Software Processes* execution, so that the stability and the predictability of the process modeling will also be progressively increased. Fig. 1 depicts the whole process.

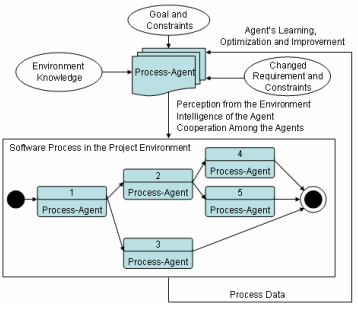


Fig. 1. The Process of OEC-SPM

A *PA* in *OEC-SPM* comprises two parts: *Infrastructure* and *Engine* (Fig. 2). The *Infrastructure* comprises three types of knowledge, which are: *Descriptive Knowledge*, *Process Knowledge* and *Experiences Library*. The *Engine* provides an acting mechanism for the *PA*, which is used to reason out the behaviors of the *PA* according to the *Infrastructure* and the environment where the *PA* resides in.

The three types of knowledge constituting *Infrastructure* determines whether the *PA* has capabilities to determine what it can do, how it to do and how many resources would be needed in order to do, in Particular:

- **Descriptive Knowledge** describes what the *PA* looks like and what it can do, it is determined by the *Process Knowledge* and the *Experiences Library* of the *PA*.
- **Process Knowledge** describes how the *PA* can proceed to realize its goals by means of *Process-Step*s organized into defined sequences.
- **Experiences Library** is constructed from the historical data generated from the previous executions of the steps by the *PA*. It can be used to estimate how many resources are likely to be required in order to achieve goals.

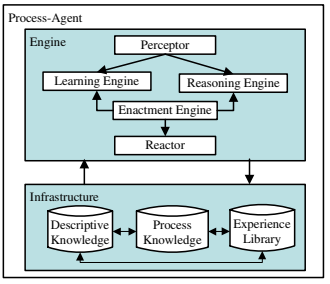


Fig. 2. The Structure of a PA

In a PA, *Infrastructure* lays all the foundations of the *PA*'s behaviors thus if we have established the *Infrastructure* we have created the *PA* already. In the *Infrastructure*, *Process Knowledge* provides an underlying determination of the goals that can be realized by the *PA* and the ways by which the *PA* will attempt to achieve the goals. Therefore the establishing of the *Process Knowledge* is a foundation that acts as an essential prerequisite to establish the *Descriptive Knowledge* and *Experiences Library*. Taking into account its fundamentality, we employ the establishing of the *Process Knowledge* as a vehicle to illustrate the creating of the PAs in this paper. The establishing of the *Descriptive Knowledge* and *Experiences Library* will not be discussed here.

Process Knowledge is captured as a group of *Process-Step*s that are the abstract representation of the *PA*'s tasks, which brings *PA* a particular knowledge level. We define *Process Knowledge* as $PK = \{st_1, st_2, \ldots, st_n\}$. Each *Process-Step* (*PS*) st_i in *PK* is an 8-tuple, $st_i = (SID_i, SD_i, R_i, SCRM_i, IP_i, OP_i, IMP_i, PRI_i)$, here:

(1) **SID_i** is the identification of the *Process-Step*;
(2) **SD_i** is the form of natural language, informal, descriptive words of the *PS*;
(3) **R_i** is the role being played by the *PA* while executing the *PS*, e.g., if the type of the *PS* is review, then R_i is "*QA*".
(4) **$SCRM_i$** is the *PS*'s control rule model. It comprises pre-conditions and post-conditions, such as constraint specifications on the process elements (e.g. the existence of artifacts or resource constraints etc.) The *PS* can be executed only if all preconditions are satisfied, and the *PS* can be successfully completed only if all postconditions are satisfied; thus the $SCRM_i$ controls the behaviors of the *PS* and conditions under which it will be executed.
(5) **IP_i** is *PS*'s input parameters, such as the artifacts needed for executing the *PS*;
(6) **OP_i** is *PS*'s output parameters, such as the artifacts produced by the *PS*'s execution;
(7) **IMP_i** describes the way the *PS* is implemented. A *PS* can be directly implemented by *PA* (*DIRECT*), or assigned to other PAs as a cooperative goal (*SUBPROCESS*).

3 The Definition of HPD

HPD is the data relating to the projects that now are already accomplished and are checked and accepted. *HPD* can come from various sources, in this paper, we focus on *HPD* that is acquired from the software systems that integrate software project management facilities (we call this kind of system the *SPMS)* for the advantage that the *SPMS*s can provide abundant information e.g. project information, task information, human information…etc we need. Works on making use of other types of sources can refer to [12], [13] and so on.

According to our surveys on various *SPMS*s, we found that the structures of the data about software projects in these systems are generally displayed similar. Therefore we define a common structure of the *HPD* by a series of common objects and relationships generalized from the concrete *SPMS*s, so that we can express the universality of the *SPMS* easily. Fig. 3 depicts the structure. In the structure,

- **Project** represents the character of the project e.g. the project name and other explanatory information.
- **Task** represents the task information including
  (1) *Name*. The name of the Task.
  (2) *Description*. The description of the Task.
  (3) *Start Time*. The actual start time of the Task.
  (4) *End Time*. The actual end time of the Task.
  (5) *Type*. The type of the Task, which is a number that represents the concrete Task types (e.g. requirement analysis, software implementation, testing, quality assurance, process definition…etc.). If two Tasks have the same Type, we call the two tasks are *similar tasks*
  (6) *Input Artifacts*. The artifacts that are needed for performing the Task.
  (7) *Work Products*. The products produced by the Task execution.
  (8) *Implementation*. The implementation style of the Task (*DIRECT* or *Sub_Process*), which is a kind of the intermediate information derived from the creation of *PA*s. It will be explained with the details in section 5.2
  (9) *Order*. The order of the Task, it is a number derived from the time information (*Start Time & End Time*) of the Project context. That is, in a same Project, if a Task holds a smaller Order, it will always begin earlier than the Tasks that hold bigger orders.
  (10) *Kind*. The Kind of the Task, which can be determined by the time sequence of the Task's children. Particularly, its value is "*Parnell*" if the Task has children with execution time periods overlapped each other (can determined by the *Start Time* and the *End Time* of the children

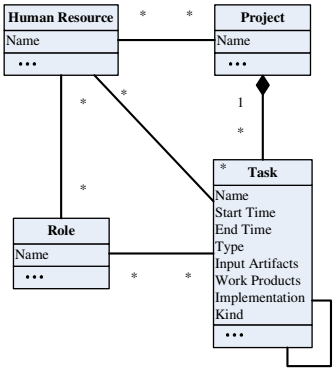


Fig. 3. The Structure of HPD

Tasks), or its value is "*Sequential*". Besides, the value can also be manually assigned.

- **Human Resource** represents the human resources that are involved in the Projects and the Tasks.
- **Role** represents the roles defined by the *SPMS*. It could be QA, SEPG, Software Engineer, Senior Manager or other kind of roles relating to the project.
- **Relationship between Project and Tasks** indicates the Tasks that belong to Project.
- **Relationship between Tasks** indicates the hierarchy of the Tasks e.g. parent task, children tasks, left brother task, right brother task…etc.
- **Relationship between Project and Human Resources** indicates the Human Resources who participate in Project.
- **Relationship between Task and Human Resources** indicates the Human Resources who perform Tasks.
- **Relationship between Role and Human Resources** indicates the role that Human Resources belonging to.
- **Relationship between Task and Roles** indicates the roles that perform Tasks.

4 The Process of Creating PAs in the Tool

The process of creating *PA*s in the tool is constituted by two sequential phases: 1. Generating *HPD* from the concrete *SPMS*s; 2. Creating *PA*s from the *HPD*. Fig. 4 depicts the whole process:

Fig. 4. The Process Overview

Phase 1. Generating HPD from the Concrete *SPMS*s. In this phase, the tool selects the historical data from the *SPMS*s and then converts the data into the objects and relationships of *HPD*. Processes in this phase are data source-dependent.

Phase 2. Creating PAs from the HPD. In this phase, the tool creates *PA* from the *HPD* generated by Phase 1 via five steps (Fig. 5), which are completely data source-independent:

Step 1. Dividing *HPD* into groups by the Human Resources and making sure each *HPD* group contains all the *HPD* relating to a Human Resource. This is because only all *HPD* bound up with a Human Resource can implicate his/her capabilities comprehensively, which also facilitates the follow-up processes that are human-centered.

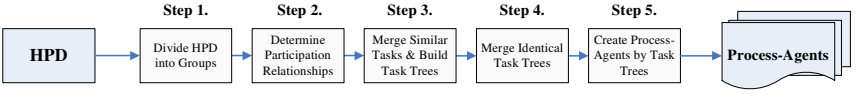


Fig. 5. The Five Steps to Create PAs from the HPD

Step 2. Determining the relationships between the Human Resources and the Tasks in each *HPD* group, making it clear in which Tasks the human has actually participated and in which didn't. The objectivity of the step is to find out what Tasks can be performed by the Human Resource itself and, what Tasks can be performed only by other Human Resources, so that we are able to explicate the coordination relationships between Human Resources.
Step 3. Merging similar Tasks that belong to different Projects in each *HPD* group, and constructing all the Tasks in the *HPD* group into a Task tree. Essentially, the similar Tasks implicate the same type of *PS*, so the Tasks need to be merged in order NOT to result in redundant PSs creations. Besides, *PK* is represented by a *PS* tree, hence constructing the Tasks into a Task tree can bring an ease to explicate the relationships between Tasks and *PS*s.
Step 4. Merging identical Task trees in different *HPD* groups. Taking into account that the identical Tasks tree in different *HPD* groups implicate a same *PK*, the identical Task trees need to be merged in order to prevent creating different *PA*s with a same *PK* (according to the definitions of PA in OEC-SPM, different *PA*s in a software organization are representing different Organizational-Entities who have different capabilities on the basis of different *PK*, so that it is not appropriate to allow different PAs to have the same *PK* in one software organization).
Step 5. Create *PA*s by the Task trees.

Details of the process will be illustrated in the next section by an example.

5 The Application of the Tool

In this section we present an example of applying the tool to create the PAs from the historical data of a software quality management system *SoftPM*.

SoftPM is a commercial platform for software quality management that provides comprehensive and effective supports for project management, process management and quality management in coordinated software development process. It has been applied in many areas and organizations in China, such as the national software industry parks, 863[1] software incubators and more than 200 software companies. *SoftPM* helps them define the standard and project's processes, establish and maintain the process assets library, perform project management and quality assurance tasks, collect the data for measurement, measure and evaluate the status of process performing and so on. The data of historical projects in *SoftPM* is stored in a database containing a series of data-tables and foreign-keys. The tool creates PAs by generating *HPD* from the database and then creating PAs from the *HPD*, the details of the process are given below.

5.1 Generating HPD from the Database of SoftPM

In this phase, the tool examines all the historical data in the database, based on which it generates *HPD* according to the data mappings depicted in Table1.

[1] 863 Program, National Hi-Tech Research and Development Plan of China.

Table 1. Data Mappings between SoftPM Database and HPD (Of all the mappings, only the portion of the mappings that is necessary for establishing the *Process Knowledge* of the *PA*s are displayed here)

	Data-Tables and Foreign-Keys in SoftPM Database		**Objects and Relationships in HPD**	
1	*project_info*	→	Project	
2	*task_info*	→	Name	Task
			Description	
			Start Time	
			End Time	
			Input Artifacts	
			Work Products	
3	*Tasks_belong_to_SpProcess*	→	Type	
	Tasks_belong_to_SpActivity			
	Tasks_belong_to_AcType			
4	*Parent_Task*	→	Kind	
	task_member			
	task_info			
5	*bs_employee*	→	Human Resource	
6	*Task_in_Project*	→	Relationship between Project and Tasks	
7	*Parent_Task*	→	Relationship between Tasks	
	task_info			
8	*project_member*	→	Relationship between Project and Human Resources	
9	*task_member*	→	Relationship between Task and Human Resources	
10	*user_task_group*	→	Role	
			Relationship between Role and Human Resources	
			Relationship between Task and Roles	

Each " " in Table1 explicates that the value of the object(s) and relationship(s) in the *HPD* are derived from the data-table(s) and foreign-key(s) in the *SoftPM* database.

5.2 Creating PAs from the HPD

In this phase, the tool creates *PA*s from the *HPD* generated by the primary phase via five steps (as mentioned in section 4).

Step 1. Dividing *HPD* into groups by the Human Resources.

(1) Creates a dataset *DS* for each Human Resource in the *HPD*, and we call the Human Resource is the *Human Resource of the DS*. Each *DS* contains the data relating to all the projects that the Human Resource was participated in as well as all the Tasks that belong to the projects. A *DS* would be null if relevant Human Resource did not participate in any projects.

(2) Removes all *DS*s that are null.

(3) Set the *Implementation* of Tasks to "*DIRECT*" in all the *DS*s.

After the step is accomplished, each *DS* would contain a set of Task trees acquired from the *SoftPM* database and each Task trees would belong to the different projects.

Step 2. Determining the participation relationships between Human Resources and Tasks in each *DS*. In this step, the tool traverses each Task in each *DS*, if a Task is not a Task participated by the *Human Resource of the DS*, the tool set the *Implementation* of the Task to "*Sub_Process*".

Step 3. Merging similar tasks that belong to different projects in each *DS*, and construct the Tasks into a Task Tree.

(1) Traverses each Task deep firstly in each *DS*, compares each Task with other Tasks in the same *DS*, if found two Tasks that belong to different projects are *similar tasks*, merges the tow Tasks by the algorithm given below.
   a) Reserves the Task *t* that is at the deeper level of the Task tree (the level of a Task in Task tree can be determined by the Relationship between Tasks).
   b) Create a copied Task *tt* containing all information of *t* and create a copied Task *tt'* containing all information of the unreserved Task *t'*.
   c) Remove all the children Tasks of *t*.
   d) Set the Kind of *t* to "Choice".
   e) Set the parent Task of *tt* and *tt'* to *t*.
   f) Set the Work Products of *t* to the union of the Work Products of *tt* and *tt'*.
   g) Set the Roles of *t* to the union of the Roles of *tt* and *tt'*.
   h) Removes *t'*.

(2) Repeats (1) until no similar Tasks can be found in the same *DS*.

(3) Check each *DS*, if the tasks in a *DS* are not yet constructed into one Task tree, create a new root Task *rt* for the *DS* and then.
   a) Set the Human Resources of *rt* to the union of the Human Resources of all the original root Task of the original task trees in the *DS*.
   b) Set the Work Products of *rt* to the union of the Work Products of all the original root Task of the original task trees in the *DS*.
   c) Set the Roles of *rt* to the union of the Roles of all the original root Task of the original task trees in the *DS*.

After step 3 is accomplished, all the Tasks in each *DS* would be constructed into Task trees.

Step 4. Merging identical task trees in different *DS*s.

Primarily we give two definitions before introducing to the step.

DEFINITION 1. Supposes T_1 and T_2 are two Task trees, t_i is a Task in T_1, the level of t_i in T_1 is n; t_j is a Task in T_2, the level of t_j in T_2 is m. For any t_i, if we can always find a unique t_j that is the similar Task of t_i which makes $t_i.n = t_j.m$, $t_i.Order = t_j.Order$ and for any t_j we can also find a unique t_i that is the similar Task of t_j, making $t_j.n = t_i.m$, $t_j.Order = t_i.Order$, we call T_1 and T_2 are the *identical Task trees*.

DEFINITION 2. Supposes T_1 and T_2 are the *identical Task trees*, t_i is a Task in T_1, the level of t_i in T_1 is n; t_j is a Task in T_2, the level of t_j in T_2 is m. If t_i and t_j are the similar Tasks and $t_i.n = t_j.m$, $t_i.Order = t_j.Order$, we call the t_j is the *corresponding task* of t_i.

Now we can present the processes in the step.

(1) Select arbitrary tow *DS*s from all the *DS*s, if the Task trees in the two *DS*s are the *identical Task trees*, perform the processing as follow
   a) Arbitrarily reserves a *DS*, for each task t_i in the reserved *DS*, find the *corresponding task* t_i in the unreserved *DS* and then
      i. Set the Human Resources of t_i to the union of the Human Resources of t_i and t_j.
      ii. Set the Work Products of t_i to the union of the Work Products of t_i and t_j.
      iii. Set the Roles of t_i to the union of the Roles of t_i and t_j.
   b) Remove the unreserved *DS*.

(2) Repeat (1) until NO identical Task tree existing in different *DS*s.

Step 5. Creating PAs by the final Task trees.

(1) Create a *PA* with an automatically generated *PA* name for each Task tree.

(2) Create a *PS* tree for the *PA* with the same hierarchy with the Task tree, making that each *PS* in the *PS* tress corresponds to a Task in the Task tree.

(3) Traverses the *PS*s, for each *PS*s and for its corresponding Task *t*
   a) Set the value of *SID* of *s* to a generated unique id.
   b) Set the value of *SD* of *s* to the Description of the *t*.
   c) Determined the *SCRM* of *s* by the hierarchy information and sequential information of *t* in the *DS*, that is
      - Make the hierarchy context of *s* consist with the hierarchy context of *t*.
      - Determine the sequence context of *s* by the time sequence context of *t*.

(4) Set the values of *IP* of *s* to the Input Artifacts of *t*.

(5) Set the values of *OP* of *s* to the Work Products of *t*.

(6) Set the value of IMP of s to the Implementation of *t*.

6 The Result of the Tool

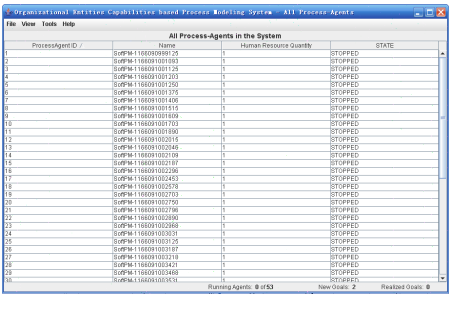


Fig. 6. The PAs Created by The Tool

We have applied the tool to a domestic software organization (due to confidential reasons, the name of the organization is not introduced here) at the CMMI Level 4, the organization has being adopted *SoftPM* as *SPMS* for 6 years and has accumulated a large amount of valuable historical data. The tool has created 53 PAs (Fig. 6) based upon the organization's data and the software processes of the organization have been represented by the *PK* of these PAs as expected.

For Example, the *PK* of *PA* with ID = 44 (Fig. 7) in the results explicates one of the requirement analyzing processes of

the organization, the *PK* of *PA* with ID = 12 (Fig. 8) explicates one of the software implementation processes, the *PK* of *PA* with ID = 46 (Fig. 9) explicates one of the testing processes, the *PK* of *PA* with ID = 35 (Fig.10) explicates one of the quality assurance processes and, the relationships among these processes can be determined by the *PK* of *PA* with ID = 5 (Fig.11), which produces the organizational software processes definition.

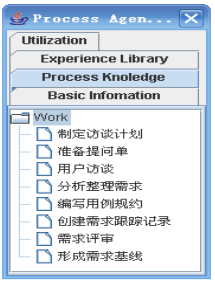


Fig. 7.

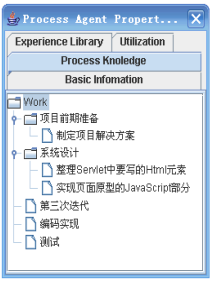


Fig. 8.

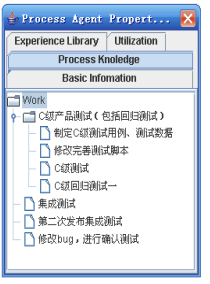


Fig. 9.

With the *PA*s created, the organization is enabled to generate its software processes self-adaptively with *OEC-SPM* owing to the process assets organized in the *PA*s, which will greatly improve the predictability and the stability of the organization.

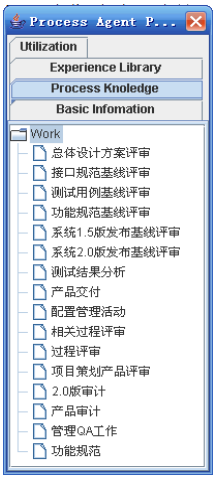


Fig. 10.

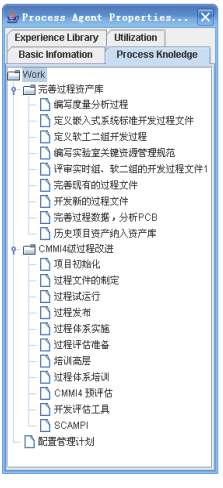


Fig. 11.

7 Conclusions and Future Work

The tool developed by us provides a strong support for creating the key element for OEC-SPM - *PA* by making use of the *HPD*. With the aid of the tool we significantly eliminate the dependence on human for creating PAs and also make the adopting of the OEC-SPM with ease and efficiency.

However, to create *PA*s by the tool needs a complete database of projects is managed at the present, taking into account it is not always realistic to have such a strong restriction, in the future works we will enhance the tool with comprehensive data pre-process operations including removing noise or outliers if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time sequence information and known changes, as well as deciding DBMS issues, such as data types, schema, and mapping of missing and unknown values[14], in order to increase the tool's usability.

Meanwhile, according to the result of the tool, although different software processes of the organization have been nicely presented by different *PA*s, the *PK* of the *PA* is not cohesive enough, which means the *PA*s have not been perfectly aggregated (for example, the PK in Fig. 8 represents a software implementation process. We found it also contains a *PS* referring to a software testing process, which is

unexpected). In the future works, we will enhance the tool by employing processes of extracting *DK* and *EL* that takes into account performance, experiences and other aspects of the human in the software development process to recognize *PA*s that present the same behavior pattern, so that we can make the knowledge and the *PA*s are further aggregated.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082, 60473060; the National Hi-Tech Research and Development Plan of China under Grant No. 2006AA01Z185, 2006AA01Z19B; the National Key Technologies R&D Program under Grant No. 2005BA113A01.

References

1. R. Conradi, A. Fuggetta and M. L. Jaccheri: Six Theses on Software Process Research. Proceedings of the 6th European Workshop on Software Process Technology, 1998, 100-104
2. C. G. Gianpaolo Cugola: Software Processes: A Retrospective and a Path to the Future. Software Process: Improvement and Practice, 4, 3, 1998, 101-123
3. R. Balzer: Keynote On "Current State and Future Perspectives of Software Process Technology". Proceedings of the 7th European Workshop on Software Process Technology, 2000,220
4. J. Xiao, L. J. Osterweil, L. Zhang, A. wise and Q. Wang: Applying Little-Jil to Describe Process-Agent Knowledge in Softpm. SPW/ProSim 2006, Shanghai China,2006,214-221
5. X. Zhao, M. Li, Q. Wang, K. Chan and H. Leung: An Agent-Based Self-Adaptive Software Process Model. Journal of Software, Vol. 15, No. 3, 2004, 348-359
6. X. Zhao, K. Chan and M. Li: Applying Agent Technology to Software Process Modeling and Process-Centered Software Engineering Environment. The 20th Annual ACM Symposium on Applied Computing(SAC'05), Santa Fe, New Mexico, USA,2005,1529-1533
7. Q. Wang, J. Xiao, M. Li, M. W. Nisar, R. Yuan and L. Zhang: A Process-Agent Construction Method for Software Process Modeling in Softpm. SPW/ProSim 2006, Shanghai China,2006,
8. L. Osterweil: Software Processes Are Software Too The 9th international conference on Software Engineering Monterey, California, United States 1987,2-13
9. L. J. Osterweil: Software Processes Are Software Too, Revisited: An Invited Talk on the Most Influential Paper of Icse 9. 1997,540-548
10. A. E. Hassan, R. C. Holt and A. Mockus. Proc. 1st Int'l Workshop Mining Software Repositories, 2004, http://msr.uwaterloo.ca/msr2004
11. Q. Wang and M. Li: Software Process Management: Practices in China. SPW 2005, 2005,317-331
12. K. A. Schneider, C. Gutwin, R. Penner and D. Paquette: Mining a Software Developer's Local Interaction History. IEE Seminar Digests, 2004, 917, 2004, 106-110
13. D. M. German: Mining Cvs Repositories, the Softchange Experience. IEE Seminar Digests, 2004, 917, 2004, 17-21
14. U. M. Fayyad, G. Piatetsky-Shapiro and P. Smyth: The Kdd Process for Extracting Useful Knowledge from Volumes of Data. Commun. ACM, 39, 1996, 27-34

Appendix: OEC-SPM

*O*rganization-*E*ntity *C*apability Based *S*oftware *P*rocess *M*odeling (OEC-SPM), which mainly aims at the software process particularities, is a modeling method presented by ISCAS to model standard processes. OEC-SPM defines an organizational entity that holds certain capabilities as a Process-Agent and regards the Process-Agents as the core elements and the basic units of the software process. Process-Agents produce the concrete software development processes and the production processes via the proactive and autonomous reasoning based upon their goals, knowledge, experiences and capabilities under a defined environment containing project goals and constraints so as to provide software project development with effective supports and proper decisions. Owing to its full consideration of the capabilities of process executors, OEC-SPM has the merits of producing software processes with good predictability, which resolves the instability and the uncontrollability of the software processes.

Safety Critical Software Process Improvement by Multi-objective Optimization Algorithms

Mario Brito and John May

Safety Systems Research Centre, University of Bristol, Queen's Building,
Bristol BS8 1TR - United Kingdom
{Mario.Brito,J.May}@bristol.ac.uk

Abstract. One of the main concerns in safety critical software development is to identify a path through the software development lifecycle that will allow the software artefact to meet the target safety integrity level (SIL) at an acceptable cost. In our previous work we modelled aspects of the software development process recommended by IEC61508-3 software safety standard. In general, there are a number of paths that one can follow in order to comply with a target SIL. The path that one chooses to follow will undoubtedly effect the costs of the software development. In this paper we study a series of optimization algorithms that can be used to improve the software development process by optimization of two objectives, development costs and confidence in claimable integrity. Our analyses show that the non-dominated sorting genetic algorithm (NSGA) is the best performing algorithm in the search for these optimal processes.

Keywords: Software Safety standards; Bayesian belief networks, Genetic Algorithms.

1 Introduction

The development of safety critical software is typically guided by software safety standards such as IEC61508-3 [1]. This standard recommends the techniques to be applied whilst developing a safety function that will monitor and control the risk posed by the operation of the main system (e.g. this can be a railway management system or a nuclear power station management system). The software development process recommended by the IEC61508-3 follows the V diagram for a software life-cycle, which is described as a collection of phases. The techniques to be applied throughout the development lifecycle are selected based upon the safety function target SIL, which is determined in the system hazard analysis. The SIL varies from 1 to 4 where SIL 4 is the highest integrity. For a given SIL target there is number of combinations of techniques that can be applied in each phase of the development lifecycle. Consequently, there are different paths that the decision maker can select in order to develop a software product that complies with the SIL target. In previous work we captured a part of the software development process recommended by IEC61508-3 in a prototype Bayesian Belief Network (BBN) [2]. Throughout this

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 96–108, 2007.

paper the term 'process model' is used when we are making reference to the BBN representation of the software development process. Because this type of model is executable it may also be called a simulation model. The use of BBNs injects a high degree of transparency into the software development process, makes it easier to identify different paths of the software development lifecycle and estimate the integrity level that one can claim if one chooses to follow a particular path. This 'expert system' was designed based upon interviews with software engineers working in the development of the IEC61508 safety standard.

In this paper we present a decision support system (DSS) that we embedded in our expert system to expand its functionality. The DSS allows a project manager (user) to perform cost efficiency analysis whilst complying with the target SIL. The proposed approach has two advantages: 1) It provides support for software project risk management; and 2) It allows organizations to perform detailed self assessment of their own process. Point 2 is important because it is common for organizations to create their own process. However in order to certify their product they must build an argument stating that their process comply with recommendations made by the standard. The proposed DSS will help organizations to build a more robust argument as to why they should adopt a process (or not). It allows the project manager to choose which process to follow to best control two key attributes software integrity and its associated development costs. The proposed DSS uses a type of Multiple Objective Evolutionary Algorithms (MOEAs) called a Non-Dominated Sorting Genetic Algorithm (NSGA) [3].

The BBN-based expert system has several input nodes. Example of such input nodes are for instance, the experience of the development staff, the power of the techniques applied, the number of project review meetings and the intensity at which the development techniques were applied. Use of the expert system typically involves the decision maker specifying a state for each node in a subset of the BBN nodes. There are in total 28 input nodes for the BBN that models phase 1 of the software development lifecycle recommended by IEC61508-3 "software requirements specification". Each node has on average 5 possible states. The 'power of the formal method' node provides an example of a node and its states: {very poor, poor, medium, good, very good}. Instantiation of the states of all the input nodes defines a particular 'scenario' or 'path' or 'process' of development, that will have an associated effectiveness (in terms of the claimable integrity), and also associated costs.

MOEAs have the same working principles as the single objective GA [4]. The first commonality between the two approaches is that both are based on the random creation of an initial population of possible solutions (also referred as individuals). Depending on the nature of the problem, the fitness of each individual is established by seeing how well the individual maximizes or minimizes the pre-defined objective function. The next generation of individuals is then created by manipulating the fittest individuals of the earlier generation using mutations and crossovers. For a normal GA algorithm there is one objective function only, whilst for a multiple objective problem there are two or more objective functions. Thus in the presence of conflicting objectives the optimization algorithm will look for a set of optimal solutions, which are said to form the Pareto front [1].

[1] The Pareto front is the set of solutions that are optimal with respect to two or more objectives.

The optimization algorithm was implemented in Visual C 6.0 and it communicates with the expert system through the Hugin Application Interface (API) [5],[6]. In brief, this optimization algorithm collects 'rigid evidence' (this is evidence relating to facts that are fixed for a given project, that are captured in terms of specified values for BBN input nodes and that constitutes a set of constraints for the optimisation algorithm) and runs "what-if" scenarios with the uninstantiated input nodes until it finds the most cost efficient set of values at those nodes. The algorithm can ask "what-if" queries. For instance, if the intensity at which formal methods were applied were increased, say from verifying a few key properties of the software requirements to verifying all required properties. Similarly, "what-if" we increase the number of the project review meetings? Given a set of user-specified fixed factors (constraints) the optimization algorithm will run all possible remaining scenarios in order to find the most cost efficient solution or set of solutions.

The remainder of the paper is organized as follows. In section 2 we give a background on the type of expert system proposed. In Section 3 we compare different MOEAs namely, MOGA, NPGA, SPGA and NSGA in order to select the algorithm that will support the proposed DSS. In section 4 we present examples illustrating the application of the proposed method. Section 5 presents our analysis and conclusions.

2 The Process Model

The use of BBNs continues to progress in the field of software dependability. Research projects using this approach include *FASGEP*, *Datum, SHIP, DeVa*. The general approach for these projects uses the underlying assumption that errors are introduced during development and models of this phenomenon will allow the project manager to assess the level of the problem. This is important aim since if necessary, the manager can then take preventive measures to remove errors or otherwise mitigate against the effects of errors on system dependability [7],[8],[9],[10]. Most of these BBNs were developed using expert opinion. Methods borrowed from social sciences are usually applied throughout the elicitation exercise in order to reduce bias [11],[12]. The validation of BBNs based expert systems is done by giving to the expert system unseen scenarios and seeing if its predictions match those of the human expert. A thorough discussion of validation of a BBN based expert system is given by Cockram [13].

A BBN is a type of graphical probabilistic model (GPM) in which nodes represent random variables (continuous or discrete) and arrows represent causal influence that one variable (parent node) has upon another variable (child node). The strength of the connections between subsets of nodes is defined in condition probability tables (CPTs). Typically, when using a BBN one inserts evidence at a node by specifying its value; and the evidence is then propagated through the network updating the belief in the states of the other nodes. Details about propagation algorithms for Bayesian networks are given in [14],[15].

The process model used in this paper is an attempt to capture part of the software development process recommended by the IEC61508-3 software safety standard. The model has been reviewed by experts working in the development of IEC61508-3. The model consists of two networks; a single phase network is used to capture the set

of activities undertaken in each phase of the development lifecycle and a skeleton network is used to capture the interactions between all phases of the software development lifecycle. The single phase network is designed to be generic, in the sense that nodes in particular positions in the graph structure are different in general, but have the same 'type'. Similarly, there may be a different number of nodes of a given type in different phases [2].

The central purpose of the single phase network is to estimate the likely criticality of outstanding errors introduced in the current phase of the software development. This is estimated in the 'significance of outstanding errors…' node (for an example see Fig. 5). This node has the following discrete states, {intolerable, undesirable, tolerable, neglibible}. The probability distribution for this node is obtained based on estimates for the quality of the development techniques and estimates for the quality of the review techniques involved in the phase. The quality of development is estimated based on the rigour at which the development techniques were applied, the complexity of the design task, the competence of the staff involved and an 'application factor'. The latter node is required due to the nature of IEC61508 safety standard. The standard is meant to be adapted to diferent industrial sectors and these may follow subtly different beliefs as regards what is considered rigourous software development. The quality of review is estimated based on predictions for states of the competence of the staff involved, the independence level between the staff carrying out the review and those involved in the design, the rigour at which the review techniques were applied and also the relevance of the review techniques for that particular phase. Depending of the current phase of the project the relevant verification technique might be project review meeting, code review, code walk-through, dynamic testing, formal proof and so on. The power of a particular verification technique for a particular phase is measured with the 'power of the verification technique' node, this node has the following states: {very poor, poor, moderate, good, very good}. The review at phase 2 is not limited to finding errors created in that phase. It can also find errors that are relevant to phase one. More generally, reviews at any later phase can find errors that are relevant to any earlier phase. This creates an interesting feedback mechanism that has to be captured in the BBN model and for this purpose we introduced the larger 'skeleton' network. This network captures interaction between different phases of the software development process, Fig. 1.

In principle, the calculation of confidence in integrity claims can be conducted at any point in the lifecycle, for example, after just one phase has been completed. It is then updated as subsequent phases are performed and the BBN grows dynamically to describe the work performed. This network is used to model interactions of two types. The first type of interaction is the one discussed above, where error finding in later phases effects integrity claims for earlier phases. The second type of interaction is the combination of integrity claims of consecutive phases into one overall claim. Fig. 1 presents the 'skeleton' network that computes the combined effects of different phases. This skeleton estimates a probability distribution for 'Overall integrity after phase i'. All these nodes have the following states: {SIL1, SIL2, SIL3, SIL4}.

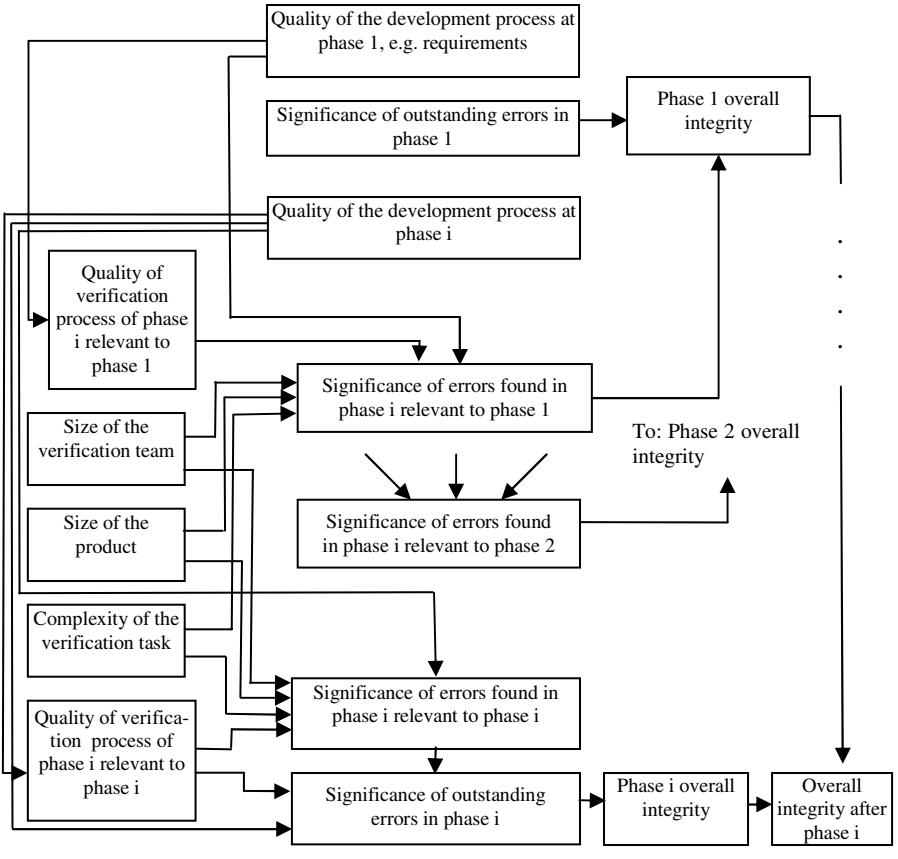


Fig. 1. Generic BBN Multi-Level structure for several phases of the safety software development lifecycle

3 Meta-heuristics for Decision Support

The proposed approach is presented in Fig. 2. This diagram contains five key elements; the 'project' element represents the information that is known with certainty about a specific project; these are the constraints of the project that are rigid and cannot be changed during process optimisation. For any given set of input evidence concerning a particular process, the BBN (process model) will compute the confidence that the target SIL can be claimed. In addition our approach also makes use of a database that contains the costs of adopting a given process. In summary, in order to use this system the decision maker 'Manager' will first identify the project constraints; these might be the project size, complexity or the type of application. These constraints will specify the states for nodes that cannot be used by the DSS for the optimization of the software development process. Given the identified constraints the optimization algorithm will run different scenarios. For each scenario the algorithm reads the SIL claim and also the cost of the product development. The SIL probability

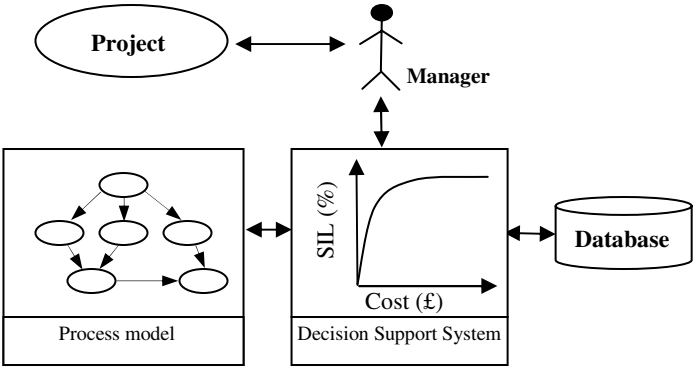


Fig. 2. Framework of the general approach to risk management

distribution is computed by the BBN whereas the cost of a particular technique is read from the database.

The proposed DSS uses a meta-heuristics optimization algorithm. Meta-heuristics optimization algorithms provide a powerful approach for optimum search. An obvious advantage of such algorithms is that they do not need to assess all possible solutions of the search space in order to find a global optimum or approximate global optimum.

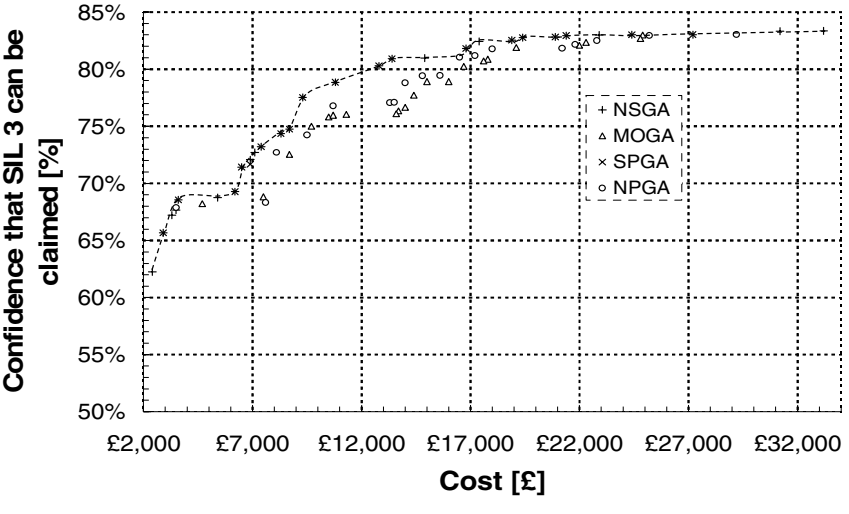


Fig. 3. Non-dominated solutions after 100 seconds running time

These algorithms have been successfuloply applied in a wide range of subjects [4]. Multi-objective Evolutionary algorithms (MOEAs) are among the most effective multi-objective optimization algorithms. Several MOEAs have been developed in the past ten years. Among these the MOGA, NPGA, SPGA and NSGA are considered to be the most effective [16],[17],[18],[3] respectively. We implemented these four

algorithms and demonstrated that for a specific set of constraints and many examples of input evidence they all converged to the same Pareto front. We chose to implement NSGA because this algorithm is the fastest to get to that Pareto front. The NSGA algorithm is described in [3]. Fig. 3 illustrates the solutions found by each of the four optimization algorithms after 100 seconds run. The NSGA managed to find the most number of non-dominated or optimal solutions. The algorithm finds all non dominated solutions found by SPGA (second best algorithm) plus 10 non-dominated solutions/processes. For our particular problem the slowest algorithm is MOGA.

4 Using the Decision Support System

In this section we provide two case studies. The first study presents the case where one intends to optimize phase one "software requirements specification" of the software development lifecycle. The second case study presents the scenario where one has completed phase one of the software development lifecycle and intends to optimize phase 2 "software architecture specification".

4.1 Case Study 1 - Optimization of the Software Requirements Specification Phase

This case study analyses the software requirements specification phase of the software development lifecycle. It is assumed that the project manager has a clear idea of the size of the project and its complexity. The project size is 'small' and the complexity of the design and verification activities was considered 'fair'. It is also assumed that he/she knows the number of people required for the job. The project manager now wants to know which development techniques to apply, the required qualification of the staff and the type of verification techniques to apply. The target node (the node whose probability distribution we are interested in) for this case study is the 'Phase 1 overall integrity'. We will consider two SIL targets, SIL 3 and SIL 4, the latter requiring the application of more powerful techniques.

Fig. 4 presents the Pareto front obtained for this case study. Each data point represents a process, i.e., a combination of techniques applied, the intensity at which they were applied and also the competence of the personnel involved in the development and in the review activities. Data point 1 in Fig. 4 represents the cost-optimal process to follow in phase one in order to attain 83% confidence that SIL 3 can be claimed. The cost of the associated process is £2,300. In terms of techniques, this process (or scenario) involves the application of: a powerful formal method at a low intensity (in practice this might mean use of a formal method to provide a few key validated system properties, or maybe just the use of formal specification without any formal validation activities); a 'very good' semiformal method at a 'low' intensity; a 'moderate' verification method, such as a formal design review meeting at a 'low' intensity; development staff with satisfactory training and moderate qualifications, but lacking in experience (i.e. 'low' technical knowledge and 'low' experience); highly experienced verification staff (experience node was set to 'moderate' and technical knowledge node set to 'good'); and a high level of independence between the design team and the review team.

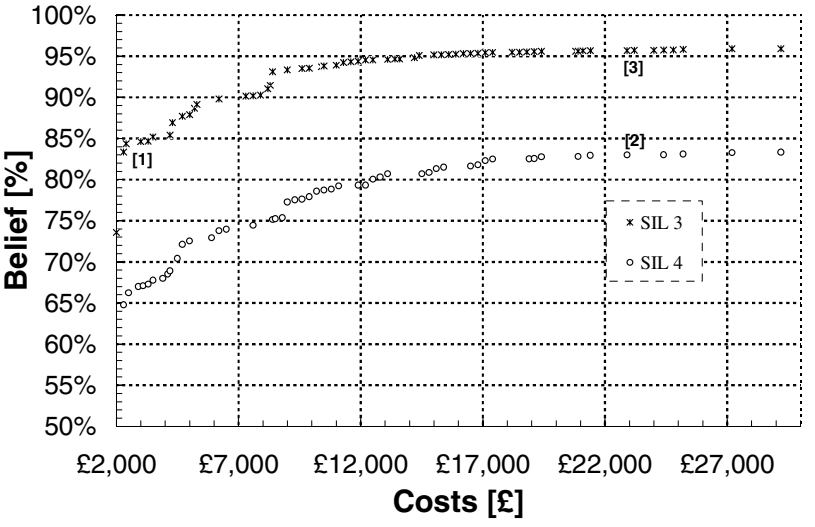


Fig. 4. Non-dominated solutions after 2000 generations

If the project manager wanted to have the same level of confidence (83%) that the software could claim SIL 4 instead of SIL 3 then he would have to follow the process corresponding to data point 2 in Fig. 4. For this process, a 'very good' formal method was applied at a 'very high' intensity and a 'good' semi-formal method at a 'very high' intensity. The qualifications of the design staff are satisfactory and the experience and technical knowledge is 'high'. The verification activities followed in this process are identical to the verification activities followed in the process for data point 1. However the qualifications, training, experience and technical knowledge of the personnel involved in the verification process is 'high'. The independence level between the two teams is also 'high'. The process present in data point 2 is similar to the process present in data point 3. If one was to follow the process present in data point 3 then one would attain 96% confidence that SIL 3 could be claimed. On the whole the processes in data points 1 and 2 mirror the findings presented in industrial reports. Concerning conformance to SIL 4, both Smith and Rivett in [19],[20] respectively argue that a formal specification should be carried out for the complete system, which in our example is addressed by the process represented by data point [2]. For SIL 3 however the two authors hold different views; whilst Smith argues that a semi-formal specification for the complete system, Rivett suggests that a formal specification should be presented for merely those functions that ought to meet SIL 3. In our example the optimal process (present in datapoint [1]) two techniques (formal and semi-formal specification methods) are applied at a low intensity.

4.2 Case Study 2 - Optimization of the Software Architecture Design Phase

The scenario presented in this section assumes that the project has reached the start of the 'software architecture design' phase. That is, a set of procedures has already been applied in phase 1. The question we address is 'which procedures should we now

apply in phase 2 in order to achieve compliance in the most cost efficient way?' For this case study we assumed that the organization that will use the software needs to comply with SIL 3. The target node is the 'Overall integrity after phase 2'. Fig. 5 is a screenshot of BBN tool Hugin 6.6. This figure illustrates our interpretation of the process model recommended by IEC61508-3 for the "software architecture specification" phase. At the top left corner of the figure are the nodes concerning the techniques recommended by IEC61508-3 for the software architecture specification.

Fig. 5. Hugin screenshot of the process model for the 'Software Architecture Design' phase of the IEC61508 software development lifecycle

Because of the interactions discussed earlier, the answer to the question will be dependent on what has happened in phase 1, in addition to the techniques available in phase 2. The following assumptions for the development in phase 1 were made: formal methods were applied at a 'high' intensity, whereas semiformal methods and computer aided specification tools were applied at a 'moderate' intensity. The staff involved in the development was also involved in the review. The competence of the staff was 'moderate'. This yields the following distribution for the confidence of the

overall SIL that can be claimed for phase one {SIL1:2.84, SIL2:12.46, SIL3:28.66, SIL4:56.04}. This means that the probability/Belief/confidence that SIL 4 can be claimed given the process evidence is P(SIL 4 | Process evidence) = 0.56. After phase 1 one could say with a belief (confidence) of 85% (28.66+56.04) that the software complies with SIL 3. The optimization algorithm investigated scenarios for phase two in order to identify those processes that maximize the belief that SIL 3 can be claimed whilst minimizing the costs. The Pareto front obtained for this case study is presented in Fig. 6: see the diamond data points.

The practical question that arises concerns the trade-off between integrity gain and increase in project costs. For instance, considering the diamond data point 1 and 2, the amount of integrity gained by following the process recommended in data point 2 perhaps does not justify the increase in costs. The main differences between the process recommended for data point 1 and data point 2 are as follows. Concerning the development process, for data point 2, a ' very good' formal method was applied at a 'very high' intensity, a 'very good' semi-formal method was applied at a 'very high' intensity and a 'very good' computer aided specification tool was applied at a 'very high' intensity, structured methods were not applied. For data point 1, a 'very good' formal method was applied at a 'low' intensity, a 'very good'semi-formal method was applied at a 'low' intensity and a 'very good' structured method was applied at a 'low' intensity. Considering the review, in both data point 1 and 2, a 'very good' review was applied at a 'very high' intensity. In addition the competence of the staff involved in development is higher for the process followed in data point 2 than it is for the staff involved in the process present in data point 1.

The competence of the staff involved in the process presented in data point 1 is considered to be moderate. These factors explain the increase in costs from data point 1 to data point 2. However the difference in confidence that the product complies with SIL 3 remains small. In this case, the reason for this is because the development in phase 1 introduces constraints on the maximum integrity that can be claimed at later phases. In some cases, the only way to increase the integrity claim after phase 2 is by simulataneously increasing the integrity at phase 1 and at phase 2. Because phase 1 has been completed, the only way to increase its integrity is by finding errors in later phases that are relevant to it (or to revisit, and therefore change the process). However it is plausible to assume - and this is built into the model - that it is difficult to find some errors introduced in phase 1 using techniques in phase 2, although this depends on the nature of the error. This is why, even if one attempts to invest an extra £10k (point 2) in phase 2, the gain in integrity is not significant.

To further illustrate this point, we considered another scenario where we introduced further constraints into phase 1. For this example we considered that formal methods were not applied at phase 1 of the software development lifecycle. The remaining nodes of phase 1 were populated with the same evidence as for the previous example in this case study. This yields the following distribution for the overall integrity that can be claimed for phase 1 {18.09, 20.48, 14.47, 46.96}. Given this probability distribution, after phase 1, one can say with 61% confidence that the software product complies with SIL 3. Again we ran the optimization algorithm in order to optimize phase 2 so that we could find those processes that would maximize our belief that SIL 3 can be claimed whilst minimizing the additional costs.

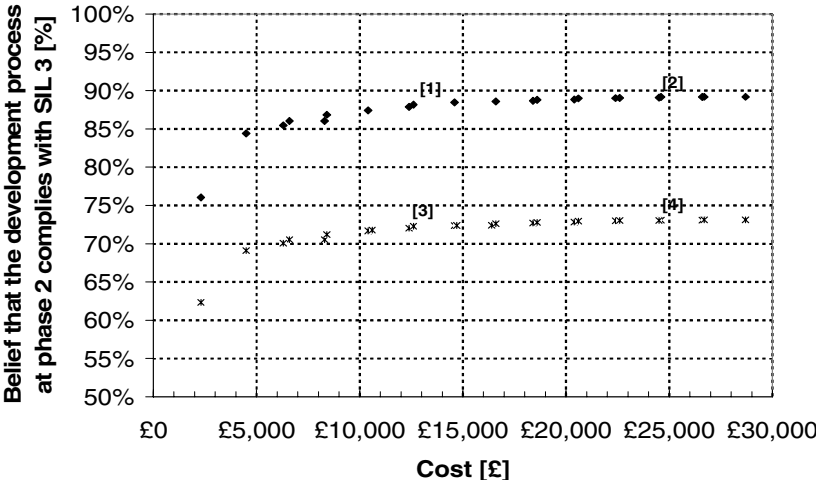


Fig. 6. Non-dominated solutions after 500 generations

The Pareto front obtained for this scenario is represented in Fig. 6 with the cross shape data points. Again the Pareto front shows that there is a point (point 3) where diminishing returns applies; no matter how much one invests in the development the increase in the overall SIL is almost insignificant (point 4). The differences between the techniques recommended in data points 3 and 4 are as follows. Concerning development, point 4 applies a 'very good' formal method at a 'moderate' intensity, a 'very good' semi-formal method at a 'very high' intensity, a 'good' computer aided specification tool at a 'moderate' intensity and structured method were not applied. The process followed in data point 3 is identical to that followed in data point 1. Both processes followed the same review activities. They both applied a 'very good' review technique at a 'low' intensity. Similarly to the example within this case study here overall integrity that can be claimed for a later phase is limited by the quality of the development in an earlier phase.

5 Discussion

The BBN encodes an understanding of how development processes should affect integrity claims. The BBN we used is not proposed as the 'correct' understanding of this (controversial) proposed correlation. The purpose of a BBN in this application is to make the assumptions transparent and act as a framework and tool for experts to use to improve the accuracy and self-consistency of their model of this correlation.

Developing safety critical software is often a costly and error prone process. The proposed DSS offers an interesting method to find a cost efficient set of techniques to follow in order to meet a target SIL. This is important information to support managerial decision-making regarding two key attributes, software product integrity and development costs, and their relationship. In one organization the project manager may be able to choose to increase the software safety integrity but will want to do so

in as cost efficient manner as possible. In another organization, the project manager may choose to investigate whatever is possible in terms of integrity within a fixed budget, and use that to decide whether to go ahead with a project. The latter is a potential use of the tool in a contractual context, namely, to provide evidence to a purchaser that the required software integrity can be achieved at the quoted cost.

With example one, we discussed processes present in the Pareto front obtained if one is targeting SIL 3 and SIL 4 for phase 1 of the development lifecycle. The results capture the simple idea that in order to have an effective and cost efficient process one ought to employ an experienced team to carry out review activities. Perhaps controversially, the BBN as built suggests that the experience of the personnel involved in the development process (for requirements capture) does not necessarily need to be high, provided that they have satisfactory training and good qualifications. This is clearly a point on which one might question the knowledge encoded in the BBN.

In our second example, we considered the case where the project is at phase 2 of the software development lifecycle, and the target integrity is SIL 3. The key observation is that in this case the BBN encodes the sensible phenomenon that the overall integrity that can be claimed after phase 2 is constrained by development in phase 1. This is specially highlighted when we presented our second Pareto front, illustrated with the crosses data points in Fig. 6. Here we considered the case where the development in phase 1 is poor. The only way to improve the integrity assessment for phase 1 whilst in phase 2, is to use phase 2 review techniques that are able to find phase 1 errors. However, it is reasonable to assume that due to the different nature of the errors introduced at phase 1, some errors would only be found in other phases of the development lifecycle that were not considered in case studies presented in this paper.

The proposed DSS is based on genetic algorithms however it might be possible to improve the performance of the DSS using a different type of meta-heuristic optimization algorithm such as simulated annealing or tabu search. Simulated annealing is known for facilitating proof of convergence. The only drawback is that this algorithm is suitable for single objective optimization problem; further research would be needed in order to adapt simulated annealing to our multi-objective optimization problem [21]. Tabu search on the other tackles an important issue in global optimization, namely, the multiple evaluation of a solution. Such algorithm might provide a faster trajectory to the Pareto front. A further potential enhancement to the current approach would be to add another objective to our DSS, for example time (or effort). The DSS would then search for those solutions that maximize the confidence in the product integrity whilst it minimizes the costs and the time taken to develop the product.

A potential drawback is the huge subjectivity involved in building the BBN. Whilst this subjectivity certainly exists, the nature of the reasoning performed lies behind current international software safety standards, where it is in a gross form lacking transparency. This notwithstanding, it is the basis of current best practice. The probabilistic reasoning lying behind these standards should be made more explicit, and the use of BBNs, whilst it probably can not remove much of subjectivity, does inject much needed formalism into process based dependability regulation, and does ensure coherency i.e. experts can not propose self-contradicting data.

Acknowledgments. We would like to thank our partners the UK Health and Safety Executive and Stirling Dynamics Ltd for their support.

References

1. IEC61508. 1998-2000. Functional safety of electrical/ electronic/ programmable electronic safety-related systems parts 1-7. Published by the International Electrotechnical Commission (IEC), Geneva Switzerland
2. Brito, M., May, J.: Gaining Confidence in the Software Development Process Using Expert Systems. In J. Gorski (ed.), Computer Safety, Reliability and Security. Lecture Notes in Computer Science Vol. 4166. Springer. Procs of the 25th International conf. on Computer Safety, Reliability and Security (SAFECOMP 2006), Gdansk, Poland, 26-29 September (2006).113-126
3. Srinival,N.,Deb,K.: Multi-objective function optimization using non-dominated sorting genetic algorithms.Evolutionary Computational Journal 2(3), (1994), 221-248
4. Goldberg, D. E.: Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley. (1989)
5. Hugin A/S: http://www.hugin.com
6. Hugin Expert A/S. 1990-2005. Hugin API Reference Manual version 6.4
7. Hall, P. et al.: Integrity Prediction during Software Development. Safety of Computer Control Systems. (SAFECOMP'92), Computer Systems in Safety-Critical Applications, Procs of the IFAC Symposium, Zurich, Switzerland, October 28-30, (1992), pp. 239-244
8. Littlewood, B., Wright, D.R.: Proceedings of the 14th International Conference on Computer Safety (SafeComp'95), Springler (1995). pp 173-190
9. Delic, K.A., Mazzanti, F., Strigini, L.: Formalising a software safety case via belief networks, Proceedings DCCA-6, Sixth IFIP International Working Conference on Dependable Computing for critical Applications, Garmisch-Partenkirchen, Germany. (1997)
10. Fenton, N.E., et al: Assessing dependability of safety critical systems using diverse evidence. *IEE Proceedings Software Engineering* **145** 1 (1998), pp. 35–39
11. Morgan, M. G., Henrion, M.: Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis, Cambridge University Press, Cambridge, UK. (1990)
12. Savage, L. J.: Elicitation of Personal Probabilisties and Expectations. Journal of the American Statistical Association, vol. 66, no 336 (1990), pp. 783-801
13. Cockram, T. Gaining confidence in software Inspection using a Bayesian Belief Model. Software Quality Journal, vol. 9, no. 1, (2001), pp. 31-42
14. Pearl, J.: Probabilistic reasoning in intelligent systems, Morgan Kaufmann (1988)
15. Spiegelhalter, D.J., Dawid, A.P., Lauritzen, S.L., Cowell, R.G.: Bayesian Analysis in Expert Systems. Journal of Statistical Science, vol .8, no. 3, (1993), pp. 219-283
16. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms,(1993), pp. 416-423
17. Horn, J, Nafploitis, N., Goldberg, D.: A niched Pareto genetic algorthm for multi-objective optimization. In Procs 1st IEEE Conf. on Evolutionary Computation, (1994), pp. 82-87
18. Zitzler, E., Thiele, L.: An Evolutionary algorithm for multi-objective optimization: The strength Pareto approach. Technical report 43, Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) (1998)
19. Smith, D., Simpson. K.: Functional Safety – A straightforward guide to applying IEC61508 and related standards. Elsevier (second edition), ISBN: 0750662697 (2004)
20. Rivett, R.S.: Emerging Software Best Practice and how to be compliant. Proceedings of the Sixth International EAEC Congress (1997)
21. Aarts E, Korst, J.: Simulated Annealing and Boltzmann Machines – A Stochastic Approach to Combinatorial Optimization and Neural Computing. John Wiley & Sons, (1989)

Representing Process Variation with a Process Family

Borislava I. Simidchieva, Lori A. Clarke, and Leon J. Osterweil

Laboratory for Advanced Software Engineering Research (LASER),
University of Massachusetts at Amherst, 140 Governors Drive, Amherst, MA 01003
{bis,clarke,ljo}@cs.umass.edu

Abstract. The formalization of process definitions has been an invaluable aid in many domains. However, noticeable variations in processes start to emerge as precise details are added to process definitions. While each such variation gives rise to a different process, these processes might more usefully be considered as variants of each other, rather than completely different processes. This paper proposes that it is beneficial to regard such an appropriately close set of process variants as a process family. The paper suggests a characterization of what might comprise a process family and introduces a formal approach to defining families based upon this characterization. To illustrate this approach, we describe a case study that demonstrates the different variations we observed in processes that define how dispute resolution is performed at the U.S. National Mediation Board. We demonstrate how our approach supports the definition of this set of process variants as a process family.

Keywords: process families, process variation, process variants, process instance generation, software product lines.

1 Introduction

Process definitions are used as vehicles for improving coordination, communication, automation, and efficiency in teams that are developing software [19, 31]. Increasingly, process definitions are also being used to improve the functioning of teams in domains as diverse as manufacturing [10], medicine [7, 18], business [14, 40], and science [2, 33]. In all of these domains one can readily find processes that are widely used to discipline the way in which key aspects of the work are to be carried out. In earlier work we have begun to define the specific processes of interest in these domains; as we have elaborated these processes to lower levels of precise detail, however, we have started to observe that different team members often perform the process in ways that differ from each other. Although the differences may seem to be primarily differences in detail, process details can matter a great deal. Thus, it is necessary to consider how each of these differences produces a process variant, which is indeed a different process.

The existence of a proliferation of different processes would seem to complicate efforts to improve coordination, automation, and improvement in that it raises the question of which process is to be used to gain these improvements. Our observation is that some of the differences may indeed be profound, but that many differences might best be thought of as variations on a high-level process that is generally agreed

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 109–120, 2007.

upon. If this is the case, then a large set of different variations might be thought of, and defined as, a process family, and that the process family might itself then be used as the basis for coordination, automation, improvement, training, etc.

In this paper we begin exploring the validity and applicability of the idea of process families. We attempt to characterize the sorts of variation that might be allowed within a family, and ways in which such variation might be represented. We validate our ideas by means of a case study, exploring the variation that we have observed in the course of defining the process of conducting a mediation at the U.S. National Mediation Board, and evaluating a specific approach to representing some of the principal forms of variation.

2 Related Work

Some particularly good summaries of work in software families, product lines, and vatiation are [30, 41]. Svahnberg et al. [38], in addition, present a taxonomy of different variability realization techniques. Jacobson et al. [20] describe some commonly used techniques to support software reuse, where variability is a main issue including inheritance, extension and extension points, parameterization, templates and macros, configuration and module interconnection languages, and generation of derived components. Other approaches include conditional compilation and dynamic binding [13], aspect- and feature-oriented programming [24, 35].

Generation approaches (e.g. [4, 8, 25]) seem particularly relevant to our work. In [8], the authors discuss the relation of feature models to various generative programming techniques such as inheritance and parameterization. In [34], the authors propose using component generators to support dynamically configurable components in software product lines. Moreover, these approaches often use a configuration specification as the basis for generation. This is similar to the notion of diversity interfaces introduced in the Koala model [39], where mechanisms such as switches, modules, and dynamic bindings of components capture variability and diversity. Jarzabek's XVCL (XML-based Variant Configuration Language) language [21] also describes systems in terms of variations and uses a generator to bind the variation points to specific variants. Work on decision models uses specifications to guide generation. KobrA [3] seems to have strong similarities to our proposed approach. KobrA extends UML models with decision models to describe the variability of components. In KobrA, each variation point is related to decisions and each component is associated with a decision model in addition to its structural, behavioral and functional models. A decision model is a list of decisions, a set of possible resolutions to each decision, and the possible effects of each resolution on the UML diagrams. The concept of decision models is also used in FAST [41] to support instantiation of domain models. Feature-oriented approaches have also been proposed to model variability. The FORM method [27] develops reusable and adaptable domain artifacts by using a feature model using AND/OR graphs where AND nodes indicate mandatory features and OR nodes indicate alternative features. A similar feature model is proposed by Griss et al [16]. Feature graphs, however, can become quite complex and unmanageable even for domains of reasonable sizes.

Our work is related to previous work in collaboration and group support systems, such as Group Systems ThinkTank or Facilitate.com, that support group decision

making. Such tools implicitly define a process family by offering configuration options (e.g., are contributed ideas anonymous?) and by letting session "owners" change configurations dynamically (e.g., enabling categorization). We believe that our approach of providing vehicles for explicit representation of the process offers clear advantages. Groupware systems fall into three broad categories: 1) Systems that are "process-agnostic" such as Groove, WebEx, SameTime, or Caucus, 2) domain-specific tools, like the group support systems mentioned above, and 3) groupware toolkits (e.g, Lotus Notes) that support building groupware tools with a programmer-specified embedded process. Neither the "process-agnostic" nor the domain-specific tools support explicit representation of the process being executed, and while they may passively support a range of processes, they cannot support understanding the relationships between them. Similarly, while groupware construction tools support explicit coding and thus conformance of a single member to a process family, they cannot support clear representation of, and thus reasoning about, the family.

We strongly concur with Briggs, who argues that collaboration research should focus on "technology supported collaboration processes" instead of "collaboration technology" [6]. This idea is echoed in [26] that discusses the "emerging field of Collaboration Engineering" which is "an approach that designs, models, and deploys repeatable collaboration processes".

Much literature addresses modeling of software development processes, with an increasing focus on modeling workflows [14, 28, 40], business processes, and service architectures [1, 12]. There is far less focus on processes for government applications, and on negotiation and dispute resolution. Many approaches are based upon the use of a flowgraph model of the process [2, 29]. Others use such formalisms as finite state machines [17, 23] or Petri Nets [11, 15]. In some cases, the formalism recognizes the need to also model artifacts [37], often by using simple type systems. In fewer cases, the need to model resources and agents is also recognized, and again these models are usually simplistic [5, 9].

3 Approach

We propose applying the software product family approach to processes as a way to handle process variation. By creating a number of processes, which are all variations of one *metaprocess*, or alternatively, by creating one metaprocess to span an entire group of related processes, a process family is effectively generated. For several processes to be members of the same family, they must be sufficiently similar, i.e., contain a common core that is identical or slightly different across processes. Often, it may be difficult to identify a core but these processes may still belong to the same family if, at a higher level of abstraction, they are determined to be the same.

We hypothesize that all necessary process instances can be generated from a framework, perhaps with the aid of some sort of specification of required process goals. The fundamental approach to doing this is through composition of components that deal with three distinct principal process dimensions. This approach (illustrated in Figure 1) is beneficial because it would allow for reuse of components across the process family and more importantly, the generation of new members of the family by simply compiling elements from common repositories.

We have experimented with the use of the Little-JIL process definition language [42, 43] as a vehicle for representing process families. Little-JIL is unusual in its clear separation of three concerns in process definition, namely the need for definition of 1) individual process steps and their coordination, 2) the behaviors of the agents that perform steps, and 3) the structure of the collection of artifacts that are produced and consumed by the steps. A complete Little-JIL process definition consists of one of each these three types of definition components. This is a particularly promising basis for modeling and defining process families, because each selection of a different set of these three components will generate a different process. In our work we have initialized the set and structure of steps with a fixed process, which we call the "Coordination Metaprocess," we then made varying augmentations with elaborative steps, while also making different selections of agent behaviors and artifact structures. This allowed for substantial process variation, and the totality of all such variants is what we call a process family.

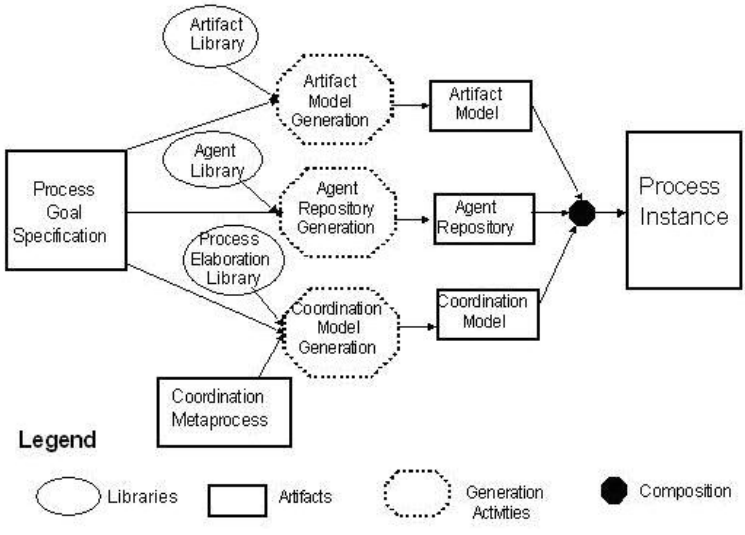


Fig. 1. Procedure for generating process instances

As indicated in Figure 1, selected process elaboration instances could be drawn from a library to add elaborative details to the Coordination Metaprocess. The resulting coordination model instance is then composed with selected agent and artifact specifications. Selection of specific components from the appropriate libraries might be driven by the process goal specification. Understanding how this is done requires a short explanation of principal features of Little-JIL.

A Little-JIL process definition is a hierarchical decomposition of steps, each of which is executed by a specified agent. Steps communicate with each other by exchanging artifacts. Thus, a Little-JIL process definition consists principally of three parts: 1) a coordination model that is a structure of steps represented by a "coordination diagram" such as shown in Figures 2 and 4, 2) a repository of agents,

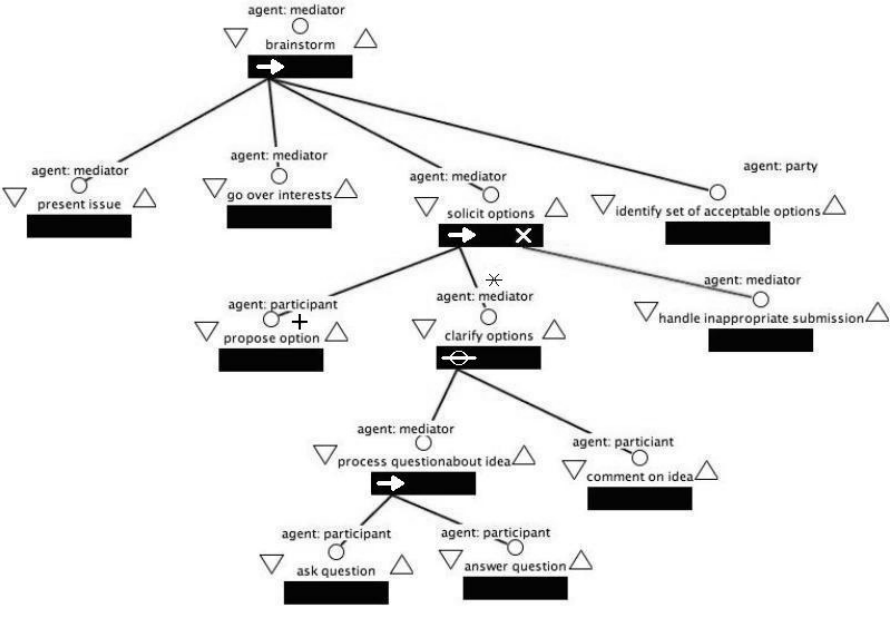


Fig. 2. The Little-JIL coordination diagram representing the "Brainstorm" process

one of which is selected for late binding to execute a step (each step in the coordination diagram has an agent), and 3) a library of artifacts used, created, and transmitted by the steps of the process. Binding different agents and different artifact definitions to a given coordination model is thus a way to achieve process variation.

Figure 2 is a visual representation of a Little-JIL coordination model of a simplified brainstorming process. The coordination model is a hierarchy of process steps, each of which is depicted by a black rectangular "step bar". The step name is located above the step bar, which is accompanied by a set of badges that denote several semantic features. The behavior of a step is defined to be the behaviors of its children (the steps below the parent, connected to it by edges emanating from the left side of the step bar define the workflow, and the steps, connected to the parent by edges emanating from the right side of the bar are exception handlers). Leaf steps (i.e., those having no children) have no behaviors defined by the coordination model. Their behaviors are the behaviors of the step's assigned agent, enabling process variation in a way that will be illustrated.

Non-leaf steps contain a sequencing badge (imbedded in the left of the step bar), which defines the order in which its sub-steps execute. For example, a "Sequential" step (right-arrow in the "brainstorm" step in Figure 2) indicates that its sub-steps are executed in left-to-right order. In Figure 2, "solicit options" is a Sequential step, indicating that "propose option" and "clarify options" must be executed in this order.

Note that the step "propose option" is connected to its parent by an edge annotated with "+" (a Kleene Plus). This indicates that "solicit options" has one or more copies of "propose idea" as its child(ren). The tag "agent: participant" specifies that any agent executing this step must be of type "participant". This obliges the agent

repository to assign agents able to execute "participant" functions to each of these child steps. Thus the activity "solicit options" is defined to be one or more participants proposing options. Similarly, a * (Kleene Star) annotation means that the child step can be executed any number (zero or more) of times.

The "clarify options" step has a circle with a slash through it on the left of its step bar, which indicates that it is a "Choice" step. A Choice step is carried out by having its agent (in this case a mediator) make a choice between the various defined alternatives, each of which is defined to be a substep. Thus, in this case, the agent chooses between "process question about idea" and "comment on idea" as the way in which "clarify options" is to be executed. There are no other allowable alternatives.

Note that in this Coordination Model, the step "solicit options" consist of "propose option" and "clarify options." While the details of "propose option" are not specified (as it is a leaf step), the Coordination Model does specify that this step is capable of throwing an exception. This is inferable by the presence of an X on the right of the step bar of its parent, indicating that the parent, "solicit options," incorporates as part of its definition a subprocess that defines how to handle the throwing of an exception by any of its children. In this case, the X is connected to a child step, "handle inappropriate submission", presumably indicating that when the process of doing "propose option" results in the detection of offensive material, then the "handle inappropriate submission" step is invoked to delete that contribution. This exception handler is defined to be "complete," therefore the flow of execution continues as though the "solicit options" step has thereby been concluded.

Of additional importance is the way in which artifact flow is defined in Little-JIL. In the Coordination Model shown in Figure 2, the step "go over interests," for example, has a defined outgoing parameter, *interest list*, which is passed to its parent, the "brainstorm" step. Specifications of all the additional artifact flows in Figure 2 are elided here to reduce the complexity of the example.

Using Little-JIL, we can model process variation and generate instances of different processes within a family through several techniques. We have thus far identified three such process instance generating techniques:

Agent Behavior: By modifying the behavior of the agents executing the process (or parts of it), we create a variant of the process. Different selections from among different agent behaviors at execution time will create different process variants, but, moreover, additional variation is possible through the use of different semantics for defining the agent repository itself. For example, we propose to consider substituting for the previously described model in which objects have static collections of methods, a model in which the collection of methods used to define agent types is dynamic, responding to changes in process execution. This would take some control of the methods usable by an agent out of the agent's hands, thus creating the capability for additional process variation.

Task Elaboration: By "clipping on" small elaboration processes onto a leaf step, we can specify how to execute it differently. Since leaf steps contain no details specifying how they are to be executed, this decision is entirely up to the execution agent bound to the step. By adding elaborative substeps, the agent is mandated to execute the step as defined by the behaviors of these children.

Artifact Structure: By selecting artifacts from a library of different artifact structures, featuring differences in semantics, structure, and content, we can create different process instances.

4 Case Study

In an ongoing research collaboration with the National Mediation Board (NMB), we have been developing a process definition for online dispute resolution (ODR) to be used by mediation professionals [22, 32]. We suggested that a rigorous process definition could be used to bring ODR technology into NMB, by indicating how computer technologies could be incorporated into these processes.

We initially believed that there was one single process to be defined and that this process had a well-defined goal, namely an agreement. But this project made it clear that our earlier belief was naïve and one process could not encompass all the variations introduced by the individual mediators. We found that not only do different mediators employ different processes from one another but an individual mediator might change the process, depending on the perceived effectiveness of the current process execution and changing group dynamics. These processes however, all seem to bear important familial relations to each other. Thus what the NMB context seems to call for is a process family, rather than a single process. As part of the case study, four descriptions of the "Brainstorm" process were elicited from four different mediation professionals. All four have attended the same training and must follow the same metaprocess prescribed by the NMB (partially outlined in Figure 2). We attempted to use the approach described above to see if all four elicited processes elicited could be best comfortably thought of as a process family.

5 Results

The "Brainstorm" coordination process shown in Figure 2 will now be taken as the Coordination Metaprocess that is to be used as the basis for a process family. We apply the three instance generation techniques outlined in section 3 to demonstrate how variation can be introduced to span a large set of variants, including all of the four different processes elicited from NMB personnel.

Agent Behavior Variation: Recall that it seems necessary to support variation due to differences in the ways in which different agents perform an activity, perhaps under different circumstances during an ODR process execution. Thus, for example, different levels of anonymity might be desirable under different brainstorming circumstances, and this can be specified by defining differences in the ways in which different agents deal with the artifacts they must process. Figure 3 contains two different formal definitions of how an agent may deal with such artifacts. In this example, an agent is considered to be an instance of a type (in this case, the type is "Mediator"), and the definition of the type is in terms of the methods that it can execute. We assume that the various methods are the various ODR process activities that the agent may be called upon to execute.

In this example, two different subtypes of agents of type Mediator are defined. Both definitions include a specification of how the step "Propose Options" is to be executed by detailing exactly how to execute the method "options list.add". The agent instance Mediator-1 of subtype Fully-Anonymous Mediator is defined so that it will never add to an options list any information about the contributor of a list item. On the other hand, the Mediator-2 agent instance of the Partially-anonymous Mediator subtype is defined to identify an options list item contributor only if the contributor is of type Mediator. Statically specifying either subtype will assure that the corresponding agent behavior is always executed, thus providing process variation. The possibility of execution time subtype binding affords the possibility of more variation in behavior. Clearly, one can populate an agent library with specifications of how each of the needed agent types is to execute each of the steps to which it might ever be bound in a full family of processes. In some cases, the agent specification might be null, in which case the agent would have no restrictions on its behavior.

```
Fully-Anonymous Mediator is-subtype-of Mediator:
Propose Options:
     for (Option opt: options) {
           options list.add(new Option(opt.what)); }
  Instances: Mediator-1{anonymous:yes}
Partially-anonymous Mediator is-subtype-of Mediator:
Propose Options:
     participant is-a mediator {
           for (Option opt: options) {
                 options list.add(new Option(opt.who,
opt.what)); }
Propose Options:
     for (Option opt: options) {
           options list.add(new Option(opt.what)); }
  Instances: Mediator-2{anonymous:partially}
```

Fig. 3. An example of items from the agent repository

Or, as in the case shown in Figure 3, different agent subtypes might have different mandated behaviors, perhaps for the different steps to which they might be bound as agents or perhaps in response to different execution state details. The organization and structure of an agent repository is the subject of current research [36]. Thus, Figure 3 shows only one example of how this repository might be organized and defined. It is not clear that a type inheritance hierarchy will necessarily be used. It is conceivable that other agent definition approaches might be used at least to specify parts of agent behaviors. Subsequent research is expected to shed light on which agent definition formalisms, and which agent library organization strategies, seem most effective in supporting needed agent-based variation.

Task Elaboration: To demonstrate the task elaboration technique, we elaborate the "present issue" step that is a leaf in the Coordination Metaprocess in Figure 2. Until the process fragment of Figure 4 is bound to the "present issue" step of the process in Figure 2, the way this step is to be executed is determined solely by the agent bound to it, subject to any restrictions or directions specified in the agent's definition.

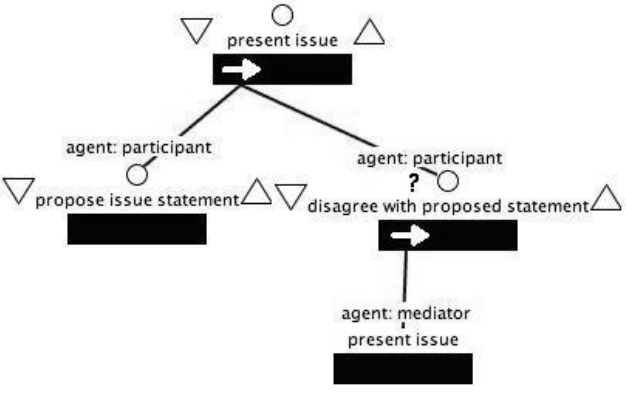


Fig. 4. "Present issue" process fragment elaboration

Once the process fragment of Figure 4 is bound, however, a new process instance (i.,e. a new member of the process family) is created. The new process instance differs from the previous process instance in that "present issue" now mandates that an issue statement is created and discussed by all participants iteratively until all agree to the statement (which means that the optional step "disagree with proposed statement" will not be executed, and "present issue" will not be called again), as directed by the process in Figure 4. Other process fragments could be defined to create other variations on this process. By late-binding different process fragments, new process instances are created and incorporated into the family. While late binding of step structures to leaf steps is not currently possible with the present version of the Little-JIL interpreter, this feature is to be included in future versions.

Artifact Structure Variation: As previously mentioned, the step "go over interests" in the Brainstorm process in Figure 2 has an outgoing parameter, *interest list*. The *interest list* is an unordered collection of the interests of the parties who are bargaining. Depending on which mediator professional is leading the mediation, he or she may choose to keep all interests together without duplicates, or alternatively, they can choose to keep each party's interests separate, without duplicates within the party.

This variation can be easily achieved by changing the artifact structure of *interest list*—by adding or removing an *author party* annotation to the structure of the *interest list* and by removing duplicates based on a predetermined comparison (e.g. if an *author party* annotation is present, two interests are the same only if both *contents* and *author party* are the same).

6 Discussion

Through applying the three outlined techniques for creating process variants through process instance generation (changes in the agent behavior, artifact structure, or task elaboration fragments), we have been able to generate multiple instances that span most of the variations in the four processes elicited for the case study. By applying more than one technique simultaneously, much larger families can be generated.

This approach also provides an important vehicle for reuse in process definition by treating similar processes as a process family built from a common core (the Coordination Metatprocess) and a set of additional components, which augment the core. Moreover, by applying these techniques, it is easy to fine-tune large processes by switching components depending on the execution circumstances.

Although this approach to process families seems to be very promising, a considerable variety of additional work is also suggested. Initially, instance generation is to be done manually, based upon selection from libraries of agent repositories, artifact models, and process coordination elaboration models, respectively. Eventually, we expect that automation will at least facilitate, if not completely replace, manual selections from these libraries, although it is likely that human customization will always be needed to produce the three components to be composed into the final process instance.

Finally, it is important to gain a stronger sense what constitutes a process variant. As noted above, selecting different choices during execution seems quite different from creating different variants. Our case study did not provide us with much insight into how to distinguish between these two notions. It might be useful to address this problem by establishing formal metrics for determining the degree of similarity between processes, and perhaps use such metrics to guide decisions about what constitutes a viable variant.

Acknowledgements. The authors express their gratitude to Daniel Rainey of the National Mediation Board for providing the details of the ODR metaprocess. The authors also gratefully acknowledge Alexander Wise, Norm Sondheimer, Ethan Katsh, Leah Wing, Allan Gaitenby, M.S. Raunak, and Matt Marzilli for their support and insights about online dispute resolution, process formalisms, and Little-JIL.

This research was supported in part by the U.S. National Mediation Board and the US National Science Foundation under Award Nos. CCR-0204321 and CCR-0205575. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. National Mediation Board, the U.S. National Science Foundation, or the U.S. Government.

References

1. Alonso, G., Agrawal, D., Abbadi, A.E., Kamath, M., Gunthor, R., Mohan, C.: Advanced transaction model in workflow context. Proceedings of the 12th IEEE International Conference on Data Engineering, Proc. 12th International Conference on Data Engineering, New Orleans, February 1996 (1996) 574-581
2. Altintas, I., Berkeley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S.: Kepler: An Extensible System for Design and Execution of Scientific Workflows. Proceedings of the 16th International Conference on Scientific and Statistical Database Management, Santorini Island, Greece (2004) 423-424
3. Atkinson, C., Bayer, J., Muthig, D.: Component-based product line development: The KobrA approach. Proceedings of the The First International Software Product Line Conference, Denver, CO (2000) 289-309

4. Batory, D., O'Malley, S.: The design and implementation of hierarchical software systems with reusable components. ACM Transactions on Software Engineering and Methodology **1** (1992) 355-398
5. Belkhatir, N., Estublier, J., Walcelio, M.L.: ADELE-TEMPO: an environment to support process modelling and enaction. Software Process Modeling and Technology (1994) 187-222
6. Briggs, R.O.: On theory-driven design and deployment of collaboration systems. Int. J. Hum.-Comput. Stud. **64** (2006) 573--582
7. Clarke, L.A., Chen, Y., Avrunin, G.S., Chen, B., Cobleigh, R., Frederick, K., Henneman, E.A., Osterweil, L.J.: Process Programming to Support Medical Safety: A Case Study on Blood Transfusion. Proceedings of the Software Process Workshop 2005, Beijing, China. Springer-Verlag (2005) 347-359
8. Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools, and Applications. Addison-Wesley (2000)
9. Dami, S., Estublier, J., Amiour, M.: APEL: A Graphical Yet Executable Formalism for Process Modeling. Automated Software Engineering International Journal **5** (1998) 61-69
10. Deming, W.E.: Out of the crisis. MIT Press, Cambridge, MA (1982)
11. Emmerich, W., Gruhn, V.: FUNSOFT Nets: a Petri-Net based Software Process Modeling Language. IWSSD '91: Proceedings of the 6th International Workshop on Software Specification and Design, Como, Italy (1991) 175-184
12. Foster, H., Uchitel, S., Magee, J., Kramer, J., Hu, M.: Using a Rigorous Approach for Engineering Web Service Compositions: A Case Study. SCC '05: Proceedings of the 2005 IEEE International Conference on Services Computing (2005) 217-224
13. Gacek, C., Anastasopoules, M.: Implementing product line variabilities. Proceedings of the 2001 Symposium on Software reusability, Toronto, Ontario, Canada (2001) 109-117
14. Georgakopoulos, D., Hornick, M.F., Sheth, A.P.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases **3** (1995) 119-153
15. Ghezzi, C., Mandrioli, D., Morasca, S., Pezze, M.: A Unified High-Level Petri Net Formalism for Time-Critical Systems. IEEE Transactions of Software Engineering **17** (1991) 160-172
16. Griss, M., Favaro, J., d'Alessandro, M.: Integrating Feature Modeling with the RSEB. Proceedings of the 5th International Conference on Software Reuse (1998) 76-85
17. Harel, D., Naamad, A.: The {STATEMATE} semantics of statecharts. ACM Transactions on Software Engineering and Methodology **5** (1996) 293--333
18. Henneman, E.A., Cobleigh, R., Frederick, K., Katz-Basset, E., Avrunin, G.S., Clarke, L.A., Osterweil, J.L., Andrzejewski, C.J., Merrigan, K., Henneman, P.L.: Increasing Patient Safety and Efficiency in Transfusion Therapy Using Formal Process Definitions. University of Massachusetts, Amherst (2006)
19. Humphrey, W.S.: Managing the software process. Addison-Wesley, Boston, MA (1989)
20. Jacobson, I., Griss, M., Jonsson, P.: Software Reuse: Architecture, Process and Organization for Business Success. Addison-Wesley Professional (1997)
21. Jarzabek, S., Zhang, H., Zhang, W.: XVCL: XML-Based Variant Configuration Language. Proceedings of the International Conference on Software Engineering, ICSE'03, Los Alamitos, CA. IEEE Computer Society Press (2003) 803-811
22. Katsh, E., Osterweil, L., Sondheimer, N.K.: Process Technology for Achieving Government Online Dispute Resolution. Proceedings of the National Conference on Digital Government Research, Seattle, WA (2004)

23. Kellner, M.I.: Software Process Modeling Support for Management Planning and Control. Proceedings of the First International Conference on the Software Process, Redondo Beach, CA (1991) 8-28
24. Kiczales, G., Lamping, J., Mandhekar, A., Maeda, C., Lopes, C.V., Loingtier, J., Irwin, J.: Aspect-Oriented Programming. Proceedings of the European Conference on Object-Oriented Programming. Springer-Verlag (1997) 220-242
25. Knauber, S.J.a.P.: Synergy between Component-Based and Generative Approaches. Proceedings of ESEC/FSE-7, Toulouse, France (1999) 2-19
26. Kolfschoten, G.L., Briggs, R.O., Vreede, G.-J.d., Jacobs, P.H.M., Appelman, J.H.: A conceptual foundation of the thinkLet concept for Collaboration Engineering. International Journal of Human-Computer Studies **64** 611-621
27. Kyo, C., Kang, S.K., Lee, J., Kim, K., Shin, E., Huh, M.: FORM: A Feature-Oriented Reuse Method with Domain Specific Reference Architectures. 143-168
28. Leymann, F., Roller, D.: Workflow-Based Applications. IBM Systems Journal **36** 102-123
29. Mayer, R.J., al, e.: IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications. Knowledge Based Systems, Inc. (1992)
30. Northrop, L.: Software Product Lines--Practices and Patterns. Addison-Wesley (2002)
31. Osterweil, L.J.: Software Processes Are Software, Too, Revisited. Proceedings of the 19th International Conference on Software Engineering, Boston, MA (1997) 540-558
32. Osterweil, L.J., Sondheimer, N.K., Clarke, L.A., Katsh, E., Rainey, D.: Using Process Definitions to Facilitate the Specifications of Requirements. Department of Computer Science, University of Massachusetts, Amherst, MA (2006)
33. Osterweil, L.J., Wise, A., Clarke, L.A., Ellison, A.M., Hadley, J.L., Boose, E., Foster, D.R.: Process Technology to Facilitate the Conduct of Science. Proceedings of the 2005 Software Process Workshop, Beijing, China. Springer-Verlag (2005) 403-415
34. Pavel, J.N.S., Royer, J.-C.: Dynamic Configuration of Software Product Lines in ArchJava. Proceedings of the Third International Software Product Line Conference (2004) 90-109
35. Prehofer, C.: Feature-Oriented Programming: a Fresh Look at Objects. ECOOP '07. Springer-Verlag (1997)
36. Raunak, M.S., Osterweil, L.J.: Effective Resource Allocation for Process Simulation: A Position Paper. Proceedings of the International Workshop on Software Process Simulation and Modeling, St. Louis, MO (2005)
37. Suzuki, M., Katayama, T.: Meta-Operations in the Process Model HFSP for the Dynamics and Flexibility of Software Processes. Proceedings of the First International Conference on the Software Process, Redondo Beach, CA. IEEE Computer Society Press (1991) 202-217
38. Svahnberg, M., Bosch, J.: A Taxonomy of Variability Realization Techniques. Software Practices and Experience **35** 705-754
39. van Ommering, R., Kramer, J., Magee, J.: The Koala Component Model for Consumer Electronics Software. IEEE Computer **33** 78-85
40. Weigert, O.: Business Process Modeling and Workflow Definition with UML (1998)
41. Weiss, D.M., Lai, C.T.R.: Software product-line engineering: a family-based software development process. Addison-Wesley (1999)
42. Wise, A.: Little-JIL 1.5 Language Report. Department of Computer Science, University of Massachusetts, Amherst, MA (2006)
43. Wise, A., Cass, A.G., Lerner, B.S., McCall, E.K., Osterweil, L.J., Sutton, S.M.: Using Little-JIL to Coordinate Agents in Software Engineering. Proceedings of the Automated Software Engineering Conference, Grenoble, France (2000)

An Algebraic Approach for Managing Inconsistencies in Software Processes*,**

Qiusong Yang[1,3], Mingshu Li[1,2], Qing Wang[1], Guowei Yang[1,3], Jian Zhai[1,3], Juan Li[1], Lishan Hou[1], and Yun Yang[4]

[1] Laboratory for Internet Software Technologies,
Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
{qiusong_yang,mingshu,wq,yangguowei,zhaijian,lijuan,
houlishan}@itechs.iscas.ac.cn
[2] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
[3] Graduate University of Chinese Academy of Sciences, Beijing 100039, China
[4] Faculty of Information and Communication Technologies,
Swinburne University of Technology
Hawthorn, VIC 3122, Melbourne, Australia
yyang@ict.swin.edu.au

Abstract. To produce quality software and evolve them in an economic and timely fashion, enactable software process models are used for regulating development activities with the support of Process-Centered Software Engineering Environments (PCSEEs). However, due to the dynamically changing development environment, the developers do not always follow the process model in presence of unforeseen situations. As human with creativity and variant nature, each developer has his or her own way of doing development that may not be allowed by the process model. As a result, various inconsistencies arise in software processes and then the authority of the process model will be undermined. In this paper, we propose an algebraic approach to promote the efficient management of inconsistencies. With the approach, potential inconsistencies can be precisely detected and valuable diagnostic information is available to help process designers efficiently locate the detected inconsistencies. The effectiveness of the approach is demonstrated by experimenting it on an example process.

Keywords: Software Engineering, Software Process, Inconsistency, Verification, Algebraic, PCSEE, TRISO/ML.

1 Introduction

As for the software process literature, the researches mainly focuses on software process modelling and software process improvement. The research on the software

* This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082, 60473060; the National Hi-Tech Research and Development Plan of China under Grant No. 2006AA01Z185; the National Key Technologies R&D Program under Grant No. 2005BA113A01.

** One of the authors, Yun Yang, gratefully acknowledges the support of K. C. Wong Education Foundation, Hong Kong.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 121–133, 2007.

process modelling involves devising notations for expressing process models, enacting the models within PCSEEs, and providing concrete guidance on the actual development process. To discuss the enactment mechanisms for PCSEEs, Dowson [1] clarifies the three domains of software processes: *process definition* (or *process model* in this paper) contains characterizations of processes or fragments of processes expressed in some notation; *process performance* encompasses the actual activities or actions conducted by human agents and non-human agents in the course of a software project; *process definition enactment* (or *process enactment* for short) encompasses what takes place in a PCSEE to support process performance governed by process definition.

In an ideal world, the process enactment can obtain timely and correct feedback from the process performance to know what actual activities or actions are conducted. A software process model describes an ideal process for development and provides procedures to handle possible exceptions. However, the feedback from process performance to process enactment is subject to the variant nature of human and tends to be delayed, ignored, or even erroneous [1]. In addition, it is impossible to define an ideal software process in advance and specify procedures to manage all unforeseen situations. As a result, the *environment-level inconsistency* [2] will occur when the process performance is not properly reflected in the process enactment.

When a software process is modelled, a set of properties or invariants can be specified to characterize the correctness of process models. When a property is violated in process performance, an inconsistency will arise. This type of internal inconsistency in process performance is called *domain-level inconsistency* in [2]. A domain-level inconsistency does not necessarily result in an environment-level inconsistency. If the process model successfully predicts the domain-level inconsistency in the process performance, the process enactment will take corresponding actions and the process performance is still faithfully reflected in the process enactment.

As shown in Figure 1, the ultimate goal of PCSEE is to make the process performance governed by the process model. PCSEEs provide mechanisms to enact the process model and components to interact with the environment so that the process model is enforced in the process performance. However, the process performance may deviate from the process model as a result of the existence of inconsistencies. The "performance model" in Figure 1 denotes the underlying model that governs the process performance,

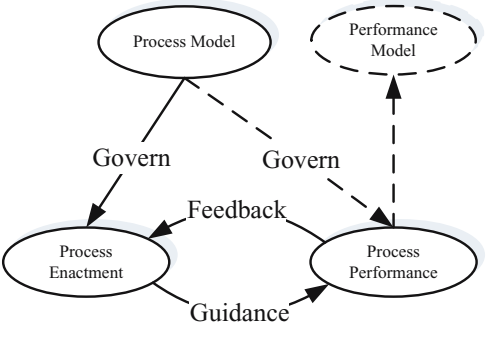


Fig. 1. Process Inconsistencies

which may be discovered from the logs and events of process performance. If a domain level inconsistency occurs, the invariant that is violated in the process performance should not be satisfied by the performance model. Correspondingly the performance model will be "different" or inconsistent with the process model if an environment-level inconsistency exists between the process performance and the process enactment.

In this paper, an algebraic approach based on the polyadic π-calculus is proposed to detect existing inconsistencies and help process designers efficiently locate and resolve the inconsistencies. To mask the mathematical intricacies of the polyadic π-calculus, we present a graphical modelling language in Section 2. In Section 3, the graphical language is mapped onto the polyadic π-calculus and the mapping rules are given. Section 4 describes the methods for detecting domain-level and environmental inconsistencies. To demonstrate the effectiveness of the approach and show how to help process designers efficiently locate and resolve inconsistencies, a case study is presented in Section 5. The work related to the research in this paper is presented in Section 6 and the paper is concluded in the last section.

2 Visualizing Software Processes with TRISO/ML

A visualization support will help to fully make use of the rigid operational semantics of the polyadic π-calculus without considering its underlying mathematical intricacies. TRISO/ML (TRidimensional Integrated SOftware development model/Modelling Language) is proposed for supporting the TRISO Model advocated in [3][4]. With rather simple and concise graphical notations in Figure 2, the language provides powerful abstractions of control flow, data dependency, and resource usage in software processes.

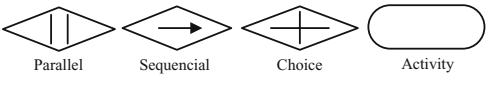


Fig. 2. Graphical Notations

Definition 1 (Software Process in TRISO/ML). *In TRISO/ML, a software process is defined as a tri-tuple:* $(\mathcal{V},\ \mathcal{E},\ \mu)$ *where:*

- $\mathcal{V}$ *is a set of nodes, as the union of* $\mathcal{C} \cup \mathcal{A}$. $\mathcal{C}$ *represents the set of nodes controlling the sequencing of activities. Each node in* $\mathcal{C}$ *is in type of either* $Parallel$, $Sequencial$, *or* $Choice$. $\mathcal{A}$ *denotes the set of activities that are carried out in a software process. An activity can be hierarchically decomposed into a set of sub-activities through the nodes in* $\mathcal{C}$, *which regulates the sequencing of the sub-activities.*
- $\mathcal{E} \subseteq (\mathcal{C} \times \mathcal{A} \cup \mathcal{A} \times \mathcal{C})$ *is a set of directed edges. If* $e \in \mathcal{E}$, *and* $s(e) = a \in \mathcal{A}$, *then there does not exist any other edge* e', *with* $e' \in \mathcal{E}, d(e') \neq d(e),\ and\ s(e') = a$ *(where* $s, d : \mathcal{E} \rightarrow \mathcal{V}$, *denoting the starting and ending node associated with a directed edge, respectively). It states that an non-terminal activity can only be decomposed once. The out-degree of a node in* $\mathcal{C}$ *is greater than two.*

– $\mu : (\mathcal{A} \cup \mathcal{C} \times \mathcal{A}) \rightarrow Attr$ *maps each element in* $\mathcal{A} \cup \mathcal{C} \times \mathcal{A}$ *to a set of attributes. For an edge* $e \in \mathcal{C} \times \mathcal{A}$*, its labelled attributes specify the data exchanges between an activity and one of its sub-activities connected by* e*. The data exchanges include the items that an sub-activity receives from its parent activity before its execution and the items that an sub-activity sends to its parent activity after its completion. For two activities that no one is the ancestor of the other, the data exchange between them is implemented as communication along channels. As for each activity node belonging to the set* $\mathcal{A}$*, the input/output channels and data transmitted along those channels can be declared as attributes. In addition, the actor to perform an activity is also an attribute of the activity.*

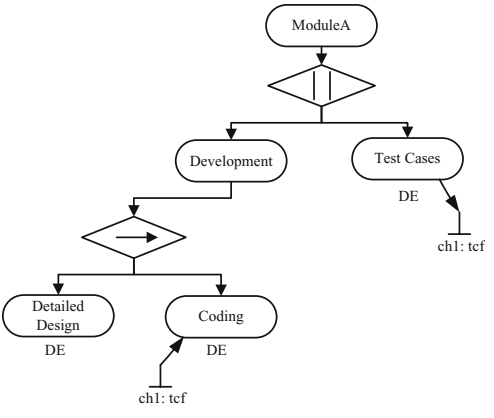


Fig. 3. An Example Process Model

An Example Process. The example presented in this section aims at demonstrating the modelling of software processes in TRISO/ML. It will be used as an example throughout this paper. In the example, the development of a module is assigned to a developer. He or she is responsible for specifying detailed design, coding and testing of the module. As the test-driven methodology is applied, the developer is required to devise test cases before the writing of code. In Figure 3, the example process is modelled with the graphical notations. The temporal order and data dependency between the activity *Test Cases* and the activity *Coding* is implemented through the synchronized communication on the channel $ch1$.

In Figure 4, one developer may abide by the test-driven approach, in which the temporal order and data dependency between *Coding* and *Test Cases* are satisfied. The performance model is consistent with the process model in Figure 3. On the contrary, another developer may be customized to the traditional development method, other than the test-driven methodology. He or she begins to write code before test cases are available. The test cases are written and used for testing after the source code is completed. The corresponding performance model is shown in Figure 5. In the model, the activity *Coding* does not wait for a message indicating the availability of test cases.

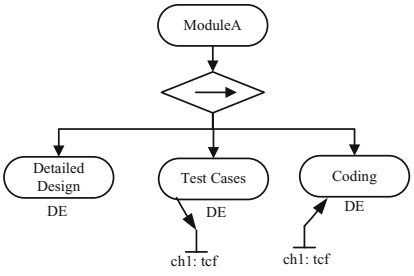


Fig. 4. Performance Model I

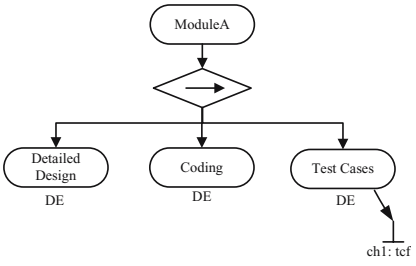


Fig. 5. Performance Model II

3 Mapping TRISO/ML onto Polyadic π-Calculus

In this section, the syntax and the reduction semantics of the polyadic π-calculus is firstly introduced. Then, the rules for mapping each construct of TRISO/ML onto the polyadic π-calculus are provided.

3.1 Polyadic π-Calculus

The π-calculus comes in two basic styles: the *monadic* calculus [5], where exactly one name is communicated at each synchronization, and the *polyadic* calculus [6], where zero or more names are communicated. We choose the polyadic calculus, because it is more elegant to use for modelling purposes and allows a notion of sorts [6][7].

Definition 2 (Polyadic π-calculus). *The syntax of the polyadic π-calculus is given in the following BNF equations [7]:*

$$\begin{aligned} P &:= M \mid P|P' \mid (\nu z)P \mid !P \\ M &:= \mathbf{0} \mid \pi.P \mid M + M' \\ \pi &:= \overline{x}\langle\widetilde{y}\rangle \mid x(\widetilde{z}) \mid \tau \mid [x = y]\pi \end{aligned}$$

Briefly, $\mathbf{0}$ is *inaction* representing a process which can do nothing; the *prefix* $\pi.P$ can perform the output, input, silent τ or match action, thereby evolving into P; the *sum* $M + M'$ offers the *choice* of M or M'; the composition $P|P'$ – "P par Q" – places the two processes together and they will be concurrently active and act independently, but can also communicate; the $(\nu x)P$ – "new x in P"– restricts the use of the name x to P and it declares a new unique name x, distinct from all external names, for use in P. As for the output and input prefixes, the intended interpretations of them are that $\overline{x}\langle\widetilde{y}\rangle.P$ can send the tuple $\widetilde{y}$ via the co-name of x and continue as P, and that $x(\widetilde{z}).Q$ can receive a tuple $\widetilde{y}$ via the name x and continue as $Q\{\widetilde{y}/\widetilde{z}\}$. The *unobserved prefix* $\tau.P$ can evolve invisibly to P. τ is the internal action of a process and not visible to the external viewer. The *match prefix* $[x = y]\pi.P$ can evolve as $\pi.P$ if x and y have the same name, otherwise the process acts as $\mathbf{0}$.

The reduction system of the given polyadic π-calculus is defined as:

Definition 3 (Reduction). *The reduction relation $\longrightarrow$ over processes is the least relation satisfying the following rules [7]:*

$$R-INTER\ \ \frac{}{\overline{x}\langle\widetilde{y}\rangle.P_1 \mid\ x(\widetilde{z}).P_2 \longrightarrow P_1|P_2\{\widetilde{y}/\widetilde{z}\}}$$

$$R-TAU\ \ \frac{}{\tau.P+M \longrightarrow P} \qquad R-PAR\ \ \frac{P_1 \longrightarrow P_1'}{P_1|P_2 \longrightarrow P_1'|P_2}$$

$$R-RES\ \ \frac{P \longrightarrow P'}{(\nu z)P \longrightarrow (\nu z)P'} \qquad R-STRUCT\ \ \frac{P_1 \equiv P_2 \longrightarrow P_2' \equiv P_1'}{P_1 \longrightarrow P_1'}$$

where $|\widetilde{y}| = |\widetilde{z}|$ for the $R-INTER$ rule.

In the definition, $\equiv$ represents the *structural congruence* among processes. The operational semantics of the polyadic π-calculus becomes simple under the $R-STRUCT$ rule.

3.2 Mapping Rules

In this section, the rules for mapping a software process modelled in TRISO/ML onto the polyadic π-calculus processes are provided. All constructs of TRISO/ML defined in Section 2 are covered by them. With these rules, the interpreting procedure becomes rather straightforward and mechanical.

Rule 1. For an actor with the unique identifier ac, the polyadic π-calculus process for it has the following form:

$$A_{ac} \stackrel{def}{=} assign_{ac}(start, end).\overline{start}.end.A_{ac}|A_{ac}$$

The process A_{ac} waits on channel $assign_{ac}$ for channels $start$ and end. When the actor wants to begin to perform the activity, the process will send an empty message through the received $start$ channel. The actor will receive an empty message from the end channel when all the sub-activities of the assigned activity have been finished. Having accomplishing an activity, the actor will be ready for another task.

Rule 2. For an activity $a \in \mathcal{A}$, it receives $\{b_{11}, \cdots, b_{1m}\}, \cdots, \{b_{l1}, \cdots, b_{ln}\}$ from the channels $\{chi_1, \cdots, chi_l\}$ and $\{p_1, \cdots, p_u\}$ from the channel $ex_{a_p\_a}$, sends $\{c_{11}, \cdots, c_{1s}\}, \cdots, \{c_{r1}, \cdots, c_{rt}\}$ through the channels $\{cho_1, \cdots, cho_r\}$, and returns $\{q_1, \cdots, q_v\}$ to its parent a_p through the channel $ex_{a\_a_p}$. Then, the polyadic π-calculus process A_a for the activity is:

$$\begin{aligned}
A_a &\stackrel{def}{=} (\nu\ i_1, \cdots, i_l, i_o)(I_{a_s}\langle i_1, \cdots, i_l\rangle\ |E_a\langle i_1, \cdots, i_l, i_o\rangle \mid O_a\langle i_o\rangle)\\
I_a &\stackrel{def}{=} (i_1, \cdots, i_l).chi_1(b_{11}, \cdots, b_{1m}).\bar{i}_1\langle b_{11}, \cdots, b_{1m}\rangle|\cdots|\\
&\quad\ chi_l(b_{l1}, \cdots, b_{ln}).\bar{i}_l\langle b_{l1}, \cdots, b_{ln}\rangle.ex_{a_p_a}(p_1, \cdots, p_v)\\
O_a &\stackrel{def}{=} (i_o).i_o(c_{11}, \cdots, c_{1s}, \cdots, c_{r1}, \cdots, c_{rt}, q_1, \cdots, q_v).\\
&\quad\ \overline{cho_1}\langle c_{r1}, \cdots, c_{rs}\rangle.\cdots.\overline{cho_r}\langle c_{r1}, \cdots, c_{rt}\rangle.\overline{ex_{a_a_p}}\langle q_1, \cdots, q_v\rangle
\end{aligned}$$

The process A_a is the concurrent combination of I_a, E_a, and O_a. The process I_a receives data from prescribed channels and the channel connecting to its parent activity, then sends the received data to the process E_a through private channels. Acting as a relay station, I_a ensures that the communication on any input channel can be carried out immediately and deadlocks will not arise as the result of the mismatch between the order of input and the order of manipulation. When an activity and its sub-activities are completed, it will output data to other activities and its parent activity, as shown by the process O_a. The execution of the activity is modelled by the process E_a, whose definition is given by the following rules.

Rule 3. For a non-terminal activity $a \in \mathcal{A}$, it is refined to w sequential activities, $a_1, \cdots, a_w$. Each sub-activity may specify the information exchanges with its parent. For example, the wth sub-activity will receive $\{p_{w1}, \cdots, p_{wj}\}$ from the activity a and returns $\{q_{w1}, \cdots, q_{wk}\}$. The activity will be assigned to the actor with the unique identifier ac. Then the E_a process for the activity a is:

$$
\begin{aligned}
E_a = {} & (i_1, \cdots, i_l, i_o).i_1(b_{11}, \cdots, b_{1m}).\cdots.i_l(b_{l1}, \cdots, b_{ln}).ex_{a_p\_a}(p_1, \cdots, p_u). \\
& trigger_a.\overline{assign_{ac}}\langle start_a, end_a\rangle.start_a.\overline{ex_{a\_a1}}\langle p_{11}, \cdots, p_{1h}\rangle.\overline{trigger_{a1}}. \\
& ex_{a1\_a}(q_{11}, \cdots, q_{1i}).triggered_{a1}.\cdots.\overline{ex_{a\_aw}}\langle p_{w1}, \cdots, p_{wj}\rangle.\overline{trigger_{aw}}. \\
& ex_{aw\_a}(q_{w1}, \cdots, q_{wk}).triggered_{aw}.\overline{triggered_a}.\overline{end_a}. \\
& \overline{i_o}\langle c_{11}, \cdots, c_{1s}, \cdots, c_{r1}, \cdots, c_{rt}, q_1, \cdots, q_v\rangle
\end{aligned}
$$

In this rule, each output variable must be bounded by certain input prefix. As an example, for $\forall t,\ 1 \le t \le w$: $\{p_{t1}, \cdots, p_{t.}\}$ is a subset of $\{b_{11}, \cdots, b_{1m}\} \cup \cdots \cup \{b_{l1}, \cdots, b_{ln}\} \cup \{p_1, \cdots, p_u\} \cup \{q_{11}, \cdots, q_{1i}\} \cup \cdots \cup \{q_{(t-1)1}, \cdots, q_{(t-1).}\}$. All the following rules are also subject to this constraint. Firstly, the process E_a withdraws the relayed input from the process I_a. Then, the activity a is assigned to the prescribed actor when the activity is triggered by its parent activity. The actual execution of the activity will not begin until the actor decides to do so. As the activity a is a non-terminal node, the process E_a then sequentially triggers the execution of its sub-activities. When the activity is completed, it will notify its parent and release the assigned actor. At last, the obtained data will be sent to the process O_a for output.

Rule 4. For a non-terminal activity $a \in \mathcal{A}$, it is decomposed into w concurrently combined activities. Then the E_a process for the activity a is:

$$
\begin{aligned}
E_a = {} & (i, q, i_1, \cdots, i_l, i_o).(\nu k_{a1}, \cdots, k_{aw})i_1(b_{11}, \cdots, b_{1m}).\cdots.i_l(b_{l1}, \cdots, b_{ln}). \\
& ex_{a_p\_a}(p_1, \cdots, p_u).trigger_a.\overline{assign_{ac}}\langle start_a, end_a\rangle.start_a.(E_1|E_2) \\
E_1 = {} & (\overline{ex_{a\_a1}}\langle p_{11}, \cdots, p_{1h}\rangle.\overline{trigger_{a1}}.ex_{a1\_a}(q_{11}, \cdots, q_{1i}).triggered_{a1}. \\
& \overline{k_{a1}}.\overline{k_{a1}}\langle q_{11}, \cdots, q_{1i}\rangle) \mid \cdots \mid (\overline{ex_{a\_aw}}\langle p_{w1}, \cdots, p_{wj}\rangle.\overline{trigger_{aw}}. \\
& ex_{aw\_a}(q_{w1}, \cdots, q_{wk}).triggered_{aw}.\overline{k_{aw}}.\overline{k_{aw}}\langle q_{w1}, \cdots, q_{wk}\rangle) \\
E_2 = {} & k_{a1}.k_{a1}(q_{11}, \cdots, q_{1i}).\cdots.k_{a1}.k_{wa}(q_{w1}, \cdots, q_{wk}). \\
& \overline{triggered_a}.\overline{end_a}.\overline{i_o}\langle c_{11}, \cdots, c_{1s}, \cdots, c_{r1}, \cdots, c_{rt}, q_1, \cdots, q_v\rangle
\end{aligned}
$$

The process E_1 triggers the sub-activities concurrently and the E_2 process collects results from sub-activities and output them to the process O_a. The enforced synchronization on channels $k_{a1}, \cdots, k_{aw}$ ensures that the process E_2 is executed after the process E_1 even under the condition that there is no activity passing data back to the activity a.

Rule 5. For a non-terminal activity $a \in \mathcal{A}$, it is decomposed into w sub-activities, which are combined together through the choice operator. Then the E_a process for the activity a is:

$$\begin{aligned}
E_a &= (i, q, i_1, \cdots, i_l, i_o).(\nu k) i_1(b_{11}, \cdots, b_{1m}). \cdots .i_l(b_{l1}, \cdots, b_{ln}). \\
&\quad ex_{a_p_a}(p_1, \cdots, p_u).trigger_a.\overline{assign_{ac}}\langle start_a, end_a\rangle.start_a.(E_1|E_2) \\
E_1 &= (\overline{ex_{a_a1}}\langle p_1, \cdots, p_h\rangle.\overline{trigger_{a1}}.ex_{a1_a}(q_1, \cdots, q_j).triggered_{a1}.\overline{k}.\overline{k}\langle q_1, \cdots, q_j\rangle) + \cdots \\
&\quad +(\overline{ex_{a_aw}}\langle p_1, \cdots, p_h\rangle.\overline{trigger_{aw}}.ex_{aw_a}(q_1, \cdots, q_j).triggered_{aw}.\overline{k}.\overline{k}\langle q_1, \cdots, q_j\rangle) \\
E_2 &= k.k\langle q_1, \cdots, q_j\rangle.\overline{triggered_a}.\overline{end_a}.\overline{i_o}\langle c_{11}, \cdots, c_{1s}, \cdots, c_{r1}, \cdots, c_{rt}, q_1, \cdots, q_v\rangle
\end{aligned}$$

Rule 6. For a terminal activity $a \in \mathcal{A}$, it is not decomposed further. Then the process E_a for the activity a is:

$$\begin{aligned}
E_a &= (i, q, i_1, \cdots, i_l, i_o).i_1(x_{11}, \cdots, x_{1m}). \cdots .i_l(x_{l1}, \cdots, x_{ln}).ex_{a_p_a}(p_1, \cdots, p_u). \\
&\quad trigger_a.\overline{assign_{ac}}\langle start_a, end_a\rangle.start_a.\overline{ex_{a_ap}}\langle p_1, \cdots, p_h\rangle.\overline{triggered_a}.\overline{end_a}. \\
&\quad \overline{i_o}\langle c_{11}, \cdots, c_{1s}, \cdots, c_{r1}, \cdots, c_{rt}, q_1, \cdots, q_v\rangle
\end{aligned}$$

Rule 7. The software process is defined as the concurrent combination of activities and actors:

$$SP = A_{a1} \mid \cdots \mid A_{am} \mid A_{ac1} \mid \cdots \mid A_{acn}$$

To analyze or simulate the software process, sometimes an additional process modelling the environment is needed to make the system closed. The process is named Env and it is concurrently combined with the process SP. It can be simply defined as:

$$Env = \overline{trigger_{root}}.triggered_{root}$$

where $root$ denotes the root activity of a software process.

4 Detecting Inconsistencies with Polyadic π-Calculus

In this section, the detections of domain-level and environment-level inconsistencies are implemented based on the two types of analyses, respectively.

4.1 Domain-Level Inconsistencies

The domain-level inconsistency is the violation of process model invariants by the performance model. At the same time, the process performance model can be mechanically transformed into polyadic π-calculus expressions. Thus, the detection of domain-level inconsistencies can be efficiently implemented through the model checking of polyadic

π-calculus. With the modal μ-calculus, not only the local properties but also the enduring and long term properties can be specified [8]. Specifically, the following properties can be used for detecting some enduring inconsistencies:

Control Flow. The control flow describes the sequencing of activities. The activities in a software process tends to be highly concurrent. The set of possible traces of a process may be too large to be efficiently managed when the scale of the project is in large. The following type of domain-level inconsistencies may be mechanically detected:

- $\mu Z.\langle - \rangle tt \wedge [-act]Z)$, the action act is eventually carried out
- $\mu X.[act_a](\mu Y.[act_b](\nu Z.[act_b]ff \wedge [-]Z) \wedge \langle - \rangle tt \wedge [-act_b]Y) \wedge \langle - \rangle tt \wedge [-act_a]X$, the sequence $\cdots, act_a, \cdots, act_b, \cdots$ is eventually executed in a process
- $\mu X.[act_a]([act_b]tt \wedge [-act_b]ff) \wedge \langle - \rangle tt \wedge [-act_a]X$, the sequence $\cdots, act_a, act_b, \cdots$ is eventually executed in a process
- $\nu Z.[act_a](\mu Y.\langle - \rangle tt \wedge [-act_{b\_r}]Y) \wedge [-]Z$, whenever the action act_a happens, the action act_b eventually happens
- $\nu Z.[act]ff \wedge [-]Z$, the action act will never happen in a process

Data Dependency. The data dependencies among the activities in a software process are modelled as communication along channels in the polyadic π-calculus. As the content of artifacts is difficult to be modelled directly, the communication is mainly used for passing the state of activities and providing additional information for actors to make decisions. The synchronized communication also reflects the data dependency between activities. With the modal μ-calculus, the following type of inconsistencies in data dependencies may be detected:

- $\nu Z.[\overline{ch}](\mu Y.\langle - \rangle tt \wedge [-ch]Y) \wedge [-]Z$, whenever there is an input on the channel ch, the output on the channel ch eventually happens
- $\nu Z.\langle - \rangle tt \wedge \langle - \rangle Z$, the satisfaction of this property shows that there is no deadlock in the process, that is to say there is no deadlocks resulting from erroneous data dependencies.

Race Condition. As for the modelling of software process in the polyadic π-calculus, the actors in software processes are shared resources. In theory, the recursive definition with concurrent self-combination in Rule 1 enable an actor to do infinite tasks simultaneously. It is equivalent that there are infinite copies of the same actor. Although it has a elegant form, the process has infinite states and its capability has to be limited to meet the requirements of finite-sate verifications. One of the solutions is to replace the process in Rule 1 with finite copies. For example, if an actor can accept at most two tasks at the same period, the process for an actor is defined as:

$$A_{ac} \stackrel{def}{=} assign_{ac}(start, end).\overline{start}.end.A_{ac} | assign_{ac}(start, end).\overline{start}.end.A_{ac}$$

As actors are commonly shared resources, the allocation of tasks should be balanced. The property featuring whether a race condition arise can be defined as:

$$\mu X.[assign_{ac}][assign_{ac}]tt \wedge \langle - \rangle tt \wedge [-assign_{ac}]X$$

It states that the fact that the actor ac may be assigned two tasks at the same time eventually becomes true. If it is regulated that no one can be assigned more than one task at the same time, the satisfaction of the property will result in an inconsistency.

4.2 Environment-Level Inconsistencies

As for the detection of environment-level inconsistencies, it is surveyed from the relationship between the process model and the performance model. If the process model and the performance model show the same behaviors, no environment-level inconsistency occurs. Although the process model and the performance model do not have the same behaviors, there is still no environment-level inconsistency if the behaviors of the performance model can be accomplished by the process model.

In the process algebra literature, there are extensive studies on the equivalence and partial order relationships on processes based on their behaivors[6] [7]. The weak observational equivalence (or weakly bisimulation equivalence) and may preorder , which will be used later, are defined as:

Definition 4 (Observational Equivalence)
Let $(\mathcal{S}, \mathcal{A}, \rightarrow)$ be a labelled transition system, where $\mathcal{S}$ is a set of states (or processes), $\mathcal{A}$ is a set of actions, and $\rightarrow \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation. τ is a transition label which is not externally visible. The weak transition is defined as:

- *$q \Rightarrow^{\varepsilon} q'$ denotes $q = q_0 \rightarrow^{\tau} q_1 \rightarrow \tau \cdots \rightarrow^{\tau} q_n = q', n \geq 0$*
- *$q \Rightarrow^{\alpha} q'$ denotes $q \Rightarrow^{\varepsilon} q_1 \rightarrow^{\alpha} q_2 \Rightarrow^{\varepsilon} q_2, \alpha \neq \tau$*

Then, the relation $\mathcal{S}$ is a weak bisimulation relation if whenever $q_1 \mathcal{S} q_2$ then:

- *$q_1 \rightarrow^{\alpha} q_1'$ implies $q_2 \Rightarrow^{\alpha} q_2'$ for some q_2' such that $q_1' \ \mathcal{S} \ q_2'$*
- *$q_2 \rightarrow^{\alpha} q_2'$ implies $q_1 \Rightarrow^{\alpha} q_1'$ for some q_1' such that $q_1' \ \mathcal{S} \ q_2'$*

q_1 and q_2 are observationally equivalent, or weakly bisimulation equivalent, if $q_1 \ \mathcal{S} \ q_2$ for some weak bisimulation relation $\mathcal{S}$ [9].

Definition 5 (May preorder). *Let $t \in (\mathcal{A} - \{\tau\})^*$ be a sequence of visible actions, $\Rightarrow^t$ is a weak transition, and $L(p) = \{s \in (\mathcal{A} - \tau)^* | \exists p'.p \Rightarrow^t p'\}$ is the language of p. Then the process p is the may preorder of the process q, if $L(p) \subseteq L(q)$ [10].*

Correspondingly, the process performance and the process enactment are environmental inconsistent if the process model and the performance model are not observational equivalent and the performance model is not the may preorder of the process model. The may preorder is used for detecting environment-level inconsistencies when only partial of the process model is executed.

5 Case Study

In this paper, CWB-NC [11] and SPIN [12] are employed for analyzing the obtained polyadic π-calculus expressions. To use these existing mature tools, it is necessary to interpret the verification of software processes as a problem accepted by the corresponding tool. The diagnostic information provided by the tools is shown to be valuable for process designers to reconcile the process enactment and the process performance.

Domain-level Inconsistency. In the example, the activity *Coding* depends on the generated test cases produced in the activity *Test Cases* . The activity *Coding* can not begin

to execute until it receives the notification from the activity *Test Cases* . The property characterizing the data dependency is expressed by the following modal μ-calculus:

$$\mu X.[ch1](\mu Y.[\overline{ch1}](\nu Z.[\overline{ch1}]ff \wedge [-]Z) \wedge \langle - \rangle tt \wedge [-\overline{ch1}]Y) \wedge \langle - \rangle tt \wedge [-ch1]X$$

It states that the activity *Coding* will wait for the notification of the availability of test cases, and the activity *Test Cases* will eventually produce the required artifacts.

In SPIN, the property can be described as: $\langle\rangle p \&\& \langle\rangle q$ (equivalent to $\langle\rangle p \&\& \langle\rangle p \rightarrow \langle\rangle q$), where p is defined as "E_CO[e_co_id]@receive" and q is defined as "E_TE[e_te_id]@send" through the definition macro of Promela. The predicate p state that the statement labelled with *receive* of the process E_CO, representing the E_a process of the *Coding* activity, can be executed immediately. The statement labelled with *receive* will wait for receiving the notification of the availability of test cases from the channel $ch1$. The predicate p is interpreted analogically.

When being verified in SPIN, the property will hold in the processes shown in Figure 3 and Figure 4. It denotes that the process model is correct w.r.t the specified data dependency and no domain level inconsistency arises during the performance of Figure 4. However the property is violated in the the second performance shown in Figure 5. SPIN tells that the performance ends in an invalid state and the trace leading to the invalid state can be repeated through guided simulation. The trace is valuable to process designers when he or she want to locate and resolve the inconsistency. He or she can learn how the inconsistent internal state is reached and thus smart decisions can be made on how to modify the process model or adjust the behaviors of developers.

Environment-level Inconsistency. When being input into CWB-NC, the process model defined in Figure 4 is the may preorder the performance model in Figure 5. The action associated with the activity *Development* is declared as internal behaviors. However, the performance in Figure 5 is not the may predorder of the process model. CWB-NC gives the Hennessy-Milner formula that discriminates one from another. The formula that given by CWB-NC is:

$$[[trigger\_CO]]ff$$

where the action $trigger\_CO$ symbolizing the start of the activity *Coding*. The formula states that the activity *Coding* is unreachable in the performance model shown in Figure 5, but not in Figure 3. In actual, their is a deadlock in the performance model. The diagnostic information can help process designers efficiently locate and resolve the environment-level inconsistency.

6 Related Work

The modelling of software processes has been one of basic subjects in the software process literature. While software process modelling and software process enactment have been discussed extensively, software process analysis has not been discussed to the same extent. Amongst the very few software process modelling approaches that deal with software process model analysis, [13] analyzes the static and dynamic properties

of software processes modelled in FUNSOFT. Those properties are closely related the analysis techniques developed for Petri net. [14][15] analyze processes written in Little-JIL with the data flow analysis tool, FLAVERS [16]. [17] verifies the process in Little-JIL and the interpreter of Little-JIL with LTSA. These researches mainly focuses on the model checking of processes, that is the domain level inconsistencies discussed in this paper. In addition, most of existing languages are designed for enhancing the understanding of processes and supporting the enactment of process models, other than for formal analysis or verification.

In [2], the author gives a formal framework for clarifying the concept of the two types of inconsistencies and describing the relationship between them. But the author says little about how to detect inconsistencies and what support should be provided to locate and resolve them. In [18], the author describes a PCSEE tolerating the existence of inconsistencies. However, the correctness of process models can not be checked mechanically due to the PLAN language used in the PCSEE. It is possible that an invariant is violated even if no deviation has been performed if an incorrect process model is enacted. In addition, no more help is provided to process designers when the reconciliation is carried out, other than the articulated operations are listed. [19] provides an approach to manage inconsistency using viewpoints, which is analogous with viewpoints for requirement engineering. The analysis of processes expressed with viewpoints has to be manually conducted. The effectiveness of the approach highly depends on the experience of the analyst and is limited to processes with small scale.

7 Conclusion

As a conclusion, it is inevitable that inconsistencies will arise in software processes. The consequence of the inconsistencies is that the process performance will deviate from the process model and that the performance may be completely out of the control of PCSEEs. An algebraic approach is proposed in this paper to effectively detect both domain-level and environmental level inconsistencies. Valuable information is provided to help process designers efficiently locate and resolve detected inconsistencies. The effectiveness of the approach is demonstrated through an example problem.

In this paper, it is assumed that a process model has been discovered from the process performance. It is possible that the discrepancy between the discovered performance model and the process performance results in inconsistencies, although the process performance is consistent with the process model. Under this circumstance, the diagnostic information obtained during the process for detecting inconsistencies can still help process designers locate the source of discrepancy.

References

1. Dowson, M., Fernström, C.: Towards requirements for enactment mechanisms. In Warboys, B., ed.: EWSPT. Volume 772 of LNCS., Springer (1994) 90–106
2. Cugola, G., Nitto, E.D., Fuggetta, A., Ghezzi, C.: A framework for formalizing inconsistencies and deviations in human-centered systems. ACM Trans. Softw. Eng. Methodol. **5**(3) (1996) 191–230

3. Li, M.: Expanding the horizons of software development processes: A 3-D integrated methodology. In Li, M., Boehm, B.W., Osterweil, L.J., eds.: ISPW. Volume 3840 of LNCS., Springer (2005) 54–67
4. Li, M.: Assessing 3-D integrated software development processes: A new benchmark. [20] 15–38
5. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes – part I and II. Journal of Information and Computation **100** (1992) 1–77
6. Milner, R.: The polyadic π-calculus: a tutorial. In: Logic and Algegra of Specificatio, Springer-Verlag (1993)
7. Sangiorgi, D., Walker, D.: The π-calculus: a Theory of Mobile Processes. Cambridge University Press (2001)
8. Stirling, C.: Bisimulation, modal logic and model checking games. Logic Journal of the IGPL **7**(1) (1999) 103–124
9. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
10. Cleaveland, R., Hennessy, M.: Testing equivalence as a bisimulation equivalence. In: Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems, London, UK, Springer-Verlag (1990) 11–23
11. Cleaveland, R., Li, T., Sims, S.: The concurrency workbench of the new century: user's manual. SUNY at Stony Brook. (2000)
12. Holzmann, G.J.: The model checker spin. IEEE Trans. Softw. Eng. **23**(5) (1997) 279–295
13. Bröckers, A., Gruhn, V.: Computer-aided verification of software process model properties. In: CAiSE '93: Proceedings of Advanced Information Systems Engineering, London, UK, Springer-Verlag (1993) 521–546
14. Cobleigh, J.M., Clarke, L.A., Osterweil, L.J.: Verifying properties of process definitions. In: International Symposium on Software Testing and Analysis. (2000) 96–101
15. Raunak, M.S., Chen, B., Elssamadisy, A., Clarke, L.A., Osterweil, L.J.: Definition and analysis of election processes. [20] 178–185
16. Cobleigh, J.M., Clarke, L.A., Osterweil, L.J.: FLAVERS: A finite state verification technique for software systems. IBM Systems Journal **41**(1) (2002) 140–165
17. Lerner, B.S.: Verifying process models built using parameterized state machines. In: ISSTA '04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis, New York, NY, USA, ACM Press (2004) 274–284
18. Cugola, G.: Tolerating deviations in process support systems via flexible enactment of process models. IEEE Trans. Softw. Eng. **24**(11) (1998) 982–1001
19. Sommerville, I., Sawyer, P., Viller, S.: Managing process inconsistency using viewpoints. IEEE Trans. Softw. Eng. **25**(6) (1999) 784–799
20. Wang, Q., Pfahl, D., Raffo, D.M., Wernick, P., eds.: Software Process Change, International Software Process Workshop and International Workshop on Software Process Simulation and Modeling, SPW/ProSim 2006, Shanghai, China, May 20-21, 2006, Proceedings. In Wang, Q., Pfahl, D., Raffo, D.M., Wernick, P., eds.: SPW/ProSim. Volume 3996 of Lecture Notes in Computer Science., Springer (2006)

Cost Estimation and Analysis for Government Contract Pricing in China

Mei He[1,3], Ye Yang[2], Qing Wang[1], and Mingshu Li[1,4]

[1] Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
{hemei,wq,mingshu}@itechs.iscas.ac.cn
[2] Center for Systems and Software Engineering, University of Southern California, 941 W. 37th Place, SAL 330, Los Angeles, CA 90089 USA
yangy@Sunset.usc.edu
[3] Graduate University of Chinese Academy of Sciences, Beijing 100039 China
[4] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China

Abstract. Software cost estimation methods and their applications in government contract pricing have been developed and practiced for years. However, in China, the government contract process has been questioned in some aspects. It is largely based on analogy to past experience and/or expert judgment, with a lack of informed decision making supported by mature estimation methods. Moreover, two primary stages of the contract review process for technical and finance contents are disjointed, which greatly limits the accuracy and efficiency of government investment decision. To improve cost estimation and assessment practices in Chinese government contract pricing, we propose the COnstructive GOvernment cost MOdel (COGOMO), which provides guidance and insights for formal cost estimation. This model emphasizes the importance of accumulating knowledge from both government and industry data repositories, and leverages to establish an industry benchmarking reference model for local government contract pricing. It integrates multiple classical research results in addition to COCOMO II, and establishes the first formal model on software cost estimation and analysis for Chinese government context. A list of suggestions is also discussed for government's further improvement on estimating practices.

Keywords: Cost estimation, Government contract pricing, Cost analysis.

1 Introduction

Software estimation is an integral part of a mature software process, and reliable estimates help the perfection of software practices in terms of predictability and manageability [1-4], while poor estimation is listed as one of the two most common causes of runaway projects[5].. Though mature tools and processes for cost estimation have been in market for over 20 years, difficulty is still being reported for the government to effectively price the costs within software development programs [6].

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 134–146, 2007.

In government contract pricing, analysts are expected to produce better predictions in order to control project investment, where cost estimation techniques have been widely adopted and are playing an increasingly important role. For example, the southernSCOPE method [7] is adopted in Australia, and Contract Pricing Reference Guide [8] is deployed in Defense Procurement and Acquisition Policy in the USA.

In China, software industry has been experiencing great-leap-forward development in the past 20 years, especially after the promulgation of No. 18 Document by the State Council in year 2000. During this period, the size of software industry was increased from 23.8 billion RMB (~$2.9 billion) in 2000 to 390 billion RMB (~$48.8 billion) in 2005. Meanwhile, the central government has invested 4 billion RMB (~$0.5 billion) in software, and spent over 30 billion RMB (~$3.8 billion) on purchasing e-government products [9]. State government has adopted many preferential policies to encourage the development of domestic software organizations, and the amount of funds keeps on expanding. However, the performance in government contract projects is not very satisfactory. Some primary reasons include: (1) Government contract pricing is largely determined by subjective judgment, without support from past experience or formal estimation tools; (2) Government records of historical contract projects are rarely collected and maintained in a consistent and centralized way, and available data is insufficient in quantity and mostly incomplete in quality in order to be directly reused; and (3) There is a lack of contract pricing benchmark for government contractors as reference when preparing for proposals. This may lead to unexplained costing items or irresolvable cost discrepancies between proposals and the actual cost.

In this paper, we present an approach named COnstructive GOvernment cost MOdel (COGOMO) to address the above issues. COGOMO provides guidance and insights to formal cost estimation, emphasizes importance of accumulating government knowledge base, and leverage to establish an industry benchmarking reference for local government contract pricing.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 analyzes existing problems in our government contract pricing. In Section 4, 5, and 6, the COGOMO approach is described, and its implementation and application in one typical government department are presented and discussed. Finally, section 7 concludes our achievement and points out future works.

2 Related Work

Among the large number of cost estimation models proposed over the last 20 years, COCOMO [4] is a well known and widely used one, especially, its effort estimation formula (shown bellow) is considered to be a general and typical format. As shown in formula (1), COCOMO II [10] takes the project size and cost driver values as input, known as an essential idea for most effort estimation model, which is also applied in our modeling method.

$$PM = A \times Size^{E} \times \prod EM_i \quad \text{while } E = B + 0.01 \times \sum SF_i \tag{1}$$

While SEER [11] and PRICE [12] are the other two leading commercial software cost estimation models, they are often used in together with COCOMO II by government organizations such as NASA to drive and compare cost estimates [13].

At the same time, governments abroad have taken some strategies in contract pricing. For example, Victorian State Government in Australia developed a new approach called southernSCOPE. It allows businesses to purchase software development on a dollar per function point basis, which can be compared to a cost per square meter basis used in the construction industry. As another example, for large US government contracts, "costing analysts" break the whole cost into direct cost and indirect cost, and multiply direct costs by various rates to obtain the total cost [1].

These ideas are proved helpful in their government. Nevertheless, the definition and understanding of cost are not the same as ours, and our experience and data resource are quite different. For example, insurance and proposed profits are all excluded in our government's contracts.

In addition to these comparisons, we have also investigated and referred to researches on such as role sets of software process in RUP (Rational Unified Process) [14], effort distribution differences among different application types of software in SPR (Software Productivity Research) [15], and industry benchmarking analysis in ISBSG (International Software Benchmarking Standard Group) [16].

3 Problem Description

Since China first adopted market-oriented economic reform over 20 years ago, the forms of government sponsorship have shifted from top-down assignment to bottom-up application or tender contracts. Especially in recent years, the government emphasizes the way of government procurement to choose contract undertakers, typically in science and technology sponsored projects and e-government projects. Both application forms submitted by applicants and tender documents filled by contract undertakers basically consist in two parts: one is called technique or content part and the other is called price or finance part. In this paper, we will use the terms "Content Tender Document (CTD)" and "Finance Tender Document (FTD)" to refer to those two parts respectively. CTD describes what product or services can be provided by bidder, mainly involving implementation design, function description, technical details, expected effort, while FTD details how much it costs and how this money will be spent which usually includes the cost such as equipment procurement, required development effort and related cost, and service effort and related cost.

Currently, the government contract pricing generally includes the following steps: identifying subject matter, calling for bids in the forms of CTDs and FTDs; reviewing CTDs and FTDs by different government expert groups independently, drawing comprehensive scores with respect to each bid based on feedback from independent reviews, and making final contract decisions as depicted in Fig. 1.

Several problems will inevitably take place in this review process. For example, if one costing item of "performing inspections abroad three times" is listed in FTD, reviewers of FTD will just make sure whether the cost is reasonable every time and it is exactly three times planned rather than twice. However, in reality, it may be very likely that there is no need to perform inspection abroad according to the requirements listed in CTD. Hence, the current way of judging bids is often being questioned.

After the investigation, we find a key issue is that two parts of review are lack of collaboration, and information is rarely communicated and exchanged between the two

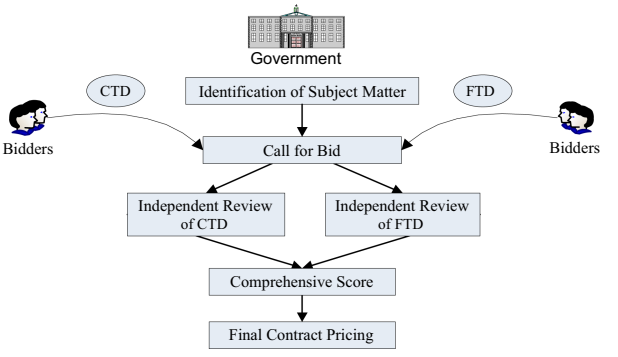


Fig. 1. Typical steps during government contract pricing in China

groups. Frequently, this leads to seriously biased cost estimation and analysis results. As a typical example, the proposed effort estimation and labor rate in some FTDs are seldom compared with the proposed work product in CTDs.

To address these problems and improve the government contract decision process, the COGOMO model is proposed to provide guidance and insights to formal cost estimation and analysis in government software contract pricing in China.

4 Overview of COGOMO

COGOMO is developed based on the COCOMO II model with respect to Chinese government project characteristics. It emphasizes the importance of accumulating knowledge from both government and industry data repositories, and then leveraging on such knowledge to establish an industry benchmarking reference for local government contract pricing.

As illustrated in Fig. 2, the three elements in COGOMO include: an established Government Knowledge Base (GKB), which will provide information to support the following two parts of work; an effort estimation model, and a cost analysis module, which are concerned in reviews of CTDs and FTDs respectively. The figure also shows that these two parts are integrated through estimated effort which aims to bridge the current gap between reviews of CTDs and FTDs.

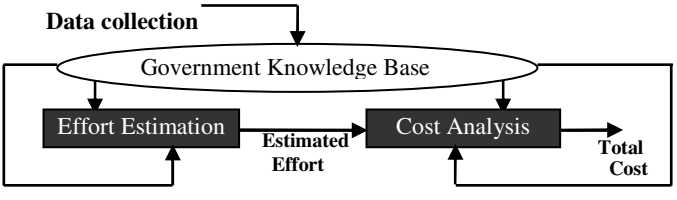


Fig. 2. Overview of COGOMO

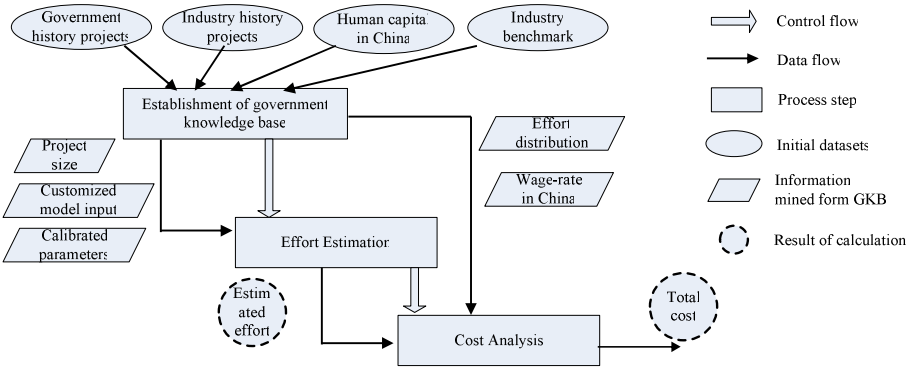


Fig. 3. COGOMO based cost estimation process

Based on COGOMO, a specific cost estimation process, as depicted in Fig. 3, can be set to elaborate the three steps. Step 1, establishing the GKB; step 2, modeling effort estimation relationship and step 3, analyzing total cost.

Step 1: establishing GKB.
The establishment of GKB is based on our empirical study planned, performed, and analyzed on international and domestic data from both industry and government software projects. In our study, particular emphasis is laid on data analysis from the following four sources to derive an effective GKB:

1) Government historical projects, which can help to analyze the requirements of government contract, composition of expenditures and project types all through the ages;
2) Industry historical projects, which can reflect software development conditions across organizations in China;
3) Human resources in China, which can illustrate labor classification like role sets in software process RUP [14] and provide local industry level information like incomes for software personnel;
4) Industry benchmark, which can make a reference for model calibration and government assessment based on local industry benchmark.

Step 2: modeling effort estimation.
In this step, effort estimation is modeled based on COCOMO II, whereas some revisions need to be made to meet the requirements for government contract pricing in China. According to the knowledge obtained from government and industry historical projects in GKB, default COCOMO II model drivers are tailored and model parameters are locally calibrated.

In practical applications of effort estimation, users input project size, rating of cost drivers, then the model will calculate estimated effort as output.

Step 3: analyzing total cost.
Since staff cost often dominate the overall software project cost [2], the total cost can be divided into the labor cost and other non-labor cost.

At first, on the foundation of effort distribution analysis and wage-rate information gained from industry historical projects and human resource survey in GKB, the labor cost at industry income level can be gained; then, taking other non-labor cost into account, the total cost can be received finally.

Taking this 3-step approach as guideline, we applied it in the process of cost estimation modeling for government sponsored project contract pricing in Beijing Municipal Science & Technology Commission (BMSTC for short). The following context will describe the implementation of our approach step by step in detail.

5 Modeling the COGOMO

5.1 Establishing Government Knowledge Base

Government historical projects. Data of 152 historical projects was collected from BMSTC, which contain the information of project name, description, and expenditure items. Analysis on this dataset led us to the following two findings:

- First, all the 152 government sponsored projects are classified as 5 general attributes: development type, development language, platform, application type, and architecture. For the future, this classification scheme will be refined while new projects are added to GKB so that further analysis can be performed such as comparing differences and similarities among different types of projects.
- Secondly, based on statistical analysis using expenditure record of government historical projects, it is concluded that all reasonable expenditures composing total cost are primarily from 8 categories: labor cost, investigation cost, tenancy cost, traveling and communication cost, printing Cost, collaboration cost, energy and material cost, and others (such as cost for software purchases, project inspection).

Industry historical projects. To collect industry historical projects, we designed projects questionnaire, and distributed to 12 domestic software organizations.

In the questionnaire, we design 28 questions with 6 categories of issues including general information, personnel, development environment, effort, sizing, and actual

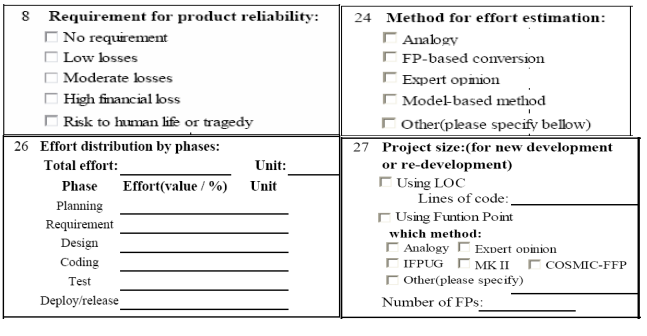


Fig. 4. Questionnaire sample questions

cost composition, referring to information gathered in COCOMO [10] and ISBSG [16] data collection forms,. For the convenience of repliers, only 16 of them are answering questions and the others are multiple choice questions; for the understandability of questions, we specify the description for each choice. For example, as to product reliability, descriptions (shown in Fig. 4) in 5 options are corresponding to the 5 levels (very low, low, nominal, high, very high) for RELY in COCOMO.

Up till now, 16 responses from 7 software organizations were received, and the size distributions are shown in Fig. 5.

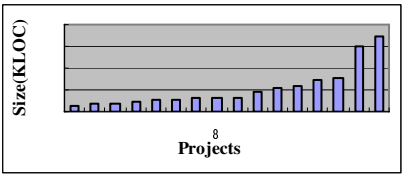


Fig. 5. Size distribution of 16 projects

Human Resources in China. To collect labor rate data in software development, we interviewed six experts all with more than 10 year experience in software administration or government projects assessment. Combined with further references such as survey reports of domestic consultation company, the labor rate pattern in Chinese software industry is summarized in Table 1.

Table 1. Labor rate pattern in Chinese software industry. (Unit: USD/Month).

Type		Wage rate
Technical	Requirement analysis / acquirement	1000-1500
	Architecture analysis	875-1250
	Implement (Programming)	375-500
	Database design	1000
	Test personnel	375-1000
	Technology support	500
Supervisory	Project manager	1000-1875
	Quality assurance	375-1000
	Configuration management	375-625
	Product manager / Problem principal	1875-3125
System management		375-625
Document editor		375-500
Other (e.g. researcher)		750

Industry Benchmark. Currently, there is no formal existing industry benchmark database for software industry in China, and ISBSG (International Software Benchmark Standard Group) sample datasets including 501 projects data are introduced. In the sample datasets, there are 11 categories of information attributes for each project; they are rating, sizing, effort, productivity, schedule, quality, grouping attributes, architecture, documents & techniques, project attributes, and size other than

FSM (Functional Size Measurement). For the purpose of our study, we are interested in a subset of such attributes as shown in Table 2.

Based on our approach as guideline, we can carry on effort estimation and cost analysis for sponsored projects in BMSTC using current achievement gained in GKB.

Table 2. Attributes included in our study from ISBSG datasets

Sizing	Count Approach	***Architecture***	Architecture
	Functional Size		Web development
	Adjusted Function Points	***Effort***	Normalized Work Effort
Grouping Attributes	Development Type	***Project Attributes***	Development Platform
	Application Type		Language Type
Documents & Techniques	Development Techniques		Primary Programming Language

5.2 Modeling Effort Estimation

In Effort Estimation, based on classical effort formula in COCOMO II (refer to formula 1 in Section 2), our work are mainly focusing on the selection of model input and calibration of model constant parameters with respect to GKB.

Tailoring for COGOMO. [17] proposes a reduced parameter modeling approach which improves model accuracy by dropping insignificant cost drivers and leveraging on reasoning between organization characteristics and its historical project costs and schedules.

In our study, a subset of COCOMO II cost drivers is selected in accordance with government requirements in contract pricing and analysis results of GKB. The principles for our selection are:

Principle 1: To be significant. Some factors having common value for most projects are not included. This principle helps in excluding TEAM, DOCU, SITE, and SCED drivers.

Principle 2: To be organization-equal. The factors selected can only distinguish different projects while shielding the difference between organizations. This helps us to further drop RESL, PMAT, ACAP, PCAP, APEX, PLEX, PCON, and LTEX drivers.

Principle 3: To be accessible. They can be accessed and measured at early phases of projects in most software organizations. PREC and TOOL are eliminated from COGOMO according to this principle.

Principle 4: To be appreciable. The government can evaluate the veracity of information offered by the applicant organizations. This principle confirms the deletion of RESL, TEAM, and TOOL by previous principles.

Finally, 8 of COCOMO II cost drivers are left to be included in COGOMO effort estimation model, including FLEX, RELY, DATA, CPLX, RUSE, TIME, STOR, and PVOL. (For more information about the definition and rating levels for cost drivers, see [10]).

Calibrating COGOMO. To increase the effort prediction accuracy of COGOMO, ISBSG sample datasets (the only industry benchmark data in GKB by now) are used to calibrate its model parameters.

Local calibration is performed by running linear regression on project actuals (i.e. actual size and effort) and fitting the data into such an equation as shown below. Newly calibrated model constant A' and B', can be derived from equations of $A' = e^{\beta_0}$ and $B' = \beta_1$.

$$Ln(PM) = \beta_0 + \beta_1 \times Ln(Size) \tag{2}$$

In our case, 501 data points have been used for the local calibration at one time, and the points disseminate seriously (see Fig. 6). This is largely because different project types always make big differences, so the parameters are further calibrated according to project classification discussed in "government historical projects" in Section 5.1.

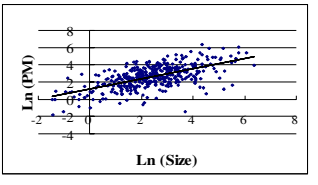


Fig. 6. Regression without classification

Linear regression is run on each type of projects in ISBSG, and calibrated A' and B' are obtained respectively. Fig. 7 and Table 3 demonstrates part of our calibration result. In them, enhancement and new development describe different development types. Since there is no cost driver ratings information in the dataset, the current r-square values and accuracies are not very high, these are expected to be improved as we collect more government projects for the future calibration.

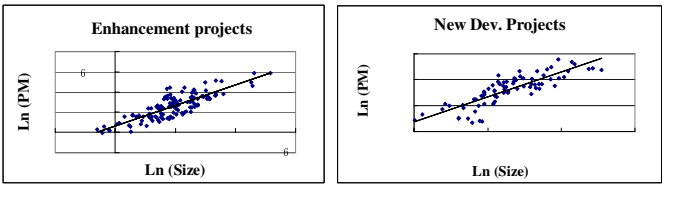


Fig. 7. Examples of linear regression with projects classification

Table 3. Examples of calibrated parameter and prediction accuracies with project classification

	A'	B'	r-squares	PRED(.30)	
				Before classification	After classification
Enhancement	1.61	1.01	0.70	33%	37%
New Development	1.81	0.96	0.71	36%	42%

Since our calibration result is based on much more industry benchmark data than before, it should be more convincing to represent industry benchmark level.

5.3 Analyzing Total Cost

In COGOMO, the total cost is established based on labor cost calculated from effort estimate and labor rate pattern, and other dominant non-labor cost.

Analysis of Labor Cost. Based on the labor rate pattern in GKB, total effort estimate obtained from COGOMO effort estimation model is broken down into various work types to get each cost respectively, and the summation of them is total labor cost.

Since the current data we have do not include any information on effort distribution, our labor cost analysis module mainly relies on existing literature. Among several researches on effort distribution, SPR (Software Productivity Research) [15] reports effort distribution on the basis of different application types, which is also an information item in GKB; the government can use SPR's data easier.

Meanwhile, it is found from the data of SPR that MIS (Management Information System) has special distribution values while others differ slightly. Hence, only MIS and Non-MIS projects are partitioned for simplification. In the future, as the information of applicant or tender documents is improved and increased continuously, we can analyze effort distribution using our own data, and adjust current values.

Finally, referring to activity types identified above and labor types demonstrated in Section 5.1, we set a mapping relationship between work and labor types, as shown in Table 4. The summation of labor cost on each type of work is total labor cost.

Table 4. Work types and labor types mapping with relative wage-rate (Unit: USD/Month)

Work Type	% of total effort		Labor Type	Wage-rate
	MIS	Other		
Requirement	3.7%	4.1%	Requirement analysis/acquirement	1000-1500
Design	7.7%	22.0%	Architecture analysis	875-1250
Coding	18.6%	23.3%	Implement (Programming)	375-500
CM	1.3%	1.8%	Configuration management	375-625
Documenting	4.4%	6.0%	Document editor	375-500
Test	53.3%	30.5%	Test personnel	375-1000
PM	11.0%	12.3%	Project manager	1000-1875

Establishment of Total Cost. Total cost estimation is established according to the portion of labor cost. At present, a total of 88 data points are used to derive the relationship between labor cost and total cost estimation, excluding the projects which are of types such as hardware development, technical investigation and research reports, outsourcing and society service.

In our study, SPSS Ver. 11.0 was used for statistical analysis [18]. Fig. 8 shows the result of Kolmogorov-Smirnov normality test for labor cost versus total cost, which is significantly normal distribution. In the meantime, the regression on total cost and non-labor cost data also show a very strong correlation, as illustrated in Fig. 9. Hence, using labor cost value and explicit proportional relationship, we can establish estimated total cost eventually.

		Labor cost %
Normal Parameters(a,b)	Mean	39.328182%
	Std. Deviation	14.9760055%
Kolmogorov-Smirnov Z		.603
Asymp. Sig. (2-tailed)		.860
a Test distribution is Normal. b Calculated from data.		

Fig. 8. Nrmality test for labor cost vs. total cost

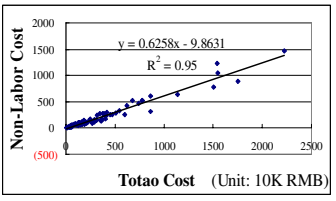


Fig. 9. Non-labor cost vs. total cost

6 Case Study of COGOMO

In 2006, the COGOMO approach was introduced to BMSTC and experimented in cost estimation of government sponsored software projects in BMSTC as initial practical application. We got some constructive feedbacks from BMSTC:

1. The tool developed based on our model passed the acceptance test by government experts group, and has been appointed to assist BMSTC in the new round of government sponsored projects contract pricing.
2. Based on our analysis results, government experts found some bias existing in previous experience. For example, one of their previous intuitive rules is that different development language would impact cost significantly, but in fact the difference is quite marginal.
3. The actual problems met in research led the government realize that some ignored information like project size is a key factor in cost estimation and control, so the required information in applicant form will be revised immediately.
4. Due to the essential necessity of government knowledge base in future improvement, e.g. calibration of estimation model, adjustment of industry benchmark, enhancement of resource management, and so on, our government has launched on establishment of CSBSG (Chinese Software Benchmarking Standard Group).

With respect to the feedbacks from BMSTC, we are continuously growing the GKB and refining the COGOMO approach by incorporating further guidelines and intelligence in support of government contract pricing process improvement. Moreover, we also provided some recommendations for our government policy in standardizing contract pricing. For example, reviewers of tender documents can adopt some parameterized cost estimation models such as COGOMO, and certain required information in CTDs and FTDs sections should be traceable and consistent.

7 Conclusion and Future Works

There is an increasing concensus on estimation as an integral part of software development life cycle, not only in providing rationale for early investigation of project feasibility, but also in facilitating informed decision-making for effective monitoring and control of project progress.

Taking cost estimation in government contract investigation process as initial practical application, this paper proposes the COGOMO approach to address the

increasing difficulty experienced by Chinese government organizations in effectively pricing the cost during its contract pricing process.

The COGOMO approach establishes a Government Knowledge Base using accessible datasets; provides a tailored and calibrated effort estimation model based on COCOMO II using appropriate industry and government data, and supports total cost analysis to obtain labor cost and other non-labor cost based on effort estimation. The initial application of COGOMO approach shows the improvement on cost estimation practices.

Consequently, our research group will also take part in this series of further work, and we plan to continuously enhance our approach on the basis of more datasets accumulated.

Further works to improve our model include calibrating key parameters and statistically analyzing effort distribution across activities with larger local industry data sample, taking the risk factor into account, and further investigating assessment of other non-labor cost.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082, 60473060; the National Hi-Tech R&D Plan of China under Grant No. 2006AA01Z182; the National Key Technologies R&D Program under Grant No. 2005BA113A01. Also, we appreciate all the help offered by the members in the joint research group between ISCAS-iTechs Lab and USC-CSSE (especially to Fengdi Shu, Rong Yuan, Da Yang, Shujian Wu, Zinan Tang and Yuxiang Wan).

References

1. Stutzke, R.D.: Estimating Software-Intensive Systems: Projects, Products and Processes. Addison Wesley Professional (2005)
2. Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous & Practical Approach. 2nd. Edition, PSP Publishing Company, Boston (1997)
3. Boehm, B.W., Papaccio, P.N.: Understanding and Controlling Software Costs. IEEE Transactions on Software Engineering, Vol. 14, No. 10. (1988) 1462-1477
4. Boehm, B.W.: Software Engineering Economics. Prentice Hall, (1981)
5. Glass R.L.: Facts and Fallacies of Software Engineering, Addison Wesley Professional (2002)
6. Vogelhut, C.J.: Contract Software for Government Acquisition Systems and Potential Ethical Concerns. (1998) http://www.nps.navy.mil/wings/acq_topics/sw_acq_ethics.htm
7. southernSCOPE - Avoiding Software Budget Blowouts. (2005) http://www.egov.vic.gov.au/ndex.php?env=-innews/detail.tpl:m1816-1-1-7:l0-0-1-:n832-0-0.htm
8. Contract Pricing Reference Guide. http://www.acq.osd.mil/dpap/contractpricing/
9. 2005-2006 Annual Report on China's Software Industry
10. Boehm, B.W., et al.: Software Cost Estimation with COCOMO II. Prentice Hall, NY(2000)
11. SEER-SEM, http://www.galorath.com
12. PRICE. http://www.pricesystems.com/.
13. NASA Handbook. http://ceh.nasa.gov/downloadfiles/pdfs/gregoryltrcostmgmt.pdf

14. Understanding RUP roles. (2005) http://www-128.ibm.com/developerworks/rational/library/apr05/crain/index.html
15. Jones, C.: Software Assessments, Benchmarks, and Best Practices. Addison-Wesley Longman Publishing Co., Inc., Boston, MA (2000)
16. ISBSG. http://www.isbsg.org
17. Chen, Z.H., Menzies, T., Port, D., and Boehm, B.W.: Feature Subset Selection Can Improve Software Cost Estimation Accuracy. PROMISE, St. Louis, Missouri (2005)
18. SPSS Inc.: SPSS 11.0 for Windows Student Version. Prentice Hall (2001)

A Multilateral Negotiation Method for Software Process Modeling*

Nao Li[1,3], Qing Wang[1], Mingshu Li[1,2], Shuanzhu Du[1], and Junchao Xiao[1,3]

[1] Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China
[2] Key Laboratory for Computer Science, The Chinese Academy of Sciences, Beijing 100080, China
[3] Graduate University of Chinese Academy of Sciences, Beijing 100039, China
{linao,wangqing,dusz,xiaojunchao}@itechs.iscas.ac.cn, mingshu@admin.iscas.ac.cn

Abstract. Currently most software process modeling approaches are predefined, not automatically adaptive to different software projects, and provide little support for development team formation with task and resources allocation in real environments. Based on our ten-year working experience for software organizations, we propose an agent-based multilateral negotiation model MNM-PA to support dynamic software process modelling and ease the work of team formation. MNM-PA brings the following advantages: (1) the software processes are not predefined; (2) the software processes are for given projects and with development teams, allocated tasks and task constrains. MNM-PA is an extension of the classic one-time biding contract net protocol. It defines the main components to model a complete negotiation process for software process construction, especially including the negotiation strategies. MNM-PA is implemented and experimented in a software process management tool namely SoftPM, which is used in more than 100 software organizations in China.

Keywords: Negotiation, Software process modeling, Agent.

1 Introduction

Software process modeling (SPM) has been evolved for approximately 20 years. Facing easily changing software processes, SPM is usually required with some extent of flexibility in order to adapt different software projects. However, most the SPMs applied in real software organizations are predefined, which results in much work of manual adaptation. They also lack the supporting for development team formation with task and resource (time, ability etc.) allocated in real environments, which can ease the work of developers, especially the project managers.

* This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082, 60473060, 60673121; the National Hi-Tech Research and Development Plan of China under Grant No. 2006AA01Z185, 2006AA01Z19B; the National Key Technologies R&D Program under Grant No. 2005BA113A01.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 147–158, 2007.

Basing on our ten-year experience of helping software organizations to construct software processes, we propose an agent-based software process modeling method. The whole idea is to apply agent-based automated negotiation technologies to dynamically form a development team with allocated task and resource for a given software project in real environments. Such a constructed team can be treated as either a real software process to be executed or a decision making support result that can be further adjusted by real developers. Our previous work [1][2][3] on this method focus on agent construction. The agent in our method represents the entities involved in software processes, which we usually call (software) process agent. They are developers, teams, or organizations. The method for the agent construction is first to collect the entity data from the experience database in our software process management tool namely SoftPM [4][5], which is used in more than 100 software organizations, and then construct the agent corresponding to that entity according to the collected data. The agent contains the capability of that entity, the sources and the knowledge it has, etc.

The focus of this paper is the negotiation between process agents. The main aim of the negotiation is to allocate the tasks of a software project to the applicable process agents with applicable resource, such as profit, work time, etc. In [7], a general negotiation model NM-PA to support such negotiation is given and the focus is how to achieve a flexible system design. Since it is general, concrete negotiation forms are not defined, such as unilateral or bilateral, etc. Based on NM-PA, this paper proposes a special multilateral negotiation model, namely MNM-PA. MNM-PA emphasizes the multilateral characteristics of the negotiation, defines a multilateral negotiation protocol and the corresponding multilateral negotiation strategies.

2 Related Work

The classic contract net protocol (CNP) [8] is a main research framework for non-centralized task allocation. An obvious shortcoming of CNP is that the encounter only happens once (one-time bidding). In software process modeling and many the other applications agents need to interact more than once. To this end, some research extends CNP to adapt different applications. However, CNP-based negotiation with multi encounters result in multilateral (one-many) negotiation, which is been realized important for real applications but little research addresses (most negotiation research study the bilateral e.g. [9]). [10] works on multilateral negotiation and its focus is negotiation strategies. However, its work is not complete. First, the definition of negotiation thread is bilateral and does not show any multilateral characteristics. Second, the given negotiation strategies also only addresses bilateral negotiation, i.e. only handling one message between two agents in each encounter, rather than the multi messages in each encounter, which is the cases in multilateral negotiation.

Our multilateral negotiation protocol is an extension of CNP and the multilateral negotiation strategies are designed to correspond to the protocol. Our negotiation model is a complete multilateral negotiation model.

3 Multilateral Negotiation Model MNM-PA

The negotiation process of the process agents is a one-to-many multilateral negotiation between a negotiation initiator and many responders (we call the negotiation participants except the initiator the responders). Therefore, it consists of many bilateral negotiations, each of which is happened between the initiator and one responder. The initiator is the process agent who allocates the tasks of a given software project. It negotiates with each of the responders with regard to some negotiation objectives, i.e., the attributes of the tasks, such as price, effort, quality, etc., until it chooses one and allocates the tasks to it (for simplicity, we only discuss one task negotiation in this paper but the main idea is same when applied to many tasks). MNM-PA models such negotiation processes. Fig. 1 visualizes MNM-PA by describing the main components in MNM-PA and the relations between them.

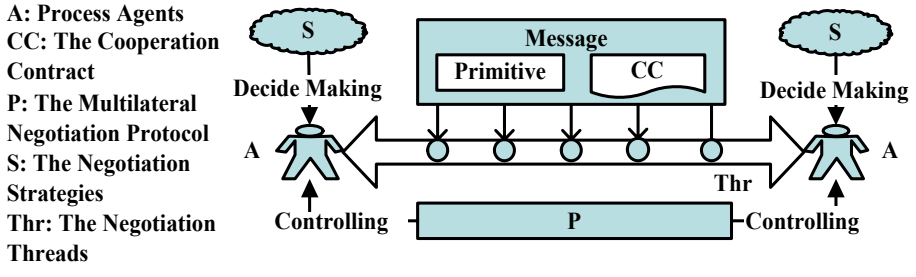


Fig. 1. MNM-PA

Fig. 1 illustrates a bilateral negotiation between two process agents. An MNM-PA based negotiation process consists of many such bilateral negotiations between the negotiation initiator and responders. When it starts, it mainly includes two components: the messages the negotiating agents send and the controlling the agents comply with. In MNM-PA, the messages mainly include two types of information: one is the negotiation primitives and another is the negotiating cooperation contract. The negotiation primitives are defined in our previous work in [7], but as our work proceeds, we change some primitive definitions. Table 1 illustrates the new primitive definitions. The cooperation contract includes the negotiation objective the agents negotiate, i.e., the attributes of some tasks, such as price, effort, quality, etc. The messages the negotiating agents send in each bilateral negotiation consist of a negotiation thread. This is, in an MNM-PA based negotiation, the number of the negotiation threads is same as the number of the bilateral negotiation. E.g., if there are 10 bilateral negotiations between the negotiation initiator and the responders, there are 10 negotiation threads. The length of each negotiation thread increases as the many bilateral negotiation processes proceed. MNM-PA defines the concept of the negotiation interaction to represent the multilateral characteristic of the one-many negotiation. A negotiation interaction represents one round of the interaction between the negotiation initiator and all the responders, i.e., the initiator sends messages to all the responders and then receives all the responding messages from them with regards to those messages it sends. Thus, it consists of the messages from different

Table 1. Negotiation primitive

Propose_N	Propose a negotiation request
Accept_N	Accept a negotiation request
Reject_N	Reject a negotiation request
Propose_CC	Send the initial CC
Accept_CC	Accept the CC received last without further modifications
Modify_CC	Modification to the CC received last
Terminate_N	Terminate the current negotiation thread

negotiation threads. In different negotiation threads the messages are not labeled by a sequential time point but also by a sequential "interaction" number, which denotes which interaction they belong to.

The controlling is done by the multilateral negotiation protocol and the negotiation strategies. The multilateral negotiation protocol is the public constrains all negotiating agents must comply with. The negotiation strategies are private. Each agent has their own negotiation strategies to make decisions. In the following of this section the details about them will be presented.

3.1 Multilateral Negotiation Protocol

The multilateral negotiation protocol defines the public constrains all negotiating agents must comply with. It can be visualized by a state chart (Fig. 2).

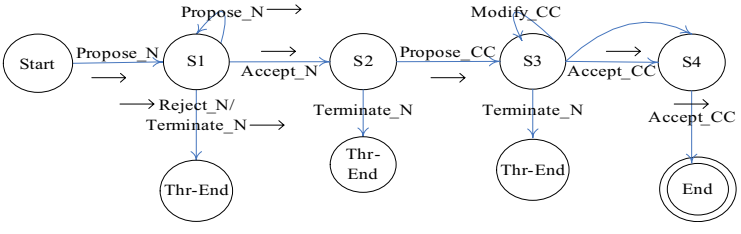


Fig. 2. Multilateral Negotiation Protocol ($NPrim_i \rightarrow$ denotes that the primitive is sent by the negotiation initiator, $\rightarrow NPrim_i$ denotes that the primitive is sent by the responder, and $NPrim_i$ denotes that it could be sent by either of them)

As Fig. 2 illustrates, negotiation agents have seven types of negotiation states. Start is the start state, End is the end state, S_1, S_2, S_3, S_4 is the middle states, Thr-End is the end state of a negotiation thread. The arrows in Fig. 2 denote the state transition. Two additional explanations are: (1) the state transition from S_3 to S_4 could be without any negotiation primitives sent/received; (2) in state S_4 when the negotiation initiator sends an *Accept_CC* to a process agent, it also sends *Terminate_Ns* to all the others and thus the whole negotiation process ends, i.e., the negotiation transits to the state End.

According to the above protocol, an MNM-PA based negotiation process has there phases: the starting phase, the negotiating phase and the ending phase. From state Start to S_2 is the starting phase, where the negotiation initiator in state Start and

S_1 tries to set up the negotiation relationship with other process agents by sending the negotiation request *Propose_Ns* to them and starts several negotiation threads. When it receives the primitive *Accept_Ns* from some process agents who accept the negotiation request, it builds up one multilateral negotiation. If the negotiation initiator receives *Reject_Ns* from some process agent, the negotiations thread between them end.

From state S_2 to S_3 (including from S_3 to S_3) is the negotiations phase. In state S_2 and S_3, in all bilateral negotiations, the initiator and the responders alternate making and sending the values of the task attributes until (1) at least a negotiation responder sends *Accept_CC* indicating acceptance of the task attribute values sent by the negotiation initiator, and then negotiation enters into state S_4, or (2) at least one cooperation contract received by the negotiation initiator is acceptable according to the initiator's negotiation strategy and in this case no negotiation primitive needs to be sent and negotiation goes to state S_4. Notice that the state transmitting from S_3 to S_4 is decided by the negotiation initiator.

From state S_4 to state End is the ending phase, where when the negotiation initiator sends an *Accept_CC* to a negotiation responder to indicate allocating the task to it, and sends *Terminate_Ns* to all the other negotiation responders, the negotiation process ends in terms of the given task.

In the start phase, the negotiation initiator can terminate any negotiation thread by sending *Terminate_N*, and the negotiation responders can terminate their negotiation threads by sending *Reject_Ns*. In the negotiating phase, both the two parts can terminate the negotiation thread by sending *Terminate_N* to the other.

3.2 Multilateral Negotiation Strategies

The multilateral negotiation strategies in MNM-PA mainly apply the evaluation method and some tactics. Negotiating agents have their own evaluation functions and tactics. They use them to privately make decision on their negotiation behaviors. The evaluation functions are used to evaluate the other agents and the cooperation contract the other agents offer. The tactics are used to computer the cooperation contract the agent offers to the other agents. The tactics are designed to be able to adjust in terms of some environment variants. In this section we give the negotiation strategies in each negotiating phase mentioned above.

In the starting phase, the negotiating agents make decisions based on the evaluation of the other process agents. According to the evaluation result, the negotiation initiator chooses the process agents it wants to negotiate with and then send the negotiation request to them; the negotiation responders make decide whether or not to accept a negotiation request once they receive it from a negotiation initiator.

Strategy-Starting. Let $M_{a\to b}^{tn+1}$ denote the message process agent a will send to process agent b at time t_{n+1}, $Vpa_a^b(K_a^t)$ denote the evaluation value towards b by a at time t (K_a^t is the knowledge of a at time k, $Vpa(K)$ is the evaluation function about process agents), $Satisfied_a(Vpa_a^b(K_a^t))$ denote that whether or not $Vpa_a^b(K_a^t)$

satisfies the evaluation criteria of a, then the negotiation strategy in the negotiation starting phase is ($t_a^{\max}$ denotes the time in the future by when the negotiation must be completed for a):

$$M_{a\to b}^{tn+1}=\begin{cases}Terminate\_N, if\ \ t_n=t_a^{\max}\\ Propose\_N, if\ \ Satisfied_a(Vpa_a^b(K_a^{tn}))=True(for\ the\ initiator)\\ Accept\_N, if\ \ Satisfied_a(Vpa_a^b(K_a^{tn}))=True(for\ the\ responder)\\ Reject\_N, if\ \ Satisfied_a(Vpa_a^b(K_a^{tn}))=False(for\ the\ responder)\end{cases}$$

In the negotiating phase, the negotiating agents mainly make decisions on whether or not to accept the cooperation contract the negotiating opponent offer, and if not what cooperation contract they will offer. Specially, the negotiation initiator will decide whether or not to enter the negotiation ending phase in each interaction. The idea of the strategies in this phase is also based on the evaluation method. When a process agent a receives a cooperation contract from b at t_n, it firstly computes a new cooperation contract it will probably send to b at t_{n+1} according to its tactics, and then rates the new contract and that one it receives from b, using its scoring function. If the score of the contract from b is grater than the score of the new contract, the contract from b is acceptable. If a is a responder, it sends an Accept_CC to b. Once there is one Accept_CC from a responder, the negotiation process enters into the ending phase. If a is the initiator, the contract from b will be added to the set of the acceptable contract list and then the negotiation process enters into the ending phase. Under the case that the score of the contract from b is less than the score of the new contract, if a is the initiator, it has to wait until it collects all the other messages from the rest of the responders, and then make decision: if in this interaction there is no acceptable cooperation contract received, nor the primitive Accept_CC (this indicates the negotiation process is still in the negotiating phase), it sends the new contract to b; if a is a responder, it also sends the new contract to b. The following is a relatively formal description of the negotiation strategies:

Strategy-Negotiating. Let $Vcc_a^{tn}(cc_{b\to a}^{tn})$ denotes the evaluation value towards the cooperation contract $cc_{b\to a}^{tn}$ that the process agent b sends to a at time t_n ($Vcc(CC)$ is the evaluation function about cooperation contracts), then the negotiation strategy in the negotiating phase for the negotiation initiator is:

$$M_{a\to b}^{tn+1}=\begin{cases}Terminate\_N, if\ \ t_n=t_a^{\max}\\ Propose\_CC, CC_{a\to b}^{tn+1}, if\ \ state=S2\\ \varnothing, if\ \ V_a^{tn}(CC_{a\to i}^{tn+1}[m])\le V_a^{tn}(C_{i\to a}^{tn}[m-1])\ \ or\ \ Accept\_CC\ \ received\\ Modify\_CC, CC_{a\to b}^{tn+1}, if\ \ V_a^{tn}(CC_{a\to b}^{tn+1})>V_a^{tn}(C_{b\to a}^{tn})\end{cases},$$

and for the negotiation responder is:

$$M_{a\to b}^{tn+1}=\begin{cases}Terminate\_N, if\ \ t_{n+1}>t_a^{\max}\\ Accept\_CC, if\ \ V_a^{tn}(CC_{b\to a}^{tn})\le V_a^{tn}(C_{b\to a}^{tn})\\ Modify\_CC, CC_{a\to b}^{tn+1}, if\ \ V_a^{tn}(CC_{a\to b}^{tn+1})>V_a^{tn}(C_{b\to a}^{tn})\end{cases}.$$

In **Strategy-Negotiating**, $M_{a\to b}^{tn+1}=\varnothing$ indicates that in some interaction, there exists a responder who agrees with the most recent cooperation contract the negotiation initiator sends to it, or the initiator receives at least one cooperation contract which satisfies its evaluation criteria, and the negotiation initiator sends nothing to all the responders and the negotiation ending phase starts.

The key issues in **Strategy-Negotiating** are how to give the $CC_{a\to b}^{tn+1}$, the cooperation contract to the negotiation opponents, and how to evaluation it (i.e., to compare the new computed one with the one received). We first discuss how to compute a cooperation contract. A cooperation contract consists of negotiation objectives, which are the attributes of the task ready to be allocated by the negotiation initiator. Therefore, knowing that how to computer the negotiation objectives knows that how to gets the cooperation contract. In order to computer the negotiation objectives, negotiating agents (whether the initiator or the responders) need to firstly identify an acceptable interval of their values. E.g., for a negotiation objective o_i, an agent identifies its acceptable value range $[Max_i, Min_i]$. In MNM-PA, it is the negotiation initiator who gives the initial values of the negotiation objectives, i.e., the initial cooperation contract, which is usually given by experience (e.g. the middle value between the maximum and the minimum). Besides from the initial value, the other $CC_{a\to b}^{tn+1}$ is given according to the following algorithm:

Algorithm 1. Let ρ denote the basic tactic, τ denote the time-dependent tactic based on ρ, ς denote the resource-dependent tactic similar to ρ, which are used to computer the values of the negotiation objectives within the predetermined value ranges; let w_a^{τ} and w_a^{ς} be the weights of τ and ς respectively that process agent a identifies in a negotiation process, $w_a^{\tau}+w_a^{\varsigma}=1$, $oi_{a\to b}^{tn+1}$ denote the value of o_i ($\in\{O_1, O2, ...O_n\}$) that a is ready to send b at t_{n+1}, $oi_{a\to b}^{tn+1}x$ denote $oi_{a\to b}^{tn+1}$ is derived according to the tactic x ($\in\{\rho,\tau,\varsigma\}$), then $oi_{a\to b}^{tn+1}$ is computed by the following formula (1), or by (1) and (2) successively:

$$(1) oi_{a\to b}^{tn+1} = oi_{a\to b}^{tn+1}\rho$$

$$(2) oi_{a\to b}^{tn+1} = oi_a^{\tau}\times oi_{a\to b}^{tn+1}\tau + w_a^{\varsigma}\times oi_{a\to b}^{tn+1}\varsigma$$

The following definitions define the three tactics in **Algorithm1**.

Definition 1. Let max_a^{oi} and min_a^{oi} denote the maximum and minimum value of the negotiation objectives o_i determined by process agent a, k_a^{oi} denote a constant to determine the negotiation objectives (given by experience), V_a^{oi} denote the evaluation function of a about o_i (for simplicity either monotonically increasing or monotonically decreasing), $\{..., oi_{b\to a}^{tn-2}, oi_{a\to b}^{tn-1}, oi_{a\to b}^{tn}, oi_{b\to a}^{tn+2}, ...\}$ be the negotiation thread

between a and b be with regard to the negotiation objectives o_i, then the basic tactics ρ to give the value $o_{i_{a\to b}}^{tn+1}$ is defined as:

$$o_{i_{a\to b}}^{tn+1}\rho = \begin{cases} \min(min_a^{oi} + n \times k_{oi}^a, max_a^{oi}, o_{i_{b\to a}}^{tn}), if\ V_a^{oi} descreasing \\ \max(max_a^{oi} - n \times k_{oi}^a, min_a^{oi}, o_{i_{b\to a}}^{tn}), if\ V_a^{oi} increasing \end{cases}.$$

Definition 2. Let the value of negotiation objectives o_i determined by ρ be $o_{i_{a\to b}}^{tn+1}\rho$ (see **Definition1**), the time function be $f(t_{n+1}) = \dfrac{min(t_{n+1}, t_a^{max})}{t_a^{max}}$, t_a^{max} denote the time limit of the process agent a to finish the negotiation, $0 \le t_{n+1} \le t_a^{max}$, then the time-dependant tactics τ is defined as:

$$o_{i_{a\to b}}^{tn+1}\tau = \begin{cases} \dfrac{f(tn+1)(2max_a^{oi} - min_a^{oi}) + min_a^{oi}}{f(tn+1)+1}, o_{i_{a\to b}}^{tn}\ not\ exist, V_a^{oi} descreasing \\ \dfrac{f(tn+1)(2min_a^{oi} - max_a^{oi}) + max_a^{oi}}{f(tn+1)+1}, o_{i_{a\to b}}^{tn}\ not\ exist, V_a^{oi} increasing \\ \dfrac{f(tn+1)(2o_{i_{a\to b}}^{tn+1}\rho - min_a^{oi}) + min_a^{oi}}{f(tn+1)+1}, o_{i_{a\to b}}^{tn}\ exist, V_a^{oi} descreasing \\ \dfrac{f(tn+1)(2o_{i_{a\to b}}^{tn+1}\rho - max_a^{oi}) + max_a^{oi}}{f(tn+1)+1}, o_{i_{a\to b}}^{tn}\ exist, V_a^{oi} increasing \end{cases}$$

Definition 3. Replace the function $f(t_{n+1})$ in the **Definition2** with the function $f(t_{n+1}) = c_{oi}^a + (1 - c_{oi}^a)e^{-N^a(t)}$ ($N^a(t)$ is the number of the process agents who negotiates with the process agent a at t), then we get the resource-dependant tactics ς, i.e. $o_{i_{a\to b}}^{tn+1}\varsigma$.

As the above definitions indicate, the time-dependent tactic τ is an adjustment of the basic tactic ρ in terms of negotiation time. When time is tight, the negotiating agents will make a rapider compromise in order to reach an agreement quicker. The resource-dependent tactic ς is also an adjustment of ρ in terms of the number of participant agents in a negotiation process the agent knows. The larger number of the negotiating process agents with a process agent, the less pressure of the process agent and the smaller compromise it makes. **Algorithm1** also gives the flexibility the value of negotiation objectives can be counted by either taken the environmental elements (i.e., the time or the resource) into account or not.

After knowing how to get a cooperation contract (by **Algorithm1**), which consists of several single negotiation objectives (and their values), we need to know how to evaluate it. A cooperation contract usually consists of more than one negotiation objectives, such as the price, the time schedule or period, LOC, the quality of documents, etc. They are sometimes dependent each other, and sometimes independent. When it is the case of the former, the evaluation of them becomes complicated. Here we give an evaluation function about the cooperation function, used by us in the simple

case, i.e. the latter case that the negotiation objectives are independent each other. The algorithm mainly refers to the additive scoring system in [11]:

Definition 4. Let a cooperation contract be $CC = \{O_1, O2, ...O_n\}$, $V_a^i(o_i)$ be the evaluation function of process agent *a* to about the negotiation objectives $o_i (i \in [1, n])$ (for simplicity either monotonically increasing or monotonically decreasing), W_a^{oi} be the weigh that *a* identifies for o_i , $\sum_{i=1}^{n} W_a^{oi} = 1$, then the evaluation function of *a* about the cooperation contract CC is defined as $Vcc_a(CC) = (-1)^k W_a^{oi} V_a^i(o_i)$, where if $V_a^i(o_i)$ is monotonically increasing, then *k=0*, and if $V_a^i(o_i)$ is monotonically decreasing, then *k=1*.

When a negotiation process enters into the ending phase, the main negotiation behaviors are to choose the final task executer by the negotiation initiator and then send the negotiation result to all the responders. The idea of the strategy is to evaluate all the acceptable cooperation contracts sent from the responders and the cooperation contracts sending to the responders and receiving *Accept_CCs* from them in the interaction immediately before the ending phase, and then choose one.

Strategy-Ending. Let *i* ($\in [1, n]$) be the responder who sends the *Accept_CC* to the negotiation initiator *a*, or whose cooperation contract is acceptable, then in the negotiation ending phase the negotiation strategy for the negotiation initiator is:

$$M_{a\to i}^{tn+1} = \begin{cases} Accept\_CC_{a\to k}, Terminate\_N_{a\to i} (i \neq k), \\ \quad if\ (V_a^{cc} descreasing\ and \\ \qquad V(cc_{l\to a}) = Max(\bigcup_{i=1}^{n} V(cc_{i\to a}))(l = [1, n], l \neq \varnothing)\ or \\ \quad if\ V_a^{cc} increasing\ and\ V(cc_{i\to a}) = Min(\bigcup_{i=1}^{n} V(cc_{i\to a}) \end{cases}$$

Strategy-Ending assumes that there is no more than one best cooperation contracts. But the case there is more than one best often happens. In this case, i.e., more than one cooperation contracts are evaluated as the best by the negotiation initiator (this also includes the case when the same cooperation contract sent by the initiator is accepted by more than one responders), either do an evaluation of these responders, or give a more strict evaluation function of the negotiation objectives to do a further evaluation.

Given all the definitions above, when $t > t_{max}$, negotiation threads end, or when $V(cc^{t+1}) \leq V(cc^t)$, the negotiation process ends, therefore, in a limited time, i.e. $0 <= t <= t_{max}$, the proposed negotiation model and specified negotiation strategies can guarantee the end of a negotiation process.

With regard to the question that whether the negotiating process agents can reach an agreement about the values of negotiation objectives, obviously, a basic assumption must be satisfied: for a negotiating process agent *a* and *b*,

$[max_a^{oi}, min_a^{oi}] \cap [max_b^{oi}, min_b^{oi}] \neq \varnothing$ ($o_i \in$ CC={O1,O2,...,On}) must hold. Because process agents might use different tactics to make the values of negotiation objectives, i.e. the different values of $w_a^{\tau}, w_a^{\varsigma}$ will result in different compositional tactics, even though the above basic assumption holds, the analysis that whether a negotiation process with $(w_a^{\tau} \times \tau_a, w_a^{\tau} \times \varsigma_a)$ and $(w_b^{\tau} \times \tau_b, w_b^{\tau} \times \varsigma_b)$ can reach an agreement compositional tactics is complicated. But if they only use the basic tactic ρ, under the basic assumption mentioned above, an agreed cooperation contract can be reached.

4 An Example

MNM-PA is implemented in SoftPM, the software development environment we develop. In this section we give an example to explain how MNM-PA works. Due to the page limit, we only give how the process agents negotiate with regards to one task A whole software process for a software project can be constructed in the same way, which is finally a task "tree", i.e. the tasks and the subtasks, with the responding task executors, and the "promise" about them.

In the example there are three process agents, PM, M1, M2. For simplification and an emphasis of the negotiating phase and the ending phase, we assume they build up the negotiation relationship in the negotiation starting phase. Then, PM starts to negotiate with M1 and M2 in terms of the three negotiation objectives of the given task, which are the price, the work time and the quality.

Table 2 shows the value ranges of the negotiation objectives of the task the three process agents determine (by their own historical data) respectively.

Table 2. Value range of negotiation objectives

	PM	M1	M2
Prc(rmb/day)	400-500	400-600	350-500
Prd(day)	20-24	20-30	18-24
Qul(level)	8-10	8-9	8-9

Table 3. Weights of Negotiation Objectives

	PM	M1	M2
Prc	0.4	0.5	0.6
Prd	0.3	0.2	0.2
Qul	0.3	0.3	0.2

For simplicity, the three agents apply the formula (1) of **Algorithm1** to computer the cooperation contract, and the same parameter value of k^{o_i} (see **Definition1**): 50 for the price, 1 for the work time and 1 for the qualification. PM sends the original cooperation contract to M1 and M2 (t_i denotes the time point of the system clock):

$$CC_{pm \to m1}^{t0} = \{ prc_{pm \to m1}^{t0}, prd_{pm \to m1}^{t0}, qul_{pm \to m1}^{t0} \} = \{400, 20, 10\} = CC_{pm \to m2}^{t1}$$

When M1 and M2 receive them, they first compute $CC_{m1 \to pm}^{t2}$ and $CC_{m2 \to pm}^{t3}$, and then compare $CC_{m1 \to pm}^{t2}$ with $CC_{pm \to m1}^{t0}$, and $CC_{m2 \to pm}^{t3}$ with $CC_{pm \to m2}^{t1}$ in terms of the evaluation value based on **Definition4,** respectively.

In order to using **Definition4**, the evaluation function of each negotiation objective should be defined. In the example, $V^{qul} = Value(qul)$, and V^{prc} and V^{prd} are as the following, respectively:

$$V^{prc} = \begin{cases} 1, if\ 250 \le prc < 300 \\ 2, if\ 300 \le prc < 350 \\ 3, if\ 350 \le prc < 400 \\ 4, if\ 400 \le prc < 450 \\ 5, if\ 450 \le prc < 500 \\ 6, if\ 500 \le prc < 550 \\ 7, if\ 550 \le prc \le 600 \end{cases}, V^{prd} = \begin{cases} 1, if\ 15 \le prd \le 16; 7, if\ 27 \le prd \le 28 \\ 2, if\ 17 \le prd \le 18; 8, if\ 29 \le prd \le 30 \\ 3, if\ 19 \le prd \le 20; 9, if\ 31 \le prd \le 32 \\ 4, if\ 21 \le prd \le 22; 10, if\ 33 \le prd \le 34 \\ 5, if\ 23 \le prd \le 24; 11, if\ 35 \le prd \le 36 \\ 6, if\ 25 \le prd \le 26; 12, if\ 37 \le prd \le 38 \end{cases}$$

Obviously for PM, V^{prc} and V^{prd} is monotonically decreasing, and V^{qul} is monotonically increasing, but for M1 and M2 it is reversal. **Definition4** also need the weights of the three negotiation objectives.Table3 gives them for the three agents.

Take the example of M1, according to **Algorithm1**, $CC^{t1}_{m1 \to pm} = \{600, 30, 8\}$. Then according to the evaluation functions defined above, V^{prc} (600)=7, and V^{time} (30)=8 and V^{qul} (8)=8. For the received one, V^{prc} (400)=4, and V^{time} (20)=3 and V^{qul} (10)=10. The result is that $7 \times 0.5 + 8 \times 0.2 - 8 \times 0.3 > 4 \times 0.5 + 3 \times 0.2 - 10 \times 0.3$. Therefore, M1 sends the new computed cooperation contract to PM. Similarly, M2 also sends the new computer one. When PM receives the cooperation contracts from M1 and M2, respectively, it does the similar thing according to **Algorithm1** and **Definition4**, until the negotiation process enters the ending phase.

At the second interaction in the negotiation phrase, the evaluation of the {450,23,9} received from M2 by PM is higher than that of the {500,22,9} to be sent by PM, and the evaluation of the {550,29,9} received from M1 by PM is lower than that of what PM wants to send. Therefore the negotiation process enters the ending phase and only the cooperation contract {450,23,9} sent by M2 is used for the final evaluation. Finally PM sends "Accept CC" to M2. According to the protocol, the whole negotiation process finishes and PM allocates the task to M2, and the constrains on the task are identified in the agreed cooperation contract with the price of 450 RMB/Day, the work time of 23 days and the quality level of 9.

5 Conclusions and Further Work

This paper proposes a multilateral negotiation method for an agent-based software process modeling. Bases on MNM-PA, the process agents representing the entities involved in software processes, such as software organizations, development teams, persons etc., can negotiate in distributed environments with regard to the tasks of a given software project to reach agreements with the task allocation and the related resource allocation. Compared with the traditional software process modeling, our method provides an un-predefined software modeling and decision making support of

team formation with task and resource allocation for different software projects in reality.

MNM-PA integrates negotiation with the one-time biding of the classic CNP, thus supporting the both cooperative and competitive negotiation among the process agents. MNM-PA defines all the main components including the negotiation strategies to support a full life cycle negotiation process modeling of the process agents. In particular, the negotiation strategies consider the environmental elements thus adaptive to environmental changes.

Further work includes the experimental analysis of the proposed tactics to obtain the best or better experience values of their weights, parameter values, etc., and the comparison of the software processes constructed by our approach with those constructed by real project mangers, for further improvement of MNM-PA.

References

1. X. Zhao, M. Li, Q. Wang, K. Chan, H. Leung. An Agent-Based Self-Adaptive Software Process Model. Journal of Software, vol.15 (3), pp.348-359, 2004.
2. X. Zhao, K. Chan, M. Li. Applying Agent Technology to Software Process Modeling and Process-Centered Software Engineering Environment. In Proceedings the 2005 ACM Symposium on Applied Computing (SAC'05), pp.1529-1533.
3. Q. Wang, J. Xiao, M. Li, M. W. Nisar, R. Yuan, L. Zhang. A Process-Agent Construction Method for Software Process Modeling in SoftPM. Q. Wang et al. (Eds.): SPW/ProSim 2006, LNCS 3966, pp. 204–213, 2006.
4. Q. Wang, M. Li. Software Process Management: Practices in China. M. Li, B. Boehm, and L.J. Osterweil (Eds.): SPW 2005, LNCS 3840, pp. 317–331.
5. User Manual of Software Management Platform for CMM/CMMI/ISO9000. Institute of Software, Chinese Academy of Science, 2005.
6. P. Barthelmess. Collaborantion and Coodination in Process-Centered Software Development Environments. A Review of the Literature. Information and Software Technology, 45(13),pp.911-928,2003
7. N. Li, M. Li, Q. Wang, S. Du. A Negotiation Model in an Agent-Based Process-Centered Software Engineering Environment. In Proceedings of SEKE 2006 (The 18th International Conference on Software Engineering and Knowledge Engineering), San Francisco, USA, pp.664-669, 2006
8. T. Sanholm. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In Proceedings of the Eleventh National Conference on Artificial Intelligence, pp.256--262, 1993
9. S. Paurobally, P.J. Turner, N.R. Jennings. Automating Negotiation for M-Services. IEEE Transaction on Systems, Man, and Cybernetics-Part A: Systems and Humans, 33(6), pp.709-724, November 2003.
10. C. Sierra, P. Faratin, N.R. Jennings. A Service-Oriented Negotiation Model between Autonomous Agents. In Proceedings of 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, pp.17-35, Ronneby, Sweden, 1997.
11. H. Raiffa. The Art and Science of Negotiation. Harvard University Press, Cambridge, USA, 1982.

Distributed Global Development Parametric Cost Modeling

Ray Madachy

University of Southern California Center for Systems and Software Engineering
941 W. 37th Place, Los Angeles, CA, USA
Cost Xpert Group, Inc.
3131 Camino Del Rio N., San Diego, CA, USA
madachy@usc.edu

Abstract. Geographically distributed development processes are becoming ever more pervasive on modern software projects. Software is developed collaboratively in multiple locations around the world, and projects are being contracted out in whole or part for economic leverage. Projects are often split among distributed teams, where the teams contribute different portions of work per phase to take advantage of their skill sets and rates. Thus there is a need for new parametric cost estimation models where effort multipliers are phase-sensitive. Working with industrial partners, a unique model has been developed to better estimate globally distributed projects where work is allocated by phase, rather than along the lines of specific functionality. The distributed development model allows for work distribution by phase per team (and per module), different environmental characteristics of the teams, localized labor categories, calendars, compensation rates and currencies for costing. It also provides a generalized scheme for user-defined global lifecycle processes that include calibrated effort and schedule distributions. A representative example project shows primary inputs and some fine-grained outputs available with the model.

Keywords: Distributed development, global software development, parametric cost modeling, cost estimation, COCOMO, Detailed COCOMO, phase-sensitive effort multipliers, software lifecycles, distributed teams, labor distribution, subcontracting.

1 Introduction and Background

Economic trends are disrupting software business models as geographically distributed development is becoming ever more pervasive on modern software projects [1]. Software is developed collaboratively in multiple locations around the world, and projects are being contracted out in whole or part. This results in needs for new estimation models of distributed development processes. The Cost Xpert Group has developed a unique and innovative model to better estimate globally distributed projects with the support of the USC Center for Systems and Software Engineering (USC-CSSE).

Processes are becoming increasingly distributed by geography and company in many sectors. These new projects are typified by outsourcing, or on-shore and off-shore work. Some projects are executed 24/7 around the globe as teams handoff their

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 159–168, 2007.

work between shifts. Projects are often split among different contractor teams with the teams contributing different portions of work and skill sets per phase.

The new model adapts traditional cost estimation formulas for distributed teams by using phase-sensitive effort multipliers. A project can be defined in terms of the distribution of software work by phase per team. The unique attributes of each team are also used in the calculations for more detailed and accurate estimates. Without these new capabilities, distributed teams could not estimate and create plans with enough detail to split out the expected workloads and labor costs. Currently there are no other estimation models or tools with this capability.

It addresses the important software industry growth in developing nations such as India or China, with increasing distributed team development and off-shore arrangements with other contractors or customers.

The model is especially powerful in conjunction with an enhanced lifecycle scheme for unlimited phases. Cost Xpert has generalized its cost and schedule models for flexible user-defined lifecycle processes.

1.1 Industry Collaboration

The model has been developed based on extensive collaboration with industrial partners practicing distributed development processes. Over the last few years, affiliates of USC-CSSE and Cost Xpert customers have expressed an increasing need for model extensions covering distributed processes. Traditional parametric models such as COCOMO II [2] cannot account for the effort variance due to different teams, nor provide insight at the detailed level for project planning and execution these companies want.

Examples of major global companies helping with the model include Unisys, Wipro Technologies and Cognizant Technologies. Companies are using the model in its current spreadsheet form (until it is included in a future Cost Xpert product update), and data is also being collected for further model validation and local calibrations.

2 Model Overview

Foundations of the new model are allowing the variation of effort multipliers by phase and a separation of factors for local vs. global project attributes. The Cost Xpert model is based on COCOMO II [2], and the extension for distributed development is a modern generalization of the Detailed COCOMO model [3] that provides fully flexible user-defined phases. The new model partitions the cost drivers between local team-level and global project-level attributes. It also allows team-level labor category distributions per phase with local hourly rates and currencies to be defined.

Detailed COCOMO allows for cost driver effort multipliers to vary by phase, but the Cost Xpert model provides greater flexibility than Detailed COCOMO because the latter is defined for only four fixed waterfall phases circa 1981. The Detailed COCOMO model adds another layer of complexity on top of Intermediate COCOMO [3], upon which COCOMO II is based. The phase-sensitive multiplier framework is not been implemented in other COCOMO-based vendor estimation tools. The new model leverages the phase-sensitivity of effort multipliers to capture the variance due to different team characteristics by phase.

The distributed development model is comprised of the following elements:

- The notion of a *Team* consisting of personnel environmental factors and labor parameters. Each team is defined in the estimate by rating the people related factors and providing labor categories, rates, labor distributions and working calendar parameters.
- The Cost Xpert lifecycle model [4] which allows for user-defined phases that are transformed into a Work Breakdown Structure (WBS) for detailed project plans [5]. The lifecycle is also described in terms effort and schedule distributions across phases.
- A new team distribution capability where their distributions can be assigned to individual phases defined by the lifecycle.
- The Cost Xpert multiple module effort model which allows for size and environmental factors to be assigned to individual modules.
- A revised labor costing model accounting for localized labor distributions, rates and currencies.

Adding the dimension of cost on top of effort is an important differentiator for these global project types. The effort profiles alone don't tell the project story given the radical differences in team cost structures across the globe. These disparities are normally the bottom line economic rationale for global outsourcing strategies. When the differences in rates are accounted for then the model allows a robust range of "what if" experimentation in terms of teaming strategies. Following are more differentiators of the model for distributed, global development processes.

2.1 Work Allocation by Phase vs. Module

Large projects are more frequently distributing work among teams by phase, rather than along the lines of software components or specific functionality. For example, global companies may perform early definitive work in inception and elaboration onshore where domain expertise lies while using offshore resources for the bulk of construction. This work allocation is increasingly feasible with improved software methodologies (including model-driven approaches), documentation and toolsets that support the entire lifecycle. New group collaboration technology allows improved coordination across the distances.

The Cost Xpert distributed development model reflects this new reality in its assumptions. Individual module-level inputs are used to derive the total phase estimates and work allocation across teams is calculated per phase.

2.2 Different Working Calendars

Different working calendars can also be defined for each team. Other software cost models are limited to a single value for hours per person-month (HPM) on a project. The HPM parameter represents the standard number of working hours for an average month and is used in effort and schedule calculations.

Cost Xpert has extended its model algorithms to allow for different working calendars on a project for multiple teams. Revisions were made to allow for multiple HPMs for the different teams in the effort equations, and the schedule formulas were

modified to synchronize the integrated activities over time. The COCOMO II HPM default is 152 hours and other models use 160 hours, but some projects we have dealt with have teams operating at 190 HPM.

2.3 Effort Multiplier Variation by Phase

Parametric model effort multipliers are traditionally invariant across phases, which is a macro approximation. The phase-sensitive effort multiplier framework allows for more sophistication and fidelity of estimates whereby each environmental factor can have unique multiplier settings for each phase. Examples include the *Required Software Reliability* cost driver where the relative effort impact varies more between the low and high settings for downstream integration and test activities compared to up-front lifecycle activities [3], or a custom factor for peer reviews that increases effort in elaboration but decreases it in construction [6]. In these cases the internal multipliers can vary by phase.

Besides distributed team processes, phase-sensitivity supports another common scenarios for rating of environmental cost factors by phase. Projects with long time horizons will likely have important factors vary over the duration. These may include experience factors accounting for learning, factors for planned process and tool improvements, platform factors to account for developing hardware or planned platform changes, anticipated project/organization disruptions and other factors. For example if experience factors are averaged across a long project lifecycle instead of by phase, then the project plan will be imbalanced with understated staffing needs in the beginning and overstated levels towards the end.

2.4 Algorithm Overview

The model refines COCOMO II formulas for phase-specific effort multipliers, team work distributions and local team attributes. On top of that it extends it for labor category distributions to provide more fine-grained outputs for personnel resource planning. The standard top level effort formula for COCOMO is

$$Effort = A * Size^{B} * \prod_{i=1}^{N} EM_i \ . \tag{1}$$

Where

- *Effort* is in person-months
- *A* is a constant derived from historical project data
- *Size* is in KSLOC (thousand source lines of code), or converted from other size measures
- *B* is an exponent for the diseconomy of scale dependent on additive scale drivers
- EM_i is an effort multiplier for the i^{th} cost driver. The geometric product of *N* multipliers is an overall effort adjustment factor to the nominal effort.

The top level effort is decomposed in the new model for each phase, team, labor category, and then aggregated across the same dimensions and time periods determined by the schedule outputs. The changed effort algorithms sum the effort across

phases, where each phase accounts for effort multipliers unique per phase. The resulting effort phase distribution may differ from the default lifecycle distribution due to differences in the team cost drivers being unevenly weighted within the phases.

To get phase-level estimates, first the nominal (unadjusted) effort for each phase in the project is determined with

$$Effort_{NOM_p} = Effort\%_p * A * Size^B. \quad (2)$$

where $Effort\%_p$ is the nominal percent of lifecycle effort in phase *p*. The new model uses the team distributions per phase and their local effort multipliers to calculate an adjusted effort for each team in each phase per

$$Effort_{ADJ_{t,p}} = Effort_{NOM_p} * Effort\%_{t,p} * \prod_{i=1}^{N} EM_{t,i}. \quad (3)$$

where $Effort\%_{t,p}$ is the percent of lifecycle effort for team *t* in phase *p,* and the effort multipliers are those unique to team *t.* The adjusted effort outputs from Equation (3) are summed up across teams for each phase, matrixed with their labor distributions, then finally spread over time per the phase schedule spans to get detailed labor outputs.

An outline of the steps to provide fine-grained and top-level outputs using the revised effort equations follows:

- Calculate unadjusted project effort per phase
- For each team
  - Calculate unadjusted effort
  - For each phase
    - Distribute basic effort across phases with nominal lifecycle distribution percentage
    - Calculate adjusted effort per phase with phase-specific effort multipliers
- For each phase
  - Sum the effort across all teams
- Calculate the adjusted lifecycle effort and schedule distribution
- Calculate the normalized lifecycle effort and schedule distribution
- For each time period
  - Calculate effort for current phase
  - Decompose effort by team
- For each team
  - For each labor category
    - For each time period
      - Allocate the portion of team effort
  - Add up the effort for all labor categories
  - Add up the cost for all labor categories
- For each time period
  - Add up the effort for all teams
  - Add up the effort for all teams
- Aggregate the team results by phase for project-level phase outputs.

2.5 Project Estimation Example

This section illustrates an example usage of the model for a representative project for a large systems integrator, and shows some of the different outputs available. In this scenario a company is developing a project distributed globally across North America, Europe, China and India. The internal supply chain system will be transitioned to use at the North America and Europe sites.

The project has chosen to standardize its global lifecycle process based on the Rational Unified Process (RUP) [7], [8]. A global lifecycle based on RUP is defined for the project per Fig. 1. In this example the default RUP effort and phase distributions from COCOMO II [2] and [8] are used, but could be based on other calibrations. Additionally any number of phases could be defined for a lifecycle.

Global Lifecycle

Name Rational Unified Process

Phase	Effort (%)	Schedule (%)
Inception	5.0%	10.0%
Elaboration	20.0%	30.0%
Construction	65.0%	50.0%
Transition	10.0%	10.0%
TOTAL	100.0%	100.0%

Fig. 1. Defining global lifecycle

Fig. 2 shows the top-level project definition with how work will be distributed in terms of percentage of software by each team for the defined lifecycle phases. For simplicity in this example, we will not show individual modules but will aggregate them into a single size measure.

Project Name Supply Chain Distributed Project

SLOC 105000

Start Date Jan-07

Team	Inception	Elaboration	Construction	Transition
North America	70%	50%	5%	20%
Europe	30%	35%	5%	30%
India	0%	15%	60%	40%
China	0%	0%	30%	10%
Totals	100%	100%	100%	100%

Fig. 2. Defining size and work distribution

Each team also defines its personnel factors, labor calendars, labor categories and rates with local currencies. Examples of the personnel factors and labor categories are shown in Fig. 3 and Fig. 4. These are the subset of environmental cost factors related to people that may vary by team. The rest of the cost factors for product, platform and project attributes generally apply to the project at-large (or individual modules) and are not shown.

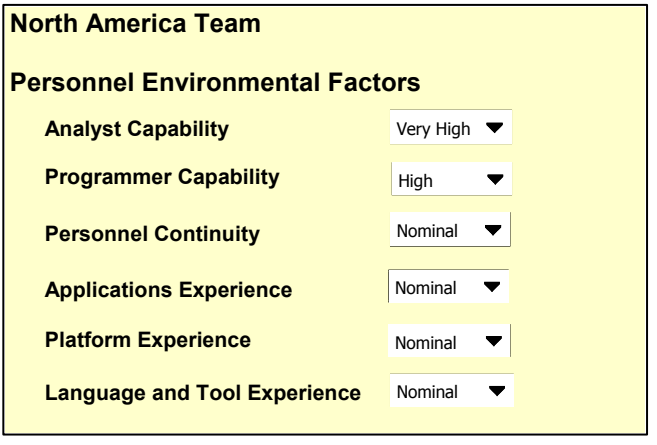


Fig. 3. Rating personnel factors for North America team

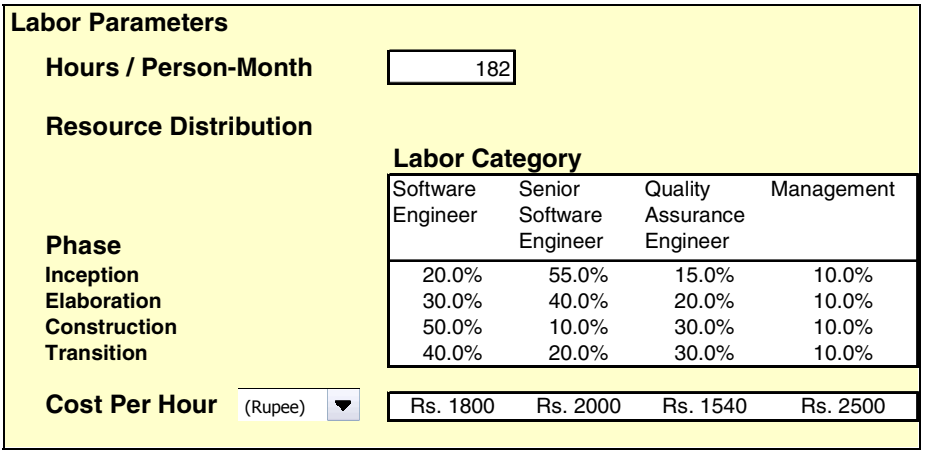


Fig. 4. Defining labor parameters for India team

Fig. 5 shows the top-level effort, schedule and cost outputs for the entire project. The costs can be viewed for any chosen currency used by the teams. Additionally the per-team slices can be displayed. The resulting effort and schedule for the project are reflected in an overall staffing profile, and project plan portions for individual teams

	Effort (PM)	Schedule (Months)	Cost (Dollars)
Inception	24.2	2.7	$ 257,550
Elaboration	105.6	8.8	$ 1,123,085
Construction	331.2	14.2	$ 3,523,819
Transition	56.1	3.1	$ 596,491
Totals	517.0	28.8	$ 5,500,945

Fig. 5. Top-level outputs

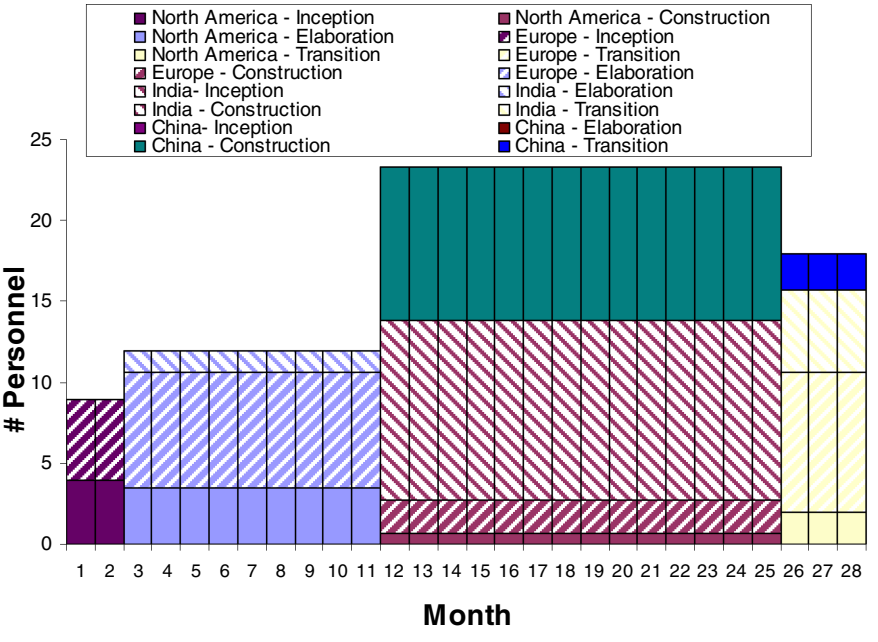


Fig. 6. Top-level staffing profile

are also available. Fig. 6 shows the overall staffing profile, with staffing decomposed by phase and team.

Estimates and staffing plans are also available on a more detailed level for each team. Fig. 7 shows the portion of the staffing plan for the North America team using the labor categories defined for that team alone. The personnel levels correspond to the work portions for the team by phase, and are adjusted for their environmental factors (i.e. cost drivers) and local calendars in terms of hours worked per month.

However the profiles of different teams may vary widely across the phases, and are necessary to have for more detailed planning. Fig. 8 shows the detailed staffing profile for the India team. It includes the specific labor categories used on that team, and the effects of the India team personnel factors on effort throughout the lifecycle.

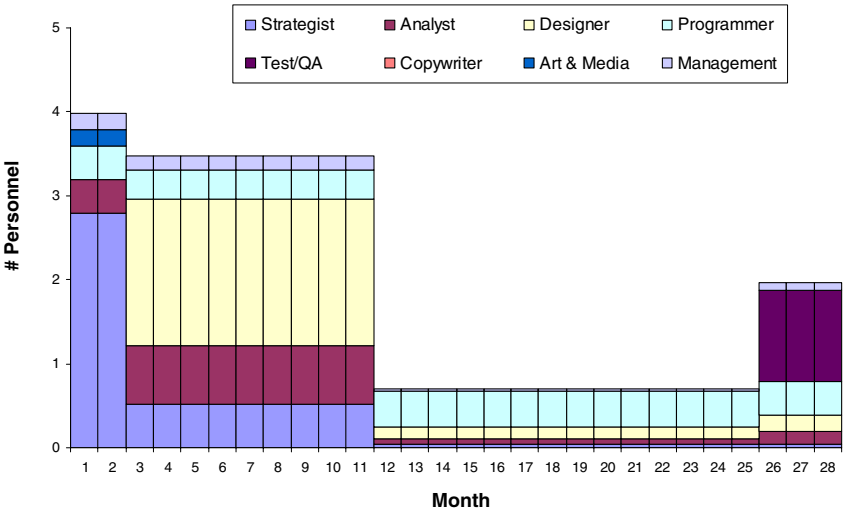


Fig. 7. Staffing profile for North America team

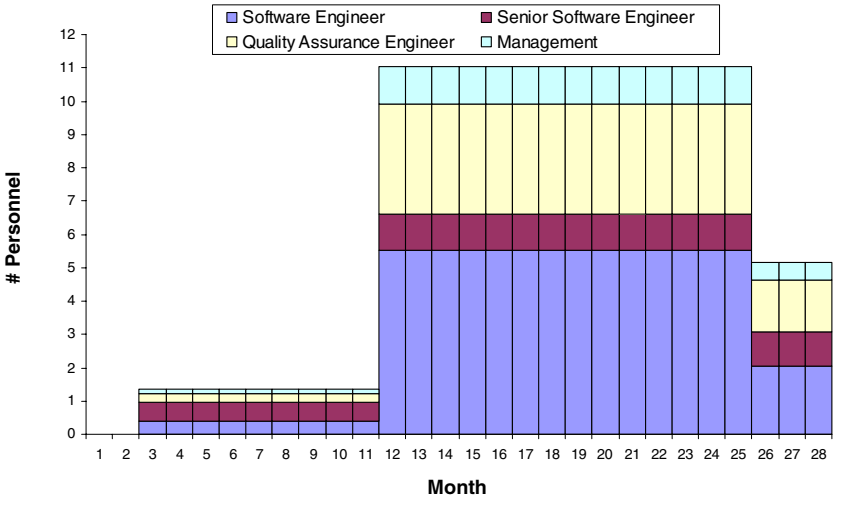


Fig. 8. Staffing profile for India team

Costs in local currency are also available. Without this team-level planning detail, the individual teams would not be able to derive executable plans for their work portions. All team plans can also be exported to Microsoft Project format for project execution.

3 Conclusions and Future Work

Globally distributed processes require new estimation models. The capabilities to allow variation of effort multipliers by phase and to account for other parametric team

differentiators are model innovations that provide substantial benefits to organizations involved in distributed software processes. They support critical business needs by enabling users to better model realistic estimation and planning scenarios. When providing more detail by phase the resulting estimates are also likely to be more accurate. These improvements support higher fidelity estimates and better balanced project plans with sufficient detail for execution.

Feedback from users and evaluators is promising, and the model is being enhanced further. It is currently implemented in a spreadsheet, with a few hardwired restrictions such as the available number of phases, teams or labor categories. It will be generalized more when it is included in future product updates of the Cost Xpert software estimation tool. A number of other features are under consideration. One is the capability for users to re-partition the cost drivers for other factors that may vary by team.

We are providing the spreadsheet model to USC-CSSE affiliate companies, Cost Xpert customers and others upon request. Interested parties may contact the Cost Xpert Group for further information and to obtain the spreadsheet (it runs on Microsoft Windows, Macintosh and Linux operating systems and requires Microsoft Excel or OpenOffice Calc). We are also soliciting additional data on distributed global projects for further model calibration and validation.

References

1. Boehm, B.: “The Future of Software Processes”, Proceedings of the International Software Process Workshop, SPW 2005, Springer-Verlag, (2005)
2. Boehm, B., Abts C., Brown A., Chulani S., Clark B.,Horowitz E., Madachy R.,Reifer D., Steece B.: Software Cost Estimation with COCOMO II, Prentice-Hall (2000)
3. Boehm, B.: Software Engineering Economics, Prentice-Hall (1981)
4. Cost Xpert Group: “Cost Xpert Lifecycle Model”, White Paper, Cost Xpert Group, San Diego, CA (2005)
5. Cost Xpert Group: “Cost Xpert Work Breakdown Structure”, Internal White Paper, Cost Xpert Group, San Diego, CA (2005)
6. Madachy R.: Software Process Dynamics, IEEE Computer Society Press (2007)
7. Kruchten, P.: The Rational Unified Process, Addison-Wesley (1998)
8. Royce W.:, Software Project Management - A Unified Approach, Addison-Wesley, (1998)

Process Mining Framework for Software Processes

Vladimir Rubin[1,2], Christian W. Günther[1], Wil M.P. van der Aalst[1], Ekkart Kindler[2], Boudewijn F. van Dongen[1], and Wilhelm Schäfer[2]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{c.w.gunther,w.m.p.v.d.aalst,b.f.v.dongen}@tue.nl
[2] University of Paderborn, Paderborn, Germany
{vroubine,kindler,wilhelm}@uni-paderborn.de

Abstract. Software development processes are often not explicitly modelled and sometimes even chaotic. In order to keep track of the involved documents and files, engineers use Software Configuration Management (SCM) systems. Along the way, those systems collect and store information on the software process itself. Thus, SCM information can be used for constructing explicit process models, which is called *software process mining*. In this paper we show that (1) a Process Mining Framework can be used for obtaining software process models as well as for analysing and optimising them; (2) an algorithmic approach, which arose from our research on software processes, is integrated in the framework.

Keywords: Software Process Mining and Management.

1 Introduction

Software and information systems are still becoming more and more complex. One of the distinguishing features of any engineering effort is the fact that process engineers create, change, update and revise all kinds of documents and files. In order to cope with the vast amount of data, documents, and files, engineers use Product Data Management (PDM) systems or Software Configuration Management (SCM) systems such as CVS or Subversion. In addition to maintaining the engineer's documents, these systems collect and store information on the process: Who created, accessed, or changed which documents?, When was a particular task completed?, etc.

The engineering processes themselves, however, are often not well-documented and sometimes even chaotic: engineering processes tend to be far less structured than production processes. In order to help engineers to identify, to better understand, to analyse, to optimise, and to execute their processes, the process data stored in the SCM systems can be used for extracting the underlying engineering processes and for automatically constructing one or more explicit *process models*. We call this *software process mining*.

Process models and software process models cover different aspects. Here, we consider the main aspects only: the *control aspect* captures the order in

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 169–181, 2007.

which tasks are executed (i. e. the control-flow), the *information aspect* captures the data, documents, and information needed and produced by a task, and the *organisation aspect* captures which persons in which role execute a task. To mine different aspects of software development processes – sometimes called *multi-perspective mining* – we need different algorithms. In order to make all these algorithms available under a single user interface, we use the ProM framework [1]. ProM provides a variety of algorithms and supports process mining in the broadest sense. It can be used to discover processes, identify bottle-necks, analyse social networks, verify business rules, etc. Moreover, ProM provides interfaces to extract information from different sources including SCM systems such as CVS and Subversion.

The focus of this paper is on providing an overview of the application of process mining to software processes. Although we do not focus on the algorithms, we discuss one process mining algorithm, which was specifically developed for software processes and integrated in ProM. Moreover, we discuss in which other ways ProM can help software engineers in dealing with their processes.

The remainder of this paper is organized as follows. First, we present related work. Then, we discuss the application of process mining in software engineering environments. Then, we provide an overview of process mining approaches and the tool support offered by ProM. In Section 5 we show the application of process mining to the Subversion logs of the ArgoUML project where we analysed five subprojects. Section 6 concludes the paper.

2 Related Work

The capabilities of using software repositories for deriving information about the software projects are being researched in the domain of *mining software repositories* [2]. Like in our approach, SCM systems are used as sources of information. They are used for measuring the project activity and the amount of produced failures, for detecting and predicting changes in the code, for providing guidelines to newcomers to an open-source project, and for detecting the social dependencies between the developers. In this area, SCMs are mostly used for detecting dependencies on the code level, whereas we make an effort at building process models and analysing them. Researchers and practitioners recognize the benefits of *software process modelling* with the aid of software repositories [3,4]. Nowadays, process improvement should be ruled by what was actually done during the software development process and not by what is simply said about it. The researchers from this domain examine *bug reports* for detecting defect lifecycles, *e-mails and SCMs* for analysing the requirement engineering processes and coordination processes between developers, their productivity and participation, etc. Although this research direction deals with software processes and their models, there is still a lack of algorithms for producing formal models.

In addition to the software process domain, the research concerning discovering the sequential patterns treats similar problems in the area of *data mining*.

The work of Agrawal and Srikant deals with discovering sequential patterns in the databases of customer transactions [5].

Since the mid-nineties several groups have been working on techniques for process mining, i.e., discovering process models based on observed events. In [6], an overview is given of the early work in this domain. The idea to apply process mining in the context of workflow management systems was introduced in [7]. However, we argue that the first papers really addressing the problem of process mining appeared around 1995, when Cook et al. [8,9] started to analyse recorded behaviour of processes in the context of software engineering, acknowledging the fact that information was not complete and that a model was to be discovered that reproduces *at least* the log under consideration, but it may allow for more behaviour. More information on recent process mining research can be found at *http://www.processmining.org*.

3 Process Mining for Software Engineering Environments

In this section, we first explain the traditional process-centered software engineering environments (PSEE). Then, we present the ideas of the *incremental workflow mining* approach.

3.1 Incremental Workflow Mining Approach

Figure 1 gives an overview of the architecture of a traditional PSEE and represents how our *incremental workflow mining* approach is integrated to this architecture: The environment consists of software repositories (SCM system, defect tracking system, etc...). The *software product* and the interaction among *practitioners* are supported and maintained by the repositories. In the traditional schema, the *Process Engineer* (project manager or department) designs the process model using his experience and existing approaches, like V-model, RUP, etc. Then, the model is instantiated and practitioners follow it during the product life cycle, indicated by the white arrows in Fig. 1. There are the following problems with this schema: The designed process model does not necessarily reflect the actual way of work in the company, human possibilities in detecting discrepancies between the process model and the actual process are limited, practitioners are not involved in the design of the process model.

The main ideas of the *incremental workflow mining* approach were described already in our previous work [10,11]. In this approach we go the other direction, it is shown with gray arrows in Fig. 1: We take the *audit trail information* (document log) of the SCM system, which corresponds to the process instances (particular executions of the process) and, using our *process mining algorithms*, derive the process model from it. Then, the process model can be analysed, verified and shown to the process engineer; he decides which changes should be introduced to the process to optimise and to manage it in a better way. Actually, the mining approach can be used not only for *discovery*, but also for *monitoring* and *improving* real software processes using the data from software repositories in general and SCM systems in particular.

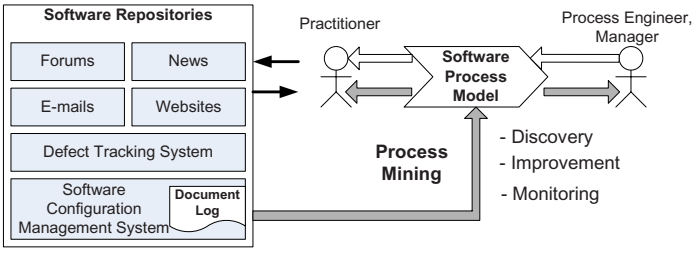


Fig. 1. Process-centered Software Engineering and Process Mining

In software engineering environments, it is usually difficult to introduce a *Process Management System* (PMS) directly from scratch. Using our approach in a *batch mode*, we gather the existing logs of several process instances and automatically generate a model from them. Our approach works also *incrementally*, i.e. as soon as new data is added to the repositories, we refine the overall process model. Following this approach, the role of the PMS changes over time: at the beginning, it is utilized only for storing the newly discovered models; after model improvements, the system can start advising the users and controlling their work in the company. We call this *gradual process support*.

3.2 Input Information

In this section, we focus on the logs of SCM systems and make our experiments with them, but the approach and the algorithms are more general: They also deal with the information derived from other software repositories.

In Table 1, we present an example of the audit trail information from an SCM system. SCM systems record the *events* corresponding to the *commits of documents*. A sequence of these events constitutes a *document log*: It contains the names of the committed documents, timestamps, and author names. Document logs with similar structure can be derived from all kinds of SCM systems, such

Table 1. Document Log

Document	Date	Author
project1/models/design.mdl	01.01.05 14:30	designer
project1/src/Code.java	01.01.05 15:00	developer
project1/tests/testPlan.xml	05.01.05 10:00	qaengineer
project1/docs/review.pdf	07.01.05 11:00	manager
project2/models/design.mdl	01.02.05 11:00	designer
project2/tests/testPlan.xml	15.02.05 17:00	qaengineer
project2/src/NewCode.java	20.02.05 09:00	developer
project2/docs/review.pdf	28.02.05 18:45	designer
project3/models/design.mdl	01.03.05 11:00	designer
project3/models/verification.xml	15.03.05 17:00	qaengineer
project3/src/GenCode.java	20.03.05 09:00	designer
project3/review/Areview.pdf	28.03.05 18:45	manager

Table 2. Filtered Log

Document
DES
CODE
TEST
REV
DES
TEST
CODE
REV
DES
VER
CODE
REV

as *CVS*, *Subversion*, *SourceSafe*, *Clear Case* and others. When we analyse the document logs, we have to identify the cases (process instances), identify the document types, abstract from the details of the log, and ignore unnecessary information. For many software projects, a *case* corresponds to a subproject or a plug-in development, in our example it corresponds to a project development (cases are separated with double lines in the tables). We detect the *documents' types* by identifying similarities of their paths and names, see Sect. 4.1 for details. The same technique is used for abstracting from the log details and for ignoring noise, i.e. ignoring exceptional or infrequent commits. However, the latter issues are also resolved on the algorithm level, see Sect. 4.2.

4 Process Mining Algorithms and Tool Support

In this section, we present the algorithms for multi-perspective software process mining. In the area of process mining, there are different algorithmic approaches, which derive the control-flow, the organization and the information models from the *event logs*. The events in these logs correspond to process *activities* produced by some PMS. In our application area, we have information about the *commits of documents* which occur in SCM systems, but generally can also occur in other systems, like PDM. All the presented algorithms are integrated as plug-ins to the *ProM* tool [1], which is described at the end of this section.

4.1 Abstraction on the Log Level

The document logs often contain either too many details or very specific document names and paths, which are not relevant for the process mining algorithms. Thus, we need a technique to abstract from the concrete names and paths or even to ignore some paths. We call this *abstraction on the log level*. The ProM tool contains a set of *filters*, which help us solving this problem.

Here, we use the remap filter, which maps the names of documents from the log to abstract names. Regular expressions specify the paths that should be mapped to abstract names. For example, if the path contains "/models/", the filename contains "design" and has extension ".mdl", then it should be mapped to "DES". Table 2 shows the result of this filter applied to the log of Table 1.

4.2 Control-Flow Mining

In this section, we describe the control-flow mining algorithms. When dealing with the control-flow, the log can be represented as a set of sequences of documents (sequences are also called cases, traces or execution logs), see Table 2.

Generation and Synthesis Approach. The approach presented in this section is a *two-step approach*: Step 1 takes a document log and generates a transition system (TS) from it; Step 2 synthesises a Petri Net (PN) from the transition system. The algorithmic details of the approach are discussed in [12]. One of the

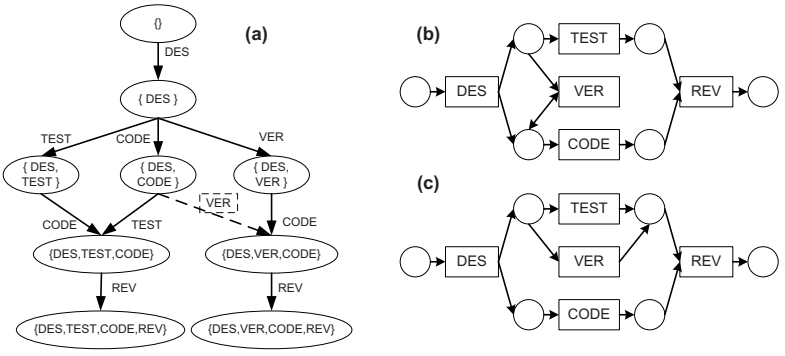


Fig. 2. Generated and Synthesis Approach: (a) Transition Systems (b),(c) Petri Nets

main advantages of the approach is the capability to *construct* transition systems and, then, to apply different *modification strategies* depending on the desired degree of generalization; we call this "clever" transition system generation or *abstraction on the model level*. Despite the fact that transition systems are a good specification technique for making experiments, they are usually huge, since they encode such constructs as concurrency or conflict in a sequential way. Thus, the algorithms developed within such a well-known area of Petri net theory as *Petri net synthesis and theory of regions* [13] are used for transforming transition systems to Petri nets, which are more compact.

The transition system shown in Fig. 2(a) with the solid arrows is *constructed* from the log given in Table 2. In this example, a *state* is defined as a *set* of documents representing the complete history of a case at a point of time. For example, for the first case, there are such states as $\{\}$, $\{DES\}$, etc. There are transitions between all the subsequent pairs of states, transitions are labelled with the names of produced documents. Using the Petri net synthesis algorithms, we generate a Petri net from the given TS, see Fig. 2(b). Events of the TS correspond to the transitions of the PN. This Petri net has the same behaviour as the TS; the concurrency of events $TEST$ and $CODE$, which is modeled sequentially in the TS, is specified more compact in the PN.

But we can also *modify* the constructed TS using some strategy. For example, the "Extend Strategy" adds transitions between two states, which were created from different traces but which can be subsequent because there is a single document which can be produced to reach one state from the other. As a result, we add one transition VER from state $\{DES, CODE\}$ to state $\{DES, VER, CODE\}$, it is shown with the dashed arrow in Fig. 2(a). A Petri net corresponding to this TS is shown in Fig. 2(c). This Petri net is more general than the first one; it allows an additional trace, namely $\langle DES, CODE, VER, REV\rangle$.

The first ideas of the generation and synthesis approach were presented in our previous paper [14], then the algorithms were significantly improved and successfully implemented in the context of ProM; the tool Petrify [15] is used in the synthesis phase. This approach overcomes many limitations of the traditional

process mining approaches; for example, it can deal with *complicated process constructs*, *overfitting* (generated model allows only for the exact behaviour seen in the log) and *underfitting* (model overgeneralises the things seen in the log). However, by now, this approach can hardly deal with *noise* (incorrectly logged events and exceptions), since we do not consider the frequencies of cases in the log; so, the other approaches that treat this problem, are presented in the next Section.

Other Approaches for Control Flow Mining. In the process mining domain a number of algorithms for control flow mining have been developed, which have different characteristics from the previously introduced approach; all these algorithms can be also applied for mining the software processes.

The *Alpha algorithm* [16] can also derive a Petri net model from an event log, however it is based on analysing the immediate successor relation between event types, i.e. documents. Another algorithm, the *Multi-phase approach* [17], creates Event-driven Process Chain (EPC) models from a log, while it first generates a model for each process instance and later aggregates these to a global model. Both the Alpha and the Multi-phase algorithms share the generation and synthesis approach's *precision*, i.e. the generated model accurately reflects all ordering relations discovered in the log.

While sophisticated filtering of logs can remove noise partially, there are also process mining algorithms which are designed to be more robust in the presence of noise. The *Heuristics Miner* [18] employs heuristics which, based on the frequency of discovered ordering relations, attempts to discard exceptional behaviour. Another approach in this direction is the *Genetic Miner* [19]. It uses genetic algorithms to develop the process model in an evolutionary manner, which enables it to also discover e.g. long-term dependencies within a process.

4.3 Mining Other Perspectives

Our generation and synthesis approach deals with the control flow, which is only one *perspective* addressed in process mining. Such information as the timestamp of an event or its originator (the person having triggered its occurrence) can be used to derive high-level information about the process also in other perspectives.

Resource Perspective. The resource perspective looks at the set of people involved in the process, and their relationships. The *Social Network Miner* [20] for example can generate the *social network* of the organization, which may highlight different relationships between the persons involved in the process, such as *handover of work*, *subcontracting* and others. The *Organizational Miner* also addresses the resource perspective, attempting to cluster resources which perform similar tasks into *roles*. This functionality can be very beneficial in a software development process, both for verification and analysis of the organizational structure. Mismatches between discovered and assigned roles can pinpoint deficiencies in either the process definition or the organization itself.

Performance Perspective. Mining algorithms addressing the *performance* perspective mainly make use of the timestamp attribute of events. From the

combination of a (mined or predefined) process model and a timed event log, they can give detailed information about performance deficiencies, and their location in the process model. If some project phase is identified as the point in the process where most time is spent, we could assign more staff to this task.

Information Perspective. The *Activity Miner* [21] can derive high-level activities from a log by clustering similar sets of low-level events that are found to occur together frequently. These high-level clusters, or patterns, are helpful for unveiling hidden dependencies between documents, or for a re-structuring of the document repository layout.

4.4 Process Analysis and Verification

Process mining is a tremendously helpful tool for managers and system administrators, who want to get an overview of how the process is executed, and for *monitoring* progress. However, in many situations it is interesting whether execution is *correct*. To answer this question, there exists a set of *analysis and verification methods* in the process mining domain. One of these techniques is *Conformance Checking* [22], which takes a log and a process model, e.g. a Petri net, as input. The goal is to analyse the extent to which the process execution corresponds to the given process model. Also, conformance checking can point out the parts of the process where the log does not comply.

Another technique is *LTL Checking* [23], which analyses the log for compliance with specific constraints, where the latter are specified by means of linear-temporal logic (LTL) formulas. In contrast to conformance checking, LTL checking does not assume the existence of a fully defined development process. Therefore, it can be used to successively introduce, and check for, corporate guidelines or best development practices.

The ProM framework also features techniques for process model analysis and verification in the absence of a log. Advanced process model analysers, such as *Woflan*, can check e.g. a Petri net model for *deadlocks* (i.e., potential situations in which execution will be stuck), or verify that all process executions complete properly with no enabled tasks left behind. Process designers find these automated tools valuable for ensuring that a defined development process will not run into problems which are hard to resolve later on.

4.5 ProM and ProMimport Tools

The ideas presented in this paper have been implemented in the context of *ProM*. ProM serves as a testbed for our process mining research [1] and can be downloaded from `www.processmining.org`. Starting point for ProM is the MXML format. This is a vendor-independent format to store event logs. One MXML file can store information about multiple processes. Per process, events related to particular process instances (cases) are stored. Each event refers to an activity. In the context of this paper, documents are mapped onto activities. Events can also have additional information such as the transaction type (start, complete, etc.), the author, timestamps, and arbitrary data (attribute-value pairs).

The *ProMImport Framework* allows developers to quickly implement plug-ins that can be used to extract information from a variety of systems and convert it into the MXML format (cf. `promimport.sourceforge.net`). There are standard import plug-ins for a wide variety of systems, e.g., workflow management systems like Staffware, case handling systems like FLOWer, ERP components like PeopleSoft Financials, simulation tools like ARIS and CPN Tools, middleware systems like WebSphere, BI tools like ARIS PPM, etc. Moreover, it has been used to develop many organization/system-specific conversions (e.g., hospitals, banks, governments, etc.). The ProMImport Framework can also be used to extract event logs from such systems as Subversion and CVS.

Once the logs are converted to MXML, ProM can be used to extract a variety of models from these logs. ProM provides an environment to easily add plug-ins that implement a specific mining approach. The most interesting plug-ins in the context of this paper are the mining plug-ins. In addition to that, there are four other types of plug-ins: *Export plug-ins* implement some "save as" functionality for some objects (such as graphs). For example, there are plug-ins to save EPCs, Petri nets, spreadsheets, etc. *Import plug-ins* implement an "open" functionality for exported objects, e.g., load instance-EPCs from ARIS PPM. *Analysis plug-ins* typically implement some property analysis on some mining result. For example, for Petri nets, there is a plug-in which constructs place invariants, transition invariants, and a coverability graph. *Conversion plug-ins* implement conversions between different data formats, e.g., from EPCs to Petri nets and from Petri nets to YAWL and BPEL. Altogether, there are *140* plug-ins for ProM.

The next section will illustrate the application of some of these plug-ins. However, since there are currently more than 140 plug-ins it is impossible to give a representative overview. One of the mining plug-ins generates the transition system that can be used to build a Petri net model. Note that for this particular approach ProM calls Petrify [13].

5 Evaluation and Applications

In order to evaluate our approach, we have chosen the *ArgoUML* project, which is an open-source UML modeling tool maintained by the *Subversion* SCM system. Since this data is freely available, it makes an excellent test case for us. ArgoUML has different subprojects with the same *file organization*; we have chosen five subprojects which implement the ArgoUML support for five different programming languages. We will use these five process instances to derive a formal model of the control-flow, to analyse the organization structure and the performance of the process, and to do some analysis and verification.

First, using the *svn log* utility provided by Subversion, we generated logs for all the five subprojects and imported them to ProM. This log consisted of about 400 commit events. The log contains project specific paths and different commits, which are not relevant for the software process. Using the *remap filter*, we replaced project specific paths with abstract names. Following the ArgoUML conventions, all the committed documents (files) containing "/src/" in

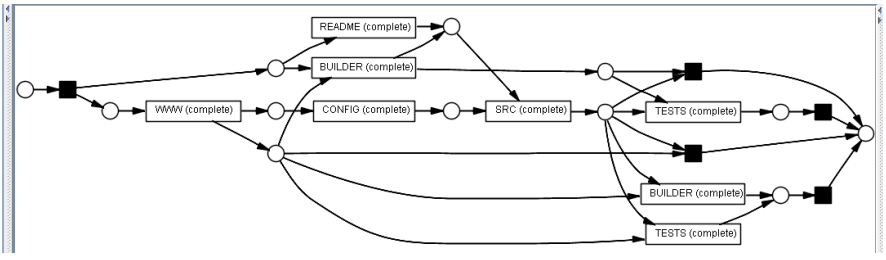


Fig. 3. Petri Net for the ArgoUML Project

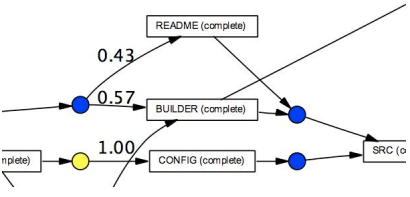


Fig. 4. Performance Analysis

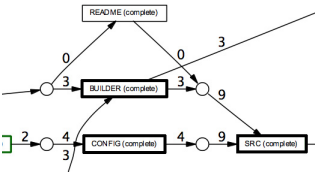


Fig. 5. Conformance Analysis

their paths and have ".java" as an extension were mapped to "SRC", all the "readme.*" files – to "README", all the files in "/tests/" – to "TESTS", the files in "/www/" – to "WWW", "build.bat" – to "BUILDER" and all the files, which names start with "." – to "CONFIG"; the other commits were ignored.

After executing the algorithms of the *generation and synthesis* approach, we obtained the Petri net shown in Fig. 3. Here, for the sake of readability, we show a simplified Petri net without loops – which was obtained by applying the "Kill Loops" modification strategy to the transition system and synthesizing a Petri net from there. Thus, the Petri net focuses on the start events, i.e. when source code development was started, when testing was started. People use to start with building web sites or editing readme files and builders, then they write code and then, they test it, sometimes builder file is changed after writing code.

The Petri net model of the development process can now be used for enhanced analysis within the ProM framework. Figure 4 shows the result of a *performance analysis* based on the mined model and the log. The states, i.e. places, have been colored according to the time which is spent in them while executing the process. Also, multiple arcs originating from the same place (i.e., choices) have been annotated with the respective probability of that choice.

Further, a *conformance analysis* can be performed using the Petri net model and the associated log. Figure 5 shows the *path coverage* analysis of the conformance checker. All activities that have been executed in a specific case (in our example we chose the C++ language support) are decorated with a bold border, and arcs are annotated with the frequency they have been followed in that case. This example shows, that the C++ team did not create a README file.

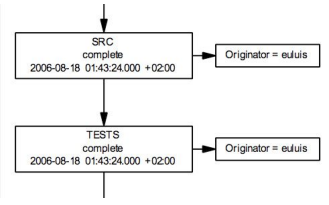


Fig. 6. LTL Analysis

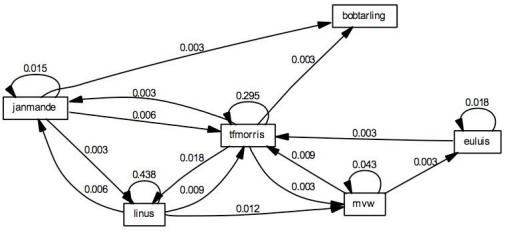


Fig. 7. Social Network

One known software engineering concept is the "four eyes principle", e.g. developers working on the source code should not write tests as well. Figure 6 shows the result of checking a corresponding *LTL* formula on the ArgoUML log. In the C++ support case, which is shown in Fig. 6, both source code and tests have been submitted by the developer "euluis", thereby violating this principle.

For determining the *social network* of a development process, it is preferable to use the original log, i.e. before it has been abstracted like explained in Section 4.1. The reason for that is, that it is also interesting when people collaborate within a certain part of the project (e.g. writing source code), while one wants to abstract from these activities on the control flow level. Figure 7 illustrates the hand-over of work between ArgoUML developers. It shows that some developers are involved only in specific phases of the project (e.g. "bobtarling" appears to work only at the end of projects), while others (e.g. "tfmorris") have a more central and connected position, meaning they perform tasks all over the process. Based on the nature of the project one may prefer different collaboration patterns, which can be checked conveniently in a mined social network like this.

Altogether, we have shown that ProM can be used to mine interesting information on a realistic software process. The filtering mechanism in combination with the kill-loop mechanism gave us a quite simple explicit process model.

6 Conclusion

In this paper, we have discussed some new algorithms for mining software and systems engineering processes from the information that is available in Software Configuration Management Systems. These algorithms are included in the ProM framework, which has interfaces to a variety of document management systems. Therefore, ProM is now an effective tool for software process mining.

For evaluation purposes, we have mined the software processes of a real project: ArgoUML. This shows that we can obtain the process models for realistic software projects. Moreover, we have shown that ProM could be used for analysing and verifying some properties of these processes.

Acknowledgments. This research is supported by EIT, NWO, the University of Paderborn and International Graduate School of Dynamic Intelligent Systems,

and the Technology Foundation STW, applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

References

1. van Dongen, B., Medeiros, A., Verbeek, H., Weijters, A., van der Aalst, W.: The ProM framework: A New Era in Process Mining Tool Support. In Ciardo, G., Darondeau, P., eds.: Application and Theory of Petri Nets 2005. Volume 3536. (2005) 444–454
2. MSR 2005 International Workshop on Mining Software Repositories. In: ICSE '05: Proceedings of the 27th international conference on Software engineering, New York, NY, USA, ACM Press (2005)
3. Sandusky, R.J., Gasser, L., Ripoche, G.: Bug Report Networks: Varieties, Strategies, and Impacts in a F/OSS Development Community. In: MSR 2004: International Workshop on Mining Software Repositories. (2004)
4. Iannacci, F.: Coordination Processes in Open Source Software Development: The Linux Case Study. http://opensource.mit.edu/papers/iannacci3.pdf (apr 2005)
5. Agrawal, R., Srikant, R.: Mining sequential patterns. In Yu, P.S., Chen, A.S.P., eds.: Eleventh International Conference on Data Engineering, Taipei, Taiwan, IEEE Computer Society Press (1995) 3–14
6. van der Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering **47**(2) (2003) 237–267
7. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Sixth International Conference on Extending Database Technology. (1998) 469–483
8. Cook, J., Wolf, A.: Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology **7**(3) (1998) 215–249
9. Cook, J., Du, Z., Liu, C., Wolf, A.: Discovering models of behavior for concurrent workflows. Computers in Industry **53**(3) (2004) 297–319
10. Kindler, E., Rubin, V., Schäfer, W.: Incremental Workflow mining based on Document Versioning Information. In Li, M., Boehm, B., Osterweil, L.J., eds.: Proc. of the Software Process Workshop 2005, Beijing, China. Volume 3840 of LNCS., Springer (May 2005) 287–301
11. Kindler, E., Rubin, V., Schäfer, W.: Activity mining for discovering software process models. In Biel, B., Book, M., Gruhn, V., eds.: Proc. of the Software Engineering 2006 Conference, Leipzig, Germany. Volume P-79 of LNI., Gesellschaft für Informatik (March 2006) 175–180
12. van der Aalst, W., Rubin, V., van Dongen, B., Kindler, E., Günther, C.: Process Mining: A Two-Step Approach using Transition Systems and Regions. BPM Center Report BPM-06-30, BPM Center, BPMcenter.org (Dec 2006)
13. Cortadella, J., Kishinevsky, M., Lavagno, L., Yakovlev, A.: Deriving Petri nets from finite transition systems. IEEE Transactions on Computers **47**(8) (1998) 859–882
14. Kindler, E., Rubin, V., Schäfer, W.: Process Mining and Petri Net Synthesis. In Eder, J., Dustdar, S., eds.: Business Process Management Workshops. Volume 4103 of LNCS., Springer (2006)

15. Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. IEICE Transactions on Information and Systems **E80-D**(3) (1997) 315–325
16. van der Aalst, W., Weijters, A., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering **16**(9) (2004) 1128–1142
17. van Dongen, B., van der Aalst, W.: Multi-Phase Process Mining: Building Instance Graphs. In Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T., eds.: International Conference on Conceptual Modeling (ER 2004). Volume 3288. (2004) 362–376
18. Weijters, A., van der Aalst, W.: Rediscovering Workflow Models from Event-Based Data using Little Thumb. Integrated Computer-Aided Engineering **10**(2) (2003) 151–162
19. van der Aalst, W., Medeiros, A., Weijters, A.: Genetic Process Mining. In Ciardo, G., Darondeau, P., eds.: Applications and Theory of Petri Nets 2005. Volume 3536. (2005) 48–69
20. van der Aalst, W., Reijers, H., Song, M.: Discovering Social Networks from Event Logs. Computer Supported Cooperative work **14**(6) (2005) 549–593
21. Günther, C., van der Aalst, W.: Mining Activity Clusters from Low-level Event Logs. BETA Working Paper Series, WP 165, Eindhoven University of Technology, Eindhoven (2006)
22. Rozinat, A., van der Aalst, W.: Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models. In Bussler et al., C., ed.: BPM 2005 Workshops (Workshop on Business Process Intelligence). Volume 3812. (2006) 163–176
23. van der Aalst, W., Beer, H., Dongen, B.: Process Mining and Verification of Properties: An Approach based on Temporal Logic. BETA Working Paper Series, WP 136, Eindhoven University of Technology, Eindhoven (2005)

Focused Identification of Process Model Changes

Martín Soto and Jürgen Münch

Fraunhofer Institute for Experimental Software Engineering
Fraunhofer-Platz 1
67655 Kaiserslautern, Germany
{soto,muench}@iese.fraunhofer.de

Abstract. Advanced software process management requires capabilities to systematically analyze differences between versions of a process model. These capabilities can be used, for instance, to support process compliance management, to learn from process evolution, or to identify and understand process variations in different development environments in order to develop generic process models such as process standards. Analyzing the differences between process models versions is a highly challenging task that needs to be based on appropriate methods and tools. Experience has shown that, beside global version comparisons, local and focused difference analyses are often needed. Example goals of such focused analyses are the identification of all process changes that are relevant for a specific role, or the identification of those process changes that are relevant for a process reassessment. This article presents a technique based on pattern-matching for such focused analysis. The technique is a component of the comprehensive *DeltaProcess* approach for difference analysis [1, 2]. We explain the underlying concepts of the technique, describe a supporting tool, and discuss our initial validation in the context of the German V-Modell XT process standard. We close the paper with related work and directions for future research.

Keywords: process modeling, process model change, process model evolution, model comparison.

1 Introduction

Process engineers working on the evolution and maintenance of process models often have the need to compare different versions of these:

- When a new release of a process standard is published, users of the standard (e.g., organizations basing their own processes on the standard) need to know what changed, in order to adapt their own models.
- When a company-wide process model is modified, employees working according to that process need to determine if they are affected by the changes.
- When a model that involves safety or mission-critical aspects is updated, reviewers and auditors can work better and more efficiently if they know exactly what was modified.
- When simultaneous variants of a model are maintained (e.g., by several projects in an organization, or by several companies adopting and tailoring a single process

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 182–194, 2007.

framework), it is useful to be able to synchronize changes between variants, which requires finding out exactly what was changed in each variant.
– When company-specific standards are originally defined, or when they evolve, it is necessary to understand the process variability, both over time as well as across the organization's project space.

One frequently asked question is whether determining where changes occurred and why is just a matter of proper change control: it could be argued that logging every change made is all that is needed. In practice, however, this is not as easy as it may appear. Maintaining change logs is tedious and difficult, and documenting the changes is often seen as unnecessary overhead to more important tasks, like actually working on improving the process model. For these reason, change logs are often missing or poorly maintained and, thus, unreliable as a source of information.

There are many other situations in which mechanisms for reliably determining the differences between process model versions can be useful. These mechanisms must be focused, and deliver results that different process stakeholders (process engineers, project managers, developers, etc.) can use for different purposes. Research-wise, process difference analysis is faced with many challenges, such as 1) defining appropriate notations for specifying difference analysis goals, 2) creating suitable comparison algorithms, and 3) designing purpose-and stakeholder-oriented presentations of the results. The work we are presenting here is part of the *DeltaProcess* approach to software process model difference and evolution analysis [1, 2]. This paper concentrates on one single aspect of the approach, namely, given a particular user's information needs, how to identify precisely those changes that provide the user with the needed information. The technique presented here for this purpose comprises three main phases: In the first phase, the models are converted into a representation that makes it easier to compare them, since it regards all types of changes uniformly. In the second phase, a so-called comparison model is produced, which comprises the contents of both compared versions. In the third phase, focused difference analysis goals are specified, and a pattern-matching system is used to look for corresponding instances in the comparison model, and produce analysis results.

The rest of the paper is organized as follows: Section 2 describes the process model comparison problem using an example to illustrate the difficulties involved. Section 3 presents our change identification technique in detail, and includes some examples of its application to common, practical situations. In Section 4, we briefly discuss our implementation of this technique, as well as our experience applying it to the German V-Modell XT [3] process standard. The paper closes with Section 5, which compares our approach to related work, and Section 6, which presents our final conclusions and outlook.

2 The Process Model Comparison Problem

To illustrate the difficulties involved in process change identification, Fig. 1 shows two revisions of a process model excerpt, which we kept deliberately small for the purposes of the example. If someone were commissioned with the task of finding *all* differences manually (i.e., by looking at the diagrams) it would probably take this person some time to find all of them, and to make sure that none is missing, however

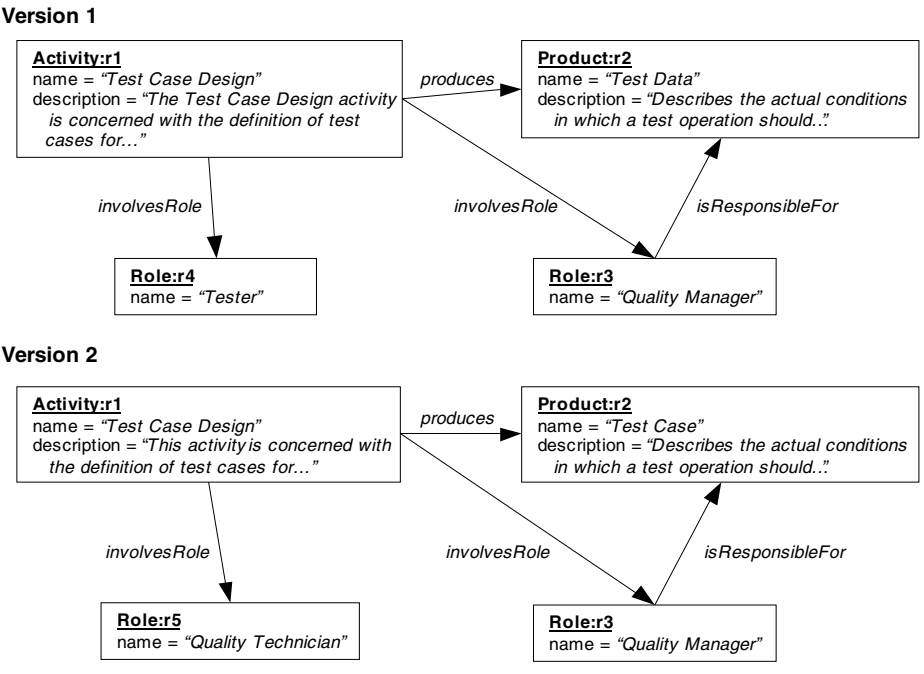


Fig. 1. Two versions of a process model (UML object diagram notation)

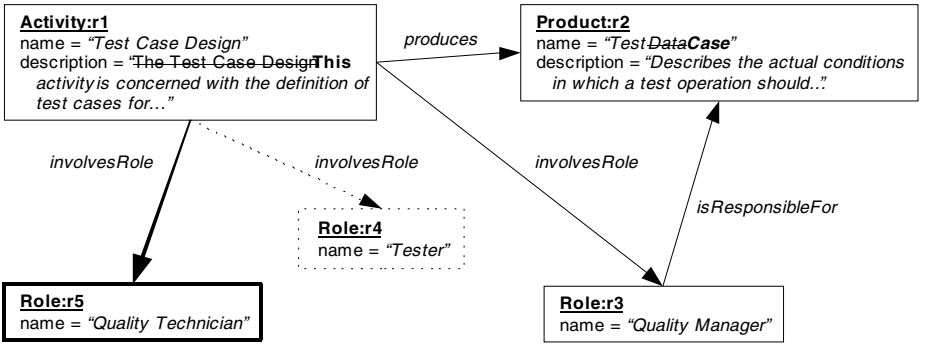


Fig. 2. A comparison of the process model revisions shown in Fig. 1

small it may be. Considering that real world process models are significantly larger than our example, it is clear that automated support is necessary to guarantee consistent and reliable comparison results.

Fig. 2 is an attempt to display all changes in the context of the changed model. Entities and relations erased from version 1 are marked using interrupted lines, whereas entities and relations new in version 2 are marked with thicker lines. Changes in entity attributes are shown by crossing deleted text off, as well as displaying new text in bold type. The following are some observations about this model comparison:

- *Changes are heterogeneous.* A number of basic change types can be directly identified from the example. They include entity additions and deletions, relation additions and deletions, and changes in the values of the various attributes.
- *Attribute changes must be interpreted according to model semantics.* Not all changes belonging to one of the basic types listed above are the same for the user. For example, if an attribute containing an integer value changes, it is probably enough to provide the user with the old and new values. On the other hand, if an attribute contains text changes, it is potentially important to determine which words were modified.
- *Relation changes must be interpreted according to model semantics.* For example, if a parent-child relationship changes, it is important to tell the user that a certain object has a new parent. If a consumes-produces relationship changes, it is important to report that some activities now produce new products, or that some products are now consumed by new activities. How this is reported may even depend on the role of the user with respect to the process. Oftentimes complete sets of simple structural changes must be grouped together and presented to the user as a unit for proper interpretation.
- *A graphical display is not enough.* Although Fig. 2 does a fairly good job of making changes obvious, the same type of display would not work if applied to a model containing hundreds or even thousands of entities. Even if the technical difficulties of producing such a large graph were overcome, finding all changes relevant to a particular task would still be difficult because of the sheer size and complexity involved.

3 Pattern-Matching Based Change Identification

In the following, we discuss our technique for model change identification. This technique makes it possible to handle a wide variety of types of changes in a completely uniform way, to flexibly define the types of changes that are considered interesting or useful (this can be based on the structure and semantics of the metamodel), and to restrict the results to only certain types of changes, or even to certain interesting portions of a model.

3.1 A Normalized Representation for Process Models and Their Comparisons

Our first step consists of representing models (and later their differences) in such a way that a wide range of change types can be described using the same basic formalism. The representation we have chosen is based on that used by RDF [4] and similar description or metadata notations. For our purposes, this notation has a number of advantages over other generic notations:

- Being a generic notation for graph-like structures, it is a natural representation for a wide variety of process model schemata.
- It has a solid, standardized formal foundation.
- As shown below, the uniformity of the notation, which does not differentiate between relations and attributes, makes it possible to describe a wide range of changes with a straightforward pattern-matching notation.

– Also as shown below, the fact that many model versions can be easily put together into a single model makes it possible to use the same pattern-matching notation for single model versions and for comparisons.

Fig. 3 shows the first revision from Fig. 1 converted to this representation. The graph contains only two types of nodes, which we will call *entity* nodes (ovals in the figure) and *value* nodes (boxes in the figure). Entity nodes have arbitrary identifiers as labels. Value nodes are labeled by the value they represent, which can belong to a basic type (string, integer, boolean, etc.)

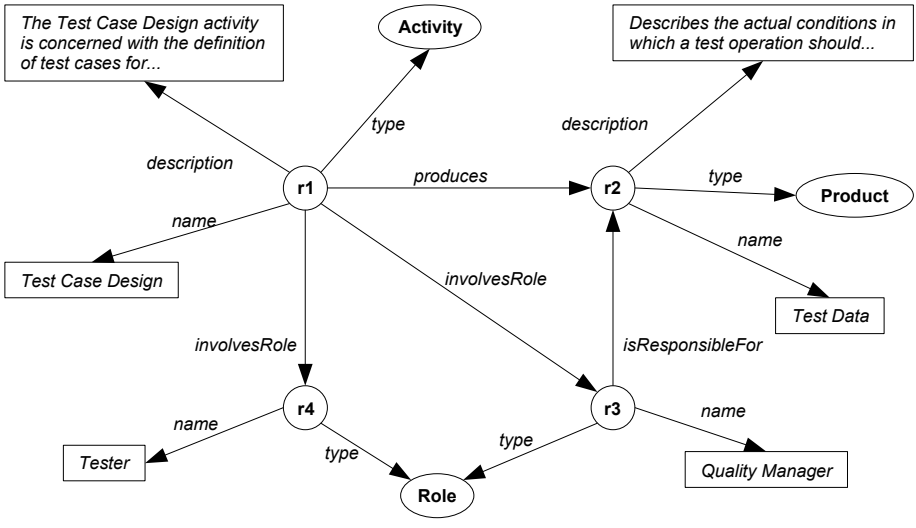


Fig. 3. A process model in normalized form

Arrows represent typed directed relations (type is given by their labels). Relations may connect two entity nodes, or may go from an entity node to a value node. It is not allowed for a relation to leave a value node. It is also not allowed for a node to exist in isolation. All nodes must be either the start or the end point of *at least* one relation. It follows that the graph is characterized completely by the set of the relations (edges) present in it, since the set of nodes is exactly the set of all nodes that are the start or the end of an edge.

The correspondence between attributed graphs (like those in Fig. 1) to this normalized form is straightforward:

– *Entities and types correspond to entity nodes.* For each entity instance and entity type in the original graph, there is an entity node in the normalized graph. There is also a *type* relation between each node representing an entity and the node representing its type.
– *Attributes correspond to entity-value relations.* For each entity attribute in the original graph, there is a relation labeled with the attribute name that connects the entity with the attribute value (that is, attributes in the original metamodel are converted into relation types). The value is a separate (value) node.

– *Entity relations correspond to entity-entity relations.* For each relation connecting two entities in the original graph, a relation connecting their corresponding entity nodes is present in the normalized graph.[1]

Fig. 4 shows the same comparison presented in Fig. 2, using the normalized notation, with changes also highlighted using interrupted and bolder lines. Formally, this graph respects exactly the same restrictions as the normalized model representation. The only addition is that edges are *decorated* to state the fact that they were deleted, added, or simply not touched. This leads us to the concept of a *comparison graph* or *comparison model.* The comparison model of two normalized models A and B contains all edges present in either A or B, or in both, and only those edges. Edges are marked (decorated) to indicate that the edge is only in A, only in B, or in both of A and B.

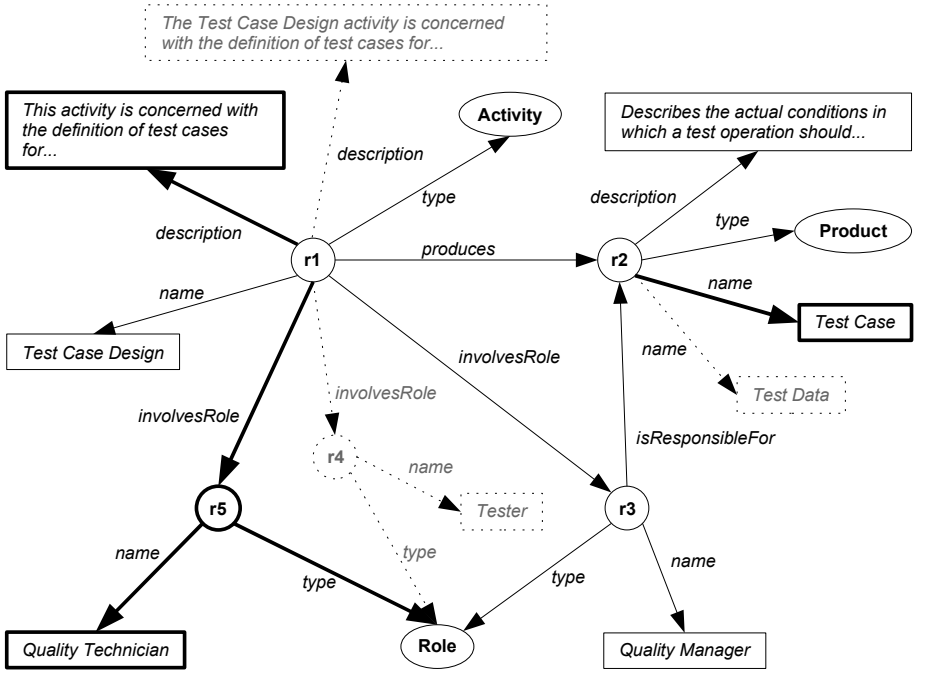


Fig. 4. A process model comparison in normalized form

The main aspect to emphasize here is the fact that all changes are actually reduced to additions and deletions of relations between nodes. This results in part from the fact that attributes are represented as relations, but also from the fact that nodes cannot exist in isolation. It is possible (and safe) to identify entity additions and deletions by looking for additions and deletions of *type* relations in the model. Also, to be fair, the comparison in Fig. 4 ignores one important aspect of Fig. 2, namely, the word level text comparison. We will deal with this limitation later in the paper (see Section 3.4).

[1] Relations with attributes can be modeled by introducing entity nodes that represent them, but the details are beyond the scope of this paper.

It is also important to point out, that the comparison model is useful only when entities have unique identifiers that remain stable after the model is changed. Although, in theory, this could be considered a strong limitation, it is seldom a problem in practice, since most practical modeling tools indeed provide unique, stable entity identifiers.

The fact that the normalized representation reduces all changes to sets of relation additions and deletions permits to describe many types of changes uniformly. The following sections discuss the mechanism that we have chosen to describe such changes in a clear and unambiguous way: a graphical pattern-matching language.

3.2 Graphical Comparison Patterns

In order to identify changes of a particular type, a so-called *graphical pattern*[2] must be defined, that matches precisely these changes. Graphical patterns are essentially comparison graphs in which some node and/or edge labels have been potentially replaced by variables. Informally, matching the pattern to a comparison graph implies replacing the variables in the pattern with actual labels from the graph, in such a way that the resulting pattern is a subset of the comparison graph. A value assignment for the variables in a pattern that results in a match is called an *occurrence* of the pattern. Normally, we are interested in the set of all occurrences of a pattern in a particular comparison graph. The following sections present examples of how to use this pattern language to identify interesting changes in process models.

3.3 Example 1: Additions and Deletions

Our first example is related to one of the simplest possible model changes: adding or deleting process entities. Fig. 5 shows four patterns that identify changes of this type with different levels of generality. The pattern in Fig. 5a) matches all additions of process activities, and for each match, sets the *?id* variable with the identifier of the new activity. In a similar way, the pattern in Fig. 5b) matches all deletions of process products. These patterns can be generalized to identify arbitrary additions and deletions:

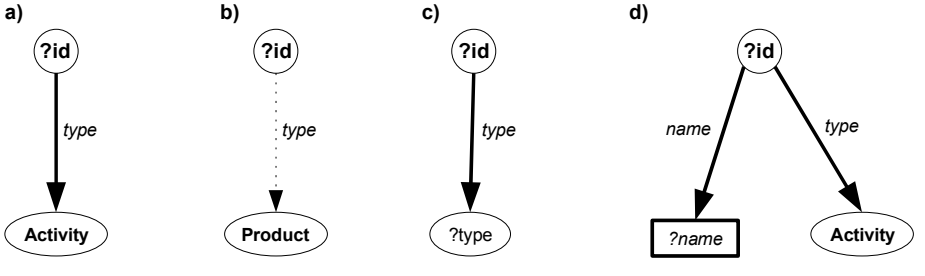


Fig. 5. Patterns for identifying entity additions and deletions

[2] Notice that our use of the word *pattern* is rooted in the standard pattern matching tradition, as related to problems like Prolog-style unification, term rewriting and regular expression search. It is not intended to relate to other uses of the word in computer science, like, for example, software design patterns.

the pattern in Fig. 5c) identifies all entity additions, and instances an additional variable with the type of the added entity. Finally, Fig. 5d) shows a pattern that not only finds new activities, but sets a variable with the corresponding name, a useful feature if the results of matching the pattern are used, for example, to produce a report.

3.4 Example 2: Changes in Attribute Values

Just as important as identifying entity additions and deletions, is finding entities whose attributes were changed. Fig. 6 shows three patterns that describe changes in attribute values. Fig. 6a) is basically an excerpt from the comparison graph in Fig. 4, which captures the fact that an attribute *description* was changed. This pattern, however, matches only the particular change shown in the example. The pattern in Fig. 6b) is a generalized version of the first one. By using variables for the entity identifier, as well as for the old and new property values, this pattern matches all cases where the *description* attribute of an arbitrary entity was changed. Notice that each match sets the value of the *?id* variable to the identifier of the affected entity, and the values of *?oldValue* and *?newValue* to the corresponding old and new values of the *description* property. The pattern in Fig. 6c) goes one step further and uses a variable for the attribute labels as well, which means it matches all attribute value changes. Notice that these patterns match once for *each* changed property in *each*

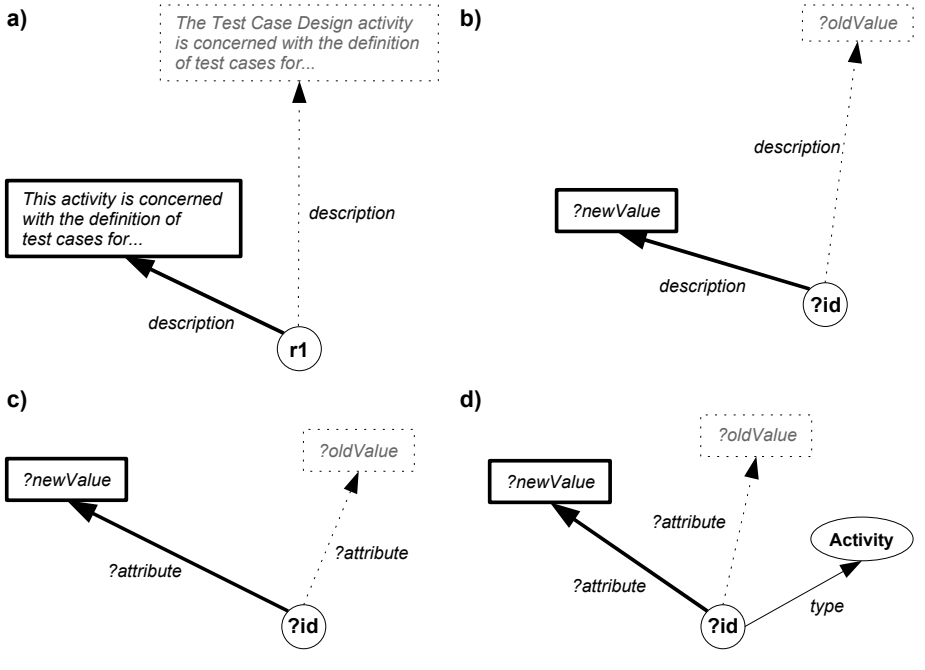


Fig. 6. Four patterns for identifying attribute value changes

object. Finally, the pattern in Fig. 6d) constitutes a specialization of its peer in Fig. 6c): it is restricted to all attribute value changes affecting process activities.

Changes identified in this way, can be fed to additional algorithms that perform attribute specific comparisons, like, for example, identifying added or deleted individual words or characters in text based attributes. These way, potentially expensive specific comparison algorithms are only applied to relevant items.

3.5 Example 3: Impact of Changes on Tool Usage

Our last example stems from a question posed to the authors by a process engineer: "How can I determine the impact of process changes on the software tool infrastructure?" The goal of this process engineer was to find out if new software development tools (or tool licenses) had to be purchased, and, if so, which people had to receive them after the new process changes were implemented.

Fig. 7 shows two patterns that could help answer this question. These patterns introduce a new language element: an edge marked with a black point (like the edge from *?toolId* to *?toolName* in Fig. 7a) matches edges in the comparison graph without regard to decoration. This means that old, new, and common edges could be matched by such an edge, as long as the labels of the end nodes and the relation label match.

By using this feature, the pattern in Fig. 7a) is able to match tools having a new *requiresTool* relation to a tool. This happens regardless of whether the tool itself is new or existed before, but was not required by the activity. The pattern in Fig. 7b) is a variation of the previous one, which matches new activities and connects them to the tools they require. These and similar patterns can be used to determine which users will eventually require new software tool licenses, in order to buy and install them timely.

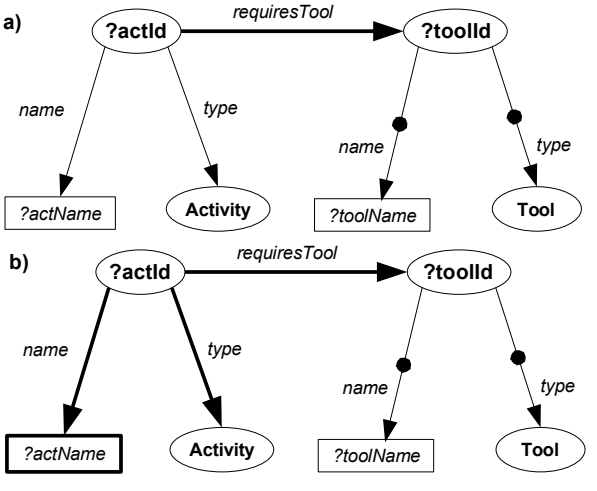


Fig. 7. Identifying the impact of activity changes on tool requirements

4 Implementation and Validation

An implementation of the pattern-matching based change identification technique presented in the previous sections is available as part of the *Evolyzer* tool (see Fig. 8), which is intended to support the *DeltaProcess* process model difference analysis approach mentioned in the introduction. We have tested our approach and tools by applying them to the various official releases of the V-Modell XT [3], a large prescriptive process model intended originally for use in German public institutions, but finding ever increasing acceptance from the German private sector. As of this writing, the *Evolyzer* tool still lacks a graphical editor for change patterns. However, patterns can be expressed as textual queries using a syntax that basically follows that of the emerging SPARQL [5] query language for RDF. Expressed as queries, patterns can be executed to find all their occurrences in a model.

The V-Modell XT constitutes an excellent testbed for our approach and implementation. Converted to the normalized representation defined in Section 3.1, the latest public version at the time of this writing (1.2) produces a graph with over 13,000 edges. This makes it a non-trivial case, where tool support is actually necessary to perform an effective analysis of the differences between versions. Our first trial with

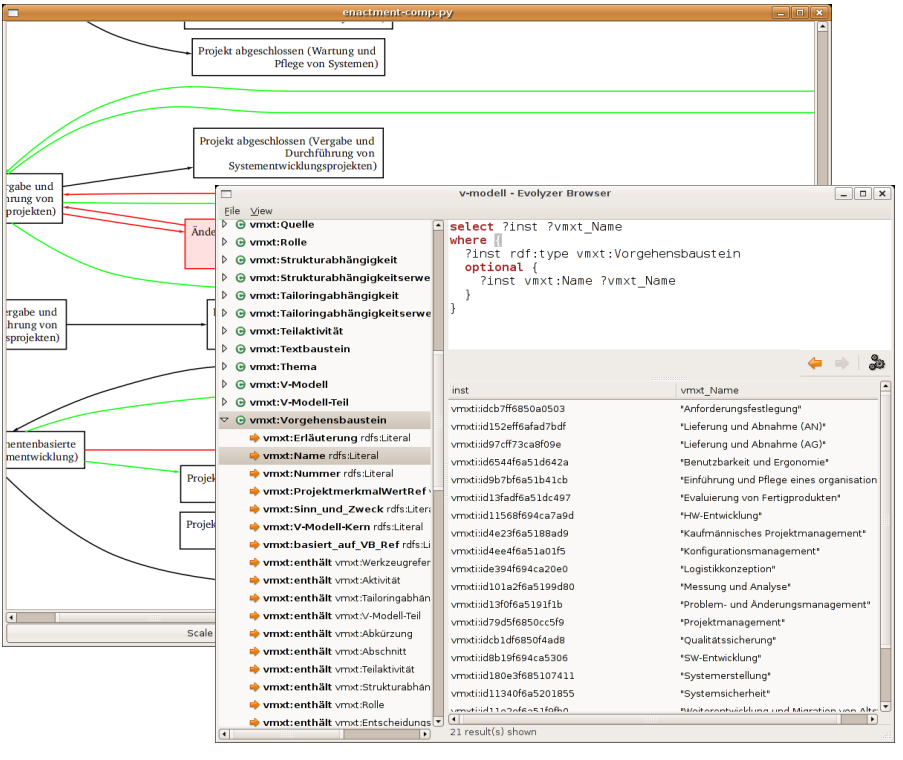


Fig. 8. The *Evolyzer* tool working on the V-Modell XT

the V-Modell XT consisted of analyzing the evolution of the "official" V-Modell XT itself: we compared the model's available public releases 1.0 (11,555 edges in our representation), 1.1 (11,822 edges) and 1.2 (13,286 edges). A comparison of versions 1.1 and 1.2, for example, yields 10,628 common edges, which indicates a common core of almost 80% of the latest version.

Using various patterns, we were able to classify the changes into groups, including added or removed entities, entities relocated inside the structure, renamed entities, corrected entity descriptions, etc. This analysis was motivated by the concrete needs of a company that has deployed a customized version of the VModell XT for use in all internal software development projects (Example 3 in Section 3.5 is also based on this work). One of the main purposes of the analysis is to use process model difference analysis to keep the company's customized model synchronized with the standard VModell XT.

Even for large cases like those just described, we consider the performance of our prototype implementation to be satisfactory. Running on a standard desktop PC, the system is able to convert an instance of the V-Modell XT into the normalized form in less than 10 seconds. Building the comparison model takes less than five seconds, and matching patterns is usually faster (of course, this depends on the complexity of the pattern). Most interesting differences can be analyzed interactively, which makes the system even more useful. Additionally, the Evolyzer framework makes it possible to feed changes identified by a pattern to further analysis algorithms. Currently is it possible to see changes in context using a graph layout system, and to produce text reports of certain types of differences.

5 Related Work

Several other research efforts are concerned in one way or another with comparing model variants syntactically and providing an adequate representation for the resulting differences.

[6] and [7] deal with the comparison of UML models representing diverse aspects of software systems. These works are oriented towards supporting software development in the context of the Model Driven Architecture. Although their basic comparison algorithms are applicable to our work, they are not concerned with providing analysis or visualization for specific users.

[8] presents an extensive survey of approaches for software merging, many of which involve comparison of program versions. The surveyed works mainly concentrate on automatically merging program variants without introducing inconsistencies, but not, as in our case, on identifying differences for user analysis.

[9] provides an ontology and a set of basic formal definitions related to the comparison of RDF graphs. [10] and [11] describe two systems currently in development that allow for efficiently storing a potentially large number of versions of an RDF model by using a compact representation of the raw changes between them. These works concentrate on space-efficient storage and transmission of change sets, but do not go into depth regarding how to use them to support higher-level tasks (like process improvement).

Finally, an extensive base of theoretical work is available from generic graph comparison research (see [12]), an area that is concerned with finding isomorphisms (or correspondences that approach isomorphisms according to some metric) between arbitrary graphs whose nodes and edges cannot be directly matched by name. This problem is analogous in many ways to the problem that interests us, but applies to a separate range of practical situations. In our case, we analyze the differences (and, of course, the similarities) between graphs whose nodes can be reliably matched in a computationally inexpensive way.

6 Conclusions and Future Work

Due to factors like model size and metamodel differences, the general problem of identifying and characterizing changes in process models is not trivial. By expressing models in a normalized representation, we are able to characterize interesting changes using a graphical pattern matching language. Graphical patterns provide a well-defined, unambiguous and, arguably, intuitive way to characterize common process model changes, as our examples show.

Our implementation of pattern queries in the *Evolyzer* system demonstrates that our pattern-based change identification technique can be used in practical situations involving very large process models like the V-Modell XT. It is important to stress, however, that the technique requires the process entities in compared models to have stable identifiers that are used consistently across versions. Thanks to the fact that common modeling tools support stable identifiers, this is often the case when comparing versions of the same model, but not when comparing models that were created independently from each other.

Acknowledgments. We would like to thank Sonnhild Namingha from Fraunhofer IESE for proofreading this paper. This work was supported in part by the German Federal Ministry of Education and Research (V-Bench Project, No. 01| SE 11 A).

References

1. Soto, M., Münch, J.: Process Model Difference Analysis for Supporting Process Evolution. In: Proceedings of the 13th European Conference in Software Process Improvement, EuroSPI 2006. Springer LNCS 4257 (2006)
2. Soto, M., Münch, J.: The *DeltaProcess* Approach for Analyzing Process Differences and Evolution. Internal report No. 164.06/E, Fraunhofer Institute for Experimental Software Engineering (IESE) Kaiserslautern, Germany (2006)
3. V-Modell XT. Available from http://www.v-modell.iabg.de/ (last checked 2006-03-31).
4. Manola, F., Miller, E. (eds.): RDF Primer. W3C Recommendation, available from http://www.w3.org/TR/rdf-primer/ (2004) (last checked 2006-03-22)
5. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL Query Language for RDF. W3C Working Draft, available from http://www.w3.org/TR/rdf-sparql-query/ (2006) (last checked 2006-10-22)
6. Alanen, M., Porres, I.: Difference and Union of Models. In: Proceedings of the UML Conference, LNCS 2863 Produktlinien. Springer-Verlag (2003) 2-17

7. Lin, Y., Zhang, J., Gray, J.: Model Comparison: A Key Challenge for Transformation Testing and Version Control in Model Driven Software Development. In: OOPSLA Workshop on Best Practices for Model-Driven Software Development, Vancouver (2004)
8. Mens, T.: A State-of-the-Art Survey on Software Merging. IEEE Transactions on Software Engineering, Vol. 28, No. 5, (2002)
9. Berners-Lee, T., Connolly D.: Delta: An Ontology for the Distribution of Differences Between RDF Graphs. MIT Computer Science and Artificial Intelligence Laboratory (CSAIL). Online publication http://www.w3.org/DesignIssues/Diff (last checked 2006-03-30)
10. Völkel, M., Enguix, C. F., Ryszard-Kruk, S., Zhdanova, A. V., Stevens, R., Sure, Y.: SemVersion - Versioning RDF and Ontologies. Technical Report, University of Karlsruhe. (2005)
11. Kiryakov, A., Ognyanov, D.: Tracking Changes in RDF(S) Repositories. In: Proceedings of the Workshop on Knowledge Transformation for the Semantic Web, KTSW 2002. Lyon, France. (2002)
12. Kobler, J., Schöning, U., Toran, J.: The Graph Isomorphism Problem: Its Structural Complexity. Birkhäuser (1993)

An Approach for Decentralized Process Modeling

Oktay Turetken[1] and Onur Demirors[2]

[1] Bilgi Grubu Ltd. ODTU Teknokent, 06531, Ankara, Turkey
{oktay@bg.com.tr}
[2] Informatics Institute, Middle East Technical University, 06531, Ankara, Turkey
{demirors@metu.edu.tr}

Abstract. This paper describes a method for organizations to perform process modeling in a decentralized and concurrent manner. The approach is based on the idea that modeling organizations' processes can be performed by individuals actually performing the processes. Instead of having a central and devoted group of people to analyze, understand, model and improve processes, real performers are held responsible to model and improve their own processes concurrently. The paper also summarizes the lessons learned by application of the method in two organizations.

Keywords: Decentralized process modeling, software process modeling, role-based process modeling, process improvement.

1 Introduction

Many people believe that process models are one of the most valuable assets of organizations. The value of this asset will increase if they are embraced by the performers, accurately reflect the executed processes and can be easily updated to reflect the required changes. A decentralized approach for process modeling can be used to achieve these multiple goals at once.

In a decentralized approach each agent in the organization models her process. The totality of the process definitions forms the organization's process-base. As the degree of the involvement of the knowledge workers who perform the processes increases, the likelihood of the model to reflect the executed process as well as the likelihood of the performers to embrace the models increases.

In addition, in a centralized approach it is generally difficult and not desired to change processes frequently once their definitions are considered stable [13]. However, software organizations should be much faster in capturing and incorporating any improvement opportunity raised in the business into the organization process-base. This goal could be achieved by a concurrent and decentralized modeling approach.

It can be argued that such an approach can not be easily implemented by large organizations which perform daily routine tasks. This might be true, yet the literature indicates that such an approach will be usable for organizations where mostly knowledge workers are integrated and collaborate for production. [5], [23]. Software engineering organizations are excellent examples.

In this study we defined a method called Plural to provide a disciplined guidance for organizations to perform modeling in a decentralized and concurrent way.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 195–207, 2007.

The responsibility of understanding, modeling and improving the processes is delegated to individuals that actually perform the processes. Each individual in the organization models the activities that s/he performs and the results are integrated to form the complete process models at different abstraction levels. Based on the resulting processes, diagrams such as the ones depicting process and role dependencies can be generated in order to provide insight into the way the organization works and can be improved.

This paper provides an overview of the Plural method and summarizes the results of its application in two exploratory cases. In Section 2, we briefly discuss the related research. In Section 3, we provide an overview of the method. Findings and lessons learned from case studies are discussed in Section 4.

2 Related Work

Process modeling and enactment approaches usually assume central specification and execution of processes. The implicit assumption on such a central structure is also present in many process redesign/improvement approaches [3], [28]. On the other hand, various studies on process modeling ([20], [14], [1]) acknowledge the importance of the involvement of the performers and even urge the empowerment of actors in order to take the responsibility to model and change their processes.

The idea of agents modeling their activities in a decentralized manner is proposed by Demirors as the "Horizontal Change Approach (HOC-A)" [4] to manage change in software development organizations. In HOC-A, process modeling and change are performed in a decentralized manner concurrently by all the members of the organization. In this sense, it is analogous to neural networks in which the overall goal is achieved collectively without direction at any specific organizational level.

For decentralized modeling, Ben-Shaul and Kaiser [2] adopt the view of 'international alliance' whereby independent countries sign 'treaties' that determine their collaboration but retain full control over their local laws. They developed the Oz environment as an enhancement to the Marvel system [12]. In general, participants of the 'treaty' explicitly specify in what ways they are willing to participate in a multi-site operation. Leonhardt et al. [15] proposes a framework for distributed and concurrent software engineering as an alternative to traditional centralized software development. They considered the use of decentralized process models to drive consistency checking and conflict resolution. Individual process models are represented locally by associating development participants with them. The process models use pattern matching on individual's development histories to determine the particular state of the development process, and utilize rules to trigger situation dependent assistance to the user. Graw et al. [8] proposes architecture for distributed process modeling and enactment based on FUNSOFT nets [9].

Role-based process modeling approaches considers an organization as a network of roles, communicating through interactions. The Riva method proposed by Ould [19] suggests a number of steps to be followed for process modeling using Role Activity Diagrams (RADs). The approach focuses on the interaction between roles which ease the modeling of concurrent engineering processes. The Interconnected Roles (IR)

framework [25] is a process description formalism based on roles, teams and processes for specifying and building distributed n-party synchronous interactions in organizations. Object-Oriented Role Analysis Method (OORAM) is an object-oriented methodology based model which uses roles to describe patterns of interacting objects [21]. A role has attributes and it may receive/send messages or invocations from/to other roles. The proposed methods and notations require a considerable degree of refinement and extension to be used for process modeling where role behaviors are created and maintained individually and independently.

Acquiring portions of the processes and merging them to form the complete model is also central in 'ViewPoints' [6], [17] or view-based approaches employed in requirements engineering and process elicitation research areas [22]. 'Each viewpoint encapsulates partial knowledge about a system and its domain - expressed in a suitable representation scheme - together with partial knowledge about the overall process of development' [17]. The framework is based on the idea that, the construction of a complex description or model involves many agents who have different perspectives or views of the artifact or system they are trying to describe [7]. However, views are outdated once they are merged into a complete model and are generally not maintained as separate entities from then on. Any change in the process is represented in the integrated model. Verlage [27] is one of the first that introduced a formalization of core requirements of view based process elicitation. Turgeon and Madhavji [26] have proposed a prototype tool called V-Elicit that helps to elicit process models from multiple sources or views. Works by Verlage and Turgeon & Madhavji are in parallel in many points. However, these works still lack support for a process modeling approach where models are developed by real performers.

In the last decade, agent-based approaches in artificial intelligence and business process management fields have received great interest for engineering complex distributed systems. Increasingly, many computer systems are being viewed in terms of autonomous agents. Proposed solutions have already been developed for many different domains and software engineering [11] and business process management (BPM) [10] are no exceptions. In agent-based approaches an *agent* is an autonomous, problem solving computer program that interacts with other agents when it has interdependencies [11]. The focus of these approaches is on the enactment of the agents' service definitions, which are presumed to be defined already. Thus, to meet the requirements posed by decentralization in modeling, these approaches need to be extended with a mechanism, a graphical notation and a tool for human agents to define the set of services they provide and their dependencies to others.

3 The Plural Method

Plural is a method to define and improve an origination's processes and to maintain its process-base. The organization goes through three phases of the process in order to establish its process-base and necessary infrastructure. Fig. 1 presents these three phases and the way each can proceed.

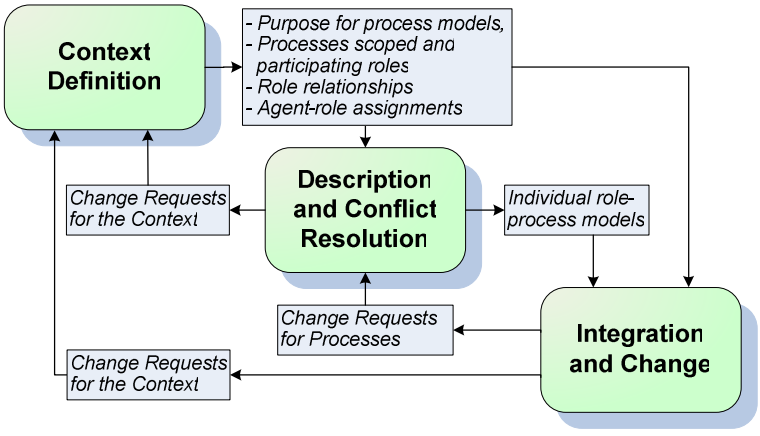


Fig. 1. The Phases of the Plural Process

In *context definition* phase, all process owners collectively define the aim and scope of the modeling process. Based on the roles each agent plays in the organization, agents start defining the activities they perform in the processes in the *description and conflict resolution* phase. They identify and resolve inconsistencies and conflicts between their definitions and others. This role based definition is considered complete after they are validated by associated peer agents and verified by the coordination team. In *integration and change* phase, complete and consistent models are merged; new models are generated and analyzed. Change requests are proposed. Based on its type, a request triggers the first or second phase and the cycle repeats itself for number of times for any request until the change is incorporated and the processes reach to their next consistent and complete state.

3.1 Context Definition

This phase sets up the organization for concurrent and decentralized process modeling. The primary goal is to achieve a structural frame of the organization in terms of a high level process network, participating roles and agents and their structural relationships. First, the process group, consisting the process owners and relevant stakeholders, determines and states the aim and objectives for modeling processes. The group, then, establishes the coordination team, which performs number of activities including monitoring and facilitating the definition process, guiding agents in modeling and maintaining the process network, verifying individual role process models, integrating and generating models for process analysis.

The group determines the processes that will be modeled and the roles that participate in those processes. The coordinators depict the coverage on a 'scope diagram' which defines the span of the entire study. Fig. 2 provides an example diagram epicting a number of processes and participating roles.

In identifying and associating the roles with processes, the inherent static relationship between these roles are revealed. For example, *configuration manager* role has (inherits) all of the responsibilities of the *project team member*, which is a more general role. The coordinators depict these relationships on a role diagram.

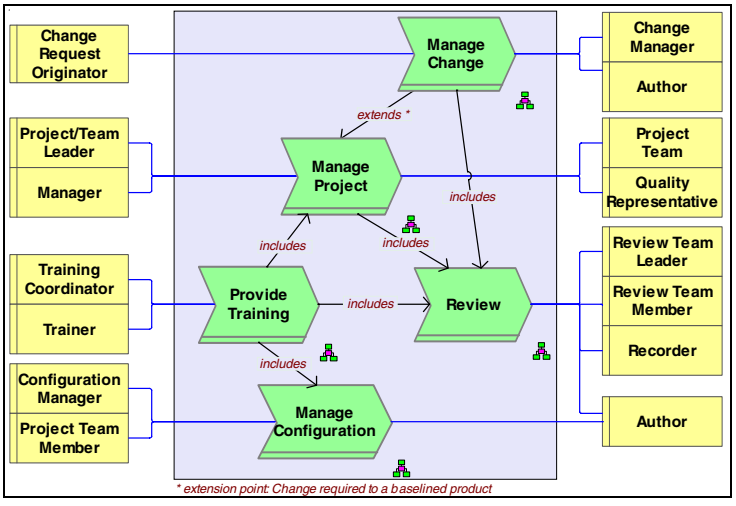


Fig. 2. A scope diagram. Include relationship represents a reusable process that is unconditionally incorporated into the execution of the other.

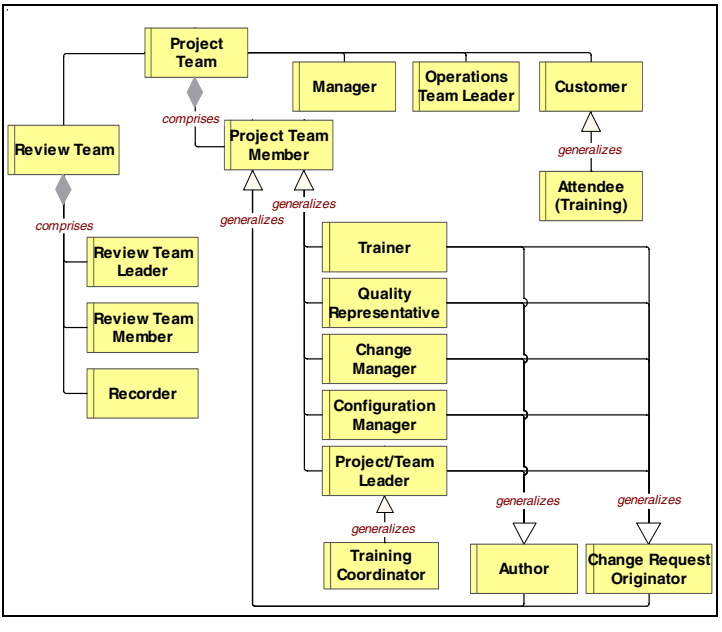


Fig. 3. Role Diagram. Similar to the class concept in Unified Modeling Language (UML) [18], relationships were in association, aggregation (composition) or generalization type.

Fig. 3 gives an example role diagram for the case given in the sample scope diagram (Fig. 2). The roles that were external to the organization (e.g. customer) or out of the scope with respect to the processes covered are called *inactive* roles. *Active*

roles, on the other hand, are the ones whose activities are modeled by an associated agent. Each active role had its own individual role-process models describing the activities it performs in the context of a specific process.

Each agent takes over the (active) roles with respect to their actual responsibilities in the organization. In addition for the development agents to be responsible from process description, peer agents that validate these definitions are also associated with roles. Agents can be assigned to multiple roles and roles can be taken over by multiple agents. The coordination team ensures that no active role is left unassigned. With respect to the scope and related assignments, the execution plan is documented and approved by all participants. The plan and the diagrams provide a framework for all participants about the responsibilities and the scope of the individual process modeling activity to be performed in the next phase.

3.2 Description and Conflict Resolution

Once the execution plan is approved, based on the schedule, the process description and conflict resolution phase may commence. The primary goal is to come up with a complete and consistent set of individual role-process models. Modeling at this stage carries role-based modeling to a further step where the real players of those roles model their processes concurrently. If deemed necessary, the coordination team performs orientation sessions to agents for the procedure to be followed, the notation, the tool and the techniques to be used, before modeling initiates.

First, each development agent determines the operations (the services that role provides) for each role for the processes they participate. Then, for each operation, they develop a description with an individual role-process diagram. Fig. 4 gives an example of an individual role process diagram.

The notation for describing the processes, is primarily based on eEPC (extended Event Driven Process Chain) [24] diagrams. eEPC diagrams are semi-formal and widely accepted in business process modeling practice. The main constructs are functions and events. An event can trigger a function or a function can produce an event so event and function combinations produce event-process chains. Data and organization view of business processes are also represented in eEPCs.

The diagram depicts the activities the role performs in that operation, the information items it requires while performing these activities and the outputs it produces. In additions to that, development agents were asked to represent the sources of the inputs and the destination of the outputs, if any, to and from its role's activities. The sources might represent other roles or items such as project repositories, folders, software tools and etc. Such representation of the interaction formed the expectations of that role from other roles. These diagrams are consistent in terms of role's expectations if, in the models of the other roles, they answer to these expectations by modeling the expected interface. For example, for the case given in Fig. 4, change manager expects to receive a change request form (1. section filled) from the change originator to start its activities. That is, it requires that information item to service that operation. This expectation is considered to be satisfied, if the change originator role, in any of its operation model declares that it delivers this item to the change manager. Otherwise there is an inconsistency between the expectations of these two roles in terms of their interface.

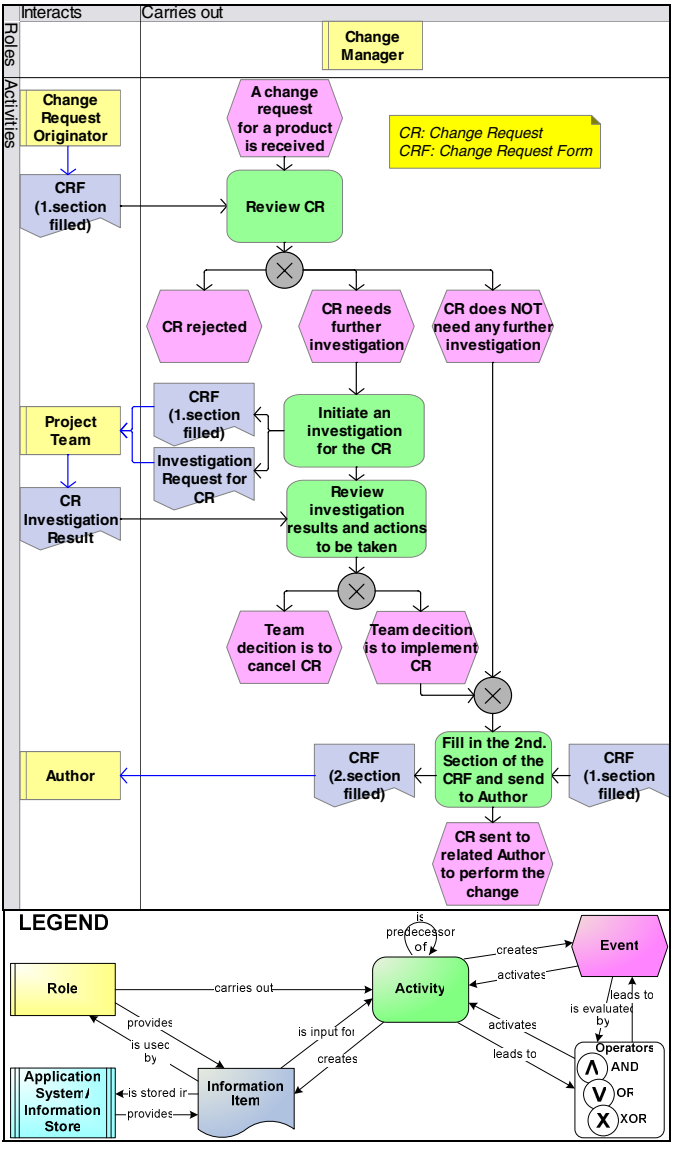


Fig. 4. Individual role-process diagram (a columnar eEPC diagram) for 'evaluate change request' operation of 'change manager' role in change management process

During process description, development agents identify inconsistencies between definitions based on the expectations of other roles. Consistency checking with respect to the expectations also requires considering aggregation and inheritance relationship between roles as well as the information items. Agents analyze the expectations and possible inconsistencies at anytime during process description. In case of an inconsistency, an agent either changes its description in order to match other's

expectations or insists on her position and communicated with other agents in order to solve the problem. Resolution is under agents' responsibility. Once inconsistencies and conflicts are resolved, the definitions are validated by peers and verified by the coordination team and the phase ends.

The primary output of this phase is a set of verified and validated individual role-process diagrams. Consistency should be established within (intra-consistency) and among (inter-consistency [17]) role-process models. However, besides these concrete outputs, individual process modeling by each agent in the organization is itself an important and a rewarding artifact of this phase. It enlightens agents about, the roles they are playing; the activities they are performing; the information they are producing and consuming; and their interaction with other roles. Successful completion of this stage implies that, many of the implicit assumptions and conflicts related with above items are uncovered, shared and solved and a common understanding of activities and artifact among agents is established.

3.3 Integration and Change

Verified and validated models implicitly or explicitly convey a great extent of knowledge, such as; what processes is carried out; which roles participates in these processes and what they perform; what information a role needs; when it needs this information as well as how it acquires it. Therefore, integration and model generation was a matter of querying and questioning the right answers to this process-base.

The first type of diagram that can be generated is the one that integrates individual role-process diagrams into a model that depicts all the activities performed in that process at the lowest level of detail. Fig. 5 depicts a diagram for the change management process. In essence, the integration for the activity-level process diagrams is performed by drawing related individual diagrams side by side and joining them with the messages the roles sent to each other and the activities they perform together.

Higher level process diagrams such as the ones depicting the role operations and message transfers can also be generated. Other types of diagrams such as processes with higher abstraction levels, their process dependencies as well as role dependencies can also be depicted. Model integration and generation is performed to obtain the macro view of the processes performed by the organization. Role dependency diagrams helped the organization to understand the interactions and interdependencies existing between roles and the implications of changing these relationships as well as to identify and compare alternative executions.

Changes to the scope including the changes on role definitions are discussed and approved by the process group including all members and reflected on the models by the coordination team. Changes related to individual role definitions, on the other hand, are directly performed by the related agent that are playing that role and reviewed by its peer agent.

The organization at this phase achieves a process-base that comprises diagrams representing the structural frame, individual role-process diagrams that are maintained by agents, and variety of generated models that visualize the way the organization works from different perspectives. Diagrams help agents to acquire the ability to understand the way processes execute, pinpoint problems and inefficiencies, identify improvement opportunities, and recommend changes and improvements.

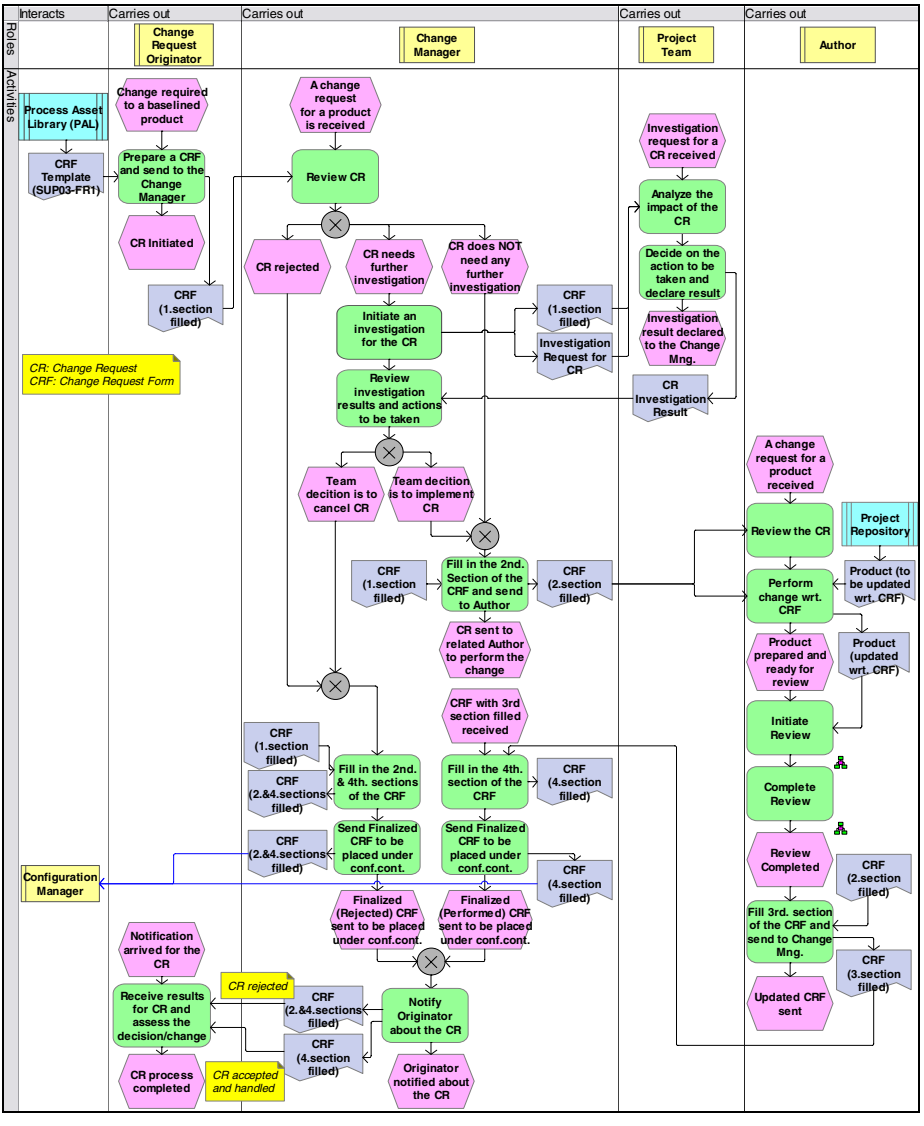


Fig. 5. An integrated activity-level process diagram for change management process

4 Case Studies and Lessons Learned

In order to evolve and observe the applicability of the method, two exploratory case studies were conducted. First study was performed in a department of a university and covered administrative processes such as student admissions, staff recruitments, etc. The second case was performed in a small software development and consultancy company and included a number of software engineering processes. The study group involved six participants in the first and four in the second case study. Almost all

agents were familiar with process modeling and related concepts in both studies. Yet, except the coordinators they were not directly acquainted with the notation and the toolset they utilized. The software organization already had procedures and guidelines for process execution written in natural language, which eased both the initiation and execution of the study. The participants, in the second case were also given a questionnaire to provide feedback on the approach followed and interviewed to elicit benefits and difficulties observed.

The tool used in these studies was the ARIS Collaborative Suite [24], which mainly supports ARIS (Architecture of Integrated Information Systems) methodology for enterprise modeling. The tool was extended with an add-on that analyzes process repository to detect and present inconsistencies between individual process models.

Table 1 summarizes the effort utilized for the studies. With the limited support of the tool, the integration took considerable amount of time for both studies, which can be significantly reduced as it can be mostly automated.

Table 1. The extent and effort utilized for the studies

	Case Study 1	**Case Study 2**
# of high level processes	12	5
# of roles	30 (13 active)	18 (15 active)
# of distinct role operations	NA	48
# of development agents	6	4
Effort:	person-hour	person-hour
Context Definition:	18	10
Definition and Conflict Resolution:	76	19
Agent1:	*9.0*	*9.0*
Agent2:	*5.0*	*5.0*
Agent3:	*20.0*	*2.5*
Agent4:	*13.0*	*2.5*
Agent5:	*11.5*	-
Agent6:	*17.0*	-
Integration:	20	5
Total	**114**	**34**

Case studies revealed number of advantages as well as limitations and key success factors. During process description and inconsistency resolution, agents identified the problems mostly related with incompleteness and ambiguities in procedures as well as implicit assumptions they hold during executions. That tacit knowledge enabled them to handle any ambiguity or fill any gap between the execution and the definition. They also reflected on how they should actually perform their responsibilities and started adapting their processes with respect to other's situations and expectations.

For several processes, agents realized that their execution was no longer adhering to the definition. For certain reasons, individuals changed the way they perform the processes but the definition stayed as it is. Moreover, the role that is responsible to perform the change was not always clearly stated. Changes to the process definitions and related artifacts were managed via the change management process and generally performed by a specific agent. However, the current state of the process, as agents noted in interviews, put a degree of bureaucracy on implementing the change itself,

which led to hesitations for initiating the change process when a necessity for change or an improvement opportunity is identified. Instead, individuals simply started altering their execution and began departing from the definition. For example, the review record that some of the agents were using incorporated more information than the standard form in their process assets library. So, this improvement chance was hindered or postponed.

The responsibilities are inherently lucid in Plural. Individual definitions are altered by related agents any time a change is necessitated. If a change did not affect role's interface, then it is a simple alteration in role's context. For example, configuration manager's alteration in the operation of 'placing under configuration control' did not affect the way other roles perform their processes, since the change does not have an impact on other role's expectations, but only affected the way the configuration manager performs its activities and produce its artifacts. However, if an update modifies its interface with the neighbor roles, then that change is incorporated in related models or it is revoked after a negotiation between parties. Such changes and their impacts are triggers that ruin consistency, and related agents should resolve them and reach to another consistent state again. The approach gave agents the responsibility to reflect any change they consider necessary not only on the way they perform their tasks but also on the artifacts they own. Being the supplier of these artifacts, they were responsible to maintain them and communicate any change with the customers of these artifacts. This might also involve negotiations with them on the content and structure.

According to the questionnaire and interviews with the participants, agents adapted to the method more rapidly than expected. They found it useful; to isolate their roles and responsibilities clearly from others and maintain them separately. All agents strongly agreed that modeling gave them a better understanding of the processes they perform and explicit modeling of their interface between other roles provided useful and important information about the process otherwise would have been omitted. Particularly for process guidance, role-based modeling was very useful since it clearly represented the responsibilities and the interface for each role.

In addition, role-based modeling by agents eased each individual to define role-based metrics and integrate the information into their process definitions. In the second case, agents were able to define when and by which means the metrics will be collected and stored so that they could also track their individual performance. Capturing each role's objectives for processes is important in understanding why the process operates as it does. This may help organization to assess process goals and the goals of its participants and help them to understand process' complexity before altering key relationships during the process change [14], [29].

5 Conclusion

As a limitation of the approach, the studies showed that the expected benefits are not fully realized if the processes being defined and are not performed or not effectively established in the organization. Another issue is related with the pattern of the organization that utilizes Plural. The software company that we performed our second case can be considered to have a relatively higher maturity in terms of its operational environment, its process stability as well as the way it considers process improvement. We

believe that, this eased the way the approach is adapted by the organization, since it fostered agent's motivation and commitment on to the study. Besides managerial and organizational issues, the diagrams utilized for the approach have some cognitive limitations. Additional information can be represented with the sacrifice in the ease of perception. For some of the processes, the process diagrams (such as student admissions, project management) were too large to fit into a regular sheet of paper which increased its complexity. Another limitation was related with the tool used. The team could not find a tool that would provide full support for the approach and that was a motive for an add-on developed onto a commercial software. However, the add-on had its own limitations and the real benefits would be gained with a tool answering the unique requirements of the approach.

We should also note that the case was performed with a limited number of participants and scope. Therefore, we have currently no data if the approach will scale-up to be used for large knowledge-based organizations with hundreds of knowledge workers and spanning a larger extent of processes across the organization.

Overall we can conclude that the study provided initial evidence of the approach's value and showed how an organization might exploit its strengths. The method helped participants to clearly define their expectations and goals and negotiate with other agents to achieve them. It provided an environment and a mechanism to define agent's expectations, unveil and discuss problems and establish a common jargon between participants while letting them to represent and keep theirs. All these hands on experiences, in turn, facilitated communication and knowledge sharing between participants. In addition, the total time for process definition and conflict resolution became the time committed by a single agent that performed its portion the longest. In other words we had process improvement cycles measured in days.

Acknowledgments. This study is supported in part by the Ministry of National Defence, Undersecretariat for Defence Industries (Republic of Turkey), KAMA-C4ISRMOS project.

References

1. Armour, Philip G.: The Laws of Software Process: A New Model for the Production and Management of Software. Auerbach Publications (2004)
2. Ben-Shaul, Israel Z. & Kaiser, G.E. A paradigm for decentralized process modeling and its realization in the Oz environment. In Proceedings of 16th International Conference on Software Engineering. Sorrento, Italia. (1994) 179–190
3. Davenport, Thomas H.: The New Industrial Engineering: Information Technology and Business Process Redesign , Sloan Management Review, 31:4 (1990) 11-27
4. Demirörs, O.: A Horizontal Reflective Process Modeling Approach for Managing Change in Software Development Organizations. Ph.D. Thesis. School of Engineering and Applied Science, Southern Methodist University (1995)
5. Drucker, P.F.: The New Society of Organizations. Harvard Business Review, (1992) 95-104
6. Finkelstein, A.; J. Kramer; B. Nuseibeh; L. Finkelstein; M. Goedicke: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. Int. J. of Software Eng. and Knowledge Eng. Vol. 2, No. 1, World Scientific Pub. Co. (1992) 31-58.

7. Finkelstein, A.; I. Sommerville: The Viewpoints FAQ. SWE Journal, Vol. 11 (1996) 2-4
8. Graw, G. & Gruhn, V. Distributed Modeling and Distributed Enaction of Business Processes. Proc. of the 5th European SWE Conference (ESEC) (1995) 8-27
9. Gruhn V. Communication support in a process-centered software engineering environment. ISPW (1994) 37-41
10. Jennings, N.R.; T. J. Norman; P. Faratin; P. O'Brien; B. Odgers: "Autonomous agents for Business Process Management" International Journal of Applied AI 14(2) (2000) 145-189
11. Jennings, N.R.; M. Wooldridge: "Agent-Oriented Software Engineering". Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (2000)
12. Kaiser, G. E. Rule-based modelling of the software development process. Proceedings of the 4th international software process workshop on Representing and enacting the software process. Devon, United Kingdom. (1988) 84 - 86
13. Kasse, T.: Practical insight into CMMI. Artech House (2004)
14. Katzenstein, G.; F. J. Lerch: Beneath the Surface of Organizational Processes: A Social Representation Framework for Business Process Redesign. ACM Transactions on Information Systems, Vol. 18, No. 4 (2000) 383-422
15. Leonhardt, U., Kramer, J., Nuseibeh, B. & Finkelstein, A. Decentralised Process Enactment in a Multi-Perspective Development Environment. Proc. of the 17th Int. Conference on Software engineering. Seattle, Washington, USA. (1995) 255-264
16. Mullery, G.: CORE - a method for controlled requirements expression. Proc. of ICSE-4, IEEE Computer Society Press (1979) 126-135
17. Nuseibeh, B.: A Multi-Perspective framework for Method Integration. PhD Thesis, Department of Computing, Imperial College, London (1994)
18. OMG, UML: Superstructure, Ver.2.0, Formal/05-07-04, Object Management Group (2005)
19. Ould, M.A.: Designing a re-engineering proof process architecture. Business Process Management Journal, Vol. 3 No. 3, MCB University Press (1997) 232-247
20. Ould, M.A.: Preconditions for putting processes back in the hands of their actors. Information and Software Technology, Vol. 45 Elsevier B.V. (2003) 1071-1074
21. Reenskaug, T.; P. Wold; and O.A. Lehne: Working with Objects: the OOram Software Engineering Method. Manning Publications (1996)
22. Sommerville, I., Kotonya, G., Viller, S. & Sawyer, P. Process Viewpoints. Proceedings of the 4th European Workshop on Software Process Technology. Springer-Verlag. (1995) 2 - 8
23. Senge, P.: It's the Learning: The Real Lesson of the Quality Movement. Journal for Quality & Participation, Nov/Dec99, Vol. 22, Issue 6. (1999)
24. Scheer, W.A.: ARIS- Business Process Frameworks. 3rd Ed., Springer-Verlag Berlin (1999)
25. Singh, B.: Interconnected Roles (IR): A Coordination Model. Technical Report CT-084-92, Microelectronics and Computer Technology Corporation, Austin, Texas USA (1992)
26. Turgeon, J.: A View-Based System for Eliciting Software Process Models. Ph.D. thesis, McGill University, September (1999)
27. Verlage, M., "Multi-View Modeling of Software Processes", In Proceedings of EWSPT3, pp. 123-127, Springer-Verlag, Grenoble, France (1994)
28. West, M.: Real Process Improvement Using the CMMI. Auerbach Publications (2004)
29. Yu, E.; J. Mylopoulos: Understanding "Why" in Software Process Modelling, Analysis, and Design. Proc. of the ICSE-16, Sorrento, Italy (1994) 159-168

A Survey of Software Development with Open Source Components in Chinese Software Industry

Weibing Chen[1], Jingyue Li[2], Jianqiang Ma[1], Reidar Conradi[2], Junzhong Ji[1], and Chunnian Liu[1]

[1] Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology, College of Computer Science and Technology, Beijing University of Technology (BJUT), Beijing 100022, China
{weibingchen,jianqiang.ma}@gmail.com
[2] Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway
{jingyue,conradi}@idi.ntnu.no

Abstract. Chinese software companies are increasingly using Open Source Software (OSS) components in system development. Integrating such components into new software systems leads to challenges related to component selection, component integration and testing, licensing compliance, and system maintenance. Although these issues have been investigated industrially in other countries, few state-of-the-practice studies have so far been performed in China and with a representative subset of software companies. It is therefore difficult for Chinese software companies to be aware of special issues, or to plan improvement of OSS-related processes. This paper describes a questionnaire-based survey in Chinese software companies of software development with existing OSS components. Data from 47 finished development projects in 43 companies have been collected. The results show that use of web search engines was the most common method to locate OSS components. Local expertise combined with requirements compliance was the most decisive factors when choosing an identified component. To avoid legal exposure, the common strategy was to use components without licensing constraints. About 84% of the components needed bug fixing or other code changes, rarely relies on support from the OSS community. However, close participation with the OSS community was rare, although most developers meant that this was important.

Keywords: CBSE (component-based software development), OSS component, Empirical study.

1 Introduction

Building new software systems by pre-fabricated components is an attractive way to achieve lower cost, shorter time-to-market, higher quality, adherence to industrial standards etc. [11]. It has recently become more and more popular to reuse Open

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 208–220, 2007.

Source Software (OSS) components in system development [2, 5, 16, 18]. Such components offer many advantages, such as free and changeable code. Indeed, many OSS components are recognized for their high reliability, performance, and robustness [17]. On the other hand, reusing OSS component (and "external" component in general) raises challenges in selecting the right component and to successfully integrate and test the selected component [12]. In addition, it is important to select and integrate OSS component with proper license, if the developed system is going to be distributed or sold to the general market [2, 17].

Many previous studies of OSS-based development are based on theoretical proposals (especially around component selection) [2, 6] and industrial case studies [5, 16]. One major survey has been performed to investigate the state-of-the-practice of OSS-based development in three European countries [11]. Although China has become a major actor to employ OSS software in industry, especially regarding software platforms like Linux, little research has been performed on the challenges of efficient reuse of OSS components in Chinese software industry.

Our questionnaire-based survey focuses on three main issues in reusing OSS components for software development in Chinese software industry, namely component selection, licensing terms, and system maintenance. We have used membership lists from a national Chinese software organization (CSO for short)[1] to achieve an almost representative subset of software companies. We have gathered information from 47 finished projects in 43 companies. The results show that use of web search engines was the most common method to locate OSS components. Local expertise combined with requirements compliance was the most decisive factors in deciding upon an identified component. To avoid legal exposure, the common strategy was to use components without licensing constraints or to package proprietary code separately. About 84% of the components needed bug fixing or other code changes, rarely relies on support from the OSS community. In addition, close participation with the OSS community in so-called OSS projects was rare on most issues, although most developers meant that this was important.

The rest of this paper is organized as follows: Section 2 describes the background. Section 3 discusses the research approach. Section 4 presents results and discussion of research questions, Section 5 contains a general discussion, and a conclusion and ideas for future work are presented in section 6.

2 Background

2.1 Concepts Used in This Study

In this study, we define a **software component** as in [10]:"Software components are executable units of independent production, acquisition, and deployment that can be composed into a functioning system." An **OSS component** is defined as a software component that: a) Is provided by the OSS community; b) Is subject to licensing constraints; c) Is not a platform software (e.g., OS like Linux, DBMS, or similar software).

[1] The name of this organization was omitted for confidential reasons.

2.2 State-of-the-Art

There have been two main kinds of empirical studies of OSS:

- *Cultural-oriented studies* concentrate on how to make new OSS software's and components, the OSS project itself and its organization as an OSS community, the participators' motivation, and the evolution of the OSS project [14].
- *Technical-oriented studies* like this one, concentrates on process issues in reusing existing OSS components to develop new software [13, 17].

The aim of this study is therefore to establish some empirical-based guidelines to make OSS-based development to run more smoothly. Typically, such a development process includes several stages, such as OSS component selection, component integration, and system maintenance.

2.2.1 OSS Component Selection

Selecting a right component is one key factor for the success of OSS-based development. Typically, the component selection process includes locating candidate components, evaluating components based on pre-defined criteria, and deciding upon components [12, 15]. Most previous studies on component selection focus on selecting COTS (commercial-off-the-shelf) components [1, 15]. Due to the peculiar nature of OSS components, the process and criteria to select OSS components are quite different with those used to select COTS components [6]. The proposed COTS component selection process may not fit OSS selection very well [6].

2.2.2 OSS Component Integration and OSS Licensing Issues

After OSS components are selected, the next step is to integrate them into the target system. To ensure the success of integrating the OSS components, the integrators need to consider not only technical issues, such as API and programming language, but also the licensing terms of the selected OSS components. There are more than 50 different OSS licenses [9]. Some licenses have strict constraints on the distribution or resale of the derived system from OSS components. For example, the GPL (GNU Public License)-type licenses do not give the licensee unlimited redistribution rights. The right to redistribute is granted only if the distribution is licensed under the terms of the GPL and includes, or unconditionally offers to include at the moment of distribution, the source code [12, 17].

2.2.3 System Maintenance

After OSS components are integrated into a software system, it is important to maintain and update those components properly for a long term use. Most technical supports from OSS communities are in the form of mailing lists and bulletin boards [12]. Since these supports are provided mainly by loosely organized volunteers, it is difficult to control the support quality. To get high quality and long-term support, one proposed strategy is to establish a long-term working relationship with the OSS community [16]. That is, the OSS component users not only download software from the community, but also upload the modified software to the OSS community [13, 16]. Such a relationship between users and the OSS community is supposed to benefit both practitioners [2].

2.3 State-of-the-Practice of OSS-Based Development in China

China is one of the major countries using OSS in information systems. The Chinese government has played an important role in the process of promoting the Chinese OSS movement. For example, The Japan-China-Korea (JCK) open alliance which announced in November 2003 is an initiative to promote OSS by cooperation [8]. Due to the Chinese government's encouragement on the use of Linux and OSS, more and more Chinese software companies start to use OSS components to develop software. No other country comes even close to the level of advancement that China has achieved in deploying OSS, particularly Linux [8]. The current scale of OSS-based development is large enough to be noticed at the global level. However, there are few empirical studies on OSS-based development in Chinese software industry.

3 Research Approach

3.1 Research Questions

This study is to investigate the state-of-the-practice of OSS-based development in Chinese software industry. We designed three research questions RQ1 to RQ3 and corresponding sub-questions.

The number of OSS components has increased dramatically these years. More than 137,000 OSS projects have been registered at sourceforge.net. Facing so many OSS components, it is difficult to select the best one to be integrated into a new system. Although researchers have proposed several structured, formal, semi-formal selection processes, and various evaluation criteria, there are few empirical studies have observed the actual selection process used by commercial developers [12]. Thus, our research question **RQ1** and corresponding sub-questions **RQ1.1** and **RQ1.2** are:

RQ1: How the OSS components were selected in practice?

- **RQ1.1.** what methods were used to locate candidate OSS components?
- **RQ1.2.** what evaluation criteria were used to evaluate OSS components?

Many studies claimed that the OSS licensing terms affect the using of OSS components. Although Ruffin [17] discussed major legal aspects of using OSS and related risks mitigation strategy, few studies have illustrated how the licensing issues are managed in practice. So the second research question **RQ2** and corresponding sub-questions **RQ2.1** to **RQ2.4** are:

RQ2: How did OSS license affect the OSS component selection and integration?

- **RQ2.1.** How well did developers understand OSS license?
- **RQ2.2.** Did developers read related OSS licensing terms?
- **RQ2.3.** Did developers encounter OSS license related troubles?
- **RQ2.4.** what strategies were used to avoid the possible OSS licensing troubles?

To get long-term technical support of the integrated OSS components, establishing a long-term relationship by engaging in the related OSS community has been proposed as a solution [7, 16]. However, this proposal lacks support from industry practices. Thus, our research question **RQ3** is:

RQ3: Did the engagement in the OSS community facilitate the maintenance of the integrated OSS components?

3.2 Research Design

To answer the research questions, we have used a survey to collect data. First, a preliminary questionnaire with both open-ended and close-ended questions was designed by reading literature. Second, a pre-study was performed to validate the quality of questions in the preliminary questionnaire and to get answers of the open-ended questions. Based on the results of the pre-study, all open-ended questions in the preliminary questionnaires were redesigned into close-ended questions. In addition, the problematic questions in the preliminary questionnaire were revised. Then, the revised questionnaire was used to collect data in a main study.

3.2.1 The Preliminary Questionnaire

The preliminary questionnaire has 5 sections. Sections 1 and 5 contain questions to collect background information of projects and the respondents. Sections 2, 3, and 4 include questions to investigate our research questions.

3.2.2 The Pre-study to Verify and Refine the Preliminary Questionnaire

The pre-study included two steps, i.e., individual interviews followed by a group discussion.

Step 1 – Individual interviews. We have interviewed 5 project managers from 5 different companies. All interviewees have solid experience with OSS-based development. Each interview was conducted by two authors of this paper. One was responsible for conducting the interview, and the other recorded answers and asked for clarification if needed. The interviews were also taped for later verification.

Step 2 – A group discussion. After the individual interviews, we revised the open-ended questions in the preliminary questionnaire to close-ended questions and made a second version of the preliminary questionnaire. We then organized a workshop with more than 30 industrial experts to verify and comment on the second version of the questionnaire. Based on comments from the workshop, we revised the questionnaire into a final version. The final questionnaire includes about 35 questions and takes about half one hour to be filled out.

3.2.3 The Main Study to Collect Data

In the main study, the data was collected by the cooperating with the CSO. In total, we got 47 questionnaires from 43 companies (4 companies filled in 2 questionnaires each). The sample selection and data collection process are as follows:

1. **Assemble the target population.** We randomly selected 2,000 companies from a database of CSO, which included about 6,000 companies.
2. **Send invitation letters by email to obtain possible participants.** We sent invitation letters by email to the 2,000 selected companies. The invitation letter introduces the survey. We specified that survey participants will be rewarded with either the final report of the survey or an annual membership of the CSO worth of 500 Chinese Yuan. We got about 200 company responses from this step and these companies were used as the original contact list.

3. **Send questionnaires by email to possible participants.** We sent questionnaires (as word files) by email to the 200 companies and asked them to select one completed software development project, which used one or more OSS components, to fill in the questionnaire. Since we cannot get the complete list of relevant projects in such a company, project selection within the company was decided by the respondents themselves. *Therefore the sample selection process was a random selection of companies, followed by a convenience sample of relevant projects within companies.*
4. **Collect filled-in questionnaires with follow up.** From the 200 companies, we first got 40 questionnaires back. To ensure the quality of the data, we excluded 10 questionnaires answered by programmers whose work experiences were less than three years. For the remaining 30 questionnaires, we contacted the respondents again by telephone to clarify possible misunderstanding and to fill in the missing data. At the same time, we contacted the remaining of 160 companies by telephone to persuade them to fill in the questionnaire. By doing this, we got 17 other questionnaires back.

4 Results and Discussion of Research Questions

In this section, we first present background information of the interviewees, participating companies, and projects. We then show the results for each research question followed by detailed discussion.

4.1 Background Information

Human respondents. Most respondents have a solid IT background. Five of them are IT managers, 13 are project mangers, 14 are software architects and 7 are senior software developers. Most of them have more than five years of software development and more than two years working experiences with OSS-based development.

Participating companies. According to number of employees, the participating companies include 7 small, 19 medium, 9 large, and 8 super large companies, as shown in Fig. 1. Comparing with the official number of employees in Chinese software companies [22], as shown in Fig.1, it shows that most of the participating companies are medium and large companies.

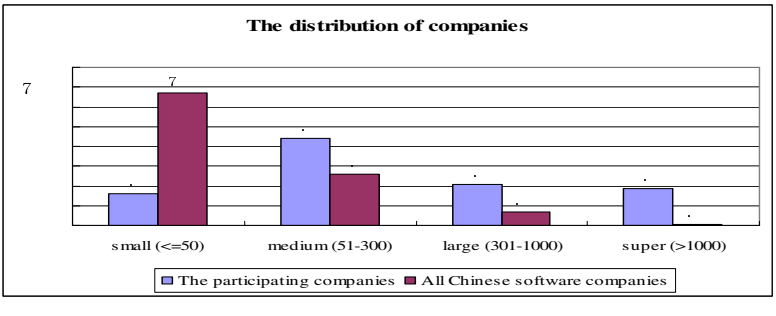


Fig. 1. The distribution of participating companies

Participating projects. Forty-six respondents filled in the actual-used effort of projects. Thirteen out of 46 projects used efforts less than 10 person-months, 18 used efforts between 10 and 100 person-months, and the remaining 15 projects used more than 100 person-months.

4.2 Investigating RQ1: How OSS Components Were Selected

Results of RQ1.1. To answer **RQ1.1**, we listed possible activities of locating OSS components from our pre-study and literature [15] as following: a) Have used it (them) before; b) From colleagues of same company; c) From friends of other companies; d) Through reading related magazines (e.g., Programmer magazine); e) Through visiting trade shows and exhibitions; f) Using search engines (e.g., Google, Yahoo); g) Visiting OSS project portals (e.g., sourceforge.net, freshmeat.com). The respondents were asked to answer whether they have performed such activities to locate OSS components or not. The results reveal that locating OSS components was mostly based either on **search engines** (e.g., Google or the search feature in Sourceforge) or **internal experience** (e.g., having used the components before, reading magazines, getting advice from internal colleagues). **External information channel**, such as getting advices from friends in other companies, was rarely used.

Discussion of RQ1.1. Previous studies have discussed the practices of selecting OSS components. In [12], the authors concluded that most companies use a manual (brute force) method, e.g., searching with Google or Sourceforge. Our data support that conclusion. However, our results show that developers used Google more frequent than Sourceforge. The authors of [12] also proposed that familiarity was the main attribute to be considered when selecting OSS components. Our results support this. As indicated in [13], companies were willing to listen to experience from other companies and were also willing to share their own experience with others. However, our results show that experience sharing between people in different organizations was not popular. The possible reason is that there is a lack of channels to share experience of using OSS components between different organizations.

Results of RQ1.2. To answer this question, we formulated possible criteria to be considered when evaluating OSS components from [3, 12] as following: a) Requirements compliance; b) Architecture compliance; c) Quality of components (security, reliability, usability etc.); d) Functionality; e) OSS licensing terms; f) Price; g) Reputation of components or supplier; h) Quality of documentation; i) Expected support from the OSS community (updating, bug fixing etc.); j) Environment to be used in (platform, hardware etc.).Respondents were asked to answer "don't agree at all", "very low", "low", "medium", "high", and "very high", or "don't know". We assigned an ordinal number from 1 to 5 to the above alternatives (5 meaning very high). The results illustrate that **requirements compliance** (i.e., with median value 4) is the most important criteria to be considered. On the other hand, **price and support** are the least important criteria to be considered (i.e., with median value 3). The importance of other criteria, such as component quality and reputation, architecture compliance, OSS licensing terms are between.

Discussion of RQ1.2. Our results confirm that one of most important criteria to be considered when evaluating OSS component is still requirement compliance, rather

than architecture compliance proposed by [12]. The authors of [10] proposed that components with more and better comments in the community or marketplace bulletin had a good chance to be selected, because they were assumed to be better tested with generally good qualities. Our data can give that conclusion further support. Although previous studies claimed that technical support was very important to ensure the success of OSS-based systems [5, 20], our data show that the possible support from the OSS community was not considered as very important during component evaluation.

4.3 Investigating RQ2: How the Licensing Terms Were Complied

Results of RQ2.1-RQ2.3. Questions related to **RQ2.1 to RQ2.3** and corresponding answers are in Table 1**. RQ2.1** and **RQ2.3** were used the same measurement as **RQ1.2**. With respect to **RQ2.2**, respondents were asked to answer "don't agree at all", "hardly agree", "agree somewhat", "mostly agree", "strongly agree", or "do not know". We assign an ordinal number from 1 to 5 (5 meaning strongly agree) to these alternatives.

Table 1. Results of **RQ2.1**-**RQ2.3**

RQs	Questions in the questionnaire	Results
RQ2.1	What was the extent of your understanding of OSS license?	The results show that most respondents did not understand OSS licenses very well.
RQ2.2	Have you read all licensing terms of the OSS component that you are using?	The results show that respondents have only partly read OSS licensing terms.
RQ2.3	Have you encountered OSS license related troubles?	21% of the respondents never encountered OSS license related troubles. The remaining respondents rarely encountered license related troubles.

Since the respondents' understanding and correct use of OSS licenses may be affected by their emphasis on licensing issues in the selection phase, we wonder whether the more the developers considered licensing terms in the selection phase, the better they understood the licensing terms. To investigate this question, we calculated the correlations between the respondents' emphasis of license criteria in the selection phase and answers of the above three questions with a *Spearman rank correlations* in SPSS 11.0.

Discussion of RQ2.1-RQ2.3. Results show that there are no **significant** correlations between the respondents' emphasis on licensing terms in selection phase and their knowledge and effort used to read these licensing terms. Surprisingly, the more developers emphasized the OSS licensing terms, the more frequently they encountered license related troubles. The possible explanation is that people did not understand licensing terms and did not take proper action to avoid possible troubles, even though they considered licensing terms as an important issue.

Results of RQ2.4. RQ2.4 deal with what actions have been used to avoid possible license related troubles. From the literature [2, 12, 16, 17], we have summarized possible strategies as following: a) Use other components without licensing constraints; b) Consult legal experts for help; c) Develop modules containing GPL-based components and with APIs exposing them, in order to avoid GPL restrictions; d) Package the proprietary code separately to avoid GPL restriction; e) Contact the OSS license's "owner" and agree on a certain license to avoid the licensing impacts; f) Place all the "derived programs" which relate to licensing issues, back to the OSS community.

We use the same measures as **RQ2.2**. The result shows that using other OSS components without license constraints was the most popularly used strategy. On the other hand, putting all "derived programs" back to OSS community was the least used strategy. The frequency of using the other strategies, such as packaging open source code with proprietary code separately and contact OSS license's "owner", was between.

Discussion of RQ2.4. From the OSS component users' perspective, the main concern on OSS licensing term is whether the system reusing OSS components is defined as a "derived programs" [2]. If so, according to many OSS licenses, the "derived work" should be published. The source code of project is a private property for business companies which hide the intellectual property (IP) from their competitors and make profits on IP investment [12]. When using OSS components, our results show that business companies would rather use components without strong licensing constraints to avoid making their code public.

4.4 Investigating RQ3: How the Maintenance Was Performed

Results of RQ3. This research question investigates how to maintain OSS-based systems smoothly. We first investigated whether developers needed to fix bugs and to change the source code. If the answer was 'Yes', the follow up questions were what they did. Results show that 44.7% of respondents needed bug fixing and 39.3% of the developers needed to change code. When they did the fixing or changing, our results show that more respondents prefer to do it themselves rather than to ask for help from the OSS community. However, respondents needed more effort (40 person-hours) on average to correct errors by themselves than by the OSS community (11 person-hours). On the other hand, respondents need less effort on average to change the code themselves (35.2 person-hours) than by the OSS community (60 person-hours).

To answer **RQ3**, we also investigated the relationship between project developers and the OSS community. We asked respondents whether there were developers (i.e., those in their projects) that have taken part in the OSS community. Only 4 respondents said 'Yes'. For the respondents with "No" answers, they were asked to select one from the following three reasons with the same measures as in **RQ2.2**. a) There was no need to take part in the community; b) Do not have enough resources (such as time, human resources, etc.); c) It was difficult to take part due to the hierarchy of the OSS community. Results illustrate that developers thought it was needed to take part in the OSS community. Due to resource limitation, such as time and cost, most of them did not join in the OSS community. However, joining in the OSS projects was not regarded as a difficult.

Discussion of RQ3. Although taking part in a corresponding community and contributing to the OSS projects and getting contributions published may not be straightforward, it proved to be helpful [13]. Our results show that most developers thought that taking part in OSS community was needed. However, there was a lack of resource to do that. Fortunately, there are many other ways to work with the OSS community. Perhaps the simplest way is to provide feedback and to report bugs to OSS projects [7, 13]. In addition, new features and possible implementation of the features can be proposed to OSS projects [13, 20].

5 General Discussion

This study summarized the practices of three key issues of OSS-based development in Chinese software industry, namely selecting OSS components, complying OSS licensing terms, and maintaining OSS components. Based on our results, we give three suggestions on facilitating the OSS-based development.

Improve the OSS search engine to facilitate experience sharing
Although several methods can be used to locate OSS components, our findings in **RQ1** show that two methods had been used most popularly, i.e., web search engines (e.g., Google) and OSS project portals (e.g., Sourceforge.net). The same findings have been reported in [12]. The advantage of using web search engines is that they are simple and fast. However, the disadvantage is that the search results are imprecise and possible huge. The advantage of using OSS project portals is that the OSS projects are centralized and classified. On the other hand, one OSS project portal can not include all OSS projects. People have to search in several portals to get all possible component candidates. The new 'Google Code Search' helps to solve the above shortcomings by combing portals of the open-source domain.

When selecting and evaluating the OSS components, experience of previous use of OSS components is valuable. Our results of **RQ1** show that, however, experience sharing was limited to internal colleagues. To facilitate experience sharing between different companies, it would have been better for 'Google Code Search' to include and structure the users' experience and comments of using components, i.e., creating an OSS community for relevant components.

Understand and comply with OSS licensing terms properly
Another important issue of reusing OSS component is OSS licensing terms [12]. It is important for companies to carefully read, understand, and comply with the license of an OSS component. Our results of **RQ2.1** and **RQ2.2** show that most developers did not read and understand OSS licensing terms properly. Although there are many OSS licenses in use (more than 50 approved by opensource.org) and the licensing terms varies, five common licenses (i.e., GPL, LGPL, BSD, AL, and MIT) [20, 21], which are simple to comply with, cover 90% of OSS projects [20]. It is may be wise for OSS users to learn and understand these most common licenses before they start to select and integrate OSS components.

Take a more active part in the OSS community
When considering maintenance of the OSS-based system, project developers may need to fix bugs of OSS components, to add or revise the components' functionalities.

Our results of **RQ3** show that developers needed more effort on debugging, than what the OSS community did. A better way might be to report bugs on bulletin boards and then letting the OSS community fix them. To change the OSS component code, our results of **RQ3** show that asking the community the changes needed more effort than doing the changes locally. The possible reason is that OSS community needs a long time to accept suggested changes.

As indicated in previous studies, one of the solutions to the maintenance of OSS-based system is to take part in OSS community [7, 16]. Some previous data show that 83% of community participants live in the Western countries and 55% of them contribute to OSS projects during working hours [14]. In contrast, our results from Chinese software industry show that only 9% of the investigated projects had dedicated developers take part in the OSS community. Thus, one of the primary tasks of the Chinese OSS movement is to mingle with the OSS community [19].

Possible threats to validity

a) Construct validity. In this study, most variables and alternatives are taken directly, or with little modification from existing literature. We did a pre-study to ensure the quality of questionnaire, and nearly 15% of the questions and alternatives in the final questionnaire were revised based on the pre-study.

b) Internal validity. We promised respondents in this study a final report or the annual membership of the CSO which worth of 500 Chinese Yuan. Most respondents took part in this survey as volunteers and selected the report as the reward. We therefore think that the respondents answered the questionnaire truthfully. However, our unit of study was a finished project. So a possible threat is that the respondents have failing memory on past projects.

c) External validity. There were more than 11,550 software companies registered in China in 2005 [22]. The CSO database contained only less than a half of them. Although we have put much effort on collecting data, we only got data from 43 companies out of our initial contact list of 2000 companies. For the remaining companies, we do not know their reasons for not participating. The respondents answered the questionnaires based on finished projects which were selected based on convenience by respondents. All the above issues may bring external threats to the conclusion of this study.

6 Conclusions and Future Work

More and more software companies are reusing OSS components in their software development projects, in China and elsewhere. Such companies need empirically-based guidelines for OSS-based development. The main conclusions of our survey are:

- Selection of OSS components is mainly based on existing web search engines, followed by local expertise for evaluation, e.g., requirements compliance and assumed component quality. The new Google code search engine (http://labs.google.com) illustrates the need for improved search support.
- OSS licensing terms are not a barrier to software companies, when reusing OSS components in system development.

- System maintenance leads in 84% of the development projects to bug fixing or other code changes in the selected OSS components, and involves the OSS community on a case-to-case basis. We recommend that the experience and knowledge around relevant OSS components is handled by an internal "component uncle", and by a more active participation with the OSS community. The latter is also expressed by the developers themselves, but not followed up - perhaps for cultural and organizational reasons?
- Finally, since China has no comprehensive, national database of software companies, it is difficult to select a random sample of participants in such surveys, even if the present one is maybe as good as we can get.

In Europe 2005, over 50% of the software companies report that they are using OSS components in own software development [4]. We do not know a similar figure for China, but have a feeling that it is lower. We therefore need further studies of the extent, challenges, problems and cost/benefits of OSS-based software development in China and elsewhere. We also need to study in what ways the use of OSS affect the software projects.

References

1. Briand, L. C.: COTS Evaluation and Selection. Proc. of International Conference on Software Maintenance, Bethesda, Maryland (1998) 222-223.
2. Brown, A. W., and Booch, G.: Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors. Proc. of the 7th International Conference on Software Reuse (ICSR-7). Austin, TX, USA, April 15-19, 2002, Springer Verlag LNCS, Vol. 2319, 123-136.
3. Dagdeviren, H., Juric, R., and Kassana, T. A.: An Exploratory Study for Effective COTS and OSS Product Marketing. Proc. of the 27th International Conference on Information Technology Interfaces, Cavtat, Croatia (2005) 644-649.
4. Evans Data Corporation, "Open Source/Linux Development Survey", 2006, http:// www.evansdata.com/survey_linux_topical.shtml.
5. Fitzgerald, B., and Kenny, T.: Developing an Information Systems Infrastructure with Open Source Software. IEEE Software, January-February (2004), 21(1):50-55.
6. Giacomo, P. D.: COTS and Open Source Components: Are They Really Different on the Battlefield? Proc. of the 4th International Conference on COTS-Based Software Systems. Bilbao, Spain, February 2005, Springer Verlag, LNCS, Vol. 3412, 301-310.
7. Holck, J., Larsen, M. H., and Pedersen, M. K.: Managerial and Technical Barriers to the Adoption of Open Source Software. Proc. of the 4th International Conference on COTS-Based Software Systems. Bilbao, Spain, February, 2005, Springer Verlag, LNCS, Vol. 3412, 289-300.
8. Kshetri, N.: Structural Shifts in the Chinese Software Industry. IEEE Software, July-August (2005), 22(4):86-93.
9. Open Source Initiative, 2005, available at http://www.opensource.org/index.php.
10. Li, J., Bjørnson, F. O., Conradi, R., and Kampenes, V. B.: An Empirical Study of COTS Component Selection Processes in Norwegian IT companies. Proc. of the Int'l Workshop on Models and Processes for the Evaluation of COTS Components (MPEC'04 Arranged in co-location with ICSE'04), Edinburgh, Scotland. May 2004, 27-30.

11. Li, J., Conradi, R., Slyngstad, O. P. N., et al.: An Empirical Study on Off-the-Shelf Component Usage in Industrial Projects. Proc. of the 6th Intl. Conf. on Product Focused Software Process Improvement, Oulu, Finland, Jun. 2005, Springer Verlag, LNCS Vol. 3547, 54-68.
12. Mandanmohan, T. M., and Rahul De': Open Source Reuse in Commercial Firms. IEEE Software, November-December (2004), 21(6):62-69.
13. Merilinna, J., and Matinlassi, M.: State of the Art and Practice of Open Source Component Integration. Proc. of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, Cavtat/Dubrovnik, Croatia (2006) 170-177.
14. Lakhani, K. and Wolf, R. G.: Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects, In: Feller, J., B. Fitzgerald, S. Hissam, K. Lakhani (eds.), Perspectives on Free and Open Source Software, MIT Press, Cambridge. (2005) 3-22.
15. Ncube, C., and Maiden, N.: Selecting COTS Anti-Virus Software for an International Bank: Some Lessons Learned! Proc. of the 26th International Conference on Software Engineering MPEC 2004, Edinburgh, Scotland, UK. (2004) 17-21.
16. Norris, J. S.: Mission-Critical Development with Open Source Software. IEEE Software, January-February (2004), 21(1):42-49.
17. Ruffin, M., and Ebert, C.: Using Open Source Software in Product Development: A Primer. IEEE Software, January-February (2004), 21(1):82-86.
18. Spinellis, D., and Szyperski, C.: How is Open Source Affecting Software Development? IEEE Software, January-February (2004), 21(1): 28-33.
19. Wang, G., and Zhang, X.: Chinese Linux Open Source Encounters Close. IT Time Weekly, Volume 22, (2004) 20-21.
20. Tuma, D.: Open Source Software: Opportunities and Challenges. Journal of Defence Software Engineering, January (2005) 6-10.
21. Ueda, M.: Licenses of Open Source Software and Their Economic Values. Proc. of the 2005 Symposium on Applications and the Internet Workshops (SAINT-W'05), January (2005) 381-383.
22. Ministry of Information of the PRC & Chinese Software Industry Association: Annual Report of China Software Industry, (2006): http://www.soft6.com/news/detail.asp? id=15759.

Empirical Study on Benchmarking Software Development Tasks

Li Ruan[1,2], Yongji Wang[1], Qing Wang[1], Mingshu Li[1], Yun Yang[1,3], Lizi Xie[1,2], Dapeng Liu[1,2], Haitao Zeng[1,2], Shen Zhang[1,2], Junchao Xiao[1,2], Lei Zhang[1,2], M.Wasif Nisar[1,2], and Jian Dai[1,2]

[1] Institute of Software, Chinese Academy of Sciences, Beijing 100080, China
[2] Graduate University, Chinese Academy of Sciences, Beijing 100039, China
{ruanli,ywang,wq,mingshu,xielizi,liudapeng,zenghaitao,zhangshen, xiaojunchao,zhanglei,wasif,daijian}@itechs.iscas.ac.cn
[3] Center for Information Technology Research, Swinburne University of Technology, Australia
yyang@ict.swin.edu.au

Abstract. Benchmarking is one of the most important methods to learn the best practices for software process improvement. However, in current software process context, benchmarking is mainly for projects rather than software development tasks. Can we benchmark software development tasks? If so, how to? Moreover, benchmarking software development tasks has to deal with multivariate and variable return to scale (VRS). This paper reports practical experience of benchmarking software development tasks under multivariate and VRS constraints using Data Envelopment Analysis (DEA). The analysis of experience data in Institute of Software, Chinese Academy of Sciences (ISCAS) indicates that the ideas and techniques of benchmarking software projects can be deployed at the software development task level. Moreover, results also show that DEA VRS model allows the developers to gain new insight about how to identify the relatively efficient tasks as the task performance benchmark and how to establish different reference sets for each relatively inefficient task under multivariate and VRS constraints. We thus recommend DEA VRS model be used as the default technique for appropriately benchmarking software development tasks. Our results are beneficial to software process improvement. To the best of our knowledge, we believe that it is the first time to report such comprehensive and repeatable results of benchmarking software development tasks using DEA.

Keywords: Benchmarking, software process improvement, performance, software development tasks, projects, data envelopment analysis.

1 Introduction

Benchmarking software projects is vital for any organization seeking to continuously improve its project management practices and identify competitive strengths and weaknesses [1]. Benchmarking in this context means to measure

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 221–232, 2007.

project performance against some established performance baselines [1]. On the other hand, recently, due to the rapid development of CMMI/TSP/PSP [2][3], the trend of software process improvement is "scaled down" to the level of individual developers [3].

Benchmarking software development tasks is vital for any developer seeking to continuously improve his task management practices and identify competitive strengths and weaknesses. In practice of project management, task is a basic unit of project and a fine-grained and detailed work breakdown for developers in a project. The need to benchmark software development tasks is clear [4]. However, in the current software process context, benchmarking is mainly for projects [1][4][5][6][7] rather than tasks. Moreover, benchmarking software development tasks is difficult to achieve, due to the multivariate and variable returns to scale (VRS, i.e., the relationship between the input and output of tasks is non-linear) [1] properties. Firstly, the program size, effort, defects, etc., are the most common input and output of tasks. The evaluation of task performance thus has to deals with multivariates. Secondly, as reported in [1], the relationship between size and effort is nonlinear. And as reported in [8], the relationship between size and defects is also nonlinear. The software development tasks thus exhibit VRS. In other words, to achieve the goal of total software process improvement (especially, PSP improvement), it is vital for organizations and developers to answer the question "*Can we deploy the idea of benchmarking software projects for benchmarking software development tasks which exhibit multivariates and VRS properties? If so, how to?*".

Furthermore, to correctly benchmark software development tasks, there are at least two key requirements and two corresponding hypothesises for us (e.g. developers) to verify.

- **Req1:** *We need to establish a task performance benchmark under multivariate and VRS constraints.* Only after the performance benchmark has been established, can developers determine which task to learn the best practices from. i.e., we must first identify the relatively efficient software development tasks.

Hypothesis 1: The relatively efficient tasks can be identified to establish the task performance benchmark under multivariate and VRS constraints.

- **Req2:** *We need to establish different reference sets for each relatively inefficient task under multivariate and VRS constraints.* Surely developers can roughly treat all the identified relatively efficient tasks derived from Req1 as a whole as the performance benchmark. However, in practical application, each identified relatively efficient task usually is of different improvement value for the inefficient one. Therefore, we must establish different reference sets for each relatively inefficient task.

Hypothesis 2: Different reference sets for each relatively inefficient task can be established under multivariate and VRS constraints.

Yet, to date, the project management literature has proposed few tools for enabling comparisons of tasks which explicitly consider the multivariate and VRS

properties of tasks. Commonly applied project performance evaluation methods, such as earned value management (EVM) [9], provide organizations with a method of systematically comparing actual performance to project/task goals. It does not take into account of the task uniqueness when performing cross-task evaluation. Statistical methods propose to compare the task performance with some theoretical optimal ones (e.g. theoretical baselines). However, as [1] recently reports, in software engineering, it seems more sensible to compare the performance with the best practice rather than with some theoretical optimal (and probably nonattaintable) performance. Furthermore, multivariate regression is unsuitable for identifying the best tasks because it tends to evaluate tasks relative to the average rather than relative to the best. Moreover, software tasks data are heteroscedastic. We could therefore wrongly tend to identify mainly the large tasks as the most productive without taking into account of the VRS properties of tasks.

Data Envelopment Analysis (DEA) developed by A. Charnes and W. W. Cooper in 1978 is a linear non-parametric programming-based performance evaluation technology. It provides a powerful approach for handling the paradigm of task uniqueness [5], multivariate and VRS [1]. [1] especially points out that "DEA is the only method complying with the two requirements (the multivariate inputs/outputs and VRS) that we consider crucial to perform correct performance assessment in software engineering." Moreover, DEA is appealing to software practitioners because it uses the best practice frontier as a benchmark rather than some theoretical baseline [1]. To date, DEA is gaining increasing interests in benchmarking research of the software process field [1][5]. But to the best of our knowledge, few research results have been reported publicly on benchmarking software development tasks which will be more beneficial to software developers than benchmarking software projects at the organization level.

In this paper, we propose a DEA-based software development tasks benchmarking method. Practical experiences on benchmarking software development tasks under multivariate and VRS constraints using DEA in ISCAS (Institute of Software, Chinese Academy of Sciences) are reported.

This paper is organized as follows. Section 2 describes the task data collection process and the data analysis tool of our study. Section 3 presents the details of the data analysis process and the empirical results. Section 4 shows the sensitivity analysis process. Section 5 summarizes our conclusions and points out the future work.

2 Data Collection and Analysis Tool

2.1 Data Collection

The organization (ISCAS) which we collect data from is one of the leading research and development organizations in China and has high software process capability maturity level (CMMI level 4). The work of the developers in ISCAS is entirely task-based and performed under strict quantitative CMMI/TSP/PSP-based processes management. In addition, the organization has developed a software process

management tool called SoftPM [10] to help software organizations and developers to record and collect the task, project and process data automatically. Moreover, the organization imposes strict task planning/reporting/auditing and defect reporting/tracing/fixing procedures with the aids of the tool. Thus, the data quality and relative completeness of the task/defect reports provided by the developers can be sufficiently guaranteed.

Firstly, we identify the input/output metrics of tasks by mining the task and defect reports. We extract task reports that at least satisfy the following rules:

1. The metrics can be automatically derived from developers'task and defect reports so as to fulfill the desire of comprehensive quantitative software process improvement.
2. The work products (e.g. Programs and Reports) are reported at the task level so as to enable the fine-grained performance evaluation.
3. Based on CMMI/TSP/PSP specification, tasks in ISCAS are divided into seven types, namely, Engineering, SPI, Management, QA, Review, Test and Self-Defined. Only tasks whose task type is Engineering are included in this study because most tasks of this type are software development tasks.
4. The output metrics must be related to the inputs so as to make reasonable and fair comparisons of the efficiency of the tasks.

We primarily use the above rules, the expert knowledge and the specification of task and defect reports to determine which input/output metrics to include in the model. The best subset regression analysis was also used to assist the expert in the selection process. We finally derive three outputs and one input (Table 1) for benchmarking software development tasks.

Secondly, we establish the task benchmarking data set by mining the task and defect report data and referring to the established evaluation metrics of tasks (Table 1). The established task data set consists of 30 completed software development tasks (Table 2). All the tasks are the engineering type, implement the same software process management package and all based on J2EE Web Applications, i.e. the task data set can be regarded as *homogenous*.

Thirdly, we transform the evaluation metrics which are undesirable inputs or outputs in DEA terminology. One metric which requires transforming in the established evaluation metrics (Table 1) is: *Program Defects*. Because an increase in an input should contribute to increased output and increasing the *Program Defects* is an undesirable output. There are several different methods for modeling undesirable outputs in DEA. The method used for this case study was

Table 1. Input and output evaluation metrics of tasks

Metric	Type	Meaning	Unit
Effort	Input	Actual effort of the task	Person Hour
Program Size	Output	Program size of the work product of the task	LOC
Program Defects	Output	Program defects found in test phase	Defects
Documents	Output	Documents specified for the task	Pages

Table 2. ISCAS Task evaluation data set

- Size: Program Size
- Defects: Program Defects

Task	Size	Defects	Documents	Effort	Task	Size	Defects	Documents	Effort
1	1579	12	6	7.5	16	620	3	7.5	6.5
2	1320	10	5	7	17	598	5	3	6.4
3	1202	8	7	7.5	18	568	3	5	5.5
4	1000	5	5	7	19	460	5	3	6.5
5	980	9	2	6.5	20	458	4	2.5	6
6	940	7	4	6	21	345	5	4	5.3
7	824	6	5	6.5	22	263	3	5	5.1
8	763	7	5	5	23	236	5	3	3
9	744	5	5.5	6	24	233	4	3	3.5
10	735	6	4	7	25	220	2	4	3
11	725	5	5.5	5.5	26	200	4	2.5	4
12	718	6	6	6	27	178	3	1	3
13	700	4	3	5.4	28	155	2	1.5	2
14	685	9	5	5	29	144	2	1	1.5
15	678	7	6.5	5.5	30	124	3	3	2

the [TRβ] [11] transformation, a common practice in the DEA literature. In the [TRβ] transformation, the undesirable output (μ_i) is subtracted from a significantly large scalar (β_i), such that all resulting (transformed) values (f_i^k) are positive and increasing values are desirable. The β_i chosen is generally a value just slightly larger than the maximum value of the undesirable output observed in the data set, since choosing a β_i value that is much greater than this maximum value can distort model results. In this application, the maximum *Program Defects* was 12 defects (see Table 2). Therefore, 14, which is slightly larger than the maximum *Program Defects* (12) by 2, was chosen as β_i. Next, all task *Program Defects* were subtracted from this β_i. Till now, the task evaluation data set has been established for further analysis.

2.2 Analysis Tool

To analyze the task evaluation dataset in Table 2, we use two classical DEA models (Table 3): the CCR model by Charnes, Cooper and Rhodes [12] and the BCC model by Banker, Charnes and Cooper [13].The CCR model's assumption is Constant Return to Scale (CRS) and the BCC model's assumption is Variable Return to Scale (VRS). CRS assumes a linear relationship between inputs and outputs [1] which is consistent with the productivity model:

$$p = \frac{y}{x}. \tag{1}$$

VRS assumes a nonlinear relationship between inputs and outputs which is consistent with cost estimation models like COCOMO. Those models generally have the following form (P = productivity, x = effort, y = FP or SLOC, B > 1):

$$x = \frac{1}{p} y^B. \tag{2}$$

Table 3. Analysis tools for hypothesises

(1) CCR Model	(2) BCC Model
min θ-$\varepsilon(\Sigma\ S_i^+ + \Sigma\ S_i^-)$	min θ-$\varepsilon(\Sigma\ S_i^+ + \Sigma\ S_i^-)$
s.t. θ_{ij_0}-$\Sigma x_{ij}\lambda_j - s_i^+ = 0, i = 1, .., m$	s.t. θ_{ij_0}-$\Sigma x_{ij}\lambda_j - s_i^+ = 0, i = 1, .., m$
$\sum y_{kj}\lambda_j$ - y_{kj_0}- $s_k^- = 0$, k = 1, .., s	$\sum y_{kj}\lambda_j$ - y_{kj_0}- $s_k^- = 0$, k = 1, .., s
	$\sum \lambda_j = 1$
$\lambda_j \geq 0, j = 1, .., n$	$\lambda_j \geq 0, j = 1, .., n$
$s_i^+ \geq 0$	$s_i^+ \geq 0$
$s_i^- \geq 0$	$s_i^- \geq 0$

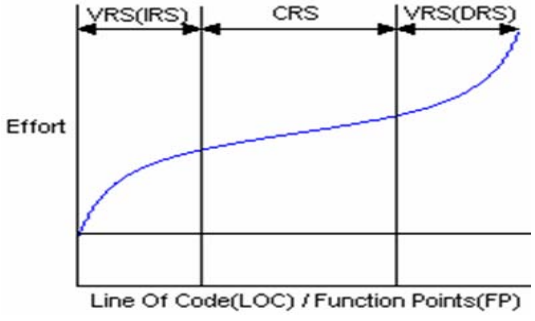


Fig. 1. CRS and VRS models

DRS (Decreasing Returns to Scale) and IRS (Increasing Returns to Scale) are two special cases of VRS. As CRS indicates the linear relationship between inputs and outputs, IRS (DRS) indicates that an increase in one unit's inputs will yield a greater (or less) proportionate increase of its outputs (Fig. 1).

3 Data Analysis and Empirical Results

In this section, we carefully discuss data analysis processes and the results for the two hypothesises described in Section 1.

3.1 Hypothesis 1: Can the Relatively Efficient Tasks Be Identified to Establish the Task Performance Benchmark Under Multivariate and VRS Constraints?

To investigate our hypothesis H1, we first use CCR and BCC models (Table 3) to quantitatively calculate the efficiency score (θ) of the tasks. The efficiency score (θ) of task is between 0 and 1. A task with efficiency score of 1 is relatively efficient. The efficiency scores obtained from these models are listed in Table 4.

From Table 1, we can see that the task performance evaluation process has multivariate inputs and outputs (one input like *Effort*; three outputs like *Programm Size, Programm Defects and Documents*). From Table 4, it can be further

Table 4. Efficiency scores of tasks obtained from BCC and CCR models

T_j	Size	Efficiency Score		T_j	Size	Efficiency Score	
		CCR	BCC			CCR	BCC
1	1579	**1.0000**	**1.0000**	16	620	0.8946	**1.0000**
2	1320	0.9186	0.9348	17	598	0.5529	0.5661
3	1202	0.9422	**1.0000**	18	568	0.7788	**1.0000**
4	1000	0.7953	**1.0000**	19	460	0.4704	0.4730
5	980	0.7561	0.7685	20	458	0.4918	0.5078
6	940	0.8217	0.8659	21	345	0.5923	0.5962
7	824	0.7626	0.8125	22	263	0.6826	0.7843
8	763	0.9593	0.9735	23	236	0.7648	0.7861
9	744	0.8375	0.9120	24	233	0.6637	0.6710
10	735	0.6104	0.6235	25	220	0.9368	**1.0000**
11	725	0.9046	0.9717	26	200	0.5128	0.5255
12	718	0.8727	0.9172	27	178	0.5298	0.5474
13	700	0.7323	0.9332	28	155	0.8475	0.8750
14	685	0.9182	0.9247	29	144	**1.0000**	**1.0000**
15	678	0.9836	**1.0000**	30	124	**1.0000**	**1.0000**

observed that the relative performance scores for each task have already been obtained under the above multivariate inputs/outputs constraints using DEA. These results reveal that the relatively efficient tasks can be identified under multivariate inputs/outputs using DEA.

From Table 4, it can be further observed that the CCR only puts T_1, T_{29} and T_{30} on the efficiency frontier. The BCC puts T_1, T_3, T_4, T_{15}, T_{16}, T_{18}, T_{25}, T_{29} and T_{30} on the efficiency frontier. By further comparison, we can find that the notable difference between the results of CCR and BCC lies in that T_3, T_4, T_{15}, T_{16}, T_{18} and T_{25} are positioned on the efficiency frontier in BCC while not recognized as the relatively efficient tasks in CCR. This result reveals that the BCC model seems to have a better capability to establish different performance benchmarks for tasks of different sizes.

For example, in the CCR model, developers can only set T_1, T_{29} and T_{30} as the task performance benchmarks. In the BCC model, developers can get more fine-grained efficiency scores (T_1, T_3, T_4, T_{15}, T_{16}, T_{18}, T_{25}, T_{29} and T_{30}) which enable developers to establish much fine-grained performance benchmarks for tasks of different sizes. To make it clearer, let us use T_{14} as an example. In BCC, developers can benchmark T_{14} on T_{15}. In CCR, developers can only benchmark T_{14} on T_1, T_{29} or T_{30}. The size difference between T_{14} and T_{15} is obviously much smaller than that of T_{14} between T_1, T_{29} or T_{30}. This result surely shows that the DEA VRS model (BCC) seems to be more appropriate to evaluate software development tasks with similar scale and ensure that relatively larger tasks are compared with other relatively larger tasks and relatively smaller tasks with relatively smaller tasks than CCR. Further, for our task data set (Table 2), by using the DEA VRS model, one task benchmarking solution may be that developers can identify $\{T_1, T_3, T_4\}$ as the performance benchmark for relatively

large tasks, {T_{15}, T_{16} ,T_{18}} for relatively middle-scale tasks, and {T_{29}, T_{30}} for relatively small tasks .

Furthermore, we calculate the average of the DEA VRS efficiency to derive quantitative improvement suggestions. The average VRS efficiency (E_{mean}), standard deviation (SD), minimum VRS efficiency (E_{min}) and the number of efficient tasks (N_{eff}) for our task data set (Table 2) are shown in Table 5.

Table 5. Average efficiency results of ISCAS task data set

	N	E_{mean}	**SD**	E_{min}	N_{eff}
VRS	30	0.8323	0.1813	0.4730	9

From a process improvement perspective, these average efficiency figures tell us that there is a potential for improvement of such tasks between 10 to 20 percent compared with the best practices tasks.

To sum up, from the above analysis results, we can conclude that although development tasks are much fine-grained compared with projects at an organization level and has multivariates and VRS properties, the relatively efficient tasks can be identified as the performance benchmark using DEA. Moreover, the DEA VRS model seems to be more appropriate for benchmarking software development tasks with the merits of dealing with multivariate and VRS properly and enabling developers to establish different task performance benchmarks for tasks of diffident scale. Moreover, at the aid of DEA VRS efficency, the potential of quantitative improvement of tasks can be further provided for developers.

The above results confirm our hypothesis H1, i.e.:

> The relatively efficient tasks can be identified as the task performance benchmark under multivariate and VRS constraints using DEA. We also find that DEA VRS model seems to be more appropriate for benchmarking software development tasks.

3.2 Hypothesis 2: Can Different Reference Sets for Each Relatively Inefficient Task Be Established Under Multivariate and VRS Constraints?

After the relatively efficient tasks have been identified given that H1 holds, tasks have thus been clearly classified into relatively efficient and inefficient ones. Surely developers can treat all the relatively efficient tasks as a whole as the performance benchmark for each inefficient task. As each identified efficient task will have different improvement value to the inefficient one, it is surely vital for developers to establish different reference sets for each different inefficient task.

By carefully investigating the analysis tools (Talbe 3), we define a reference set of the task T_j {j = 1, ..., n} as:

$$RS_j = \{T_{jr} : \lambda_{jr} \neq 0, r = 1, ..., n\}$$

It should be noted that each task T_{jr} in the reference set RS_j is relatively efficient. For convenience, we also call each efficient task T_{jr} in the reference set as a *peer* of the task T_j. The corresponding λ_{jr} (calculated from Table 3) of the peer T_{jr} is called *peer weight*. The *peer weight* indicates the importance of the peer T_{jr} to the given task T_j. Via the peers $\{T_{jr}\}$ and their weights $\{\lambda_{jr}\}$, developers can further determine which peer (efficient task) is of the biggest improvement value to the task T_j and thus need to be learned more from. Table 6 shows the reference relationships (the peer set and the peer weight) among tasks in Table 2.

For example, in Table 6, developers can find that T_7 derives a reference set of tasks $\{T_1, T_{30}\}$ in CCR result. By further investigating the peer weight of each peer in the above reference set $\{T_1, T_{30}\}$, developers can determine to choose T_{30} as the most suitable task to learn best practices from because T_{30} has the biggest peer weight (0.74) in the peer set $\{T_1, T_{30}\}$. Similarly, in BCC result, developers can determine to emulate best practices from T_{25} by comparing the peer weight among the reference set $\{ T_1, T_3, T_4, T_{25} \}$ of T_7. The reference set and the most valuable task to emulate for the other tasks in Table 6 can be derived in a similar way.

To sum up, by investigating the reference relationships using DEA, developers can establish different reference sets for each relatively inefficient task. Moreover, with the aid of peer weights of the peers, developers can further find which task is of the biggest improvement reference value to his own personal software process.

The above results confirm our hypothesis H2, i.e.:

Different reference sets for each relatively inefficient task to borrow best practices from can be established under multivariate and VRS constraints by investigating the task reference relationships using DEA.

4 Sensitivity Analysis

DEA identifies best practice rather than the average or say the best 10 percent, which makes the techniques very sensitive to extreme observations. It is therefore necessary to do a sensitivity analysis of outliers. There are several techniques (e.g., superefficiency and analysis of reference units) each with their strengths and limitations depending on the purpose of the DEA analysis. The purpose of our DEA-based task analysis is twofold: first to identify best practice tasks as well as the reference tasks for individual tasks and second, to determine the average efficiency of the software development tasks to quantify the overall potential for performance improvement. Based on these two purposes, the simplest and probably most reasonable sensitivity analysis is to remove all the frontier tasks one by one and study the effect on the mean efficiency [1].

Our task data set has nine tasks $\{T_1, T_3, T_4, T_{15}, T_{16}, T_{18}, T_{25}, T_{29}, T_{30}\}$ (see Table 4) on the VRS frontier. We do sensitivity analysis by removing each of the nine tasks one at a time. We then compare E_{mean} in Table 5 and Table 7.

We observe that none of the frontier tasks are extreme in the sense that their removal hardly influence the average efficiency. i.e., there is still a potential

Table 6. Reference relationships of ISCAS task data set

- ES: Efficiency Score;
- P:Peer;
- PW: Peer Weight

	CCR			BCC				CCR			BCC		
T_j	ES	P	PW	ES	P	PW	T_j	ES	P	PW	ES	P	PW
1	1.0000	1	1.0000	1.0000	1	1.0000	14	0.9182	1	0.3594	0.9247	1	0.2494
2	0.9186	1	0.8180	0.9348	1	0.7803			30	0.9479		15	0.3577
		29	0.1970		4	0.0658						30	0.3929
					29	0.1539	15	0.9836	1	0.3075	1.0000	15	1.0000
3	0.9422	1	0.6857	1.0000	1	1.0000			30	1.5526			
		30	0.9619				16	0.8946	1	0.2329	1.0000	16	1.0000
4	0.7953	1	0.5758	1.0000	4	1.0000			30	2.0342			
		29	0.2620				17	0.5529	1	0.3161	0.5661	1	0.2910
		30	0.4277						29	0.5187		25	0.1417
5	0.7561	1	0.5916	0.7685	1	0.5826			30	0.1949		29	0.5373
		29	0.3181		29	0.4174	18	0.7788	1	0.2715	1.0000	18	1.0000
6	0.8217	1	0.5510	0.7685	1	0.5826			30	1.1237			
		29	0.4024		29	0.4174	19	0.4704	1	0.2286	0.4730	1	0.2263
		30	0.0972						29	0.3087		29	0.3394
7	0.7626	1	0.4638	0.8125	1	0.3125			30	0.4398		30	0.4343
		30	0.7390		3	0.0833	20	0.4918	1	0.2183	0.5078	1	0.1881
					4	0.1249			29	0.6240		4	0.0396
					25	0.4792			30	0.1887		25	0.1337
8	0.9593	1	0.4180	0.9735	1	0.3469						29	0.6386
		30	0.8307		15	0.1934	21	0.5923	1	0.1350	0.5962	1	0.0640
					25	0.2825			30	1.0624		15	0.2309
					30	0.1773						30	0.7051
9	0.8375	1	0.3882	0.9120	3	0.4470	22	0.6826	1	0.0423	0.7843	16	0.2857
		30	1.0570		4	0.1005			30	1.5820		25	0.7143
					16	0.0167	23	0.7648	1	0.0841	0.7861	1	0.0754
					25	0.4358			30	0.8317		29	0.1131
10	0.6104	1	0.4123	0.6235	1	0.3959						30	0.8114
		29	0.1917		25	0.3127	24	0.6637	1	0.0769	0.6710	1	0.0734
		30	0.4430		29	0.2501			29	0.0646		29	0.1101
					30	0.0414			30	0.8246		30	0.8165
11	0.9046	1	0.3739	0.9717	3	0.4775	25	0.9368	1	0.0411	1.0000	25	1.0000
		30	1.0855		4	0.0426			30	1.2512			
					16	0.0071	26	0.5128	1	0.0548	0.5255	1	0.0478
					25	0.4728			29	0.2315		29	0.3217
12	0.8727	1	0.3531	0.9172	1	0.2012			30	0.6466		30	0.6305
		30	1.2938		15	0.2436	27	0.5298	1	0.0296	0.5474	1	0.0237
					16	0.2824			29	0.9117		29	0.9763
					25	0.2727	28	0.8475	1	0.0082	0.8750	25	0.1667
13	0.7323	1	0.3730	0.9332	1	0.0103			29	0.7997		29	0.0833
		29	0.7712		4	0.6322			30	0.2171			
					29	0.3575	29	1.0000	29	1.0000	1.0000	29	1.0000
							30	1.0000	30	1.0000	1.0000	30	1.0000

Table 7. Results of sensitive analysis of ISCAS task data set

-Task: The removed task
-E_{mean}: Mean of E_{VRS}

Task	E_{mean}
1	0.8408
3	0.8266
4	0.8303
15	0.8289
16	0.8370
18	0.8266
25	0.8327
29	0.8444
30	0.8364

improvement of around 20 percent (see Table 7). This result verified that our DEA-based evaluation (Table 4, Table 6) of ISCAS software development tasks (Table 2) are reasonable.

5 Conclusions and Future Work

This empirical study focuses on the question of "Can we benchmarking software development tasks under multivariate and VRS constraints? If so, how to?".

The analysis of experience data within Institute of Software, Chinese Academy of Sciences (ISCAS) using Data Envelopment Analysis (DEA) indicates that the ideas and techniques of benchmarking software projects, which is more beneficial to organization's software process improvement, can be deployed at the software development task level, which is more beneficial to developers' personal software process improvement. And we find that our DEA-based approach is helpful to benchmark software development tasks under multivariate and VRS constraints. Moreover, results also reveal that the DEA VRS model allows developers to gain new insight about how to identify the relatively efficient tasks as task performance benchmark and how to establish different reference sets for each relatively inefficient task under multivariate and VRS constraints. We thus recommend DEA VRS model be used as the default technique for appropriately benchmarking software development tasks. Our results are beneficial to total quantitative software process improvement (especially, PSP improvement). To the best of our knowledge, we believe that it is the first time to report such comprehensive and repeatable results of benchmarking software development tasks using DEA.

Our future work will concentrate on the following topics. Firstly, metrics of software development tasks can be further extended based on the changing and different requirements of software organizations. We also plan to further evaluate the software development tasks using DEA to analyze the developer's personal software process change effects. Secondly, it is useful to investigate more in depth

DEA versus other benchmarking models (e.g. regression analysis) on software development tasks. Thirdly, we are developing an interactive and visual tool to provide more supports for benchmarking software tasks using DEA.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082, 60473060, 60673121; the National Hi-Tech Research and Development Plan of China under Grant No. 2006AA01Z185; the National Key Technologies R&D Program under Grant No. 2005BA113A01. One of the authors, Yun Yang, gratefully acknowledges the support of K. C. Wong Education Foundation, Hong Kong.

References

1. Stensrud, E., Myrtveit, I.: Identifying High Performance ERP Projects. IEEE Transactions on Software Engineering **29**(5) (2003) 398–416
2. Humphrey, W.S.: Introduction to the Team Software Process. Addison Wesley Professional (1999)
3. Stark, J.A., Crocker, R.: Trends in Software Process: The PSP and Agile Methods. IEEE Software **20**(3) (2003) 89–91
4. Katrina D. Maxwell, P.F.: Benchmarking Software Development Productivity. IEEE Software **17**(1) (2000) 80–88
5. Farris, J.A., G.R.V.A.E.L.G.: Evaluating the Relative Performance of Engineering Design Projects: A Case Study Using Data Envelopment Analysis. IEEE Transactions on Engineering Management **53**(3) (2006) 471– 482
6. ISBSG: Worldwide Software Developmentthe Benchmark. Technical Report Release 5, International Software Benchmarking Standards Group (1998)
7. Myrtveit, I., Stensrud, E.: Benchmarking COTS Projects Using Data Envelopment Analysis. In: Sixth International Software Metrics Symposium. (1999) 269–278
8. Malaiya, Y.K., Denton, J.: Module Size Distribution and Defect Density. Software Reliability Engineering (2000) 52–71
9. Institute, P.M.: A Practice Standard for Earned Value Management, New Square.PA (2005)
10. Qing, W., Li, M.S.: Measuring and Improving Software Process in China. In: Proceedings of the 4th International Symposium on Empirical Software Engineering (ISESE'05, Australia (November 2005) 17–18
11. H., S.: Undesirable Outputs in Efficiency Valuations. European Journal of Operation Reseach **132**(2) (2001) 400–410
12. Charnes, A., Cooper, W.W., Rhodes, E.: Measuring the Efficiency of Decision Making Units. European Journal of Operation Research **2** (1978) 429–444
13. Banker, R.D., Charnes, A., Cooper, W.: Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment. Management Science **30** (1984) 1078–1092

An Empirical Study on Establishing Quantitative Management Model for Testing Process

Qing Wang [1], Lang Gou [1, 2], Nan Jiang [1, 2], Meiru Che [1, 2], Ronghui Zhang [1, 2], Yun Yang [1, 3], and Mingshu Li [1]

[1] Institute of Software, Chinese Academy of Sciences
[2] Graduate University of Chinese Academy of Sciences
{wq,goulang,jiangnan,chemeiru,zhangronghui, mingshu}@itechs.iscas.ac.cn
[3] CITR, Swinburne University of Technology, Australia
yyang@ict.swin.edu.au

Abstract. Frequently, effort of defect detecting and fixing are counted into software testing activities/phase. Current leading software estimation methods, such as COCOMO II, mainly estimate the effort depending on the size of software product and allocate testing effort proportionally. It can not predict detecting and fixing effort accurately. In fact, testing effort is significantly influenced by the quality of other software development activities. These lead to the difficulty of the testing effort to be estimated accurately. It is a challenging issue for quantitative software process management. In this paper, we propose an empirical method to identify performance objectives, establish performance baseline and establish quantitative management model for testing process. The method has been successfully applied to a software organization for their quantitative management of testing process.

Keywords: Software measurement, Quantitative process management, Testing process, Process performance baseline.

1 Introduction

Testing is an important method for quality control. It is also an important process that needs to be managed quantitatively for high maturity organizations. However, quantitative management model of testing is complex because it is constrained not only by the size of product, but also by the quality of implementation activities, such as design and coding. The more defects, the more effort is needed to fix and verify them. How to estimate the effort and the defects related data, establish performance baseline and quantitative management model of testing process is challenge. In fact, many software projects delay due to the slippage of the testing activities.

Many estimation methods focus on estimating or predicting the effort and defect separately. For example, COCOMO II [1] is a famous cost estimation method with a family of extension models respecting to different types of development needs. COQUALMO [1] is one of these models which can be used to estimate quality of software product in terms of defect density. But it does not consider the interrelationship between defect and effort. The testing effort comes mainly from the general percentage of the total estimated effort.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 233–245, 2007.

As we know, even though there are many verification activities during the software development lifecycle, testing is still the most common and important method to detect and fix defects. The defects detected in testing are injected not only from coding, but also from requirements analysis and software design. In this paper, we propose an empirical method of establishing quantitative management model for testing process. Based on the method, the performance objectives of testing process are identified. Then some statistical techniques are used to analyze the data related to these objectives and some interesting empirical results are found. The method has been successfully applied to a software organization and appears very useful in helping software organizations quantitatively manage testing process.

Institute of Software, Chinese Academy of Sciences (ISCAS) is a research and development organization in China which is appraised and rated at CMMI maturity level 4. ISCAS developed a toolkit called SoftPM [2][12] which is used to manage software project and has been deployed to many software organizations in China. The data used in this paper comes from 16 projects based on facilitating SoftPM.

In this paper, the empirical method of establishing quantitative management model for testing process is presented in Section 2. The application of establishing quantitative management model for testing process is discussed in Section 3. Section 4 presents how to quantitatively manage testing process in a software organization based on the method. Related work is discussed in Section 5. Section 6 summarizes our conclusions and points out future work.

2 Empirical Method

This section will present our empirical method of establishing quantitative management model for testing process. The three steps of the method are to: (1) identify the performance objectives (P-Objs) to be managed quantitatively and construct data samples; (2) establish the process performance baseline (P-BL) for the identified P-Objs; and (3) analyze the correlations between the identified P-Objs.

2.1 Identify P-Objs and Construct Data Samples

Normally, effort of defect detecting/fixing and defect injected phase are sensitive data that we should consider for testing process. A general assumption is that the effort of defect detecting and fixing should consume a certain percentage in total development effort, and the effort of defect fixing is influenced by the defect number and the defect injected phase. In our method, three P-Objs have been identified including:

(1) Percentage of Detecting Effort (PDE): Detecting effort means the effort for all detecting activities including test planning, test case preparing, test implementation and fix verifying. PDE is the percentage of the detecting effort in the total effort.

(2) Defect Injection Distribution (DID): In general, many software organizations collect defect data for quality control. There are always some defects injected in early phases which are only detected until the testing activities even in the high maturity organizations. In our method, three primary phases, namely requirements, design and coding, are used to classify the corresponding injected phases for each defect. The corresponding percentages of defects injected in the three phases are denoted as

DID_R, DID_D and DID_C respectively. The principles of assigning the injected phase are described as: a) defect injected in the requirements phase: a defect that is due to poor requirements, such as inconsistency and unclear requirements; b) defect injected in the design phase: a defect that is due to poor design, such as unclear interface, misunderstanding of requirements and incomplete data verification; and c) defect injected in the coding phase: a defect that is due to poor coding, such as incorrect words in a Web page and inconsistent code against requirements or design.

(3) Percentage of Fixing Effort (PFE): Fixing effort data means the effort for all defect fixing activities including defect analysis and fixing. PFE is the percentage of the fixing effort in the total effort.

2.2 Establish P-BL of Identified P-Objs

P-BL is the basis for quantitative process management. It is established based on the statistical analysis of historical data. There are many methods and techniques, such as BSR (Baseline-Statistic-Refinement) [4] and SPC (Statistical Process Control) [6][7] which can be used to establish P-BL.

Defect fixing is an important activity of software development which demands certain effort. In International Software Benchmark Standard Group (ISBSG) (www.isbsg.org), the fixing effort is collected and counted in rework effort. However, many effort estimation methods do not pay sufficient attention to the effort of defect fixing; instead, just include it in the testing activities. Frequently, defect detecting is performed by test team, and defect fixing is performed by development team. Estimating their effort separately is helpful for organization to plan their human resource and schedule. In addition, the fixing effort is strongly correlated with the number and injected phase of defects. Splitting them and establishing their P-BLs are very useful to manage testing process quantitatively.

For high maturity software organizations, the defect related process performance, such as defect injection, defect removal, and defect density, also has some common and stable properties. Many methods discuss the defect removal ratio and defect density. These are very useful and easy to understand. Here we focus on the defect injection and the correlation between the defects and effort needed to fix them.

2.3 Analyze Correlation Between P-Objs

In the testing activities, it is the common knowledge that the earlier a defect is injected, the more effort is needed to fix it. In contrast, the later a defect is injected, the less effort is needed to fix it. So, defects injected in an earlier phase, such as the requirements phase, have the effort of increasing the defect fixing effort, whereas, defects injected in a later phase, such as the coding phase, have the effort of decreasing the defect fixing effort.

After constructing defect related data samples, software organizations can discover some more precise correlation between defects and fixing effort. Our method is based on this hypothesis. There are some statistical methods which can be used to analyze the correlation between DID and PFE, such as multiple regression analysis [11]. After the correlation between DID and PFE has been analyzed, the regression equation between DID and PFE can be used to refine the estimation of fixing effort after

testing. The outcome can provide a guideline to estimate the effort of defect fixing based on the defects and the distribution of injection phases. So, after testing, the project managers could re-estimate and re-plan their fixing effort effectively. The factors of regression equation could be refined and calibrated based on the historical data of software organizations. Then it can be more applicable in these organizations.

3 Establish Quantitative Management Model for Testing Process

Based on the empirical method presented in Section 2, we collected testing process data from 16 Web-based system development projects and concluded some empirical results. These 16 projects came from two closely-related software organization entities. The two entities have the self-governed process management system and were rated at CMMI maturity level 3 and moving towards CMMI maturity level 4 in the period of our data collection.

3.1 Data Sample

Web-based development techniques have been widely applied in China. The projects' characteristics are described in Table 1.

Table 1. Features of Web-based system development projects

Category	Features
Architecture	Browser / Server
Development process	Iterate development and testing; Use prototype and documents to confirm requirements; Refine through minor release software
Development cycle	Less than one year
Quality goal	Business software with high quality
Process maturity	Upper CMMI maturity level 3.

All the 16 projects were successful projects with little schedule overruns, and they all performed the requirements, design, coding and testing processes. Table 2 summarizes the brief information about the projects.

We collected the PDE, DID and PFE data from the 16 projects as presented in Section 2.1. These data were reported by engineers and were collected in SoftPM [2][12]. Table 3 shows the total effort, detecting effort, and PDE of the 16 projects. Unfortunately, the detecting effort for individual requirements, design and coding phases were not recorded. Hence we could only collect the total detecting effort for the projects.

For the 16 projects, all the defects considered were detected in the testing activities. These defects were classified into four categories: critical defects, serious defects, non-critical defects and cosmetic defects. In this paper, we only describe the total defects collected without distinguishing them. Table 4 shows the defects injected in three primary phases of the 16 projects.

Table 2. Brief information about the 16 Web-based system development projects

Proj.	# of staff	Schedule (Months)	Size (KLOC)	Application domain	Process performance
1	12	12	151.9	Application system integration	
2	15	7	173.1	Tool development	
3	5	6	18.2	Tool development	
4	14	7.5	311.2	Application system integration	Successful projects with little schedule overruns.
5	6	4	76.9	Website design and development	
6	3	3	45.9	Website design and development	
7	6	2	17.0	Information management	
8	14	9	280.4	Tool development	
9	4	5.5	45.0	Tool development	All performed requirements, design, coding and testing processes
10	12	7	55.5	Information management	
11	9	5	60.7	Tool development	
12	4	2	19.6	Information management	
13	11	9	90.4	Application system integration	
14	12	4.5	250.5	Application system integration	
15	5	6	80.0	Information management	
16	4	7	30.0	Tool development	

Table 3. Total effort (Labor Hour), detecting effort (Labor Hour) and PDE of the 16 projects

Proj.	Total effort	Detecting effort	PDE (%)	Proj.	Total effort	Detecting effort	PDE (%)
1	7048	1396	19.8%	9	3397	693	20.4%
2	11614	3734	32.2%	10	7114	1070	15.0%
3	3143	785	25.0%	11	6864	1684	24.5%
4	11177	3624	32.4%	12	1205	265	22.0%
5	2926	609	20.8%	13	14683	2684	18.3%
6	1313	182	13.9%	14	6579	2117	32.2%
7	1560	354	22.7%	15	4230	940	22.2%
8	10865	2566	23.6%	16	1934	401	20.7%

Table 4. Defects injected in each phase of the 16 projects

Proj.	Requirements		Design		Coding	
	# of defects	DID_R	# of defects	DID_D	# of defects	DID_C
1	19	10.1%	41	21.8%	128	68.1%
2	118	15.6%	153	20.3%	483	64.1%
3	33	17.8%	61	33.0%	91	49.2%
4	251	18.7%	412	30.6%	682	50.7%
5	27	20.5%	44	33.3%	61	46.2%
6	17	13.3%	35	27.3%	76	59.4%
7	15	15.6%	28	29.2%	53	55.2%
8	135	14.1%	322	33.6%	501	52.3%
9	32	12.2%	82	31.2%	149	56.7%
10	15	13.9%	29	26.9%	64	59.3%
11	92	18.3%	116	23.1%	295	58.6%
12	12	14.0%	26	30.2%	48	55.8%
13	18	11.8%	36	23.5%	99	64.7%
14	53	11.0%	142	29.5%	286	59.5%
15	78	17.6%	114	25.7%	251	56.7%
16	20	13.9%	42	29.2%	82	56.9%

Table 5 shows the total effort, fixing effort, and PFE of the 16 projects.

Table 5. Total effort (Labor Hour), fixing effort (Labor Hour) and PFE of the 16 projects

Proj.	Total effort	Fixing effort	PFE (%)	Proj.	Total effort	Fixing effort	PFE(%)
1	7048	762	10.8%	9	3397	420	12.4%
2	11614	2370	20.4%	10	7114	1403	19.7%
3	3143	632	20.1%	11	6864	1325	19.3%
4	11177	2839	25.4%	12	1205	185	15.4%
5	2926	536	18.3%	13	14683	2214	15.1%
6	1313	108	8.2%	14	6579	824	12.5%
7	1560	245	15.7%	15	4230	802	19.0%
8	10865	1521	14.0%	16	1934	264	13.7%

3.2 P-BL of Identified P-Objs

First, we analyze the PDE data in Table 3. The XmR (individuals and moving range) control chart [6] is applied. Assume that the sequence of data sample is X_i, the moving range (mR) is:

$$mR_i = |X_i - X_{i-1}| \quad \text{i =2…n}$$

According to the theory of statistics, we can get the upper control limit (UCL), central line (CL), and lower control limit (LCL) for mR-chart and X-chart as follows:

$$UCL_{mR} = 3.268\overline{mR},\ CL_{mR} = \overline{mR},\ LCL_{mR} = 0$$

$$UCL_x = \overline{X} + 2.660\overline{mR},\ CL_x = \overline{X},\ LCL_x = \overline{X} - 2.660\overline{mR}$$

The XmR chart control limits for PDE data are shown in Table 6. We construct the XmR chart in Fig. 1 by using the PDE data in Table 3 and control limits in Table 6. As shown in Fig. 1, for both the mR-chart and X-chart, all data points distribute between the upper control limit and the lower control limit. The process appears stable. The CL_x (22.9%) can be considered as the P-BL of PDE to be used to estimate the effort of defect detecting and schedule of testing process during project planning.

Then, we analyze the DID data in Table 4. Similarly, we use the XmR control chart to analyze the distribution of DIDs. Table 7 shows the XmR chart control limits and Fig. 2 shows the XmR control charts for DID_R, DID_D and DID_C. For the three

Table 6. XmR chart control limits for PDE data

UCL_{mR}	CL_{mR}	LCL_{mR}	UCL_x	CL_x	LCL_x
22.9%	7.0%	0	41.5%	22.9%	4.2%

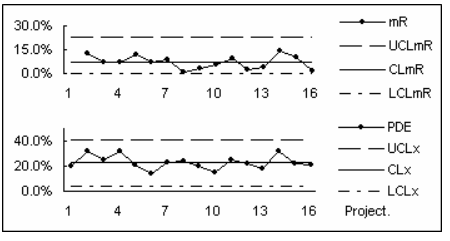
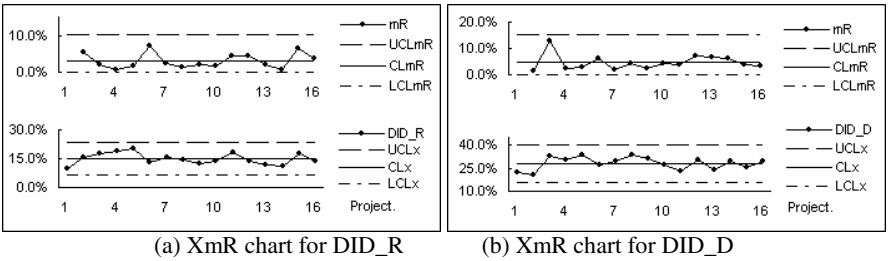


Fig. 1. XmR chart for PDE data of the 16 projects

XmR charts in Fig. 2, all data points distribute between the upper control limit and the lower control limit in both mR-chart and X-chart. Hence, the DID_R, DID_D and DID_C were converged and the distribution of defect injection appears stable. Therefore, the 14.9%, 28.0%, 57.1% can be accepted as the P-BLs of DID_R, DID_D and DID_C respectively to be used to estimate the distribution of defect injection.

Table 7. XmR chart control limits for DID data

DID	UCL_{mR}	CL_{mR}	LCL_{mR}	UCLx	CLx	LCLx
DID_R	10.2%	3.1%	0	23.2%	14.9%	6.6%
DID_D	15.1%	4.6%	0	40.3%	28.0%	15.8%
DID_C	15.9%	4.9%	0	70.0%	57.1%	44.2%



(a) XmR chart for DID_R (b) XmR chart for DID_D

(c) XmR chart for DID_C

Fig. 2. XmR charts for DID_R, DID_D and DID_C data of the 16 projects

Table 8. XmR chart control limits for PFE data

UCL_{mR}	CL_{mR}	LCL_{mR}	UCLx	CLx	LCLx
15.1%	4.6%	0	28.6%	16.2%	3.9%

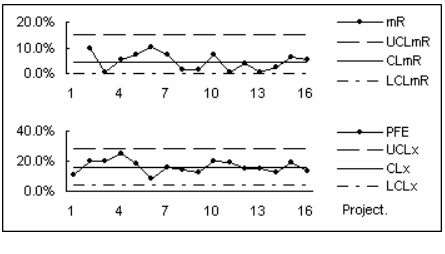


Fig. 3. XmR chart for PFE data of the 16 projects

Finally, we analyze the PFE data. Similarly, we use the XmR control chart. Table 8 shows the XmR chart control limits and Fig. 3 shows the XmR control chart for PFE. As shown in Fig. 3, the PFE also appears stable and converged. In this case, the CL_x (16.2%) can be treated as the P-BL of PFE to be used to estimate the effort of defect fixing and schedule of testing process during project initial planning.

3.3 Correlation Between P-Objs

As mentioned in Section 2.3, DID will influence the PFE. In this section, we analyze the correlation between PFE and DID_R/DID_C. PFE and DID_D are uncorrelated. Fig. 4 is the scatter diagram of DID_R, DID_C and PFE data based on Table 4 and Table 5. In Fig. 4, as expected, PFE increased with DID_R, which means that DID_R and PFE have positive correlation; and PFE decreased with DID_C, which means that DID_C and PFE have negative correlation.

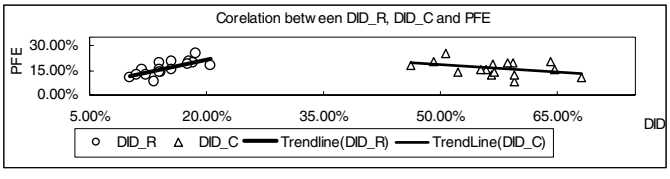


Fig. 4. Correlation between DID and PFE

In detail, we analyze the multiple correlations between DID_R, DID_C and PFE by using multiple linear regression. Let X_R, X_C, Y denote the data set on DID_R, DID_C and PFE of the 16 projects based on Table 4 and Table 5 respectively. By performing linear regression on independent variables X_R, X_C and dependent variable Y using Matlab 6.1 (http://www.mathworks.com), we first derive the binary linear regression equation as follows:

$$Y = -0.1597 + 1.3712 * X_R + 0.2065 * X_C$$

Then, an F test [11] is performed. As calculated by Matlab 6.1, we get the F statistic F = 9.5484. Let *n* denotes the number of data points which is equal to 16, and *k* denotes the number of independent variables which is equal to 2. At the confidence level α=0.05, the critical value of $F_{\alpha=0.05}(k, n-k-1) = F_{\alpha=0.05}(2, 13) = 3.81$. It is clear that $F_{\alpha=0.05}$ (2, 13) < F. Therefore, the correlation between DID_R, DID_D and PFE is linearly prominent. The regression equation between DID_R, DID_C and PFE can be used to adjust the estimation of defect fixing effort after testing.

4 Manage Testing Process Quantitatively

We applied our empirical method on an ongoing project of the organization to estimate, plan and manage its testing process quantitatively. The P-BLs and correlation established above plus some other baselines to compose the quantitative management model of the organization process management system. The steps of applying the quantitative management model for testing process are: (1) based on the

P-BL of the P-Objs, estimating the defect detecting effort, defect fixing effort and number of defects injected in each phases during the project planning; (2) through the testing activities, collecting the defect related data and re-estimating the effort of defect fixing when the actual P-Objs has abnormality.

4.1 Initial Estimation

As mentioned earlier, the organization was rated at CMMI maturity level 3 and was moving to CMMI maturity level 4. It had some P-BLs in place, such as detected defect ratio and software productivity. The detected defect ratio refers to (defects / code-size) where the defects are detected in testing activities (except the unit testing). The indicator of detected defect ratio is used to control the quality of software before submitted for testing. The software productivity is the mean productivity (total-size / total-effort) which can be used to estimate the total effort. Besides these, we added the empirical results presented in Section 3 to optimize project management. The new extended P-BL of Web-based system development projects in the organization is shown in Table 9 with some new quantitatively control objectives defined. An ongoing Web-based system development project (project No.17) was selected in the organization. Table 10 summarizes the brief information about the project.

Table 9. Extended P-BL of Web-based system development projects

Detected defect ratio - DDR	Software productivity - Prod	PDE	DID_R, DID_D, DID_C	PFE
4.01 Defects/KLOC	2.3 KLOC/Labor Month	22.9%	14.9%, 28.0%, 57.1%	16.2%

Table 10. Brief information about project No.17

# of staff	Plan schedule (Months)	Plan size	Development approach
8	4	67 KLOC	Increment and Iteration

Project No.17 was planned to complete the whole software product through two iterations. Each iteration implemented half of the product functions. Before the first iteration started, the project manager and skilled engineers estimated the sizes of both

Table 11. Estimation for each iteration of project No.17

Estimation	1st Iteration	2nd Iteration
Size (KLOC)	30	37
Schedule (Months)	2	2.4
Total defects detected in testing activities (Size*PRDE)	120	148
Defects injected in requirements (Total defects*DID_R)	18	22
Defects injected in design (Total defects*DID_D)	34	42
Defects injected in coding (Total defects*DID_C)	68	84
Total effort (Labor Month) (Size/Prod)	13.0	16.1
Detecting effort (Labor Month) (Total effort*PDE)	3.0	3.7
Fixing effort (Labor Month) (Total effort*PFE)	2.1	2.6
Development effort (Labor Month) (Total effort*(1-PDE-PFE))	7.9	9.8

iterations. Then, the total defects detected in the testing activities were estimated by using formula: Size*DDR; the total effort was estimated by using formula: Size/Prod. After that, the estimation for both iterations could be elaborated further, as shown in Table 11. Based on the estimation, project manager of project No.17 established a project plan of both iterations and performed against it.

4.2 Tracking and Re-estimation

During the testing activities of the first iteration, the defects injected in the requirements, design, and coding phases were 46, 30 and 62 respectively. Correspondingly, DID_R (X_R), DID_D and DID_C (X_C) were 33.3%, 21.8% and 44.9%. Compared to the P-BL in Table 9 and the control limits in Table 7, DID_R was higher, which means more defects were injected in the requirements phase. Given this abnormality, the project manager did some further analysis. As mentioned earlier, the defect fixing effort should be greater due to larger number of defects injected in the requirements phase. The PFE (Y) was re-estimated based on the regression equation ($Y = -0.1597 + 1.3712 * X_R + 0.2065 * X_C$). The new PFE was 38.7%. The re-estimated fixing effort was extended from 2.1 labor months to 6.9 labor months. With the effort increasing, the schedule of the first iteration had to be delayed by 10 work days and one engineer was added. After the first iteration, the actual data were collected in Table 12.

Since the actual DID of the first iteration was higher, the processes of requirements development, requirements management, especially the requirements review may have some problems. In reality, there could be many possible causes leading to the

Table 12. Actual performance data of the project No.17

Actual performance data	1st Iteration	2nd Iteration
Size (KLOC)	28	34
Schedule (Months)	2.5	1.9
Total defects detected in testing activities	138	132
Defects injected in requirements	46	18
Defects injected in design	30	40
Defects injected in coding	62	74
Total effort (Labor Month)	20	14
Detecting effort (Labor Month)	4.4	3
Fixing effort (Labor Month)	7.1	2.5
Development effort(Labor Month)	8.5	8.5

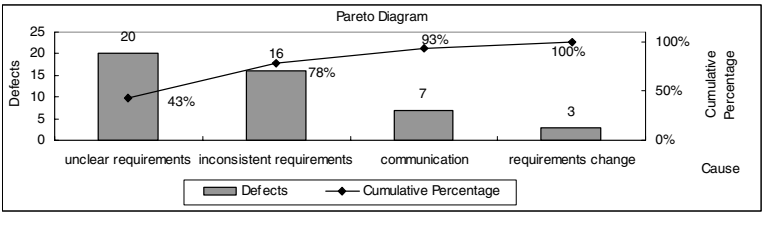


Fig. 5. Causal analysis of poor requirements phase

poor quality of the requirements phase. We analyzed all 46 defects injected in the requirements phase, and used a Pareto diagram to rate the major causes, as shown in Fig. 5. In Fig. 5, almost 80% of the 46 defects were due to the first two causes: unclear requirements and inconsistent requirements. Based on the causal analysis, the organization improved the requirements review process.

During the second iteration, the defects related data were collected (Table 12). The defects injected in the requirements, design and coding phases were similar to the estimation. Therefore, we did not need to re-estimate the effort of defect fixing. The second iteration was completed on time according to the initial schedule, and the actual performance (Table 12) was similar to the estimation (Table 11). Hence, the testing process of the second iteration is normal and stable.

In the future, when the organization has more project data of testing process, they can refine the P-BLs for PDE, DID and PFE. The experience from this case study validates that the empirical method presented in Section 2 is helpful for improving quantitatively managing testing process. And the empirical method can be used in initial estimation, tracking the process performance, identifying abnormality of process, analyzing the causes, re-estimating the fixing effort and improving the process to keep it controllable.

5 Related Work

COCOMO (COnstructive COst MOdel) II [1] is a widely-used estimation model, which allows one to estimate the total effort of a project depending on the estimated size. It provides two sets of empirical results on effort distribution for both waterfall and RUP lifecycle phases, which can be used to estimate effort of each phase including testing activities proportionally. COCOMO II can not predict the effort of defect detecting and fixing accurately. COQUALMO (COnstructive QUALity MOdel) [1] is a quality model extension to COCOMO II. It is used to estimate defects injected in different activities, and defects removed by defect removal activities. COQUALMO does not associate the defects with the effort of defect fixing.

Software Productivity Research (SPR) [3] (http://www.spr.com) is a provider of consulting services to help companies manage software development processes. SPR collected data from about 9,000 projects and reported the percentages of testing effort for system software, military software, commercial software, MIS and outsourcing software respectively. Osamu Mizuno et al. [13] develop a linear multiple regression model of estimating the testing effort. In the model, the testing effort can be obtained from the design effort and review effort, and also influenced by historical data factors. Both of them do not distinguish the effort of detecting from the effort of fixing in the testing activities.

The Rayleigh model [5][8][9] is based on Weibull's statistical distribution. Supported by a large body of empirical data, it is found that the defect detecting or removal patterns follow Rayleigh's curve. In this way, the Rayleigh model can be used for predicting the potential software defects [10]. It can be concluded from the Rayleigh model that there are some defects injected in early phases left to later phases such as the testing activities.

The related work above shows that the defects related data have been paid much attention by both academia and industry. In addition, there are much research on defect distribution and testing effort. Unfortunately, the above methods do not distinguish the effort of defect detecting from the effort of defect fixing. They also do not present mechanisms to adjust the effort of defect fixing based on the defect distribution. In this paper, we focus on identifying more performance objectives to indicate the relationship between the effort and defects, which is valuable for quantitative testing process management.

6 Conclusions and Future Work

In this paper, we propose an empirical method of identifying performance objectives (P-Objs), establishing performance baseline (P-BL) and establishing quantitative management model for testing process. From the empirical study, we find that the defect injection distribution (DID) of the requirements, design and coding phases have common and stable properties for high maturity software organizations. In addition, the percentages of the detecting effort (PDE) and fixing effort (PFE) are also similar. With the analysis of multiple regression, some correlations emerge between the effort of defect fixing and the defect injection distribution. Based on the method, a software organization established quantitative management model for testing process, and quantitatively controlled an ongoing project. Through the application, we can conclude that the empirical method is effective in quantitatively managing testing process. The method also provides helpful insights for project managers to make the detailed estimation for testing process, such as the distribution of defect injection, the effort for detecting and fixing defects.

As future work, the empirical method addressed in the paper can be refined with more studies and practices in different application domains. Some other factors should be considered. For example, the differences in project size, personnel capability and project type may affect the P-BLs of PDE, DID and PFE.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082, 60473060; the National Hi-Tech Research and Development Plan of China under Grant No. 2006AA01Z182; the National Key Technologies R&D Program under Grant No. 2005BA113A01. One of the authors, Yun Yang, gratefully acknowledges the support of K. C. Wong Education Foundation, Hong Kong.

References

1. Boehm, B.W., Horowitz, E., Madachy, R., Reifer, D., Clark, B.K., Steece, B., Brown, A.W., Chulani, S., Abts, C.: Software Cost Estimation with COCOMO II. Prentice Hall PTR (2000)
2. Wang, Q., Li, M.: Measuring and Improving Software Process in China. Proceedings of the 4th International Symposium on Empirical Software Engineering, Australia (2005) 183-192

3. Jones, C.: Software Assessments, Benchmarks, and Best Practices. Addison-Wesley Professional (2000)
4. Wang, Q., Jiang, N., Gou, L., Liu, X., Li, M., Wang, Y.: BSR: A Statistic-based Approach for Establishing and Refining Software Process Performance Baseline. Proceedings of the 28th International Conference on Software Engineering, Shanghai, China (2006) 585-594
5. Kan, S.H.: Metrics and Models in Software Quality Engineering. Addison-Wesley Professional (2002)
6. A.Florac, W., D.Careton, A.: Measuring software process-Statistical process control for software process improvement. Addison-Wesley Professional (1999)
7. Jalote, P., Saxena, A.: Optimum Control Limits for Employing Statistical Process Control in Software Process. IEEE Transactions on Software Engineering VOL.28 (2002) 1126-1134
8. Norden, P.V.: Useful Tools for Project Management, Operations Research in Research and Development. New York: John Wiley & Sons (1963)
9. Putnam, L.H.: A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering VOL. SE-4 (1987) 345- 361
10. Putnam, L.H., Meyers, W.: Measures for Excellence: Reliable Software on Time, Within Budget. Prentice Hall PTR (1991)
11. Wooldridge, J.: Introductory Econometrics: A Modern Approach. South-Western College Pub (2002)
12. Wang, Q., Li, M.: Software Process Management: Practices in China. Proceedings of the Software Process Workshop, Beijing, China (2005) 317-331
13. Mizuno, O., Shigematsu, E., Takagi, Y., Kikuno, T.: On Estimating Testing Effort Needed to Assure Field Quality in Software Development. Proceedings of the 13th International Symposium on Software Reliability Engineering, Annapolis, MD (2002) 139-146

DynaReP: A Discrete Event Simulation Model for Re-planning of Software Releases

Ahmed Al-Emran[1,2,3], Dietmar Pfahl[1,3], and Günther Ruhe[1,2]

[1] Schulich School of Engineering, University of Calgary, Canada
[2] Software Engineering Decision Support Laboratory, University of Calgary, Canada
[3] Centre for Simulation-based Software Engineering Research, University of Calgary, Canada
{aalemran,dpfahl,ruhe}@ucalgary.ca

Abstract. Software release planning can be described as a process consisting of the following three phases: (i) strategic release planning, i.e., the assignment of features to subsequent releases; (ii) operational release planning, i.e., the allocation of resources to tasks within each individual release; and (iii) dynamic re-planning, i.e., the revision of plans in order to handle unexpected changes imposed on product/project managers responsible for the realization of individual releases. Example changes include the addition or removal of features and/or developers, adjustments due to overestimated developer productivity, or underestimated work volume of feature-specific tasks, and adjusted degrees of task dependencies. The research presented in this paper mainly focuses on phase (iii) in conjunction to phase (ii) of the release planning process, assuming that phase (i) has already been completed. For that purpose, we present a discrete-event simulation model called DynaReP (Dynamic Re-Planner), which can be used for operational planning and re-planning of individual software releases. The applicability, effectiveness, and efficiency of DynaReP are illustrated through a series of typical planning and re-planning scenarios.

Keywords: Software release planning, operational planning, re-planning, discrete event simulation model, process simulation.

1 Introduction

One of the key questions of incremental software development is to decide which features can be offered at which release. This decision depends on the customer needs, technological constraints, and the resources and time frame available to implement the features. This decision is very dynamic in its nature, as many planning parameters and the features themselves are under continuous change [SSA96]. As a consequence of that, we study re-planning of software releases in more detail in this paper.

Good software release planning on both strategic and operational levels is extremely important [Pen02]. A bad release plan may cause late delivery of high-value features, unsatisfied customers, budget overrun, and thus decreased competitiveness. Software release planning can be done following a three phase solution procedure: (i) Planning of releases on a strategic level, (ii) Planning individual releases on

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 246–258, 2007.

operational level for the next release, and (iii) Performing dynamic re-planning on the operational level. Software release planning on strategic level (Phase i) involves decision-making about what new features to implement in which release. This takes into account cumulative resource consumption and technological dependencies between features. Release planning on operational level (Phase ii) involves decision-making about the allocation of developers to tasks within a single release. Re-planning on operational level (Phase iii) involves decision-making about the re-allocation of developers to tasks in the face of resource changes, feature changes, and observation of planning mistakes due to wrong assumptions about feature values, effort needs, developer productivities, and task-dependency[1] relationships. All of these decision-making problems are inherently difficult [Mom04].

The research presented in this paper focuses on a simulation model – DynaReP (Dynamic Re-Planner) – emphasizing on phase (iii) of the release planning procedure, i.e., supporting the re-planning of operational release plans each time a change in planning parameters is identified during the development of a release. Although the model is capable to perform phase (ii) for initial operational plan generation, we focus on phase (iii) as there exist no effective solution to the re-planning problem. The proposed approach is applicable to any given solution of the decision-problem of phase (i). Information on existing methods supporting phase (i) can be found in [RuS05].

Planning and re-planning of software releases can be formulated as a mathematical optimization problem (see [NgR06] for planning and resource allocation). However, these formulations are static in the sense that they do not allow for dynamic re-planning as easily as the simulation-based approach does. In general, the scenarios for re-planning discussed in this paper would be either impossible to model or would need more effort to model them as part of an optimization-based approach.

The remainder of this paper is structured as follows. Section 2 provides the motivation behind this research based on existing work performed in the area of software release planning. Section 3 describes the simulation model DynaReP. Section 4 illustrates the applicability and usefulness of DynaReP with the help of a case example. Section 5 discusses issues related to planning performance and limitations. Finally, Section 6 provides conclusion and future directions.

2 Related Work and Motivation

Both simulation and optimization approaches have been proposed in the context of planning for software releases. For example, the discrete-event simulation model presented in [HRD01] addresses some of the decision-problems associated with phases (i) to (iii) of the release planning procedure. Assuming a continuous stream of new incoming requirements, the model is used to investigate potential bottlenecks within subsequent releases. Bottlenecks are associated with task overload situations, i.e., situations in which the level of available resources assigned to specific tasks is too small to process incoming new (or from previous releases postponed) requirements. The model is also used to evaluate resource allocation changes that supposedly avoid previously identified overload situations. The problem dealt with in [HRD01] is

[1] The amount of effort that has to be consumed by a task before work on a subsequent task related to the same feature can begin.

different from the problem focused on in this paper for two reasons. Firstly, it does not consider dependencies between requirements, and thus does not provide specific feature allocations to individual releases. Secondly, it does not facilitate the evaluation of specific developer allocations to tasks within individual releases.

EVOLVE* [RuN04] is a hybrid intelligent framework that was initially applied to strategic software release planning. The objective of EVOLVE* is to create synergy between computational intelligence applied to formalized problem description and the application of the knowledge and experience of human experts. For software release planning, the result is an optimal feature assignment to different releases that maximizes stakeholder satisfaction while balancing trade-offs between release time, effort, and value. The decision support system ReleasePlanner™ (www.releaseplanner.com) is based on EVOLVE* and has been introduced successfully into several companies (e.g., Siemens, Corel, Trema Laboratories). While EVOLVE* addresses phase (i) of the general release planning procedure, it does not focus on phases (ii) and (iii). That is, EVOLVE* cannot answer how, by whom, and when individual features will be realized within a single release, how long it will take to perform individual development tasks, and how to perform re-planning when necessary.

$\text{OPTIMIZE}_{\text{RASORP}}$ (Optimize Resource Allocation for Software Release Planning) [NgR06] is an optimization approach that generates simultaneously feature allocation plans for subsequent releases and operational feature implementation plans for individual releases. Thus it combines phases (i) and (ii) of the general release planning procedure. $\text{OPTIMIZE}_{\text{RASORP}}$ considers tasks associated with features, a pool of developers to carry out these tasks, the productivity of developers to perform these tasks, and mappings between tasks and developers for realization of features within releases and maximizing release value. While $\text{OPTIMIZE}_{\text{RASORP}}$ offers a guaranteed degree of optimality[2] for resource allocation for the purpose of release planning, it does not support automatic re-planning, e.g., re-allocation of developers in the middle of a release implementation due to changes in planning parameters.

"Lightweight Replanning" [ARM06] is a process model that supports the revision of feature allocations to releases by comparing already assigned features of a specific release under development with new features that are requested to be included in that release. The purpose is to help decide instantly which of the old features should be postponed to subsequent releases and replaced by new features. The re-planning capability offered by this approach exclusively focuses on release plans resulting from phase (i). Issues on a more operational level, e.g., allocation of developers to feature development tasks, cannot be addressed.

REPSIM-1 (Release Plan Simulator, Version-1) [PAR06] is a System Dynamics simulation model that can perform stability analyses on existing release plans generated in phase (ii) of the general release planning procedure. Various stability analyses types evaluate the sensitivity of existing plans to possible planning errors. Planning errors can relate to alterations in expected personnel productivity, feature and task specific work volume (effort), and degree of task dependency. Stability analyses allow release planners perform "what-if" analyses on the proposed plan that might help them to be well prepared for easier and better manual re-planning than ad hoc

[2] "Guaranteed degree of optimality" refers to a solution where its objective function value is compared to an upper bound for the best possible objective function value.

approaches in case unexpected changes occur. However, automatic re-planning is not supported.

3 The DynaReP Model

DynaReP is a discrete-event process simulation model developed using EXTEND™ (http://www.imaginethatinc.com). The following sub-sections describe first the most important model parameters, constraints, variables, and controllers. Then the model structure and underlying method is presented.

3.1 Model Capabilities

Applying DynaReP helps address the following research questions:

1. How to generate initial operational plans of single releases? This question involves defining effective allocations of developers to feature development tasks.
2. How to perform automatic re-planning? Re-planning is needed when:
   - **a.** A new feature needs to be included in a release.
   - **b.** A planned feature is excluded from the release.
   - **c.** A developer becomes unavailable.
   - **d.** A developer needs to be added to the development team.
   - **e.** The estimated task dependency is bigger/smaller than expected.
   - **f.** The work volumes of features were under-estimated/over-estimated.
   - **g.** The productivities of developers were over-estimated/under-estimated.

It should be noted that DynaReP can be applied to perform re-planning due to the occurrence of one event or any combination of the above listed events. In Section 4, some of these re-planning scenarios we will be exemplified.

3.2 Model Heuristic, Parameters, Variables, Constraints and Controllers

The heuristic used for assigning developers to feature/task-pairs essentially consists in matching the next available developer with the highest task-specific productivity to the next waiting feature with the largest effort (for a specific task). If only one developer with very low productivity is currently idle, then this mapping procedure can result in assigning a developer with low productivity to a large feature. To avoid such a worst case situation, a set of threshold variables are defined which exclude developers with productivity below a certain value to be assigned to feature/task-pairs.

In the following, model parameters, variables, constraints and controllers are described in detail. DynaReP offers of the following model parameters:

- *Initial # Feats*: The number of features planned to be implemented at the beginning of the development. One *Feature* entity per feature is created in the model. Each *Feature* entity is further decomposed into *Task* entities representing ordered tasks necessary to develop the feature. Examples of subsequent task types are *design*, *implementation*, and *test* (denoted as T_1, T_2, and T_3 respectively). The estimated volume (effort required) for each of the tasks per feature are stored in the model database and corresponding entities are initialized accordingly.

- *Initial # Devs*: The number of developers available when development starts. One *Developer* entity per developer is created in the model. Per developer productivity values for each task type are kept in the model database and corresponding entities are initialized accordingly. Productivity represents the amount of work done per time unit. For example, if the productivity of developer D_k is y and the work volume of task T_j for feature F_i is x person-weeks (PW), then D_k can perform T_j of F_i in x/y weeks. Productivity 0 for a task type implies that a developer is not able to perform that type of task.
- *Task Dependency*: Specifies the dependency between subsequent tasks in terms of percentage of task-related effort that has to be consumed before a subsequent task can begin. If *Task Dependency* is 100, then T_j (e.g., test task) of a feature F_i can start only if T_{j-1} (e.g., implementation task) of F_i is 100% complete; if it is 50, at least 50% of the preceding task needs to be completed to start the next task of the same feature. No *Task Dependency* applies to the very first task, T_1 (e.g., design task), since it does not have any predecessor task.

An important model variable is *Threshold Productivity,* a vector of productivity threshold values used to restrict the availability of developers per task type. For example, if the model has design, implementation, and test tasks, the vector has three cells. For a specific type of task, if a developer does not possess a productivity value higher than that of the corresponding *Threshold Productivity* variable, then that developer will not be allowed to carry out that type of task. DynaReP uses an optimizer construct (offered by the simulation modeling tool EXTEND™) that automatically assigns a value to each of these *Threshold Productivity* variables such that the overall duration of a calculated release plan becomes minimal. Note that values for these variables are re-assigned at each time a change is made in the planning parameters.

Independent from the value the model parameter *Task Dependency*, DynaReP maintains a model constraint *Task Precedence Relation*: this is a start-start and end-end relation between two subsequent task types T_{j-1} and T_j, such that a task of type T_j cannot start before a task of type T_{j-1} has started, and a task of type T_j cannot end before a task of type T_{j-1} has been completed.

The following model controllers are used in DynaReP to allow its users to specify the changes to be performed for re-planning:

- *Include Feats*: used to indicate a specific feature to be included in the release.
- *Exclude Feats*: used to indicate a specific feature to be excluded from the release.
- *Include Devs*: used to indicate a specific developer who joins the developer team.
- *Exclude Devs*: used to indicate a specific developer who becomes unavailable.
- *Re-join Devs*: used to indicate re-joining of a previously excluded developer.
- *Change Effort*: used to adjust under/over-estimated work volume (effort) of a specific feature-task combination.
- *Change Prod*: used to adjust over/under-estimated productivities for a specific developer-task combination.
- *Change Times*: used to indicate the time from when a new change will be effective. This also allows DynaReP to keep values of all *Threshold Productivity* variables determined at different time.

There exists another important model controller - *PTimeout*: a periodical time interval when a *Timeout Signal* will be sent to enforce releasing blocked *Task* and *Developer* entities (see section 3.3 for more details).

3.3 Model Structure and Description

DynaReP consists of ten high-level blocks, each of which is can be further decomposed. Figure 1 shows how these blocks are connected to each other with two types of connections: (i) Entity Link, and (ii) Information Link. Entity links are paths through which entities are routed from one block to another. Information links allows data passing among the blocks. Since DynaReP assigns developers to perform different tasks to realize each feature, there exist three types of entities in DynaReP: (1) *Feature/Task*, (2) *Developer*, and (3) *Coupled* (when the other two entities merged to form one entity representing a task is assigned to a developer).

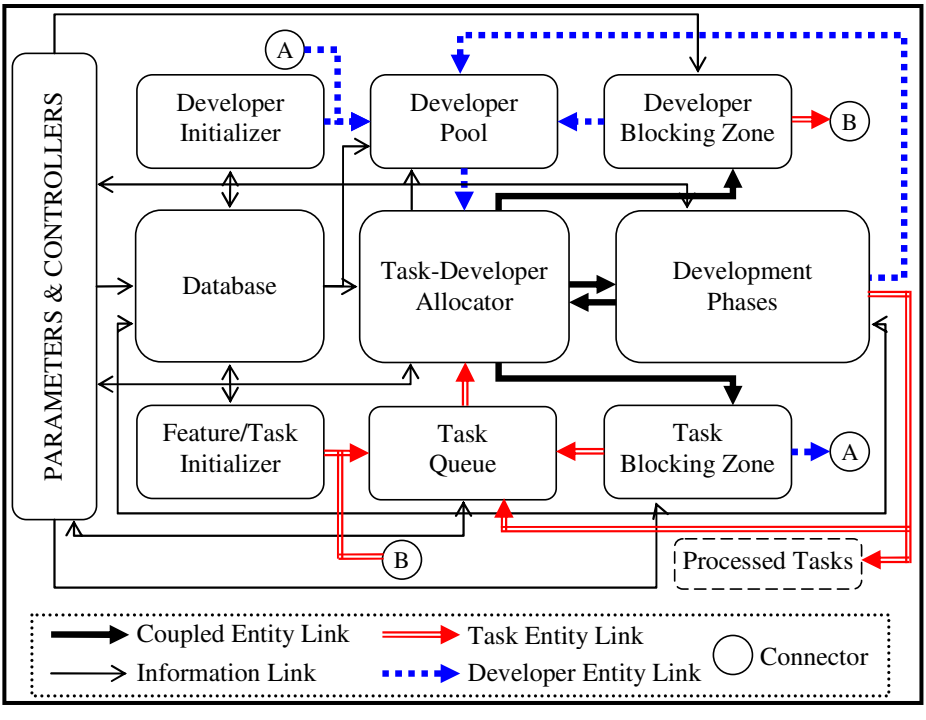


Fig. 1. High level view of DynaReP model structure

The description of the top level blocks can be summarized as follows:

- *Database:* Holds input information about features and their related task-specific effort estimates, developers and their task-specific productivity; it also stores output information, i.e., which developer is assigned to which feature/task-pair, when task started, and how long tasks were executed.

- *Feature/Task Initializer:* Creates *Feature* entities for each of the features, breaks each of them down into associated *Task* entities, initializes them with their respective information from *Database*, and sends them to *Task Queue* block.
- *Task Queue*: Holds incomplete *Task* entities, releases task entities that belong to an incomplete feature with the highest work volume, and sends them to *Task-Developer Allocator* block.
- *Task Blocking Zone:* Receives *Coupled* entities from *Task-Developer Allocator* whenever a task does not meet either of *Task Precedence Relation* and *Task Dependency*; decuples *Coupled* entities, sends *Developer* entities immediately to *Developer Pool*, and keeps *Task* entities until next *Timeout Signal* (c.f., section 3.2) to let other tasks in the queue to be considered.
- *Developer Initializer:* Creates *Developer* entities, initializes them with their respective information from *Database*, and sends them to *Task Queue* block.
- *Developer Pool*: Holds *Developer* entities, releases developers that possess the highest productivity value for the type of task that has currently arrived at *Task-Developer Allocator*, and sends them to *Task-Developer Allocator.*
- *Developer Blocking Zone:* Receives *Coupled* entities from *Task-Developer Allocator* whenever a developer does not possess productivity higher than the corresponding threshold value; decuples *Coupled* entities, sends *Task* entities immediately to *Task Queue*, and keeps *Developer* entities until next *Timeout Signal* (c.f., section 3.2) to let other developer in the pool to be considered.
- *Task-Developer Allocator:* Combines received *Developer* entities with previously received *Task* entities to simulate the assignments of developers to feature/task-pairs, checks whether their combination meets all necessary conditions, and depending on that sends the *Coupled* entity to either *Development Phases* (if conditions are satisfied) or to one of *Task Blocking Zone* and *Developer Blocking Zone* (depending on type of condition failed). Sending *Coupled* entities *to Development Phases* also causes a Timeout Signal (c.f., section 3.2) to generate.
- *Development Phases:* Simulates the real-world behavior of developers working on tasks by holding *Coupled* entities for some calculated simulation time. After completing a task, *Coupled* entity is decoupled to form separate *Task* and *Developer* entities. The *Developer* entity is sent back to the *Developer Pool* and the *Task* entity is taken out of the simulation model (shown as *Processed Tasks* in Figure 1) since the task is being processed. At last, a *Timeout Signal* (c.f., section 3.2) is generated.
- *Parameters & Controllers:* Defines and manipulates all model parameters and controllers (c.f., section 3.2).

4 Hypothetical Case Study Example for Re-planning Scenarios

In this section, we illustrate some of the re-planning capabilities of DynaReP (cf., section 3.1) in order to demonstrate the applicability and usefulness of DynaReP for re-planning of software releases. For that we chose a case example representing a hypothetical software release development situation with the following properties:

- Features to be implemented in the release: F1, F2, …, F8
- Tasks to be carried out to create each feature: T1, T2, and T3 (e.g., design task, implementation task, and test task, respectively)

- Developers available to work on each feature-specific tasks: D1, D2, …, D6
- The estimated work volume (in PW) for each feature-specific task (cf. Table 1)
- The assumed productivity of each developer for each task type (cf. Table 2)

Table 1. Features and their estimated task work volumes (effort in person-weeks)

Feature	Task Type		
	T1: Design	T2: Implementation	T3: Test
F1	3	6	6
F2	8	3	2
F3	6	10	5
F4	3	3	6
F5	5	6	4
F6	7	5	3
F7	10	5	6
F8	6	8	10

Table 2. Developers and their productivities for different task types (dimensionless)

Developer	Task Type		
	T1: Design	T2: Implementation	T3: Test
D1	1.5	2	1
D2	1	1.5	2
D3	2	1	0
D4	0	2	1.5
D5	0.5	1.5	2
D6	2	1	1

4.1 Baseline Scenario: Initial Planning

To be able to demonstrate the re-planning capability of DynaReP, we need to generate an initial plan (cf. phase ii) that assigns developers to feature/task-pairs contained in a release. This allocation is done based on the DynaReP heuristic (c.f., section 3.2). As a result, we receive a complete schedule that tells us when, by whom, and for how long each feature-specific task will be conducted. In our case example, the initial plan generated by DynaReP requires 15 weeks to complete the release. The initial plan is the base for the re-planning scenarios shown in the next subsections. As soon as an unexpected situation occurs (e.g., a new feature needs to be included and realized in a release), the initial plan needs to be altered in order to accommodate the change.

4.2 Re-planning Scenario 1: Feature Inclusion

Request for including a completely new feature that was not planned to be included within a release is a very common situation.

For our case example, we assume that the development organization started working according to the initial plan from the beginning of the release development. After three weeks, a new feature F9 is enforced by the customer to be included and implemented. We also assume that the estimated work volume for each of the tasks of F9 is

8 person-weeks (PW). Now, after specifying this situation through model controllers *Include Feats* and *Change Times* (c.f., section 3.2), DynaReP can produce an altered plan as shown in Figure 2. The revised plan contains the same schedule as the initial plan (shaded portion of Figure 2) up to the 3rd week. From the beginning of the 4th week, a new developer allocation takes place and costs an additional 1.5 weeks (i.e., a total duration of 16.5 weeks) to complete the release.

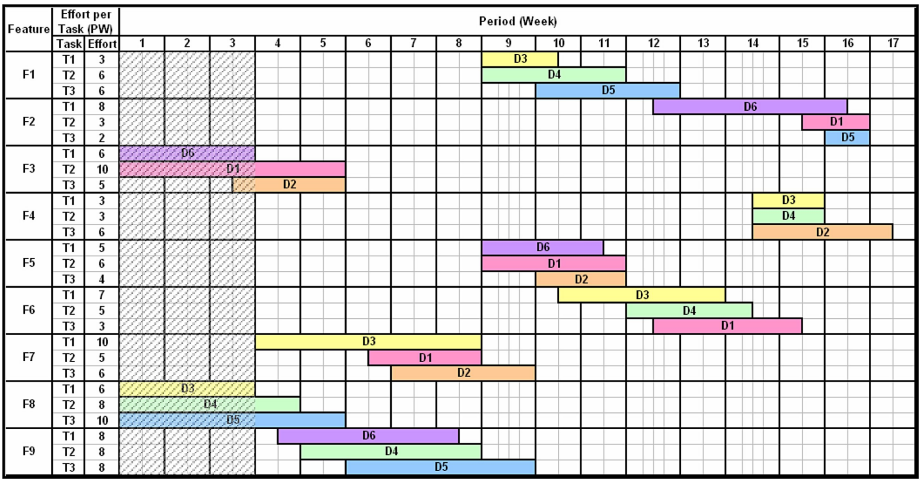


Fig. 2. A new feature F9 is included in the release under development

4.3 Re-planning Scenario 2: Underestimated Work Volume

Another very common scenario in industry is the underestimation of work volume (effort). If this occurs, it can easily happen that a developer cannot finish a task within an expected time period and the duration allocated for the task needs to be extended. Because of this extension, later tasks that were supposed to be handled by the same developer need to be adjusted as well resulting in another re-planning scenario.

For our case example, we assume that developers D1 and D2 could not finish tasks T2 and T3, respectively, of feature F3 within 5 weeks as planned. Both developers need one more week to complete their corresponding tasks. Since both of the developers possess a productivity of 2 for tasks types T2 and T3, respectively, the work volume for the tasks T2 of F3 and T3 of F3 need to be increased by 2 PW (effort = calendar time x productivity). Figure 3 shows the new plan resulting from this change (altering the plan of Figure 2 from the 5th week on). The effort values for T2 and T3 of F3 are updated to 12 PW from 10 PW and 7 PW from 5 PW, respectively. This is done by using model controllers *Change Effort* and *Change Times* (c.f., section 3.2). Note that in this re-planning scenario the altered plan indicates that no additional time is required to complete the release development.

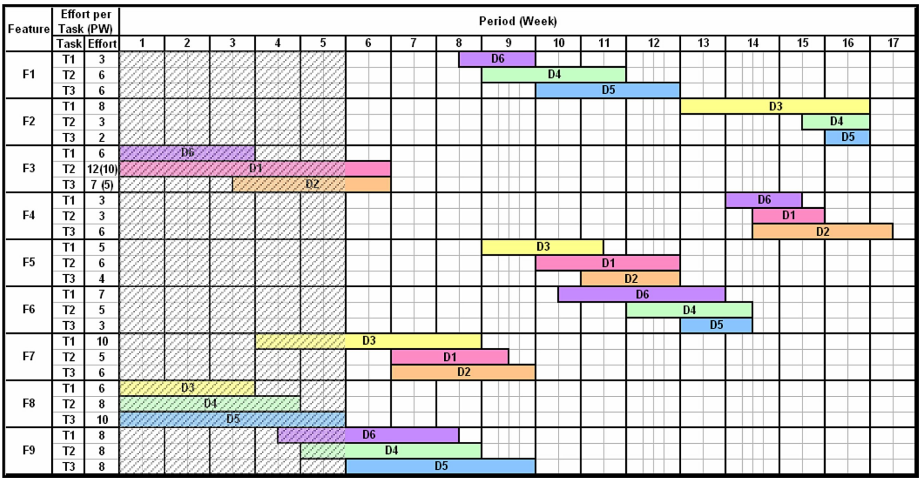


Fig. 3. Work volumes of tasks T2 & T3 of F3 were underestimated

4.4 Re-planning Scenario 3: Developer Unavailability

Unavailability of developers in a short notice is one of the most difficult problems to resolve. Possible reasons for developer unavailability are sickness, transfer (to another project), or simply leave away. In order to handle such situation, a re-allocation of developers (i.e., re-planning) is required to fill-up the gaps in the schedule induced by the unavailable developer. This re-allocation can be done either using the existing human resources (that may cause longer development time) or including (e.g., transferring, hiring) developers into the development team.

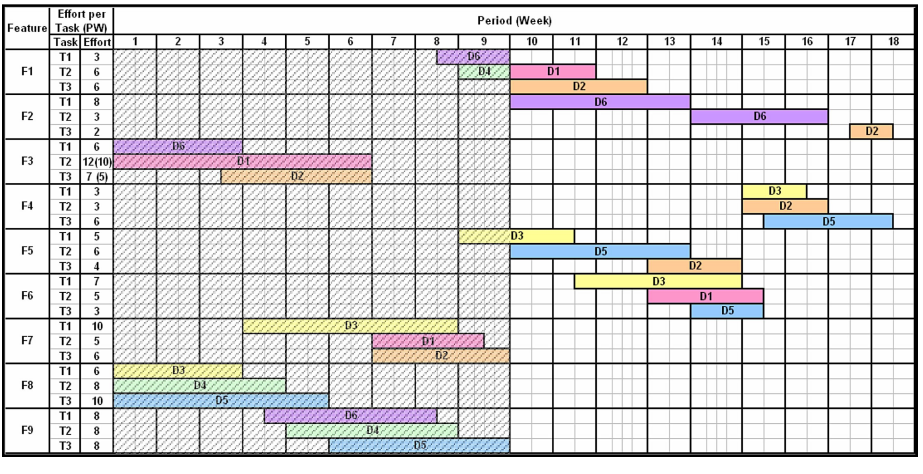


Fig. 4. Developer D4 has become unavailable from the beginning of 10th week

Figure 4 shows how DynaReP re-allocates the remaining developers to complete the release within 17.5 weeks, if developer D4 becomes unavailable starting from the beginning of the 10th week. Note that DynaReP properly handles the issue that developer D4 had not completed task T2 of F1 when forced to leave. Again, only two controllers, *Exclude Devs* and *Change Times*, are required to specify this change.

5 Discussion

In this section, first, we discuss the quality of release plans generated by DynaReP. $\text{OPTIMIZE}_{\text{RASORP}}$ [NgR06] is the closest work to the research presented in this paper. It provides allocation of resources in the context of software release planning with a guaranteed degree of optimality. Therefore, we have chosen $\text{OPTIMIZE}_{\text{RASORP}}$ as our benchmark method for evaluating DynaReP's planning quality. Since differences exist among their assumptions and objectives, necessary arrangements (not presented here due to space limitations) were made so that planning solutions can be compared fairly.

In order to conduct the quality evaluation of DynaReP, five release planning input data sets were selected. Table 3 shows the summary of the evaluation. Columns four and five show for each of the five input data sets the release development durations estimated by $\text{OPTIMIZE}_{\text{RASORP}}$ and DynaReP, respectively. The schedule times generated by DynaReP are 5% to 10% longer than the schedule time generated by $\text{OPTIMIZE}_{\text{RASORP}}$.

Table 3. Performance Comparison: $\text{OPTIMIZE}_{\text{RASORP}}$ vs. DynaReP

Cases	Number of Features	Number of developers	Development Time taken by $\text{OPTIMIZE}_{\text{RASORP}}$ [weeks]	Development Time taken by DynaReP [weeks]	Performance Differences
1	34	9	22	24	9%
2	37	12	18	19	6%
3	45	18	13	14	8%
4	47	16	15	16	7%
5	65	12	27	29	7%

This difference may be considered acceptable in the sense that accommodating changes and performing re-planning with DynaReP is quick and easy. This statement can be supported by (i) the re-planning scenarios presented in Section 4 where we observed that only two model controllers needed to be accessed per change in planning parameters, and (ii) the simulation time needed by DynaReP to generate operational plans (in these five cases, they are in the range of 1-3 minutes).

Thus, there exists a trade-off situation when deciding whether to choose $\text{OPTIMIZE}_{\text{RASORP}}$ or DynaReP. While $\text{OPTIMIZE}_{\text{RASORP}}$ can offer a guaranteed degree of optimal resource allocation, it does not easily support re-planning. On the other hand, re-planning can be achieved almost instantly by DynaReP, however at the cost of 5-10% longer schedules. Thus, if plan quality is very important and the need for re-planning is low then $\text{OPTIMIZE}_{\text{RASORP}}$ should be chosen. If re-planning is frequent, then DynaReP is the better choice.

This is to be noted that the case example presented in section 4 was kept small in order to demonstrate a variety of model capabilities in a limited space. However, the performance comparison data in table 3 signifies that the model can be used for even larger release plans and this clarifies its applicability in a real-world situation. Other validity questions are related to different estimates. For example, the productivity and effort estimates used in the case example are hypothetical. This encourages the proposed approach to be validated in an industrial environment.

Besides the fact that DynaReP cannot guarantee the generation of optimal plans there exist some other minor limitations. DynaReP generates operational plans with developers working on a single task at a time and vice versa. This makes DynaReP, for example, not applicable to projects that apply pair-programming. A work-around for this limitation is to split developers into several "virtual developers" whose productivity adds up to the original productivity of the split developer. A similar strategy can be applied to features. Finally, the current version of DynaReP does not provide a sophisticated user interface for users unfamiliar with the simulation tool EXTEND™.

6 Conclusions and Future Work

In this paper, we presented the design and evaluation of the discrete-event simulation prototype DynaReP, which provides decision-support for operational release planning and re-planning. Its importance, effectiveness and efficiency were demonstrated via a series of typical re-planning scenarios. We have shown that the DynaReP model is able to accommodate a great variety of re-planning scenarios being of great practical interest. The model is appropriate for a frequent change request environment, since it can accommodate changes and perform re-planning both quickly and easily. For planning, the results have been shown to be almost as good as the results from applying specialized optimization algorithms. Moreover, the model can handle a realistic constraint "task dependency" which is not considered by any other methods or models proposed in the context of software release planning.

Future work will focus on (i) enhancing the model heuristic in order to improve effectiveness; (ii) including feature dependency constraints that specify whether a feature must be realized before another feature; (iii) improving model usability (e.g., data input via GUI, connection to external database, etc.); and (iv) validating the proposed approach in an industrial environment. In addition, we plan to study both planning and re-planning also for stochastic variables expressing effort of feature/task-pairs and/or productivity of developers. We again expect stronger modeling capabilities from using simulation when compared to optimization.

References

[ARM06] Albourae, T., Ruhe, G., Moussavi, M.: Lightweight Replanning of Software Product Releases. Proceedings of International Workshop on Software Product Management, Minneapolis/St. Paul, Minnesota, USA (2006)

[HRD01] Höst, M., Regnell, B., Dag, J., Nedstam, J., Nyberg, C.: Exploring Bottlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulation. Journal of Systems and Software, Vol. 59, No. 3 (2001) 323-332

[NgR06] Ngo-The, A., Ruhe, G.: Optimized Resource Allocation for Incremental Software Development. TR 062/2006, Laboratory for Software Engineering Decision Support, University of Calgary (2006)

[Mom04] Momoh, J.: Applying Intelligent Decision Support to Determine Operational Feasibility of Strategic Software Release Planning. Masters thesis, Department of Electrical and Computer Engineering, University of Calgary, Canada (2004)

[PAR06] Pfahl, D., Al-Emran, A., Ruhe, G.: Simulation-Based Stability Analysis for Software Release Plans. In: Wang, Q. et al. (eds.): SPW/ProSim 2006 - Proceedings. LNCS 3966, Berlin-Heidelberg: Springer-Verlag (2006) 262-273

[Pen02] Penny, D.A.: An Estimation-Based Management Framework for Enhancive Maintenance in Commercial Software Products. Proceedings of International Conference on Software Maintenance (2002) 122-130

[RuN04] Ruhe, G., Ngo-The, A.: Hybrid Intelligence in Software Release Planning. International Journal of Hybrid Intelligent Systems, Vol. 1, No. 2 (2004) 99-110

[RuS05] Ruhe, G., Saliu, O.: The Art and Science of Software Release Planning. IEEE Software, Vol. 22, No. 6 (2005) 47-53

[SSA96] Stark, G., Skillicorn, A., and Ameele, R.: An Examination of the Effects of Requirements Changes on Software Maintenance Releases, Journal of Software Maintenance: Research and Practice, Vol. 11 (1999) 293-309

The Economic Impact of Software Process Variations*

Florian Deissenboeck and Markus Pizka

Institut für Informatik, Technische Universität München
Boltzmannstr. 3, D-85748 Garching b. München, Germany
{deissenb,pizka}@in.tum.de

Abstract. The economic benefit of a certain development process or particular activity is usually unknown and indeed hard to predict. However, the cost-effectiveness of process improvements is of paramount importance and the question how profitable certain activities are needs to be answered. Within a large-scale commercial organization, we were challanged with the task to quantify the *economic* benefit of isolated test and development environments. To answer this question we defined a generic process model based on absorbing Markov chains that allows to analyze the economic benefit of software process variations. This model exposes conflicts between process steps and reiterations of development activities and thereby provides a highly flexible tool for the investigation of the effects of changes to a development process on its overall performance. This model was used to predict the impact of isolated testing on the overall effort and duration of projects at BMW. The results obtained correspond well with the perception of experienced developers and gives a detailed explanation for the effects. Besides this, it can be used to analyze various other economic aspects of software development processes and yields an interesting alternative for cost estimation.

Keywords: Software Process Economics, Process Simulation, Industrial Application, Absorbing Markov Chains.

1 Software Process Economics

How does one determine the economic impact of selecting a certain process model? Is XP cheaper than RUP? What are the risks of the waterfall model? Does the spiral model actually yield faster time-to-system? All of these questions are of practical relevance, highly important but very difficult to answer. Surely, in a specific situation we assume that alternative A is faster (cheaper, better, ...) than B based on our invidual experience but the benefit can neither be quantified nor guaranteed.

In contrast to this, the costs of activities are usually clear and precisely documented in bills. For example, it is unclear how much money can be saved in

* Part of this work was sponsored by the BMW Group.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 259–271, 2007.

a given project setting by spending one additional dollar on model-based development techniques. The same applies for well-established activities like documentation as well as for more specialized methods like requirements engineering with formal methods.

1.1 The Value of Isolated Testing

Within a large scale industrial organization, we were challenged with the task to determine how much time and effort is saved by using isolated test and development environments for IBM mainframe (i. e. PL/I, COBOL) based commercial software projects.

While isolated testing is rather straight forward for UNIX and Microsoft Windows based software projects it is non-standard for mainframe applications since all projects share the same machine with the same infrastructure without having private copies of libraries, databases and so on. To achieve some kind of isloation most IT organizations that develop and maintain mainframe applications create some kind of software solution that enables them to develop and test multiple projects simultaneously in separated *environments* on a single mainframe.

The costs of this solution are usually easy to determine by adding space, CPU time, software licenses and support personnel. However, although it can be argued in a qualitative manner that separate testing and avoiding conflicts is useful, it is hard to quantify the *benefit*. In practice this makes it very hard to argue in favor (or against) such measures and consequently leads to decisions that lack an economically justified basis.

1.2 Approach, Contribution and Outline

Starting from our project partner's concrete questions about the economic benefit of process variations, we formulated a precise research question (Sec. 2) and investigated different approaches to answer it.

Due to a number of reasons (detailed in Sec. 6) we found that an empirical study could not satisfactorily answer this question and therefore developed new concepts to evaluate the economic effect of decisions regarding process variations (Sec. 2). These concepts are based on a probabilistic process model that uses absorbing Markov chains for the process simulation. This model advances existing process models as it renders project risks explicit and precisely describes reiterations of activities (Sec. 3). This model can be used to derive quantitative information on the cost and benefit of specific process activities.

We illustrate this with a study carried out for the BMW Group to determine the economic benefit of isolated test and development environments on mainframes (Sec. 4 and 5). We explain how our approach extends previous work (Sec. 6) and illustrate how the scope of application of the analytical model can be further broadened (Sec. 7).

2 Requirements / Situation

The object of investigation of our study was the development and test processes used by BMW Group's mainframe software development division. At BMW, several 100 software engineers develop and maintain critical business information systems with a total of 85 millions lines of PL/I and COBOL code. The division uses two separated IBM zSeries mainframes for development and operation, whereas our study focused exclusively on the development mainframe.

2.1 Mainframe Software Development

Unlike the more common workstation-based development environments, mainframes do in general not provide developers with isolated environments where they can edit, compile, link and test the code they are working on without interfering with other projects. In fact, if no additional measures are taken, all developers share the same development environment and all test data.

Due to the frequent separation of development and operation spaces of typical mainframe installation this does not pose any problems for the operation of the software, but creates servere problems for the concurrent development and test of multiple projects. Conflicts between projects can occur during almost all activities (e. g. compile, link, test) and affect almost all development artifacts (e. g. source code, libraries, test data). These conflicts are not only frustrating and time-consuming for the developers, but make sound testing almost impossible as test results can not be interpreted properly. For example, if a test case fails, it is not decidable whether it failed because of a bug or because another project changed the test data in the shared data base.

Unfortunately, isolated test spaces cannot be established for mainframes as easily as in ordinary workstation-based environments where every developer can have his own test space on an own workstation.

2.2 The CAP Isolation Mechanism

The BMW Group developed a software-based isolation technique on top of the virtualization mechanism provided by the mainframe.[1] This technique offers projects isolated test and development environments called Caps *(capsules)*. These Caps contain a complete copy of the required development environment including compilers, linkers, job control, and test databases. They thereby enable projects to develop and test in an independent, conflict-free manner until they reach a certain degree of maturity and can be integrated in the main development trunk in a special integration test phase. Caps have the additional advantage of making it easy to *reset* the complete development environment of a project to a specific state.

These advantages, however, come at a price as the initialization, operation and support of a Cap is a non-trivial task that demands significant hardware resources as well as expenses for dedicated personnel.

[1] IBM zSeries mainframes provide a coarse-grained virtualization mechanism.

2.3 Research Question

The qualitative benefit of a CAP can be explained quite easily by explaining how non-isolated development environments create expensive conflicts and contribute to poor product quality due to unreliable test results. It is, however, very hard to compare these qualitative benefits to the known quantitative costs of the CAP mechanism. Therefore the research question of the study we conducted was:

What is the economic benefit of using a CAP *for a software project?*

Note, that although this initial questions focuses on project effort, our study also analyzed the project duration to characterize the crucial time-to-system aspect. However, we cannot report on this in detail due to space constraints.

3 A Probabilistic Process Analysis Model

As explained in Sec. 6, we are convinced that is not feasible to answer the above questions on a quantitative scale by carrying out an empirical study. We therefore opted for an analytical model that abstracts from the problem under investigation and allows us to focus on the impact of CAPs on development effort and time.

This model was inspired by an observation of the analogy between software development processes and concurrent systems theory [1]. Development *activities* are similar to *tasks* executed by an operating system. In a development process the *resources* are not memory and file handles but source code, libraries and test data. Similar to the conflict that arises from a write access to the same memory address in a parallel system, a concurrent change to a program by two different projects produces a conflict in the software development process.

3.1 Probabilities and Risks

These considerations lead to a probabilistic process model that describes a development process as a system of concurrently executing tasks. The tasks of the system are the *activities* of the software process and the processors are humans (developers) executing these activities. Due to the goal of the overall process and limited resources, there are constraints on the order of the activities entailing the need for coordination. The transitions from one activity to possible succeeding activities are labeled with probabilities. Through this, there may also be *loops* in the parallel deterministic automaton describing costly rework in the development process due to failure or incompleteness at a certain stage of the process. The activities and the frequency of there execution define the cost and the duration of the project.

Fig. 1a shows a model of a simplified software process with the typical activities and transitions between them. Unlike other process models this model explicitly describes the loops (cycles) realistically found in software projects. This enables us to e. g. model the alternation between the activities *Implementation* and *Unit Test* that takes place in practice: Developers write some code,

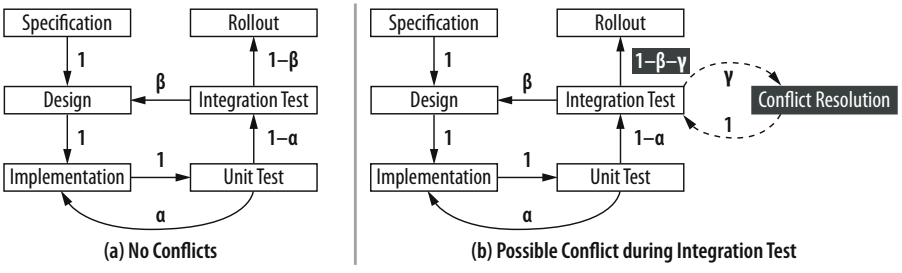


Fig. 1. Example Processes

test it (either manually or automatically) and then go back to implementing more code and/or fix existing code. They do so until they are eventually done with the implementation and all their tests pass. In addition to that, loops allow us to explicitly capture prevalent project risks that are often ignored [2]; e. g. an unlikely, but still possible, transition from the *Integration Test* to the *Specification* could be easily added to the process model. Note that the sum of the probabilities of the outgoing transitions of an activity must always be one.

Fig. 1b illustrates how resource conflicts during specific activities can be elegantly expressed through additional conflict-specific activities and adjusting the transition probabilities accordingly. For example, a conflict with another project during the *Integration Test* does not only reduce the probability that the project can proceed with the activity *Rollout* but requires the execution of the additional activity *Conflict Resolution*.

3.2 Operationalization of the Model

While this model provides an interesting abstraction of a software development process, it is does not answer the question about the benefits stated above, yet. Fortunately stochastics can help here as the process model can be viewed as a stochastic process or, more precisely, as a *discrete Markov chain* with an *absorbing state*.

A Markov chain is defined as a stochastic process with a set of states $S = \{s_1, s_2, \ldots, s_r\}$. The process starts in one of these states and moves stepwise from state to state. If the chain is in state s_i, then it moves to state s_j at the next step with a probability denoted by p_{ij}. This probability does not depend on the state history of the chain [3].

Markov chains are typicalyl represented as directed graphs very similar to the ones in Fig. 1 or as a transition matrix P that denotes the transition probabilities for every state. Working with this matrix, Markov chain theory provides powerful methods to compute a number of interesting properties of the chains. It is, for example, easy to calculate in which state the chain is expected to be after n steps when started in state s_i, or to determine the probability for moving from state s_j to state s_l in k steps.

When modeling a software process there must be an activity that does not have any transitions to other activities and thereby marks the end of the process (*Rollout* in Fig. 1). Translated to a Markov chain model this is equivalent to a terminal state s_i that has exactly one outgoing transition to itself with the probability $p_{ii} = 1$, Such a state is called an *absorbing state* and Markov chains with an absorbing state are called *absorbing Markov chains* [3].

Absorbing Markov chains are a powerful tool for analyzing processes as they provide well defined methods to determine

- the expected total number of steps until the chain reaches an absorbing state as well as
- to calculate the expected number of steps spent in each state.

Without going into the mathematical details we illustrate this for the process shown in Fig. 1a.

For the sample probabilities $\alpha = 0.95$ and $\beta = 0.2$ the absorbing Markov chain analysis yields the following expected number of visits to each state (start state *Specification*): *Specification* is expected to be carried out only once, *Design* and *Integration Test* are expected to be performed 1.25 times, and *Implementation* and *Unit Test* 25 times. The total number of steps before the chain reaches the absorbing state *Rollout* is given by the sum which is 53.5.

Figure 2 shows how different values for the probabilities α and β influence the expected total number of steps in the example process. While values close to 1 lead to an infinite number of steps in both cases, one can see that increasing β raises the number of steps stronger than increasing α as this transition occurs *later* in the process.

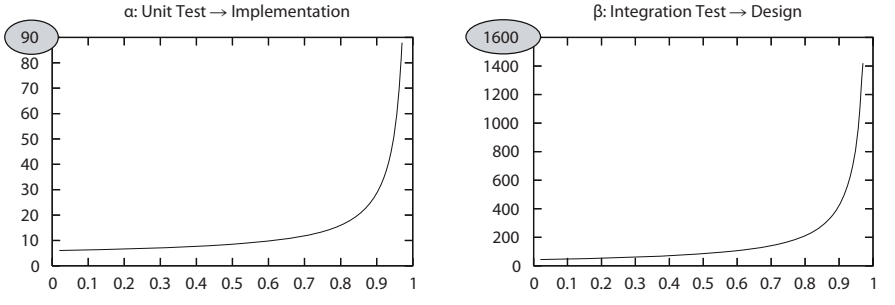


Fig. 2. Transition Probability vs Expected Total Number of Steps

3.3 Total Project Effort and Duration

The expected total number of steps represents a measure for project progress, but it still does not yet fully answer the questions about the total project effort and duration. To achieve this each process activity a is now associated with the average effort $\mathsf{eff}(a)$ and time $\mathsf{time}(a)$ needed for a single execution of the activity. The total effort and duration of a project is given by:

$$\mathsf{eff}_t = \sum_{a \in A} \mathsf{eff}(a) \cdot \mathsf{steps}(a) \qquad \mathsf{time}_t = \sum_{a \in A} \mathsf{time}(a) \cdot \mathsf{steps}(a)$$

where A is the set of all activities and $\mathsf{steps}(a)$ is the expected number of visits to activity a. Note that $\mathsf{eff}(a)$ and $\mathsf{time}(a)$ dependend on the project size.

4 Application of the Analysis Model to Isolated Testing

To apply our approach to analyze the economic benefit of isolated test and development at BMW, three fundamental pieces of information are needed:

1. transition probabilities
2. effort needed to execute for each activity
3. time needed to execution for each activity

As it is not realistic to *correctly* determine this information without investing considering empirical studies, we analyzed the two process variations (Cap and Non-Cap) in a relative manner. We therefore designed a *reference process*, calibrated it with existing empirical data and parameterized it with the probability for conflicts during development and test. Based on this reference process we designed the process models for Cap and Non-Cap development and compared them using the method presented above. This comparative approach allowed us to abstract from concrete values for the transition probabilities as well as the efforts and times needed for each activity.

4.1 Reference Process

Based on existing process descriptions and interviews with project managers as well as developers, we created the reference process model with 13 activities and 18 transitions (not presented here in its entirety due to confidentiality reasons). This model does not contain special isolation-related activities and therefore consists of the usual specification, design, implementation and test activities. It does, however, carefully distinguish between module tests and two levels of integration tests and contains explicit error analysis activities.

Eleven of the 18 transition of the model have a transition probability unequal one. Using existing process analysis data as well as interviews we estimated the probabilities and ensured that the remaining impreciseness does not bias our study results (see Sec. 5).

4.2 Calibration

To determine the effort needed for each execution of the activities, we calibrated the reference process with data from well-known empirical studies.

For example, the Markov chain analysis showed that the activity *Implementation* will be carried out 95.24 times and thereby accounts for 36.78% of the expected total 258.95 process steps. As [4] and other sources point out that implementation usually accounts for $\approx$ 20% of the total development effort, we concluded that the relative effort of each execution of *Implementation* activity in our process is 0.21%. These relative measures of effort were later on used to compare the different processes.

4.3 Parameterization

Obviously the difference between the CAP and Non-CAP development processes is determined by the number of conflicts with other projects that arise during the different activities. We expressed this by introducing the *conflict probability parameter* c and parameterized the process models accordingly. Figure 3 exemplifies this for the *Integration Test* and shows how the conflict parameter c influences the transition probabilities.

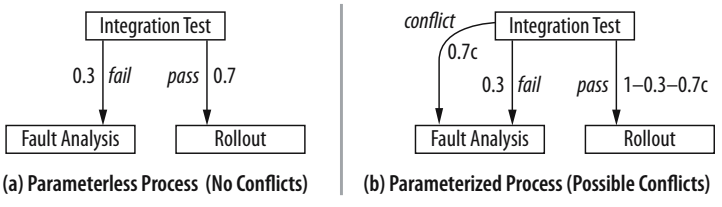


Fig. 3. Process Parameterization

4.4 CAP and Non-CAP Process Models

Based on the previously defined reference process we built specific models for CAP and Non-CAP development. The models differ as the CAP model contains specific CAP-related activities, e. g. *CAP Refresh* and the Non-CAP model explicitly describes conflict resolution activities (Fig. 4).

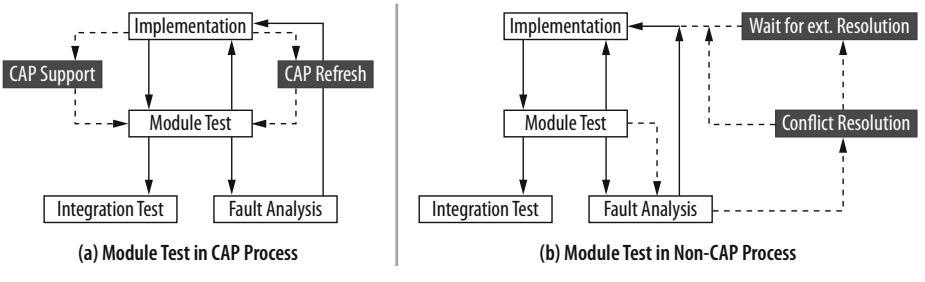


Fig. 4. Differences between CAP and Non-CAP Process (Module Test)

Please note that the CAP process, though isolated, is not fully free of conflicts as conflicts may arise during the *Integration Test* when the project leaves its CAP.

4.5 Relative Project Effort

For both processes the Markov chain analysis was carried out for different conflict probabilities and the total effort was put into relation with the same calculation for the reference process. Figure 5 shows the results in two resolutions. On the

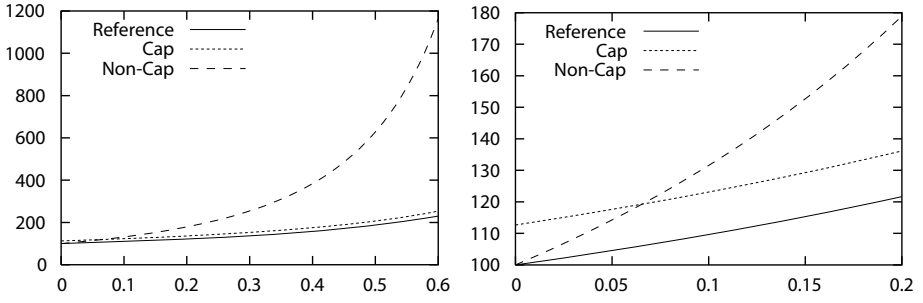


Fig. 5. Conflict Probability vs Relative Effort

left, the total effort for all three processes is shown for the conflict parameter interval $[0; 0.6]$. One can easily see that the efforts for the reference and CAP process behave in a similar way whereas the effort for the Non-CAP process increases much stronger. However, the right side with its finer resolution (interval $[0; 0.2]$) shows that for very low conflict probabilities the effort for the CAP process exceeds the effort for the Non-CAP process.

The results can be explained by analyzing the frequencies of each activity in the three process models. In the CAP and reference process an increasing conflict probability raises only the frequency of the integration test that is performed when the project leaves the CAP. In the Non-CAP process, however, the conflict probability also affects the module test. As the test activities constitute *nested loops* in the process this leads to a much stronger increase of the overall effort. It is also obvious that the CAP process has higher costs than the Non-CAP process for very small conflict probabilities as the cost for creating and maintaining the CAPs occurs independent of the conflict probability. This meets the expectation that CAPs are obsolete if there are no conflicts.

4.6 Estimation of the Conflict Probability

As the results of the process analyses show, the final decision on the economic efficiency of the CAP mechanism depends on the conflict probability parameter c. To determine the conflict parameter we analyzed the average number of dependencies among mainframe programs and examined the number of actual changes of these programs by using the configuration management system. The latter is important as program-to-program dependencies do cause conflicts only if both programs are modified at the same time.

For the analyzed period of one year we found that $\approx 55,86$ *relevant* (i. e. with possible conflict) changes occur for every program every year. Given a work year of 200 days this resolves to 0.279 relevant changes a day. As the reference process predicts about 100 test activities in a year this finally leads to a conflict probability of $0.279/2 = 0.1395$ or 14%.

Note that this does only regard program but *not* data dependencies. For data, the conflict probability is obviously dramatically higher.

5 Results and Discussion

The process analysis and the estimation of the conflict probability leads to the following conclusion:

> *Projects with an average number of dependencies save about* 20% *of total effort through using the* Cap *isolation mechanism as they avoid additional process cycles and conflict resolution activities.*

We therefore recommended to use non-isolated development only for projects with no or very few dependencies. Although we do not have a formal external validation of our results we can say that our results fully correspond with our project partners' experiences. In addition to this this recommendation was already followed before this study was conducted, as project managers intuitively chose isolated development only for projects with zero or few dependencies.

Although it is not detailed here, the difference between Cap and Non-Cap is even stronger with respect to time-to-system.

A new insight gained from this study regards the validity of test results if projects perform tests on shared data. As this drastically increases the conflict probability, enormous efforts are needed to ensure the validity of test results.

The major threats to the validity of these results is the determination of the transition probabilities and the memoryless nature of Markov chains.

Transition Probabilities. To evaluate how strongly different transition probabilities influence the results we performed a sensitivity analysis [5] to determine the transition that has the highest influence on the result. Using the variance-based *Extended FAST Method* [6] we found the transition *Module Test* → *Integration Test* to be not only the most *important* but with an *total order index* of 0.72 about three times as important as the second ranked transition. We therefore focused our analysis on the most important transition probability and found that changes to this probability do of course change the absolute efforts calculated for each process model. They do, however, *not* change the relation between Cap and Non-Cap development processes.

Memorylessness. The memorylessness of Markov chains implies that the transition probability from e. g. *Module Test* to *Implementation Test* and others is always the same, no matter how often the activities have been carried out before. As this might contradict one's intuition, we evaluated the influence of memorylessness by introducing a *process memory* in form of a compound interest function for the activity efforts. By defining a negative interest rate *(reduction rate)* we could simulate a situation where each execution of an activity demands less effort than the previous execution. Repeating the analysis for the two process models with this process memory showed again that the memory does influence the absolute results but *not* invalidate the relation between the Cap and Non-Cap development processes.

6 Related Work

Numerous *empirical studies* were conducted to answer similar process-related research questions, e. g. [7,8,9,10]. In general, empirical research generated highly valuable data that also helped us in calibrating the reference process model. However, empirical studies have a number of drawbacks that rendered them unsuitable in our situation. As it is impossible to replicate the same development project with two different processes (e. g. Cap vs Non-Cap) without changing any other influencing parameter, an empirical study would have to be carried out on *similar* projects. Due to the size and complexity of mainframe software development projects it is very hard to control their *similarity* and to correctly interpret the observations. As this could be overcome only by a significant number of repetitions of such studies, reliable results could be expected only after investing enormous amounts of time and effort [11,12,13,14].

Due to these reasons we chose to use an approach based on *process simulation*. Similar approaches where presented as early as in the 1950ies with the *Critical Path Method (CPM)* and PERT [15]. More recent approaches were presented (among others) by Drappa and Ludewig [16], Madachy [17], Podnar and Mikac [18], Zhang et al. [19] and Mockus et al. [20]. While all of these approaches served as highly valued inspirations, they are either of qualitative nature [19], too specific to their original application [20], do not consider conflict probabilities and project cycles [15,16,18] or were too fine-grained for our purpose [17]. Overviews on process simulation techniques can be found in [21] and [22].

Markov chain-based process simulation models were proposed earlier by Kulkarni and Adlakha [23], Hardie [2] as well as Minh and Bhaskar [24]. Kulkarni and Adlakha focus on the project completion time of PERT networks and do therefore analyze *acyclic* process models only. Hardie specifically includes cyclic process models and concludes that the reluctance to model project cycles is one of the main reasons for flawed predictions. However, he does not use absorbing Markov chains to calculate expected project efforts. Minh and Bhaskar extend Hardie's work by using absorbing Markov chains to determine the expected number of process steps but do not include analysis of project efforts. To our knowledge, neither of the above authors applied their approaches in an industrial context. Padberg [25] presented a model that is based on a Markov decision model to evaluate scheduling strategies. This approach does not model activities explicitly and could therefore not be used to determine the project effort in our case.

7 Conclusions and Future Work

While it is usually easy to determine the costs of specific techniques or methods applied in software development, it is almost always extremely hard to quantify the *economic* benefit of such measures. As decisions for or against such measures should be economically justified, this is a serious problem in today's software engineering practice.

To answer the question about the economic benefit of isolated test and development environments in mainframe software development we developed a stochastic process simulation that explicitly describes project risks and activity reiterations. We demonstrated how this model can be used to compare process variations and found that isolated test environments typically save $\approx 20\%$ development effort in the setting analyzed.

We believe that the incorrect predictions for project time and cost frequently encountered in practice are mainly due to project managers' reluctance to address project risks caused by unplanned reiterations of development activities. Therefore our current and future work focuses on applications of the model in the field of software project cost estimation. In this context we are working on the completion of the tool-suite that allows process design and analysis based on the methods presented in this paper.

References

1. Bacon, J.: Concurrent Systems: Operating Systems, Database and Distributed Systems: An Integrated Approach. Addison-Wesley, Boston, MA, USA (1993)
2. Hardie, N.: The prediction and control of project duration: a recursive model. International Journal of Project Management **19**(7) (2001) 401–409
3. Grinstead, C.M., Snell, J.L.: Introduction to Probability. AMS (2003)
4. Boehm, B.W.: Software Engineering Economics. Prentice Hall PTR, Upper Saddle River, NJ, USA (1981)
5. Saltelli, A., ed.: Sensitivity Analysis. John Wiley & Sons (2000)
6. Saltelli, A.: A quantitative model-independent method for global sensitivity analysis of model output. Technometrics **41**(1) (1999) 39–56
7. Williams, L., Kessler, R., Cunningham, W., Jeffries, R.: Strengthening the case for pair programming. Software **17**(4) (2000) 19–25
8. Phongpaibul, M., Boehm, B.: An empirical comparison between pair development and software inspection in Thailand. In: ISESE '06, ACM Press (2006)
9. Dingsøyr, T., Røyrvik, E.: An empirical study of an informal knowledge repository in a medium-sized software consulting company. In: ICSE '03, IEEE CS (2003)
10. Du, G., McElroy, J., Ruhe, G.: A family of empirical studies to compare informal and optimization-based planning of software releases. In: ISESE '06, ACM Press (2006)
11. Perry, D.E., Porter, A.A., Votta, L.G.: Empirical studies of software engineering: a roadmap. In: ICSE '00, ACM Press (2000)
12. Pfleeger, S.L.: Albert einstein and empirical software engineering. Computer **32**(10) (1999) 32–38
13. Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., El Emam, K., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**(8) (2002) 721–734
14. Seaman, C.: Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. **25**(4) (1999) 557–572
15. Malcolm, D.G., Roseboom, J.H., Clark, C.E., Fazar, W.: Application of a technique for research and development program evaluation. Operations Research **7**(5) (1959)
16. Drappa, A., Ludewig, J.: Quantitative modeling for the interactive simulation of software projects. The Journal of Systems and Software **46**(2–3) (1999) 113–122

17. Madachy, R.J.: System dynamics modeling of an inspection-based process. In: ICSE '96, IEEE CS (1996)
18. Podnar, I., Mikac, B.: Software maintenance process analysis using discrete-event simulation. In: CSMR '01, Washington, DC, USA, IEEE CS (2001)
19. Zhang, H., Huo, M., Kitchenham, B., Jeffery, R.: Qualitative simulation model for software engineering process. In: ASWEC '06, IEEE CS (2006)
20. Mockus, A., Weiss, D.M., Zhang, P.: Understanding and predicting effort in software projects. In: ICSE '03, IEEE CS (2003)
21. Kellner, M.I., Madachy, R.J., Raffo, D.M.: Software process simulation modeling: Why? what? how? Journal of Systems and Software **46**(2-3) (April 1999) 91–105
22. Williams, T.: The contribution of mathematical modelling to the practice of project management. IMA J Management Math **14**(1) (2003) 3–30
23. Kulkarni, V.G., Adlakha, V.G.: Markov and markov-regenerative pert networks. Operations Research **34**(5) (1986) 769–781
24. Minh, D.L., Bhaskar, R.: Analyzing linear recursive projects as an absorbing chain. Journal of Applied Mathematics and Decision Sciences (2006)
25. Padberg, F.: A comprehensive simulation study on optimal scheduling for software projects. In: ProSim '04, IEE (2004)

Deriving a Valid Process Simulation from Real World Experiences

Christoph Dickmann[1], Harald Klein[2], Thomas Birkhölzer[3], Wolfgang Fietz[1], Jürgen Vaupel[1], and Ludger Meyer[2]

[1] Siemens Medical Solutions, Postfach 3260, 91050 Erlangen, Germany
{christoph.dickmann,wolfgang.fietz,juergen.vaupel}@siemens.com
[2] Siemens CT SE 3, Otto-Hahn-Ring 6, 81730 München, Germany
{h.klein,ludger.meyer}@siemens.com
[3] University of Applied Sciences Konstanz, Braunegger Str. 55, 78462 Konstanz, Germany
thomas.birkhoelzer@htwg-konstanz.de

Abstract. This paper presents a systematic approach to develop and configure a process simulation model that relates process capabilities to business parameters in order to support process improvement projects within Siemens. The research work focuses on the systematic set up of a validated and acknowledged model that matches the company's process improvement needs by involving experts to adapt an existing mathematical framework and simulation application. The methodology consists of three complementary steps: An approved conceptual model is used as structural skeleton, quantitative parameters are derived by a prospective expert survey, and final adaptation and customization is facilitated in order to be useable for process experts themselves (instead of model developers).

Keywords: simulation; software process improvement; capability maturity model; CMMI; balanced scorecards; validation; expert survey; process knowledge; simulation model customization.

1 Introduction

Process improvement needs considerable investment and normally results in changes to critical software development steps. Hence, it requires a solid justification from a business management and software development viewpoint. Many development organizations, however, face the problem that process improvements are generally considered to be beneficial activities, although in most cases the prospective results and alternatives are not estimated or compared on a systematic quantitative basis. Existing reports of quantitative outcomes usually cover only one improvement scenario without providing insights into alternatives [11].

Quantitative process simulation is considered a means to face this problem by describing and calculating a complex real world system in a simplified way in order to enable process owners and management stakeholders to test different process improvement approaches [2], [4], [5], [6].

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 272–282, 2007.

However, most of the existing simulation approaches model software *project* performance and evaluate improvement options by comparing project performance based on different settings of the model, for a comprehensive overview see [2].

This work sets up a model on the abstraction level of the organization as a whole in which model inputs represent CMMI-based process improvement actions and model outputs represent business measures of the organization. The simulation model relates capabilities of key process areas, e.g. assessed as CMMI levels [10], with business outcomes measured by core metrics such as defined in a company's Balanced Scorecard [1]. This approach is similar to [5] but is extended to a comprehensive scope.

Such an approach crucially depends on the trustworthiness of the underlying model. Therefore, it was the objective of this research project to set up a validated and credible process simulation model for Siemens by leveraging the existing knowledge of process experts from different company units. The methodology consists of three steps:

1) An approved conceptual model defining process areas, business metrics and their relations is used as structural skeleton,
2) quantitative parameters are derived from a prospective expert survey in selected and representative development organizations, and
3) a final adaptation and customization is facilitated in order to be useable for process experts themselves (instead of model developers), thus enabling an easier, more direct and therefore better customization.

The resulting simulation model consists of two sets of entities: process areas and business metrics. The first set of entities is the basis for investments of the process improvement budget, and is essentially based on the CMMI process areas combined with special focus areas derived from the company's business needs. Those process areas are aggregated to a metrics level, which represents the second entity set. Metrics entities are calculated by using weighted paths from process level to metrics level. Those paths are quantitatively derived from the prospective expert survey mentioned above. Results of the simulation are presented using the Balanced Scorecard (BSC) methodology [1] grouped in the well-established four categories Process, Quality, Customer and Financial. These metrics represent the level of achievement of the organization's business goals, and, therefore, quantitatively indicate the benefit resulting from process improvement.

In section 2 we sketch the mathematical basis of this research work and a tool implementation (for details see [7], [9], which also present simulation results). This covers the mathematical concepts as well as the simulation implementation. Section 3 talks about the configuration approach, which is necessary for setting up a valid model. Thereby the derivation of process areas and metrics from the business needs are explained. Section 4 describes the customization approach and section 5 provides a discussion of the achievements and open issues.

2 Background

In [7] and [9] a generic mathematical process simulation framework was evaluated and presented, which is designed to simulate software development organization in the context of process changes (i.e. process improvements).

Time-discrete, nonlinear, first-order equations relate inputs $\hat{u}_t = (\hat{u}_{1,t},\ldots,\hat{u}_{n,t}) \in \mathbf{R}^n$, normalized internal state variables $\hat{x}_t = (\hat{x}_{1,t},\ldots,\hat{x}_{n,t}) \in \mathbf{R}^n$ and normalized outputs $\hat{y}_t = (\hat{y}_{1,t},\ldots,\hat{y}_{m,t}) \in \mathbf{R}^m$ as follows:

$$\hat{x}_{i,\,t+1} = (1-\lambda_i)\cdot \mathrm{n}(\hat{x}_{i,\,t}) + \lambda_i \cdot s_{i,\,t-\tau_i} \;; \tag{1}$$

$$s_{i,t} = \alpha_i \cdot \hat{u}_{i,t} + \sum_j \beta_{ij}^{(x)} \cdot \mathrm{n}(\hat{x}_{j,t}) + \sum_j \gamma_{ij}^{(x)} \cdot \mathrm{g}(\mathrm{n}(\hat{x}_{l_{ij},t}), \nu_{ij}^{(x)}, \mu_{ij}^{(x)}) \cdot \mathrm{n}(\hat{x}_{j,t}) \;; \tag{2}$$

$$\hat{y}_{i,t} = \frac{\sum_j \beta_{ij}^{(y)} \cdot \mathrm{n}(\hat{x}_{j,t}) + \sum_j \gamma_{ij}^{(y)} \cdot \mathrm{g}(\mathrm{n}(\hat{x}_{l_{ij},t}), \nu_{ij}^{(x)}, \mu_{ij}^{(x)}) \cdot \mathrm{n}(\hat{x}_{j,t})}{\left(\sum_j \beta_{ij}^{(y)} + \sum_j \gamma_{ij}^{(y)}\right)} \;; \tag{3}$$

$$\mathrm{n}(arg) = \begin{cases} 1 & \text{for} \quad arg > 1 \\ arg & \text{for} \quad 0 \le arg \le 1 \\ 0 & \text{for} \quad arg < 0 \end{cases} \;; \tag{4}$$

$$\mathrm{g}(arg, \nu, \mu) = \frac{1}{1 + e^{-\nu(arg-\mu)}} \;. \tag{5}$$

The output variables $\hat{y}_t$ are intended to represent business metrics. The internal state variables $\hat{x}_t$ represent process area capabilities, e.g. assessed as CMMI levels [10]. The inputs $\hat{u}_t$ represent investments in process areas that change model states and output values, correspondingly. These variables are determined by the structure of the organization to be modelled, e.g. which metrics are used, see section 3.2. Temporal process dynamics and inter-process dependencies are represented mainly by equations (1) and (2) and defined by the parameters $\lambda, \tau, \alpha, \beta, \gamma, \nu, \mu$. Equation (3) is the basis for calculating normalized outputs as weighted sums of model states, and (4) and (5) are auxiliary functions for normalization and gating. The gating function as specified by equation (5) is used to model that a certain capability level of one process area is a prerequisite of the effects of another process area.

The framework described by equations (1) to (5) is similar to classical system dynamics by the use of continuous state variables and time based simulation. However, time continuous differential equations are replaced by time discrete difference equations (1). The latter is a standard simulation technique in order to reduce implementation and computation complexity without loss of generality.

In [7] and [9] it was shown that this framework is able to produce interesting and plausible simulation behavior in principle. However, to derive a valid model representing real world organization behavior within this framework the following steps are necessary (see section 3 for details):

1. Identification of the real world set of internal states governing the organization.
2. Identification of the real world outputs indicating the organization's production and business.
3. Identification of the relations between states and outputs.
4. Quantification of the model parameters describing state dynamics (λ_i, τ_i), as well as relational strengths ($\alpha_i, \beta_{ij}^{(\bullet)}, \gamma_{ij}^{(\bullet)}$) and characteristics ($\nu_{ij}^{(\bullet)}, \mu_{ij}^{(\bullet)}$).

The mathematical framework defined by equations (1) - (5) is implemented in an interactive, Java-based simulation application [4].

The user interface allows entering the input investments u_t interactively at a simulation step (representing a time period). Then, the simulation can proceed for a selectable number of simulation steps and the effects of the investments onto the business parameters are graphically displayed resembling score cards, see Fig. 1. Additional reports for later analysis can be generated as well. A more detailed description and simulation results can be found in [4], [7].

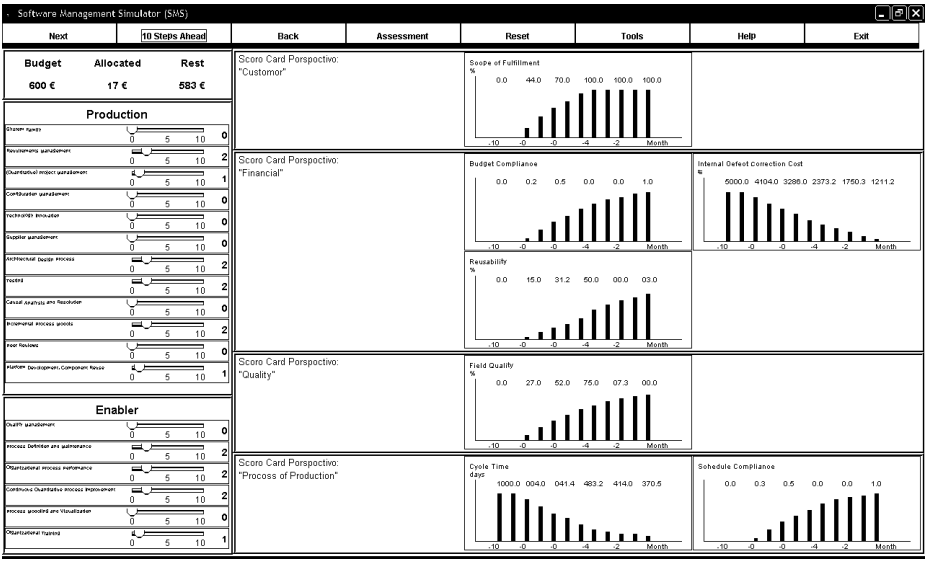


Fig. 1. Screenshot of the simulation application. The input area (investment in process areas to be improved) is on the left side, the output score card (business metrics) display in the center.

3 Deriving a Validated Model for the Simulation

3.1 Methodology

The question of model validity and simulation validation has been an ongoing challenge for most of the software process simulation efforts [3]. Methodologically, the simplest case is to prove congruence of the simulation results with real world input-output-data.

However, this type of validation is almost never achieved because of "lack of data" from software development projects. Moreover, this "lack of data" is not due to the shortcomings of the respective research effort (which might be overcome by a better approach), but poses a principal methodological problem: software producing organizations are time-varying and can not be experimentally tested for different scenarios. Therefore, real world input-output-data is anecdotic in the best case, able to support or discourage a simulation model, but will hardly suffice for strict validation.

On the other hand, most organizations have built up a substantial amount of process knowledge and experience over time. Process areas, metrics and procedures might have been originally derived from external sources like CMMI or RUP, but were also tested, used and modified during the course of many projects and organizational evolutions. In this paper, it is suggested and demonstrated, that a *valid* process simulation model can be build upon this knowledge. This is done by two steps: The structure of the simulation model is derived from an existing and approved conceptual process model and the quantitative parameters are estimated by a prospective expert survey.

As a final, not yet completed step, the overall behaviour will be assessed by a retrospective expert appraisal.

3.2 Model Structure Based on Process Knowledge

At Siemens, there is a central software engineering group which supports other organizational units with respect to software engineering methodology in general and process improvements in particular.

Within this group, the resulting process knowledge has been condensed and solidified into agreed upon and adopted sets of

- core process areas, see Table 1
- business metrics, see Table 2 and
- relationships between process areas and business metrics.

The core process areas are essentially based on CMMI combined with specific focus areas derived from the Siemens business needs. CMMI defines 22 key process areas [10]. The Siemens software process model defines 18 process areas, of which 6 process areas are general enabler processes that indirectly influence company

Table 1. Process areas used for the simulation model based on an existing conceptual model

Process Areas	
System Family	Incremental Process Models
Requirements Management	Peer Reviews
(Quantitative) Project Management	Platform Development, Component Reuse
Configuration Management	Quality Management
Technology Innovation	Process Definition and Maintenance
Supplier Management	Organizational Process Performance
Architectural Design Process	Continuous Quantitative Process Improvement
Testing	Process Modeling and Visualization
Causal Analysis and Resolution	Organizational Training

development and business by laying the ground for a systematic process-based development. The company practice and experience has resulted in focusing on this set of adapted process areas, which adequately represent the way how software development is done in this company.

The set of seven business metrics were synthesized from software development of different company units, some of which are similar to metrics that are commonly used in the software industry, like cycle time or schedule fidelity [11].

Originally, this conceptual model was developed for process improvement consulting purposes, not for simulation. Accordingly, these sets are not validated in a formal manner, but have been proven useful within different practical and project contexts and are generally acknowledged and accepted throughout the company.

This structure is used unchanged for the simulation model: The key process areas are mapped onto the internal states $\hat{x}_t$, the business metrics onto the outputs $\hat{y}_t$, and the relationships onto the existence of weights $\beta_{ij}^{(\bullet)} \neq 0$ (which still need to be quantified, see Section 3.3).

Table 2. Metrics used for the simulation model based on an existing conceptual model

Metrics	
Scope of (Requirement) Fulfillment	Schedule Compliance
Budget Compliance	Internal Defect Correction Cost
Field Quality	Reusability
Cycle Time	

3.3 Model Parameter Estimation Based on Expert Knowledge

The major difference between a conceptual process model and a simulation model (i.e. a model, which can be enacted) is the quantification of behaviour: Conceptual models often show qualitative aspects only, simulation models need complete quantification. As stated above, it is one goal of this research to quantify the existing conceptual model (see Section 3.2).

Expert surveys are an established method to estimate quantitative values, which can not just be deduced from project or production data. Therefore, a structured questionnaire (together with an accompanying motivation and instruction manual) was developed, which was sent to process experts from different organizational company units.

Specific care was taken to phrase the survey questions in the language of the process experts, not using mathematical terminology or formulas in the questionnaire. Questions were designed with seven-point scaled alternatives to be checked concentrating on one aspect of the model at a time, e.g. the time variations (agility) of one process area or the strength of a particular relation. Overall, the questionnaire contained questions on 126 model quantifications and took about 40-90 minutes to complete. All process experts provided estimates to all of these process- and business-related parameters that are well-established in the company.

The median of the answers from 18 completed interviews were mapped to the respective parameters resulting in the intended simulation model. The presentation of

these detailed company-specific results would, however, exceed the scope of this publication.

The median was used because it is less sensitive to singular outliers. As expected, the answers show some degree of variance. Causes might be: differing expert opinions, diverse work backgrounds or individual answer behavior to the questions. It might be interesting to further analyse this variance in order to distinguish subgroups. However, the statistical basis (18 samples) is not yet sufficient for such an attempt.

3.4 Next Steps

While the prospective expert survey results discussed above (i.e. process concepts, relations and relation strengths known prior to the existence of the simulation model) provide justification for the isolated mechanisms of the model (relations and dynamics), the resulting overall behaviour can only be assessed based on simulation results of the entire model. This work is currently ongoing (an can not yet be reported). It will be divided in two tasks:

- Typical (meaningful) simulation scenarios will be prepared and presented to process experts for appraisal in the sense of "This scenario conforms to my notion of real world behaviour (answer: yes or no)."
- Stochastically distributed input patterns will be applied to scan for apparently erroneous behaviour within "less used" parts of the simulation domain. This resembles the proposal in [8] using an optimization approach to search for such outliers.

4 Customization

Even a validated simulation model represents just one instance of an organization, in a more or less representative way. In order to be used in different contexts, it needs to be adapted or customized.

Of course, all simulation models can be customized in principle by simulation experts or model developers. However, the challenge is to bring this ability to the process experts themselves in order to increase speed and flexibility of model evolution as well as acceptance.

Within this project, this is achieved by two means: Flexible mapping of normalized variables and a spreadsheet-based configuration interface.

4.1 Mapping of Normalized Variables

The mathematical equations (1) - (5) combine very different variables, e.g. the capability of a process area measured in CMMI levels with metrics measured in percentage (Scope of Fulfillment) or days (Cycle Time). Moreover, values for some variables rise in case of improvements (e.g. Scope of Fulfillment) while others decline (Cycle Time). Theoretically, the translation between these units might be incorporated in the weight factors $\beta_{ij}^{(\bullet)}$. For practical configuration, however, this would blend two issues: the relative strength of an interaction and the translation or conversion of units. Experience has shown that it would require the knowledge and skills of a developer who

is well familiar with the model to comprehend and handle such interdependencies correctly.

Normalization is an acknowledged modeling technique to avoid such dimension and scale conversion and translation. Therefore, all computational model variables $\hat{u}_t, \hat{x}_t, \hat{y}_t$ are normalized to the dimensionless interval $[0,1]$: equations (1) - (5) are already formulated accordingly. All parameters uniformly refer to this scale basically representing relative weights. Thus, parameter values can be much more easily customized or adjusted by process experts without in-depth modeling experience, because unit conversion or translation issues are separated from interaction strength.

However, although such normalized variables carry all information of the simulation results, the addressees of the simulation results – process experts and managers – are not accustomed and prepared to interpret "normalized metrics" etc. This would immediately convey the impression of "not adjusted to our problem". Therefore all computational variables $\hat{u}_t, \hat{x}_t, \hat{y}_t$ are mapped to a configurable range by a simple linear transformation.

$$\text{map}(arg) = \begin{cases} m_{arg} & \text{for} \quad arg < 0 \\ m_{arg} + (M_{arg} - m_{arg}) \cdot arg & \text{for} \quad 0 \le arg \le 1 \\ M_{arg} & \text{for} \quad arg > 1 \end{cases} \tag{6}$$

The parameters m_{arg} and M_{arg} can be configured for each variable separately, e.g. as 0 and 100 for a variable measured in percentage as Scope of Fulfillment or as the minimal and maximal Cycle Time measured in days. Therefore, these parameters provide the simplest level to customize the general model to the context of a specific organizational unit.

4.2 Spreadsheet-Based Configuration

In order to promote usage as well as acceptance of process simulation, it is helpful to align the tools with the context and the experience of the customer, not the other way round. However, process experts (and managers) do neither use expert-centered simulation tools nor technologies like XML (as the model storage format proposed in [4]) on a regular base. Customization mechanisms on such a rather technology-centered level would hardly be used by these target groups.

Therefore, it was an important part of this project to create a seamless spreadsheet user interface (in this case a Microsoft Excel form) for customization of all model parameters. The goal is to enable software development process experts to adapt and customize the model by staying in their familiar tool context, i.e. spreadsheets, and not force them to use less known simulation tools or XML editors. While this seems to be a rather technical issue – the respective software modules need to be designed and programmed – it has the potential to create a new usage scenario and environment for process simulation modeling reaching beyond the current boundaries of simulation model development (i.e. direct process knowledge acquisition). First experiences of this approach (beta testers) are promising.

5 Discussion

We aim at supporting a large and globally operating process organization to better plan software process improvement projects: software development process owners and management stakeholders responsible for related budgets should be enabled to test different process improvement approaches by using a simulation tool. This simulation needs to reflect the relevant aspects of day-to-day business and performance controlling, otherwise it lacks the power to convey adequate insights to be of practical use.

In order to develop a trustworthy model that gets accepted by process experts in the company for this purpose, we used a process model that is based on experiences in software development and CMMI usage in a global software developing company. Thereby, explicit representations from implicit knowledge, experience or assumptions existing within the company were created compared to e.g. a theoretical or literature-based model. The experts were able to estimate these model parameters known from their practice. However, they were aware that their individual experience may vary from the experience in other organizational units of the company. It has been clearly communicated that this survey contributes to the model to that effect that it is a first approach of describing their work reality in a model.

Practice and experience within Siemens has resulted in focusing on a specific set of adapted process areas, which adequately represent the way how software development is done in this company. The second company-specific adaptation relates to the development and business metrics that are used to quantify the effects of investing in development or enabling processes. The dynamics of process changes following investment and the inter-process relations were quantified by getting estimates from software process experts from different units of Siemens. As a result, a complete and end-to-end mathematical model of this software-producing organization was configured. Such a complete model reveals a considerable number of additional important connections and issues, compared to individual knowledge about many of the single pieces (single processes) of such a model that are well known. To our knowledge, such real-world configuration and process modelling is not normally done in process improvement projects.

Often, process improvement evaluation projects compare software development metrics with CMMI improvement, e.g. before and after process improvement projects. Our work goes one step further in that it defines a simulation tool based on real-world quantitative estimations that allows for experimenting with different process improvement project scenarios and thereby can show effects of process investments that fit company ways of working.

Of course, the model is a high-level abstraction and simplification of reality, therefore all results and conclusions drawn from them need to be critically examined and used with due care. Nevertheless, the simulation tool can be used for organizational development: The results and insights from simulations can serve as challenge and guideline for process practitioners and management to adapt the company's processes and process metrics, as well as to change the organization, e.g. by changing the staffing of company units, by adding or changing supplier relations, etc.

In a subsequent step, we plan to generate case studies that validate the simulation tool against concrete process improvement projects from the company by comparing business performance before and after process changes.

The mathematical model that is configured by real-world values from different branches and organizational units of Siemens represents an average profile for process improvement at Siemens. About 180 parameters were set in total, of which ca. 100 can be assumed to be rather stable throughout Siemens, so that they would not need to be adapted for most of the organizational settings that are simulated.

However, other parameters need adaptations if the simulation tool is to be used in a specific company branch or unit. Therefore, a flexible customization of the simulation tool was realized.

For further validation of the model configuration, it is planned to compare simulation results with metrics and development or business performance results from company process improvement projects in a before-after manner.

The results of the experts' model paramater estimations will be fed back into the company's process model in order to reflect about assumptions and check for needed updates. This parameter data will also be used to identify models that fit more specific subgroups of the company's process environments.

In a next step, it also seems suitable to investigate how the configuration needs to be adapted to compare traditional development processes with newly emerging agile and SCRUM-based software development processes.

6 Conclusion

This work demonstrates an approach to develop valid and trustworthy simulation model. The model is drawn from an existing conceptual model and augmented by quantitative dependencies by transforming implicit knowledge of process expert into explicit quantitative relations.

This adapted process model will be used to support software development managers and process practitioners in finding strategies to change and improve existing software development practice.

References

1. Kaplan, R.S., Norton, D.P.: The Balanced Scorecard: Translating Strategy into Action. Harvard Business School Press (1996)
2. Raffo, D.M., Kellner, M.I.: Modeling Software Processes Quantitatively and Evaluating the Performance of Process Alternatives. In: K. E. Emam and N. Madhavji (eds.): Elements of Software Process Assessment and Improvement. IEEE Computer Society Press, Los Alamitos, California (1999) 297 -341
3. Raffo, D.M., Kellner, M.I.: Empirical Analysis in Software Process Simulation Modeling. Journal of Systems and Software **53** (2000) 31-41
4. Birkhölzer, T., Dantas, L., Dickmann, C., Vaupel, J.: Interactive Simulation of Software Producing Organization's Operations based on Concepts of CMMI and Balanced Scorecards. Proceedings of the 5th International Workshop on Software Process Simulation and Modeling (ProSim 2004), Edinburgh (2004) 123-132
5. Pfahl, D., Stupperich, M., Krivobokova, T.: PL-SIM: A Generic Simulation Model for Studying Strategic SPI in the Automotive Industry. Proceedings of the 5th International Workshop on Software Process Simulation and Modeling (ProSim 2004), Edinburgh (2004) 149-158

6. Raffo, D.M, Nayak, U., Setamanit, S., Sullivan, P., Wakeland, W.: Using Software Process Simulation to Assess the Impact of IV&V Activities. Proceedings of the 5th International Workshop on Software Process Simulation and Modeling (ProSim 2004), Edinburgh (2004) 197-205
7. Birkhölzer, T., Dickmann, C., Vaupel, J., Stubenrauch, J.: Towards an Interactive Simulator for Software Process Management under Uncertainty. Proceedings of the 6th International Workshop on Software Process Simulation and Modelling (ProSim 2005), St. Louis (2005) 169-174
8. Wakeland, S., Raffo, D.M.: Heuristic optimization as a V&V tool for software process simulation models. Software Process Improvement and Practice **10-3** (2005) 301-309.
9. Birkhölzer, T., Dickmann, C., Vaupel, J., Dantas, L.: An Interactive Software Management Simulator based on the CMMI Framework. Software Process Improvement and Practice **10-3** (2005) 327-340
10. CMMI Product Team: CMMI for Development, Version 1.2. CMMI-DEV, V1.2, CMU/SEI-2006-TR-008, Pittsburgh (2006)
11. Galin, D., Avrahami, M.: Are CMM Program Investments Beneficial? Analyzing Past Studies. IEEE Software **23-6** (2006) 81-87

Project Delay Variability Simulation in Software Product Line Development

Makoto Nonaka[1], Liming Zhu[2], Muhammad Ali Babar[3], and Mark Staples[2]

[1] Faculty of Business Administration, Toyo University, Japan
nonaka-m@toyonet.toyo.ac.jp
[2] National ICT Australia
{liming.zhu,mark.staples}@nicta.com.au
[3] Lero, University of Limerick, Ireland
Muhammad.alibabar@ul.ie

Abstract. The possible variability of project delay is useful information to understand and mitigate the project delay risk. However, it is not sufficiently considered in the literature concerning effort estimation and simulation in software product line development. In this paper, we propose a project delay simulation model by introducing a random variable to represent the variability of adaptive rework. The model has been validated through stochastic simulations by comparing generated adaptive rework to an actual change effort distribution, and by sensitivity analysis. The result shows that the proposed model is capable of producing reasonable variability of adaptive rework, and consequently, variability of project delay. Analysis of our model indicates that the strength of dependency has a larger impact than the number of residual defects, for the studied simulation settings. However, high levels of adaptive rework variability did not have great impact on overall project delay.

Keywords: process simulation, software product line development, product quality, project planning.

1 Introduction

Software Product Line (SPL) development can shorten the total cycle time, the duration from the beginning of core asset development to the end of product development, by achieving large-scale reuse [1]. However, effort estimation, planning, and development management for SPL are more complex and difficult than those for sequential development, because of inter-connected relationships between core assets and products, concurrency of their projects, and multiple deadline management [2]. In addition, there are still general problems with software effort estimation errors such as unplanned work [3] as well as requirements volatility [4]. The total cycle time can sometimes be longer than initially planned because of these problems.

One source of unplanned work is poor quality of software artifacts. A certain number of defects will inevitably remain in released core assets, as software testing can not demonstrate the absence of defects [5]. When residual defects in core

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 283–294, 2007.

assets are detected after their release to product projects (not to customers), corrective maintenance[1] is usually performed[2] to modify the core assets. When multiple product projects are undertaken simultaneously during core asset maintenance phase, corrective maintenance in core assets sometimes brings associated rework to all ongoing product projects that depend on the core assets, to adapt the products to the changed core assets. We call this type of rework "adaptive rework".[3]

With regard to this problem, we previously proposed a simulation model for estimating project delay in concurrent software development and conducted a deterministic simulation with fictional project data [7], which did not estimate the variability of project delay. The variability, or the level of risk of project delay is useful information [8] when a project manager wants to understand and mitigate project delay risk. Even in the literature concerning effort estimation and simulation, the level of risk of project delay in SPL development has not been considered enough [9,10,11,12]. In consideration of the variability of project delay, we set the following research questions in this paper. *How much variability of project delay in SPL development is expected when (a) the number of residual defects in core assets changes and (b) the strength of dependency changes?*

To explore these research questions, we propose a simulation model for estimating project delay and its variability by introducing a random variable to represent the duration of adaptive rework. Furthermore, we increase the expressiveness of the model by introducing inter-dependency of core assets. We conducted stochastic simulations with fictional project data with the proposed model.

The reminder of this paper is organized as follows. Sect. 2 describes the proposed simulation model which includes the previous model and the enhanced features. Simulation results and derived implications are described in Sect. 3. Sect. 4 discusses model evaluation. Sect. 5 contains a discussion and describes related work. Concluding remarks are described in Sect. 6.

2 Proposed Simulation Model

Software process analysis approaches can be categorized into the following three types [13]: analytical models such as COCOMO II [9], continuous simulation models [14,15], and discrete-event simulation models [16,17,18]. A discrete-event simulation model is suitable for detailed analyses of process and project performance [19]. As we consider sequential events concerning residual defects and adaptive rework, we apply a discrete-event simulation model to the proposed model.

[1] Corrective maintenance is defined in an IEEE standard [6] as "reactive modification of a software product performed after delivery to correct discovered faults."

[2] In actual practice, not all discovered defects will always be fixed.

[3] The meaning of 'adaptive rework' in this paper and that of 'adaptive maintenance' in an IEEE standard [6] are somewhat different. Adaptive maintenance is defined in [6] as "modification of a software product performed after delivery to keep a computer program usable in a changed or changing environment."

2.1 Primary Factors of the Simulation Model

Suppose that there is a limitation on available resources. To avoid or reduce project delay, the frequency of adaptive rework as well as its duration should be reduced. The frequency is closely correlated with the number of residual defects in core assets. The duration of each piece of adaptive rework will in practice relate to the strength of dependency between core assets and products. This assumption is partly supported by [20,21,22] showing that design complexity has a large influence on maintenance effort. The duration will also relate to what development phase it occurs in. Literature reports that the ratio of the cost of finding and fixing a defect during design, test, and field use is 1 to 13 to 92 [23] or 1 to 20 to 82 [24].

From this discussion, we select the following three factors as primary factors of the simulation model.

1. *The number of residual defects in core assets (NRD).* NRD will depend on product size, product complexity, process quality, and other factors. We assume that NRD can be estimated.
2. *The strength of the dependency (DEP).* We consider DEP between core assets and products as well as among core assets. DEP is represented as a continuous variable that ranges from 0 to 1. DEP = 0 means no dependency, and DEP = 1 means the strongest. In practice, there may be different levels of dependency for different changes, but as discussed below, we use a single DEP value to represent the worst-case dependency.
3. *Work effort multiplier (WEM).* We introduce WEM to represent the ratio of the duration of pieces of adaptive rework for each development phase in which adaptive rework occurs. We assume that each product project follows sequential processes. WEM is represented as a continuous variable that ranges from 0 to 1.

2.2 Determining Adaptive Rework

To determine the duration of each piece of adaptive rework, we first consider defect correction completion time in the core asset maintenance phase that determines the time when adaptive rework occurs. The defect correction completion time can be determined by applying a Software Reliability Growth Model (SRGM) [25]. Suppose that all residual defects in core assets are detected during core asset maintenance phase. If we draw an SRGM curve during the phase, the defect correction completion time of these defects can be determined by assigning a time to each defect along with the curve depending on reliability growth.

Next, we introduce a parameter "worst case adaptive rework" (WCAR). WCAR is supposed to represent the duration of adaptive rework in the following worst-case scenario: (1) the defect correction completion time is at the end of the product project, and (2) DEP is the strongest.

WCAR inherently has a certain distribution, because the duration of WCAR depends on what kind of defects corrected in core assets. Here we introduce a continuous random variable to represent the WCAR distribution. According to

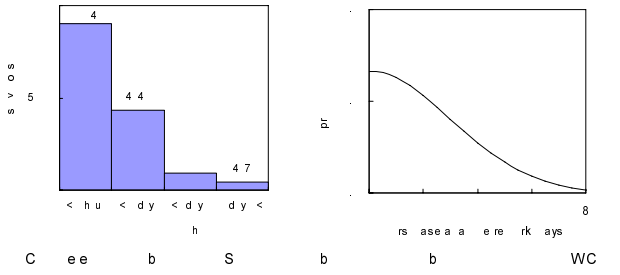


Fig. 1. An actual error correction effort histogram and a distribution for WCAR

the Software Engineering Laboratory (SEL) data subset [26], an effort distribution for error correction has a right-skewed distribution as depicted in Fig. 1 (a). To generate a WCAR distribution like Fig. 1 (a), we use the right-hand half part of a normal distribution (Fig. 1 (b), $\mu = 0$ and $\sigma = 3$, for example). Note that the range of the WCAR distribution is larger than that of the SEL data distribution, as the WCAR distribution represents worst cases of adaptive rework instead of actual change effort.

With these parameters, the duration of each piece of adaptive rework can be determined as follows. The duration of adaptive rework $\Delta r_i(d_j)$ (in months) caused by the defect d_j in the product project i is assumed to be represented by the formula

$$\Delta r_i(d_j) = \mathrm{EffDist}^{-1}_{wcari}(p) \times \mathrm{WEM}_j(t_{d_j}) \times \mathrm{DEP}_{ki} \times \epsilon, \tag{1}$$

where $\mathrm{EffDist}^{-1}_{wcari}(p)$ (in months) is the inverse function of the WCAR effort distribution probability function for the project i. Probability p is given at random. WEM for the project i is represented with $\mathrm{WEM}_j(t_{d_j})$ when the defect d_j correction is completed in core asset maintenance phase at the time t_{d_j}. DEP between the core assets k and the product i (or core assets i) is represented with DEP_{ki}. The parameter ϵ is 1 if t_{d_j} is within the period of the product project i. Otherwise, ϵ is 0.

2.3 Model Assumptions

The simulation model relies on the following assumptions:

1. Adaptive rework occurs at the time when the causal defect is corrected. Actually, this assumption is not true in practice. Defect correction delay has been observed in [27], which reported that 55% of defects were corrected within a few days, 36% within the next week, and the last 9% before customer release or in the next version.
2. The amount of adaptive rework decreases from WCAR, depending on DEP and WEM, which is partly supported by [20,21,22,23,24].
3. Adaptive rework for completed projects is not performed even though later defect corrections in dependent core assets may be performed.

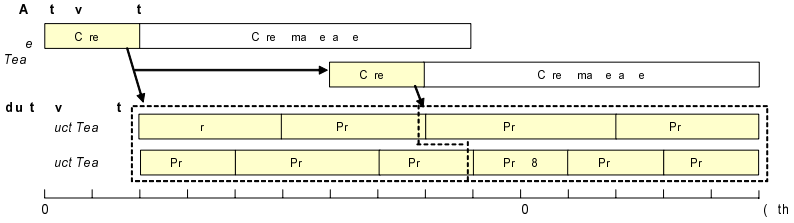


Fig. 2. Time schedule of the fictional SPL development project

4. Products are sequentially developed in planned order by an assigned team with a limited number of resources.
5. The impact of imperfect defect correction during corrective maintenance in core assets and adaptive rework is negligible, which is in practice supported by [27]. That is, it makes little difference on project delay if we do not consider defect correction effort arisen from the another defects that will be injected during those activities.

3 Simulation Results

3.1 Project Data and Parameters

A fictional SPL development project has been studied for simulation. The time schedule of the project is shown in Fig. 2. Arrows in Fig. 2 represent dependency. In this project, 10 products are scheduled to be developed by two product teams concurrently. Core assets are developed, maintained, and enhanced by a core team that is independent of the product teams. Core-2 is an enhanced version of Core-1. Prod-1 to Prod-5 depend on Core-1, while Prod-6 to Prod-10 depend on Core-2. Core-1 maintenance phase is scheduled to be finished at the same time when Prod-5 finishes. The scheduled total cycle time is 15 months. Each pair of successive product projects is scheduled without any buffers. The duration of core asset maintenance phase will be expanded in response to the delayed product projects.

Note that the absolute sizes of core assets and products are not considered here, because they do not directly affect simulation results in the proposed model. Nonetheless, size does affect NRD as described in Sect. 2.1, and DEP might be partly dependent on size.

Several patterns for NRD and DEP have been studied to explore the research questions. For the other parameters, a fixed value or a fixed model is applied.

1. *NRD:* Four patterns of NRD have been studied ranging from 10 to 40 defects in increments of 10 defects. These values are the sum of NRD in both Core-1 and Core-2.
2. *DEP:* We have studied three DEP levels of 0.2, 0.6 and 1.0.

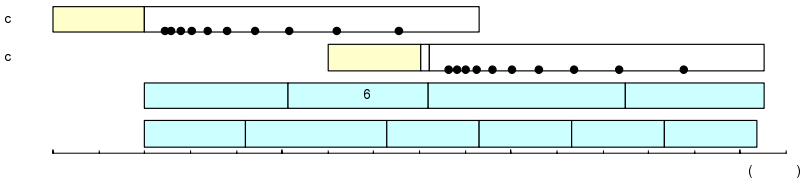


Fig. 3. A simulation result: detail view of project delay

3. *WEM:* By considering the empirical data concerning the cost of defect correction during design and test [23,24], a factor of 20 has been studied. To make the model simple, we use a linear model ranging from 0.05 to 1.0.
4. *Defect correction completion time:* Though numerous SRGMs have been proposed in the literature [25], we apply the following simple logarithmic function

$$y = 1 + \log_a x, \tag{2}$$

where y represents cumulative rate of defect detection, while x represents normalized duration of core asset maintenance phase ($0 < x \leq 1$). In this simulation $a = 20$ is used, which means that 60% of residual defects are corrected before 30% of maintenance phase, and 90% of defects are corrected before 75% of the phase, for example.
5. *WCAR:* We use the distribution pattern in Fig. 1 (b). Note that WCAR is limited up to 8 days in the simulations in order not to generate unrealistically large amount of rework, though a normal distribution has unlimited values.

3.2 Result 1: Detail View of Project Delay and Adaptive Rework

Figure 3 shows a simulation result representing how project delay occurs caused by residual defects in detail (DEP = 0.6, NRD = 20). The dots represent residual defects and their correction completion time. One can see that Core-2 development project is delayed for 0.02 months due to two residual defects detected in Core-1 maintenance phase. The estimated total cycle time is 15.39 months (i.e. a total delay of 0.39 months).

Figure 4 shows the histograms of generated adaptive rework with four combinations of NRD and DEP. Note that each boxplot has a different scale in both x-axis and y-axis. The shapes of the histograms are all skewed to the right, as the WCAR distribution is also right-skewed. The ranges of Fig. 4 (c, d) are quite smaller than those of Fig. 4 (a, b). The ranges of Fig. 4 (a, b) are still smaller than those of the WCAR distribution in Fig. 1 (b), as the WCAR distribution is assumed to have the largest WEM. In Fig. 4 (c, d), all pieces of adaptive rework are completed within one day and most of them are less than 0.2 day (two or three hours) because of weak DEP. The distributions with weak DEP are considered to be a better approximation of actual change effort distribution shown in Fig. 1 (a).

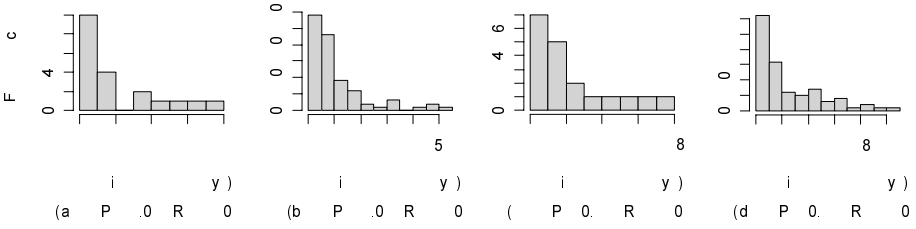


Fig. 4. Examples of generated adaptive rework histograms

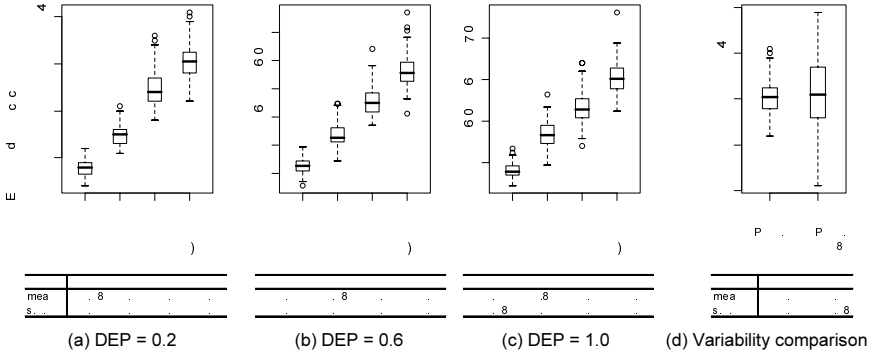


Fig. 5. Simulation results on estimated total cycle time (Note: y-scales are different)

3.3 Result 2: Variability of Project Delay

We conducted 100-run simulations for each combination of NRD and DEP. The boxplots in Fig. 5 (a, b, c) represent the simulation results. The mean and the standard deviation of each combination are shown in the table below the boxplot. Note that each y-axis has a different scale among boxplots.

The results imply that project delay and its variation can be held down if DEP and NRD are low (DEP = 0.2 and NRD = 10). Even for the worst combination in the studied settings (DEP = 1.0 and NRD = 40), the standard deviation of estimated project delay was not very large (0.20). In this case, the range for all data including suspected outliers was from 16.12 to 17.31. As the initial planned time was 15 months, the estimation error rate ranges from 1.07 to 1.15. It means that 8 percentage points of schedule estimation error has appeared in this case at most. It is considered to be in practice quite a small difference for effort or schedule estimation error. When we consider the impact of DEP on durations of pieces of adaptive rework, it sometimes bring larger durations (over one or two days) of adaptive rework as shown in Fig. 4 (a, b). However, the overall effect on project delay is trivial according to the simulation result.

The following is a detailed analysis of the simulation results.

1. *Magnitude of variability:* The standard deviations for DEP = 0.2 are quite small (from 0.02 to 0.04), and even those for DEP = 1.0 are still small (from

0.08 to 0.20). This is because most pieces of adaptive rework are distributed among smaller values regardless of DEP.

2. *Difference of variability in NRD:* The standard deviations of the same DEP slightly increase as NRD increases, because the chance to have more pieces of adaptive rework also increases. By comparing the pair of both (a, b) and (c, d) in Fig. 4, one can see that the frequencies of (b) and (d) are larger than those of (a) and (c) respectively, and that a few but large durations of pieces of adaptive rework are appeared in both (b) and (d).
3. *Difference of variability in DEP:* Similarly, the standard deviations of the same NRD increase as DEP increases. DEP has a stronger impact on variability compared to NRD, when we consider only for the studied simulation settings. This is because different DEPs generate different WCAR distributions, while different NRDs share the same WCAR distribution. The shape of a WCAR distribution is considered as a dominant factor on variability, rather than NRD.
4. *Comparison of variability:* We selected two simulation settings which have almost the same estimated total cycle time but different parameters: (A) DEP = 0.2 and NRD = 40, and (B) DEP = 1.0 and NRD = 8. Fig. 5 (d) shows the comparison results between them. To judge whether the means of both settings are the same, we used Welch's t-test at the 5% significance level. The p-value was 0.40, so we can conclude that there is no statistically significant difference on means between them. However, an F-test showed quite a small p-value $\ll 0.01$. Then we can conclude that there is a significant difference between their variances. This difference mostly comes from the different WCAR distributions, as described in the item 3.

4 Model Evaluation

Because of the nature of simulation study, it is impossible to validate all aspects of the proposed simulation model comprehensively. However, the utility of the model can be evaluated by using empirical data, even though it will not demonstrate comprehensive validation. As we do not have enough empirical data at this moment, we follow four aspects of validation and verification for simulation models [28]: conceptual model validity (between problem entity and conceptual model), computerized model verification (between conceptual model and computerized model), operational validity (between problem entity and computerized model), and data validity.

Conceptual model validity and *data validity:* The proposed model is considered to be reasonably valid under the assumptions described in Sect. 2.3, because the formula (1) is partly supported by several empirical observations as stated in Sect. 2.1 and 2.2. Data validity as input to the model is also supported by these empirical observations in terms of determining the WCAR distribution and WEM. However, there are some limitations of the model. We discuss this topic in Sect. 5.2.

Computerized model verification: We have confidence that the simulation program is accurately implemented because of our precise investigations of the

simulation results (like Fig. 3 and Fig. 4), and because of our inspections of the individual simulation runs including extreme conditions.

Operational validity: In general, operational validity is difficult to assess when no observable problem entity is available. In such a case, comparison to other models and sensitivity analysis are meaningful approaches to validate a simulation model [28]. One possible approach is to compare the proposed model to other effort estimation models. However, this approach is not applicable in this case, because both COCOMO II [9] and COPLIMO [10], a COCOMO II based cost estimation model for SPL development, do not produce variability of estimated effort. These models have a lot of parameters such as effort multipliers, but these parameters are deterministic but not stochastic.

Another possible approach is to evaluate the generated adaptive rework by the simulation program rather than total cycle time. By comparing the distributions of the generated adaptive rework in Fig. 4 (a, b) to the change effort distribution from the SEL data in Fig. 1 (a), both distributions can be subjectively judged to be similar. However, the ranges of Fig. 4 (c, d) are smaller than that of Fig. 1 (a), as DEP has a strong impact on durations of adaptive rework. At least, we can conclude that the simulation model is capable of producing reasonable adaptive rework distributions.

Sensitivity analysis is also a useful approach to demonstrate validity of the model, which we have already discussed in Sect. 3.3. It can be considered that the model has reasonable validity but some limitations described in Sect. 5.2.

5 Discussion and Related Works

5.1 Calibration for Practical Application

When one wants to apply the proposed model in practical situations, the parameters of the model have to be calibrated. NRD and WEM may be able to be estimated easily by investigating one's own organizational defect correction data. The WCAR distribution model might be generated by measuring adaptive rework caused by residual defects. However, DEP will be difficult to calibrate, though it has a stronger impact on duration of adaptive rework compared to NRD, for the studied settings.

In this paper, specific DEP metrics are not assumed. DEP might depend on attributes such as coupling between core components and product components, number of dependent product components reusing a core component, and inheritance depth between core and product components. Those attributes and measured values will be translated into DEP and calibrated by checking generated adaptive rework distributions like Fig. 4.

5.2 Limitations of the Model

The proposed model uses the calendar time scale for the duration of adaptive rework instead of the effort or cost scale, because defect correction completion

time is also represented by using the calendar time scale. Therefore, a project delay always occurs corresponding to any residual defects, even though the durations of pieces of adaptive rework are very short. In practice, such small pieces of adaptive rework may not bring delay, but instead require additional effort or cost. This is one of the limitations of the model in terms of conceptual validity.

In addition, the project delay estimated by the proposed model can not be translated into absolute effort or cost, as the current model does not use those scales directly. However, when considering relative effort or cost estimation error, the current model may be useful as it is.

Moreover, the current model does not explicitly consider resource limitation and resource allocation policies as well. Project delay will be occurred in practice when enough resource are not available. There are other sources of project delay such as unplanned work arisen from requirements change and defect correction. We are in the process of introducing these factors into the simulation model.

5.3 Effort Estimation and Simulation in SPL Development

Several studies have appeared in the literature on estimating the benefits of SPL development [10,29,30]. These studies use more macro-level analytical models compared with our model. The primary purpose of the studies [29,30] is for estimating the return on investment of SPL development compared with non-SPL development. COPLIMO [10] is a deterministic cost estimation model for SPL and does not represent uncertainty, as well as COCOMO II [9]. COCOMO-U [12] introduces uncertainty into COCOMO II, but does not mention how the model can be applied to SPL development.

Chen et al. proposed a discrete-event SPL process simulator using COPLIMO as their base cost model [11]. Schmid et al. studied SPL planning strategies through deterministic simulations [2]. These two studies have similar research questions to ours. However, these studies do not explicitly use factors such as NRD, DEP, and adaptive rework. They are also not capable of calculating the level of risk of estimated effort under uncertainty, as they are based on deterministic simulation models.

6 Conclusions

In this paper, we proposed a stochastic simulation model for estimating project delay and its variability in SPL development. The model has been validated through simulations with fictional project data, by comparing generated adaptive rework to an actual change effort distribution, and by sensitivity analysis. The result shows that the proposed model is capable of producing reasonable variability of adaptive rework, and consequently variability of project delay, even though some limitations exist. Analysis of our model indicates that the strength of dependency, or DEP, has a larger impact on durations of adaptive rework than the number of residual defects, or NRD, for the studied simulation settings. The result shows that the level of risk of project delay can be held down if DEP and NRD are quite small. It will still be held down even though DEP is strong, if

most pieces of adaptive rework do not require large effort. When we consider the impact of DEP, it sometimes bring larger durations of adaptive rework. However, the overall effect on project delay is trivial according to the simulation result.

The future work primarily involves empirical validation of the proposed model, enhancement of the model to overpass the limitations and the model assumptions which constrain the utility of the model, and calibration methods of the parameters. We are in the process of enhancing the model to be capable of estimating absolute effort overruns under specific resource allocation plan as well as its limitation. We are also trying to contact some companies to gather empirical SPL development data that are usable for model evaluation.

Acknowledgments. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 16700042, 2005. National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council. The third author was working with NICTA when this paper was produced.

References

1. Clements, P., Northrop, L.M.: Software Product Lines: Practices and Patterns. Addison-Wesley, MA (2001)
2. Schmid, K., Biffl, S.: Systematic management of software product lines. Softw. Process Improve. Pract. **10** (2005) 61–76
3. Genuchten, M.v.: Why is software late? an empirical study of reasons for delay in software development. IEEE Trans. Softw. Eng. **17**(6) (1991)
4. Subramanian, G.H., Breslawski, S.: An empirical analysis of software effort estimate alterations. J. Systems and Software **31**(2) (1995) 135–141
5. Dijkstra, E.: Notes on structured programming. In Dahl, O.J., Dijkstra, E., Hoare, C.A.R., eds.: Structured Programming. Academic Press, London (1972)
6. IEEE: Ieee std. 1219-1998, ieee standard for software maintenance (1998)
7. Nonaka, M., Azuma, M.: Software delivery estimation model for incremental and iterative development process considering undetected design defects (in japanese). In: Proc. Software Symposium 2003. (2003) 107–114
8. Jørgensen, M.: Realism in assessment of effort estimation uncertainty: It matters how you ask. IEEE Trans. Softw. Eng. **2004**(30) (2004) 209–217
9. Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., Steece, B.: Software Cost Estimation with COCOMO II. Prentice Hall (2000)
10. Boehm, B.W., Brown, A.W., Madachy, R., Yang, Y.: A software product line life cycle cost estimation model. Proc. 2004 Intl. Symp. Empirical Softw. Eng. (ISESE'04) (2004) 156–164
11. Chen, Y., Gannod, G.C., Collofello, J.S.: A software product line process simulator. Softw. Process Improve. Pract. **11** (2006) 385–409
12. Yang, D., Wan, Y., Tang, Z., Wu, S., He, M., Li, M.: Cocomo-u: An extension of cocomo ii for cost estimation with uncertainty. Lecture Notes in Computer Science **3966** (2006) 132–141

13. Donzelli, P.: A decision support system for software project management. IEEE Software **23**(4) (2006) 67–75
14. Abdel-Hamid, T., Madnick, S.: Software Project Dynamics- An Integrated Approach. Prentice-Hall, Englewood Cliffs, NJ (1991)
15. Calavaro, G.F., Basili, V.R., Iazeolla, G.: Simulation modeling of software development process. In: Proc. 7th European Simulation Symposium. Soc. for Computer Simulation. (1995)
16. Hansen, G.A.: Simulating software development processes. IEEE Computer **29**(1) (1996) 73–77
17. Antoniol, G., Cimitile, A., Lucca, G.A., Penta, M.: Assessing staffing needs for a software maintenance project through queuing simulation. IEEE Trans. Software Eng. **30**(1) (2004) 43–58
18. Padberg, F.: A study on optimal scheduling for software projects. Softw. Process Improve. Pract. **11** (2006) 77–91
19. Kellner, M.I., Madachy, R.J., Raffo, D.M.: Software process simulation modeling: Why? what? how? J. Systems and Software **46**(2-3) (1999) 113–122
20. Epping, A., Lott, C.M.: Does software design complexity affect maintenance effort? In: Proc. 19th Softw. Eng. Workshop. (1994) 297–313
21. Bocco, M.G., Moody, D.L., Piattini, M.: Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation. J. Software Maintenance and Evolution **17**(3) (2005) 225–246
22. Ramanujan, S., Scamell, R.W., Shah, J.R.: An experimental investigation of the impact of individual, program, and organizational characteristics on software maintenance effort. J. Systems and Software **54** (2000) 137–157
23. Kan, S.H., Dull, S.D., Amundson, D.N., Lindner, R.J., Hedger, R.J.: As/400 software quality management. IBM Systems Journal **33**(1) (1994) 62–88
24. Remus, H.: Integrated software validation in the view of inspections / reviews. In: Proc. Symposium on Softw. Validation, Elsevier (1983) 57–64
25. Musa, J.D.: Software Reliability Engineering. Osborne/McGraw-Hill (1998)
26. SEL: Sel (software engineering laboratory) data. http://www.cebase.org (1997)
27. Defamie, M., Jacobs, P., Thollembeck, J.: Software reliability: assumptions, realities and data. In: Proc. 1999 Intl. Conf. Softw. Maintenance (ICSM'99). (1999) 337–345
28. Sargent, R.G.: Validation and verification of simulation models. Proc. 31st Conf. Winter Simulation (1999) 39–48
29. Cohen, S.: Predicting when product line investment pays. Technical Report Techinical Report CMU/SEI-2003-TN-017, Software Engineering Institute, Carnegie Mellon University (2003)
30. Böckle, G., Clements, P., McGregor, J.D., Muthig, D., Schmid, K.: Calculating roi for software product lines. IEEE Software **21**(3) (2004) 32–38

Modeling Risk-Benefit Assumptions in Technology Substitution

Antony Powell, John Murdoch, and Nick Tudor

[1] YorkMetrics Ltd, IT Centre, York Science Park, Heslington York YO10 5DG, UK
[2] Department of Computer Science, University of York, Heslington, York YO10 5DD, UK
[3] QinetiQ, Systems Assurance Group, Malvern Technology Centre, St Andrews Road Malvern, Worcestershire, WR14 3PS, UK
antony.powell@yorkmetrics.com, jm@cs.york.ac.uk, njtudor@qinetiq.com

Abstract. This paper describes the application of comparative simulation models to reason about the economic risks and benefits of adopting new methods and tools for software development. It addresses three questions: (i) can technology substitution be modeled with sufficient confidence? (ii) what modeling strategy is most appropriate? and (iii) what are the outcomes of modeling technology substitution on an industrial case study? The end goal is to develop models that support economic evaluations that are necessary and sufficient to support technology substitution decisions. Such models will help developers and managers to assess the value of a new technology and employ strategies to de-risk its adoption.

Keywords: simulation, system dynamics, technology adoption.

1 Introduction

1.1 Principles of Technology Substitution

The adoption of a new development technology (tool, process or lifecycle) is a risky endeavor. This is particularly so where the consequences of failure are high, such as in the development of safety critical software. Here, a poor technology decision can lead to substantial corrective costs, late delivery, or even failure of the end product.

As a result, many organizations tend to follow excessively conservative development strategies, particularly for high-integrity software products. Improvements in technology are limited and tend to finesse existing development methods and processes in a piecemeal manner. Arguments for their adoption on a new project are based on relatively minor changes to existing practices and performance.

The technical and economic comparisons needed to support technology adoption are straightforward when the substitution is 'like for like'. However, when more radical technologies become available, like formal methods, the safety critical domain is slow to capitalize, despite potentially compelling business cases. They are naturally late adopters, waiting for establish convincing results from other application domains.

Where the technology being modified has a direct influence on software safety processes themselves, it is not always practical to rely on trials in other domains. In

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 295–306, 2007.

these cases, the organizations must provide two strands of evidence to support the argument for adoption of the new technology – technical and economic.

The *technical* evaluation usually involves a stepwise comparison of the 'as-is' and 'to-be' approaches to demonstrate that the technical output is the same or better, all other things being equal. The criteria used ultimately affect product qualities such as validity, certifiability, reliability, and consistency of the output. The objective is to justify that the new process is technically 'at least as good as' its predecessor.

The *economic* evaluation builds models to predict the commercial impact on the project of introducing the new technology. The criteria used include performance factors such as cost, timescales, resources, and risk. These factors are inherently dependent on the evaluation of the technical impact of the new technology. The aim is to show the process is economically more attractive than its predecessor.

In both cases the substitution decision is based on evidence and arguments of the efficiency, effectiveness and efficacy of the new technology by use of analysis, experiments, or case studies. In practice, however, many attempts at process improvement often fail to deliver the expected improvements due to experimental errors, technical difficulties, unexpected costs, or changes in application context. Furthermore, even where the technical and process change can be technically justified, and the economic payback appears significant, the business inertia and risks are often prohibitive to technology adoption.

The substitution argument therefore presents essential difficulties for managers who must commit to a new technology based on predictions of its technical and economic impact. The aim of this paper is to explore how the substitution decision can be pragmatically modeled to provide better evidence for the business justification and decision for technology substitution.

1.2 Assessment of Technology Substitution

The advantages and disadvantages of a change in a development process can be assessed from several viewpoints. For example, we can consider the responsibilities at each of the typical 'layers' of engineering management as illustrated in Figure 1. A process (or product) innovation can be assessed in terms of (i) the added value, (ii) the added risk, and (iii) compliance with applicable constraints.

A particular process or product technical innovation has therefore to be considered from numerous technical and organizational perspectives. This is the basic concern of technology transition.

In practice organizations need models of the economic consequences of adopting the new technology, subject to an '*at least as good as*' technological substitution [1]. The business justification is based on the new technology being better, cheaper, faster and more predictable than the existing process. Whilst any model is predicated on the technical uncertainty, an economic model may help managers to decide '*is it a risk worth taking?*'

The problems of modeling this decision can be categorized as essential and accidental.

Essential problems reflect the intrinsic complexity of real world processes. The development of software is a socio-technical activity with internal and external uncertainties. The nature of software as a dynamic feedback system, with emergent

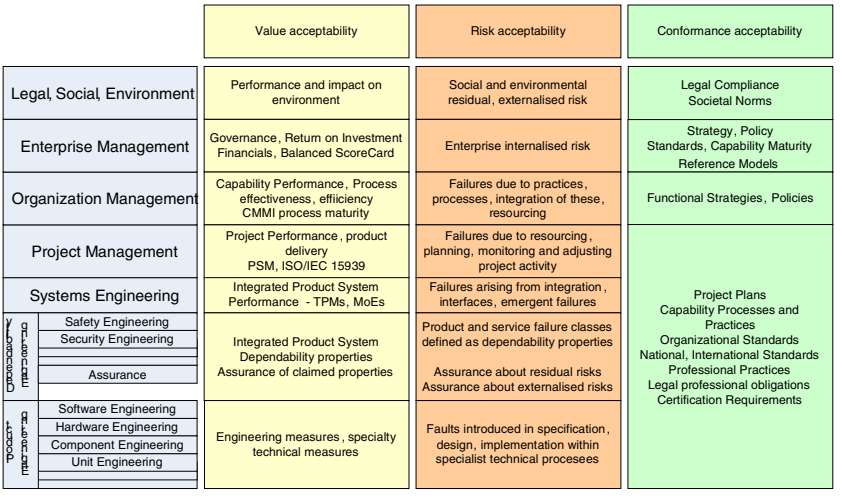


Fig. 1. A Technology Substitution Framework

behavior, makes it difficult to predict as estimation and planning affects performance, i.e. 'a different estimate creates a different project' [2].

Accidental problems reflect limitations in our understanding of the process and its likely behavior. These limitations reflect the absence of sufficient case study data due to limitations in experimental capability (e.g. prohibitive costs of experiments) and learning (e.g. appropriate data and models). For example, an unforeseen technical problem can have a significant effect on process performance. With better measures and models we should be able to reduce these problems.

The problems of accurately predicting the impact of process change are acute. In practice, the predictive accuracy of models is poor even for simple waterfall lifecycles. As Kitchenham observes 'There is no evidence that estimation models can do much better than get within 100% of the actual effort during requirements specification and 30% of the actual effort prior to coding' [3]. The reality is that practitioners and managers need to pursue more pragmatic approaches for limiting the accidental problems that surround technology adoption.

1.3 Strategies for Modeling Technology Substitution

A response to the uncertainties in technology substitution is to focus on dealing with the relative risks of a planned technology insertion. As Kitchenham observes: "*senior managers need to concentrate more on managing estimate risk than looking for a magic solution to the estimation problem*" [3]. That is, rather than trying to produce the 'perfect estimate,' our models should instead try to elicit and control the risks of technology adoption.

One approach to understanding risk is to use parametric models. The data from experiments can be used in off-the-shelf tools like PriceS, Galorath, or COCOMO, and sensitivity analysis performed. However, these parametric approaches are often based assumptions that are inappropriate when the substituted technology departs so significantly from the lifecycles on which the models are formulated.

Another approach is to perform process simulation using systems dynamics. This is attractive as it uses flows and feedback loops to model how the new technology would behave in practice. Moreover, it allows for extensive sensitivity analysis which enables different scenarios and special case conditions to be checked. This can give decision makers realistic feedback about their target field of use and envisioned roll-out of the process. These assumptions are used as a baseline against which to reason about alternative plans and monitor real performance as our understanding improves.

It is therefore possible to make a number of general observations on the nature of the modeling task:

1. *Technical Equivalence.* Any model is unreliable in the face of technical uncertainty and therefore a suitable level of technical equivalence must be established.
2. *Business Context.* Any model must clearly define the scope of the comparison to specify what is included in and excluded from the assessment.
3. *Adoption Process.* Any model must consider the expected organizational impact of tool adoption and process change (e.g. in terms of disruption etc).
4. *Explicit Assumptions.* Any model must contain an explicit dynamic hypothesis about the expected change in process behavior.
5. *Reasoning Capability.* Any model must support reasoning about alternative behaviors and planning assumptions via relative change and sensitivity analysis.
6. *Improve Evidence.* Any model must encourage the gathering and integration of new evidence of cause-effect relationships that drive performance and behavior.
7. *Model Improvement.* Any model should enable plans (assumptions) to be progressively refined over time as a series of baselines.
8. *Sufficient Detail.* Any model is inherently limited by the level at which data is collected and causal relationships can be confirmed.

These assumptions allow us to build a very primitive model to describe the modeling approach. This can then be amended over time with the support of further quantified experimental evidence of process performance. The following assumptions are made in the models developed in the next section:

1. The software development process is phased or staged; different phases apply to the two processes; different technical resources are required by different phases;
2. The coordination between phases is of types: pipeline, sequential or hybrid.
3. An important difference between the conventional and formal processes is in the area of fault generation, discovery and rework.

These variations have led us to develop a ‘plug and play’ approach to modeling the software development process. Basic building blocks are provided that can be connected together in different ways to capture features of real processes as usefully as possible.

The remainder of this paper describes a dynamic evaluation method using the general technique of broad range Sensitivity Analysis as a starting point [4]. The aim is to get models of accuracy necessary and sufficient to show relative trends in order to improve decision making and overcome barriers to adoption.

2 A Case Study of Technology Substitution

This real-world industrial case study concerns the evaluation of a conventional test-based development process (Process A) against a formal proof-based development process (Process B). The aim here is to give a simplified overview to illustrate the use of dynamic models to reason about the behavior and sensitivity of the two processes.

2.1 Step 1 – Model Boundary

Modeling the technology substitution relies on finding a common boundary at which the two processes can be compared. This assumes a level at which the inputs and outputs are broadly comparable.

Process A follows a conventional ISO/IEC 12207 waterfall lifecycle process [5] as illustrated in Figure 2. The Requirements are used to form the basis for a Top Level Design that outlines the architecture and functionality of the system. A Detailed Design is then manually Coded and Unit Tested before Integration Testing.

Process B follows a formal development process with the comparative footprint represented by the shaded box in Figure 2. The Requirements are expressed as a Formal Specification which is then Refined before being Autocoded and the output is Tested for exceptions. Both processes take a design specification as an input and result in verified code.

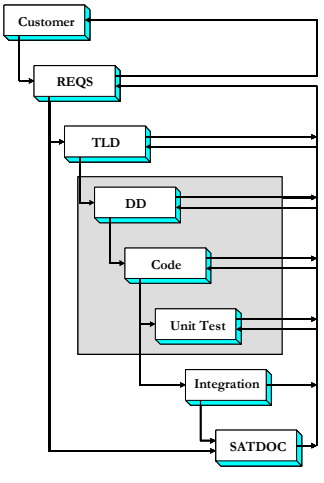


Fig. 2. Comparative Process Footprint

The substitution of the conventional process with a formal one promises benefits on a number of levels, including: (i) the direct removal of the resource-intensive hand-coding and unit testing processes could represent a very significant saving in costs and timescales, (ii) the use of an automated formal process can reduce the latency of design and code errors, illustrated by the inner loop in Figure 3, and

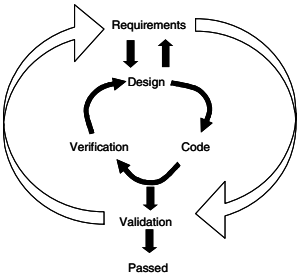


Fig. 3. Modeling Rework Loops

(iii) the rapid iteration provides early feedback of requirements problems, as represented by the outer loop.

There is, however, reluctance to move away from coding and unit testing that represent tried and tested processes with relatively known technical outcomes. An insight into the relative performance and sensitivities of the two processes is therefore essential to the evaluation.

2.2 Step 2 – Model Objectives and Assumptions

The purpose of the model is to understand the relative performance of the two processes and reason about the consequences of adopting the new process. This is an iterative process that involves eliciting assumptions about process behavior and refinement using performance data.

The basic assumption is that we model the processes in isolation, both processes starting with the same inputs, and producing common outputs. This excludes upstream and downstream activities, concurrent projects, holidays and lost time etc, in order to simplify the comparison.

The case study model therefore assumes that Process A and Process B perform the same quantity of work, using the same resource, in order to generate an equivalent output. We are solely interested in the relative effort and duration of each process.

The aim is to build simple but representative models in the first instance, followed by more advanced models as understanding increases. We only need detail at a level necessary and sufficient to capture the relative performance of the two processes.

2.3 Step 3 – Model Construction

In the case study Processes A and B each have three intermediate stages as show in Table 1 below. It is important to note that the outputs of each stage are not equivalent

Table 1. Process Stages

	Process A	**Process B**
Stage 1	Design	Formal Spec
Stage 2	Code	Refinement
Stage 3	Unit Test	Autocode/Test

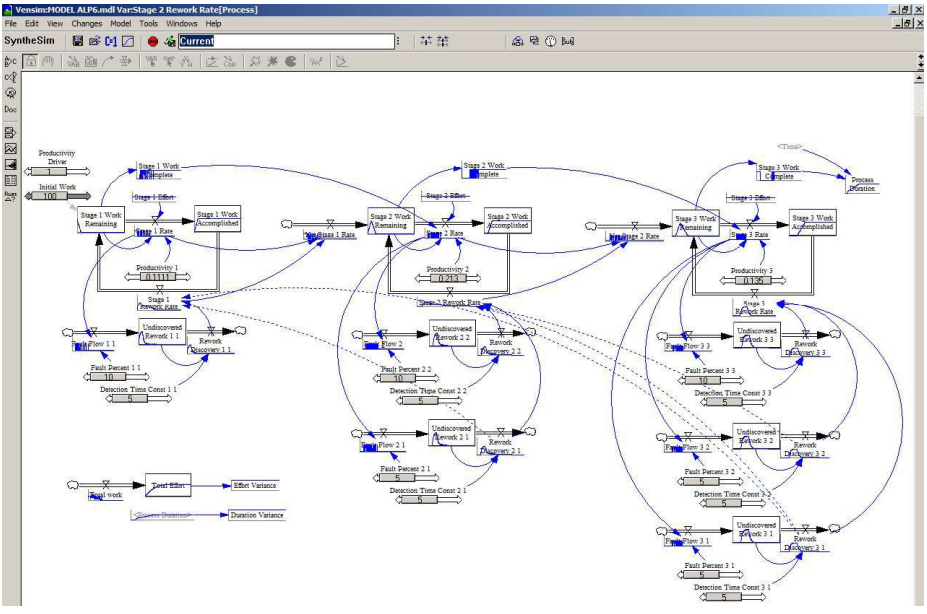


Fig. 4. Case Study Simulation Model in Vensim [6]

in both processes. The model uses flows in the individual stages as means to understand performance of each process as a whole.

The model is then constructed using the model shown in Figure 4 below; each model has three phases with associated work (units of activity, e.g. code modules) and rework flows.

For simplicity the basic model assumes that the stages are sequential rather than concurrent. The model includes the summary of effort and durations as the main output variables. Finally, Productivity Multiplier and Quality Multiplier variables are used to aid the sensitivity analysis of the Rate and Fault assumptions to be examined.

2.4 Step 4 – Data Gathering

The process of data gathering is driven by the model objectives and assumption, and must be at a level necessary and sufficient to describe relative performance.

The inputs to the model are the quantity of Work to be performed, the work Rate of each stage, the Fault percent (or breakage) per phase, and the Time to Detect Rework. The fault, detection and rework flows are expressed for all combinations of phases to account for undetected rework slipping through the process. It also models the inner and outer rework loops illustrated in Figure 3. Example data tables for productivity on Process A and Process B are as follows:

Table 2. Example Data Tables

Process A	Stage 1	Stage 2	Stage 3
Stage name	Design	Code	Test
Initial Work	100	-	-
Productivity	9 hours/unit	4.7 hours/unit	7.4 hours/unit

Process B	Stage 1	Stage 2	Stage 3
Stage name	Formal Spec.	Refinement	Autocode/Test
Initial Work	100	-	-
Productivity	2.3 hours/unit	4 hours/unit	1.6 hours/unit

An indirect benefit of constructing models with relative rather than absolute data is that it can protect commercial confidentialities. The data tables reflect relative values that can be negotiated and agreed by participants including tool vendors and users.

2.5 Step 5 – Model Runs, Sensitivity Analysis and Validation

The model was therefore run with data taken from the experiments performed on the Conventional Testing (Project A) and Formal (Project B) approaches to development as shown in Figure 5 below.

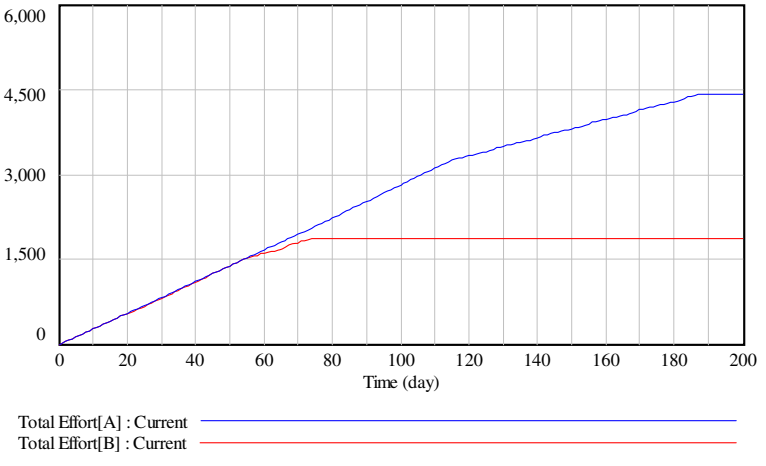


Fig. 5. Results of Model Run - Total Effort Profiles [Process A and Process B]

In this example, the results of the initial run show an overall reduction in the development cost from 4500 person-hours to 1600 person-hours and a corresponding reduction in duration from 186 days to 73 days.

Whilst the initial results might show distinct benefits, the modeling approach seeks to validate the assumptions on which the relative judgments are being made. In practice the process would be to seek more accurate performance data, refine the models and achieve a consensus on the outcomes. It must also take account of other

pragmatic considerations including skills requirements, resource leveling, schedule constraints, levels of reuse, etc.

In order to test the relative assumptions, the model was then used to perform multivariate sensitivity, for example to perform multivariate sensitivity analysis uses the Productivity Driver variable on both processes This simulates the consequence of predictions of likely productivity to be between 50% worse than expected and 100% better than expected.

The results of the sensitivity run are shown in Figure 6. The top line-graph shows the total effort forecast by the simulation of Process A. The bottom line-graph shows a sensitivity of the model of Process B to a variation in Productivity between -50% to 100% (across all stages) over 500 model runs. The sensitivity is expressed as confidence bands of 50%, 75%, 95% and 100%. The reader should note the different effort scales on the two charts.

The relative cost-benefit can then be assessed using sensitivity in effort variance (Figure 7). This shows the results of effort variance (Effort Process B – Effort Process A)

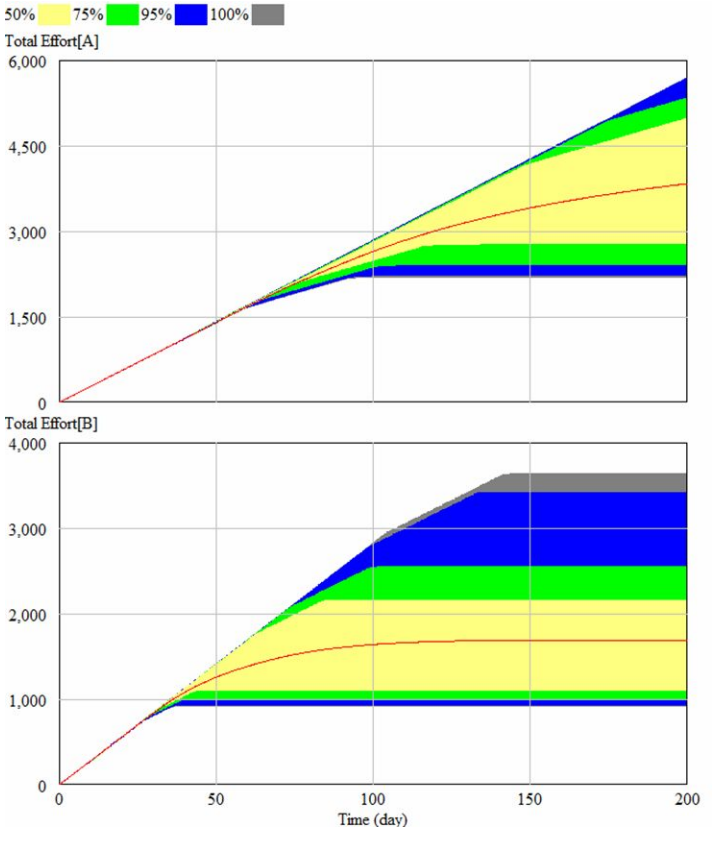


Fig. 6. Results of Sensitive Analysis

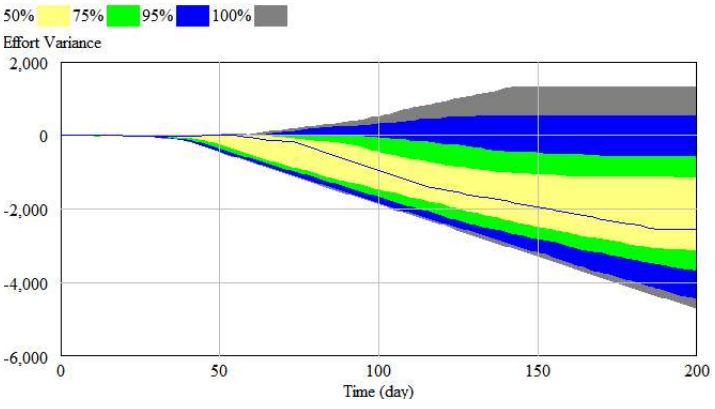


Fig. 7. Sensitivity in Effort Variance

for 500 simulation runs. The confidence bounds of 50%, 75%, 95% and 10% are colored on the chart.

The results from the example show a high degree of confidence that Process B will yield reductions in effort relative to Process A. This pay-off sensitivity supports evaluation of risks-benefit assumptions and can provide evidence of the acceptable 'risk-premium' for adopting the new technology.

The analysis also needs to take into account the relative sensitivities of fault injection, detection and rework. This is illustrated in Figure 8 which shows graph of the Undiscovered Rework (units) over time for an example model run where fault rates are assumed to be equal.

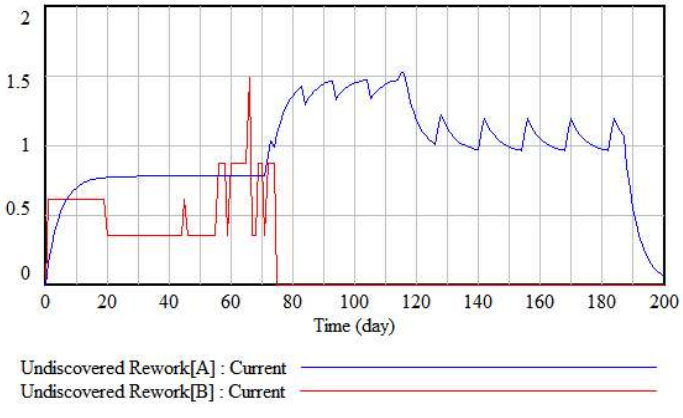


Fig. 8. Rework Strip Graph

The results illustrate a left-shift effect in Process B which benefits from earlier detection and fixing of rework. In practice the formal approach of Process B promises to reduce fault injection and detection rates even further .

Related sensitivity runs of fault injection, detection times and rework flows effectively simulate the inner and outer rework loops. These show high confidence that Process B is robust to higher rates of fault injection and breakage.

2.6 Step 6 – Model Review

A key tenet of the model is to iterate around the evidence gathering, modeling and analysis, akin to a scientific method of investigation. It is critical to keep in mind that these are models of reality that are used to progressively refine the understanding of the users. If applied correctly the modeling approach can aid understanding, but if used inappropriately it has the potential to mislead. For example, the model might indicate a reduction in project duration of 25% but applied in practice the demands of the wider program will dictate the actual duration. The results must therefore always be considered in the context of the underlying context and assumptions on which the model is constructed and applied (as illustrated in Section 1.3). This is an iterative process of providing and testing evidence to support the arguments on relative performance and refining the models accordingly.

The basic model contained in this paper illustrates the principles of using system dynamics to model the cost-benefit assumptions of technology substitution. However, more advanced models have been constructed according to the projects and technologies being evaluated [7]. The final decision to adopt the new technology would be made against the evidence provided and arguments for its impact supported by the model. The decision makers must evaluate if the model results and confidence bounds can outweigh the risk-premium of adoption.

3 Conclusions

Technology adopters must strive to understand and de-risk the substitution process through integrated technical and economic evaluation - supported by analysis, case studies and experimental evidence.

The case study presents a technological substitution argument with significant technical and economic implications. This study does not enable a decision for adoption to be made based upon expected cash value return. It does, however, give indication of relative risks and benefits, supporting the observation by Kitchenham that "senior managers need to concentrate more on managing estimate risk than looking for a magic solution to the estimation problem" [3]. This study deals with the uncertainties in technology substitution by focusing on the relative risks of a planned technology insertion. That is, rather than trying to produce the ‘perfect estimate,’ our models instead try to elicit and control the risks of technology adoption.

The development of a reliable economic or commercial model of technology substitution is therefore intrinsically limited in the presence of technical uncertainty. Without adequate justification that the processes are meeting the technical equivalence requirement, or may rely on contingent corrective action in order to achieve it, we face severe limits on our ability to build a model that will reflect the process behavior when applied in practice. A cost model based on piecemeal decomposition and comparison of constituent activities would be so unreliable and potentially misleading as to undermine the commercial argument that it is trying to support.

This work has therefore attempted to determine a pragmatic method for justifying the economic (commercial) outcomes of technology substitution in practice. This should contribute to a business case that evidences a new technology as predictably better, faster and cheaper than its predecessor.

The proposed approach concentrates on modeling the assumptions in the technology adoption decision. The model simulates the high-level flows of work in the two processes using available data and assessments of process differences. While these are gross assumptions about behavior, a model can be evolved to describe the large-scale payback curves that need to be achieved in order to gain acceptance for the process change.

This approach promises some advantages compared to conventional models:

- It makes essential technology insertion, costing and planning assumptions explicit. It encourages users to refine their understanding and decision-making, rather than relying on the point estimates of black-box cost models.
- It helps investigate the trade-offs between planning variables. It prompts users to consider the consequences of alternative scenarios, such as the implication of perturbations to the performance of the technology.
- It allows assumptions to be progressively refined over time. It treats estimation and planning as a continuous rather than one-off activity from which to understand the bounds and limitations of risk and performance.

More work is required to build more advanced models that are sufficiently reliable to help managers to determine cost-benefit curves to describe the risks and returns of technology substitution. In order to do this it is crucial to gather further experimental evidence of the behavior of the new technology. For the time being, the proposed method presents a step towards building models that are necessary and sufficient for supporting these arguments.

References

1. Galloway, A., Paige, R., Weaver, R., McDermid, J., and Toyn, I., "Technology Substitution Arguments in the Context of DO-178B," Department of Computer Science, University of York, UK, 2005.
2. Abdel-Hamid, T.K., "Impact of Schedule Estimation on Software Project Behavior," IEEE Software 3(4): 70-75 1986.
3. Kitchenham, B.A. (1998), "The Certainty of Uncertainty," European Software Measurement Conference FEMSA 98, Antwerp.
4. Raffo, D., "Evaluating the Impact of a New Technology Using Simulation: The Case for Mining Software Repositories," ProSim Workshop, St.Louis, May 2005 [4]
5. ISO/IEC, "ISO/IEC 12207 - Information Technology Software Life Cycle Processes," International Organization for Standardization/International Electrotechnical Commission, 1995.
6. Vensim 5.0, Ventana Systems Inc, Harvard, MA
7. Powell, A.L. and Tudor, N. (2005) "Modeling Technology Substitution", Report for QinetiQ plc, June 2005.

Evaluating the Impact of the QuARS Requirements Analysis Tool Using Simulation

David M. Raffo[1,2], Robert Ferguson[3],
Siri-on Setamanit[1], and Bhuricha Deen Sethanandha[2]

[1] School of Business Administration, Portland State University,
Portland, Oregon 97201, USA
{raffod,sirion}@pdx.edu
[2] Maseeh College of Engineering and Computer Science, Portland State University,
Portland, Oregon 97201, USA
bhuricha@cs.pdx.edu
[3] Software Engineering Institute, Carnegie Mellon University,
Pittsburgh, PA 15213, USA
rwf@sei.cmu.edu

Abstract. Adopting new tools and technologies on a development process can be a risky endeavor. Will the project accept the new technology? What will be the impact? Far too often the project is asked to adopt the new technology without planning how it will be applied on the project or evaluating the technology's potential impact. In this paper we provide a case study evaluating one new technology. Specifically we assess the merits of an automated defect detection tool. Using process simulation, we find situations where the use of this new technology is *useful* and situations where the use of this new technology is *useless* for large-scale NASA projects that utilize a process similar to the IEEE 12207 systems development lifecycle. The method can be applied to assessing the impact (including Return on Investment), break even point and the overall value of applying any tool on a project.

Keywords: Process Simulation, Requirement Analysis Tool, Quantitative Method, Technology Adoption.

1 Introduction

Competition in the software industry and the continuing pressure from low cost economies is pressing companies to improve their efficiency and to find ways to optimize their development and quality assurance activities, both locally and globally.

New tools and new technologies offer promise for speeding software development tasks, reducing costs and improving quality at all points along the development lifecycle. Over the years, development organizations have invested heavily in these tools with some success. But there have also been some failures. How can managers determine whether a new tool or technology will be beneficial to their development environment? Under what project conditions would it be beneficial to apply a new tool or technology and when would it not be beneficial?

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 307–319, 2007.

Process Simulation Modeling (PSM) is a technology that is increasingly used within academic and research realms to evaluate issues related to process strategy, process improvement, technology and tool adoption, project management and control, and process design.

Recent developments in PSM tools have drastically cut the costs to develop these models. Moreover, new models coupled with more systematic and repeatable methods have been developed to apply PSMs within organizations, enabling PSM to provide greater business value.

Specifically, PSM can be used to evaluate new tools and technologies. Using PSM enables an organization to:

- Plan how a new technology might be applied
- Assess the costs and benefits of the new tool or technology
- Explore alternative approaches for applying the technology

Using PSM, an organization can answer the following questions ***before*** rather than after they invest in the technology.

- What is the likely impact of applying new tools and technologies?
- What is the likely economic benefit or value of the tool or technology? What is the ROI?
- When might the tool or technology be ***useful*** and when might it be ***useless***?
- Under what conditions does the tool or technology perform best?
- What performance standards does the tool need to achieve in order to have a positive return?
- Are there better ways to apply the tool?

The technology evaluated in this paper is an automated natural language requirements analysis tool (QuARS) [1]. The developers of this technology had recently made significant breakthroughs in reducing costs and increasing the effectiveness of this technology. Is the technology now "ready for prime time" on NASA projects? This study seeks to address this question.

2 Background

In this study, we created a software process simulation model to study the impact of the Quality Analyzer for Requirements Specification (QuARS). The process simulation model allows us to conduct controlled experiments and answer various questions before we deploy the tool on the actual software project. In the following section we will provide background of both technologies.

2.1 Process Simulation Models (PSMs)

Process Simulation is a commonly used technique to improve management decisions including:

- Strategic management,
- Process planning,

- Project tracking and control,
- Process improvement and technology adoption,
- Process understanding
- Training and learning [2].

Raffo [3] used process simulation models to justify process improvement initiatives, to predict the impact of process changes before they are implemented and to assess multiple process alternatives under various business case scenarios. PSM also helps software organizations achieve higher CMMI levels. Organizations can use PSM to establish a framework for selecting core process and product metrics [4]. Raffo et al. [5] showed that PSM can provide a quantitative assessment of the risk or uncertainty associated with various process alternatives and support quantitative prediction of project level performance in term of cost, quality and schedule. Moreover, the results of PSM are very useful in determining financial performance measures such as Return on Investment, Net Present Value, etc [6]. The use of PSM led Leon [7] to new understandings of the software requirement elicitation process. Pfahl [8] evaluated applications of PSM in software project management education.

One of the advantages of process simulation models is their ability to represent software development processes at both the mirco and macro process levels. Process simulation models (PSMs) can be used to represent software development projects from narrowly focused portions of the software development life cycle to long-term product evolutionary models [2].

Two process simulation paradigms, discrete event simulation (DES) and system dynamics (SD), have been widely adopted by researchers. While the DES paradigm describes software development processes as sequences of discrete activities, the system dynamics paradigm describes the interaction between project factors in the form of levels and flows. Melis [9] developed a discrete model to evaluate the effectiveness of Extreme Programming process. Abdel-Hamid and Madnick [10] used a SD model to demonstrate why managers underestimate the resources required for a software project. Madachy [11] developed a SD model to study the impact of inspection-bases processes. Smith [12] developed an agent-based model to study open source software evolution. Martin and Raffo [13] developed a hybrid model that takes advantage of both the discrete event and system dynamics paradigms. The hybrid model can represent many different aspects of software development processes and help answer questions that are important for management decision making. This hybrid modeling paradigm has been elaborated by Setamanit, Raffo and Wakeland to evaluate issues related to Global Software Development [14].

In this study, we use a discrete event simulation model to address technology adoption questions. The PSMs can help software managers answer several tool adoption questions such as:

- Would it be better to build test suite in-house or to buy an existing one?
- Is the new tool worth the cost? This includes purchase, maintenance, training, process changes and other associated implementation costs.
- What level of performance does the tool need to achieve in order to be worthwhile?

These questions are very important to help management decide whether or not to adopt the technology. The process simulation is a cost effective approach that helps answer these questions. Without simulation, software organizations would have to take a risk in adopting the technology and learn about the impact from pilot studies and controlled experiments which can consume a lot of time and resources. Using PSMs, we can conduct virtual controlled experiments on a software project. By incorporating software metrics data, PSMs allow modelers to perform various analyses such as sensitivity analysis, design of experiments [15, 16] and statistical comparison of various software process configurations. The results of these analyses can provide useful information for management decision making.

2.2 The QuARS: Quality Analyser for Requirements Specification

Software requirements-related defects are the most common and most expensive type of defects to correct. Depending on when this class of defect is found, the cost to find and fix these defects can range between 50-100 times the effort/cost it would have taken to correct the defect in the requirements phase [17]. Therefore, it is crucial to detect as many requirements defects as early as possible. The fact that requirements documents are commonly written in natural language, make them prone to errors. There are several human intensive defect detection techniques such as inspection-base techniques and scenario-based review techniques. However, these techniques can be expensive and time consuming.

The Quality Analyser for Requirement Specification (QuARS) is an automated natural language requirements analyzer tool that identifies defects in requirements. QuARS performs expressive analysis on requirements documents and indicates potential defects based on the quality model described in [1]. A drawback of automated defect detectors is the possibility of detecting false positives. After analyzing a natural language requirements document, QuARS requires the user to review the all the faults found and to distinguish between real defects and false positives. The metrics generated by QuARS are a readability score and a defect density score for the document.

Several empirical studies have been done on QuARS. Ferguson [18] conducted a study on QuARS at the NASA Independent Verification and Validation (IV&V) facility. The results indicated cause and effect relationships between expressive defects in requirements and the final software product. Lami [19] applied QuARS to analyze NL requirements in a commercial software project. The results showed that QuARS takes less time and finds more defects of certain types than human review. However, the cost of correcting false positives was also very high.

3 GPSM-Based Evaluation Approach

The objective of this case study is to use a generalized process simulation model (GPSM) [20] to evaluate the impact of implementing QuARS.

3.1 The IEEE 12207 Model

The specific model used in this study is a model of the IEEE 12207 standard for software lifecycle process [21] as shown in Figure 1. We extended the model described

in [22] to perform this study. The model was further calibrated using NASA project data and industry standard data from [23]. The IEEE 12207 Model consists of two layers, 1) Development and 2) IV&V. The development layer represents the systems and software lifecycle phases based on the IEEE 12207 standard. It is comprised of nine phases. Each phase has one or more process steps in it. In total, there are 86 steps in the software development process. The IV&V layer represents the activities carried out by external software auditors. This layer consists of five main IV&V phases. Each phase is comprised of multiple IV&V activities that may be used to verify and validate software artifacts from the corresponding software development phases.

The results of this model were validated against the performance data from 12 large-scale NASA projects (with project size of 90 thousand lines of code (KLOC) or higher).

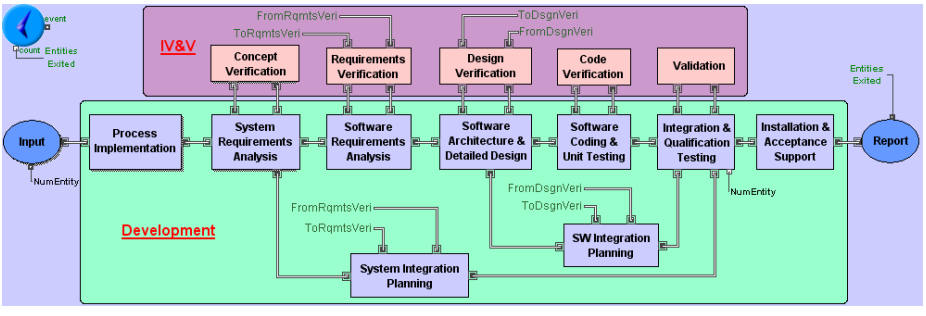


Fig. 1. IEEE 12207 Process Simulation Model with IV&V Layer

In this case study, we consider using QuARS during:

1. Quality assurance (i.e. V&V[1]) activities within the project: applying QuARS to analyze the System Requirements, Software Requirements, and then at both phases.
2. IV&V activities outside of the project: applying QuARS at Concept Verification, Requirements Verification, and then at both phases.

The key questions that we aim to answer are:

1. Does QuARS add value to the project?
2. Is QuARS more effective in V&V or IV&V mode?
3. What is the amount that the project should be willing to pay for QuARS?

The general method for using simulation to assess the impact of new technologies on a project is similar to assessing the impact of a process change [24]. The first step is to establish baseline model result. Then, we design each TO-BE process scenario

[1] Verification and Validation (V&V) activities determine whether development products of a given activity conform to the requirements of that activity, and whether the software satisfies its intended use and user needs. This determination may include analysis, evaluation, review, inspection, assessment, and testing of software products and processes.

and make appropriate changes to the model. After that, we run each TO-BE scenario model and determine the change in performance and select the "best" process option.

3.2 QuARS Assumptions

To evaluate automated defect detection tools we consider following criteria 1) PD: the probability of detecting faults, 2) Accuracy, 3) The cost of using the tool, and 4) the probability of fault positives. We use data from [18, 19] to represent QuARS capabilities. In addition to the empirical data, we also made several assumptions based on the field study at commercial firm [18, 19] as well as at NASA as follows:

- QuARS productivity is 10,000 lines of code per person-hour.
- 37% of the requirements defects are QuARS detectable. QuARS defect detection rate is 100% for QuARS detectable defects.
- Employing QuARS improves the quality of the requirements document, thus the defect detection capability at Requirements inspection improves by 5% to 15% (min = 5%, max = 15%, mode = 10%) if the QuARS detected defects are corrected prior to requirements inspection.
- The cost of training and associated software engineering process group (SEPG) activities is 1 person-month.

Employing QuARS also provides benefits to other development phases besides the Requirements phase as follows:

- Improves clarification of requirements, thus improves design productivity by 5% to 10%
- Improves Engineering design decisions, thus reduces the injection of design defects by 5% to 10%
- Improves test planning and test case generation productivity by 10% to 20%
- Improves the quality of test cases, thus reduces the injection of test case defects by 5% to 15%

4 Business Implications of QuARS

4.1 AS-IS Baseline Model Results

As discussed in the previous section, the IEEE 12207 process model baseline performance was predicted in terms of effort (or cost), duration, and latent defects (or delivered defects). The characteristics of the AS-IS model are as follow:

- The project is 100,000 lines of code.
- The industry standard data [23] were used for earned value (% effort allocated for each activity) and defect detection rate.
- Organization specific data were used for productivity and defect injection rates.

The baseline performance for the AS-IS process (without using QuARS) is shown in Table 1. Note that the data presented in this paper is marked to protect company confidentiality.

Table 1. Baseline performance for the AS-IS process

	Effort incl. IV&V	Effort	Rwrk_Efrt	IV&V Effort	Duration	Avg. Dur	Crctd_Dfcts	Ltnt_Dfcts
Mean	71,371.20	69,301.61	27,404.94	2,069.59	4,837.69	2,423.03	6,004.87	634.68
Std. Dev.	1,910.20	1,894.25	1,037.12	246.33	195.06	92.37	227.50	24.64

4.2 Scenario 1: Applying QuARS in V&V Mode at Different Phases

In this scenario, we made changes to the model to represent 3 configurations: 1a) QuARS at System Requirements phase; 1b) QuARS at Software Requirements phase; and 1c) QuARS at both phases. Figure 2 shows a flow chart of the AS-IS and TO-BE processes for configuration 1a) QuARS at Systems Requirements phase.

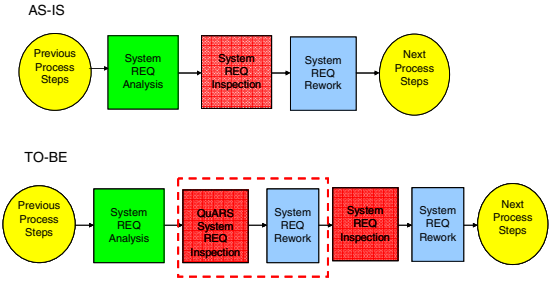


Fig. 2. Process Flow Chart for AS-IS and TO-BE Process – 1a) QuARS at Systems Requirements phase

We made changes to the model to represent each configuration, and then ran each model for 30 runs. Based on consistency tests performed during model verification and validation process, we found that with inherent variability in the model, 30 runs were sufficient for the model to produce stable results. Table 2 shows the differences of the model mean results from three configurations comparing to the AS-IS baseline performance. Note that a positive value means improvement. For example, when employing QuARS at System Requirements phase, the total effort (including IV&V effort) reduced by 1,659.07 man-hours.

Table 2. Scenario 1 Performance Comparison to the Baseline

Comparison to Baseline

	Effort incl. IV&V	Effort	Rwrk_Efrt	IV&V Effort	Duration	Avg. Dur	Crctd_Dfcts	Ltnt_Dfcts
1a) QuARS at Sys Req	1,659.07	1,669.63	1,311.82	-10.56	103.00	48.64	33.98	18.14
p value	0.00	0.00	0.00	0.87	0.05	0.05	0.56	0.00
1b) QuARS at Sw Req	5,141.86	5,127.99	4,778.59	13.87	377.28	71.50	-10.12	55.12
p value	0.00	0.00	0.00	0.83	0.00	0.01	0.86	0.00
1c) QuARS at Sys & Sw Req	5,267.99	5,284.64	4,925.64	-16.65	362.00	80.63	-9.89	58.54
p value	0.00	0.00	0.00	0.80	0.00	0.00	0.87	0.00

One can see that applying QuARS resulted in better overall project performance. In all three cases, effort spent was lower; the duration was shorter; and the quality was improved. The effort was improved because of the increase in productivity in

subsequent development phases as a result of better requirements document. In addition, QuARS allows us to detect and correct defects early in the process, which results in lower rework cost. With better and clearer requirements, the quality of the overall product also improved.

It is interesting note that applying QuARS at Software Requirements phase (1b) yielded a more significant improvement than applying QuARS at System Requirements phase (1a). When applying QuARS at Software Requirements phase, the effort decreased by almost 3,500 man-hours and the average number of latent defects reduced by more than double (37 defects), as compared to applying QuARS at System Requirements phase. Applying QuARS at both phases resulted in marginal improvement on effort and quality; however, the duration was a bit longer than applying QuARS only at Software Requirements phases.

Moreover, we experimented with the option of applying QuARS before or after requirements inspection. Although we found that applying QuARS after requirements inspection does improve the project performance as compared to the baseline, the benefit of applying QuARS after a requirements inspection is 10% to 15% lower than when applying QuARS before requirements inspection.

4.3 Scenario 2: Applying QuARS in IV&V Mode at Different Phases

For this scenario, we examined the impact of QuARS if we apply it during IV&V activities. Changes were made to the model to represent three different configurations: 2a) QuARS at the Concept Verification phase; 2b) QuARS at the Requirements Verification phase; and 2c) QuARS at both phases. Figure 3 shows flow charts of the AS-IS and TO-BE processes for configuration 2b) QuARS at Requirements Verification phase.

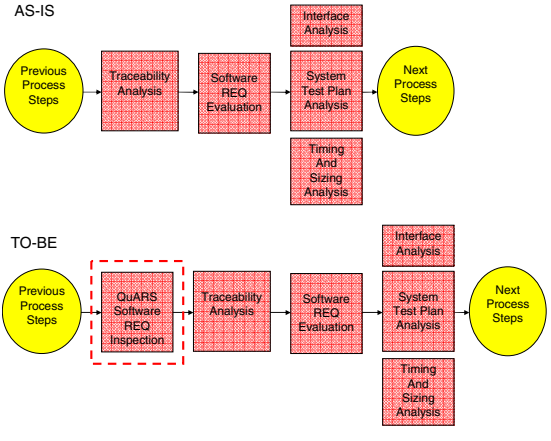


Fig. 3. Process Flow Chart for the AS-IS and TO-BE Process – 2b) QuARS at the Requirements Verification phase

We made changes to the model to represent each configuration, and then ran each model for 30 runs. Table 3 shows the differences of the model mean results from three configurations comparing to the AS-IS baseline performance.

Table 3. Scenario 2 Performance Comparison to the Baseline

Comparison to Baseline

	Effort incl. IV&V	Effort	Rwrk_Efrt	IV&V Effort	Duration	Avg. Dur	Crctd_Dfcts	Ltnt_Dfcts
2a) QuARS at Concept V	1,448.16	1,679.42	1,321.83	-231.26	114.25	68.84	31.97	17.08
p value	0.00	0.00	0.00	0.00	0.01	0.01	0.59	0.01
2b) QuARS at Requirments V	2,427.46	2,717.04	2,340.55	-289.58	190.67	64.10	18.92	28.59
p value	0.00	0.00	0.00	0.00	0.00	0.01	0.75	0.00
2c) QuARS at both	2,899.94	3,373.50	2,975.94	-473.56	236.75	97.55	10.73	35.96
p value	0.00	0.00	0.00	0.00	0.00	0.00	0.86	0.00

Similar to scenario 1, applying QuARS in IV&V did improve project performance as compared to the baseline model for all three configurations. However, the value of QuARS as an IV&V tool is significantly less than the value of QuARS as a V&V tool. The effort reduced by 2% to 4% when applying QuARS at IV&V mode, while the effort reduced as much as 8% when applying QuARS at V&V mode. The reason for this is that the secondary effects as discussed in Section 3.2 were not experienced by the project when employing QuARS in IV&V mode.

From the results of these two scenarios, we can conclude that QuARS did add value to the project by reducing effort, shortening project duration, and improving quality. However, the phase (location) that QuARS will be applied is very important. The degree of value added depends on the location that QuARS is applied. Applying QuARS at V&V model offers more benefits that applying QuARS at IV&V mode. Applying QuARS at both Systems and Software Requirements phases yield the highest benefit, but the actual sweet spot is to apply QuARS at the Software Requirements phase. In addition, QuARS should be applied before the Requirements inspection in order to capture the most benefit.

Although the analysis above concluded that QuARS does add value to the project, it didn't take into account the cost of purchasing QuARS. If the cost of QuARS exceeds its benefits, it will not be worthwhile to implement QuARS. The next step is to calculate the maximum price that the project would be willing to pay for QuARS. The analysis to answer this question is provided in the next section.

4.4 Financial Analysis

In order to weigh the projected benefits received from QuARS against the cost of implementing the tool, the first thing we need to do is to convert the project performance measures (in terms of effort, duration, and quality) to the financial measures. In addition, when evaluating future benefits, it is crucial to consider the timing of the benefits (i.e. when will we realize the benefits). Therefore, we will use present value (PV) based method to perform the financial analysis for this case study. Detailed information about financial calculation for software process improvement business case can be found at [25]. There are several key parameters required for the analysis as follows:

- The organization's internal investment rate cut-off (a.k.a. hurdle rate) is 20% annually.
- The cost of development staff is $100 per hour. The cost of IV&V staff is also $100 per hour.

- The cost to correct latent defects after release is 1.5 person-month (or $25,500 per defect).
- There are 170 work hours per month.
- Implementation costs (QuARS costs) are assumed to be incurred at time = 0, development costs can be assessed as a one time cash flow when the project is completed (time = duration), costs to fix latent defects occurs at 1 years after the project is completed (time= duration + 12 months).
- There is no benefit gain if the project completes early. Note that this is specific to the organization. Other organizations may gain benefit if the software is released early (i.e. increase in market share/revenue).

From the information above, we can write the Net Present Value (NPV) equation as follows:

$$NPV = IC + SE/(1+r/12)^{\wedge Duration} + SRW/(1+r/12)^{\wedge Duration + 12} \quad (1)$$

Where:

IC = Implementation cost (value of QuARS in our case)
SE = Saving in Effort
r = Organizational internal investment rate cut-off
Duration = Project duration (month)
SRW = Saving in rework cost

If NPV > 0, the project will benefit from using the tool. If NPV < 0, project will lose money when using the tool. To find the maximum amount that the project should be willing to pay for the tool, we want to find IC that will make the NPV equal to 0. Therefore, we assume that NPV = 0 and then solve for IC. Table 4 shows the value of QuARS for different scenarios.

Table 4. Value of QuARS

Config.	QuARS Value	
	Mean	Std Dev
1a) QuARS at Sys Req	$329,350.06	41,623.20
1b) QuARS at Sw Req	$1,012,909.55	53,732.26
1c) QuARS at Sys & Sw Req	$1,094,509.64	68,700.92
2a) QuARS at Concept V	$313,387.99	32,630.94
2b) QuARS at Requirments V	$511,362.33	39,002.30
2c) QuARS at both	$638,714.67	50,579.24

The probability that the QuARS value is higher than $0 is 100%, which indicates that QuARS helps improve project performance. The probability that the QuARS value is higher than $100,000 is also 100%. This suggests that if the total cost of QuARS implementation is $100,000, the project would gain significantly (between $213,388 and $994,510) should it decide to implement QuARS.

Note that the key parameters provided at the beginning of this section are similar to values commonly found in the industry. The advantage of simulation is that any of these numbers could be changed to reflect a different organization/situation. For example, if the project was performed offshore where the programmer wages are lower, the value of QuARS (benefit from implementing QuARS) would be lower. We found

that if the cost of staff was reduced by 50%, the value of QuARS would range between $264, 828 and $928,408.

We also varied key financial parameters such as hurdle rate, cost of staff, and cost to correct latent defects to evaluate its impact on the value of QuARS. In general, the value of QuARS decreases when hurdle rate increases, cost of staff decreases, or cost to correct latent defect decreases. In addition, we also conducted sensitivity analysis on important QuARS parameters provided in Section 3.2: QuARS Assumptions include: percent of the requirements defects that are QuARS detectable, QuARS effectiveness (detection rate), and percent improvement on Requirements inspection detection capability. However, in the interest of space, these analyses have not been included in this paper.

5 Conclusion

The primary benefits of using Software Process Simulation Models include: (a) selection of the best possible development process for specific situations and circumstances, (b) improved project planning and execution, (c) provision for an objective and quantitative basis for project decisions, (d) reduced risk when implementing process changes, (e) enhanced understanding of possible outcomes in complex processes and projects.

Software process simulation is a powerful tool for conducting what-if analysis. This can help project managers evaluate the impact of process changes or new tool implementations. In this paper, we showed how simulation can be used to evaluate a new technology, QuARS (a Quality Analyser for Requirements Specification). In addition to assessing the value of QuARS in general, we also used simulation to determine the impact of adding QuARS at different phases in the project. This analysis can help project managers identify the optimum point in the process to apply QuARS to capture full potential benefits. We found that, in general, applying QuARS resulted in better overall project performance. However, the degree of the value added depends on the insertion point and step order in which QuARS is applied. Applying QuARS at the in-project V&V level offers more benefits than applying QuARS externally to the project in IV&V mode. Applying QuARS at both the Systems and Software Requirements phases yields the highest benefit, but the actual sweet spot is to apply QuARS at the Software Requirements phase. In addition, QuARS should be applied before the Requirements inspection in order to capture the most benefit. The financial analysis presented in this paper shows how one can translate the impact of a new technology into financial value, which makes it easier to make a decision as to whether to acquire a new tool.

When using simulation to assess a new technology, we can:

- Identify the conditions where a new technology would be useful and where it would be useless
- Identify the optimum point in the process to employ new tool
- Assess the risks associated with a new technology
- Establish performance benchmarks or criteria that the vendor of a new tool would have to achieve in order for the organization to consider investing and adopting a new tool or technology.

References

1. Lami, G., Gnesi, S., Fabbrini, F., Fusani, M., Trentanni, G.: An Automatic Tool for the Analysis of Natural Language Requirements. International Journal of Computer Systems Science and Engineering, Special Issue on Automated Tools for Requirement Engineering **20** (2005)
2. Kellner, M.I., Madachy, R.J., Raffo, D.M.: Software Process Simulation Modeling: Why? What? How? Journal of Systems and Software **46** (1999) 91-105
3. Raffo, D.: Predicting the Impact of potential process changes: A quantitative approach to process modeling. In: Emam, K.E., Madhavji, N.H. (eds.): Elements of Software Process Assessment and Improvement. IEEE Computer Soc. Press, Los Alamitos, CA (1999)
4. Raffo, D., Harrison, W., Vandeville, J.: Coordinating models and metrics to manage software projects. Software Process Improvement and Practice **5** (2000) 159-168
5. Raffo, D.M., Vandeville, J.V., Martin, R.H.: Software process simulation to achieve higher CMM levels. Journal of Systems and Software **46** (1999) 163-172
6. Raffo, D., Settle, J., Harrison, W.: Estimating the Financial Benefit and Risk Associated with Process Changes. First Workshop on Economics-Driven Software Engineering Research, International Conference on Software Engineering (ICSE 99), Los Angeles, California (1999)
7. Osterweil, L.J., Sondheimer, N.K., Clarke, L.A., Katsh, E., Rainey, D.: Using Process Definitions to Facilitate the Specification of Requirements. Department of Computer Science, University of Massachusetts, Amherst, MA 01003 (2006)
8. Pfahl, D., Laitenberger, O., Dorsch, J., Ruhe, G.: An Externally Replicated Experiment for Evaluating the Learning Effectiveness of Using Simulations in Software Project Management Education. Empirical Software Engineering **V8** (2003) 367-395
9. Melis, M., Turnu, I., Cau, A., Concas, G.: A Software Process Simulation Model of Extreme Programming. The 6th International Workshop on Software Process Simulation and Modeling, St. Louis, Missouri (2005)
10. Abdel-Hamid, T.K.: Dynamics of software project staffing: A system dynamics based simulation approach. IEEE Transactions on Software Engineering **15** (1989) 109-119
11. Madachy, R.J.: System dynamics modeling of an inspection-based process. (1996) 376-386
12. Smith, N., Capiluppi, A., Fernández-Ramil, J.: Agent-based simulation of open source evolution. Software Process: Improvement and Practice **11** (2006) 423-434
13. Martin, R.H., Raffo, D.: A model of the software development process using both continuous and discrete models. Software Process: Improvement and Practice **5** (2000) 147-157
14. Setamanit, S., Wakeland, W., Raffo, D.: Using Simulation to Evaluate Global Software Development Task Allocation Strategies. Special Issue on Software Process, Software Process: Improvement and Practice (Forthcomming)
15. Wakeland, W.W., Martin, R.H., Raffo, D.: Using design of experiments, sensitivity analysis, and hybrid simulation to evaluate changes to a software development process: a case study. Software Process: Improvement and Practice **9** (2004) 107-119
16. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. The McGraw-Hill Companies, Inc., New York (2003)
17. Leffingwell, D., Widrig, D.: Managing software requirements: a unified approach. Addison-Wesley Longman Publishing Co., Inc. (2000)
18. Ferguson, R.W., Lami, G.: An Empirical Study on the Relationship between Defective Requirements and Test Failures. the 30th IEEE-NASA Annual Software Engineering Workshop (SEW-30) IEEE Computer Society Press, Columbia, MD U.S.A. (2006)

19. Lami, G., Ferguson, R.W.: An Empirical Study on the Impact of Automation on the Requirements Analysis Process. Journal of Computer Science and Technology (JCST) (Forthcomming)
20. Raffo, D., Nayak, U., Wakeland, W.: Implementing Generalized Process Simulation Models. The 6th International Workshop on Software Process Simulation and Modeling, St. Louis, Missouri (2005)
21. IEEE/EIA 12207.0-1996 IEEE/EIA Standard Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology Software Life Cycle Processes. IEEE/EIA 12207.0-1996 (1998) i-75
22. Raffo, D., Nayak, U., Setamanit, S.-o., Sullivan, P., Wakeland, W.: Using Software Process Simulation to Assess the Impact of IV&V Activities. Proceeding of the 5th International Workshop on Software Process Simulation and Modeling (ProSim'04), Edinburgh, Scotland (2004)
23. Jones, C.: Applied software measurement: assuring productivity and quality. McGraw-Hill, Inc. (1991)
24. Raffo, D., Wakeland, W.: High Value Added Ways to Apply Process Simulation with Organizations. Software Engineering Institute, Carnegie Mellon University (Forthcoming)
25. Harrison, W., Raffo, D., Settle, J., Eickelmann, N.: Technology Review: Adapting Financial Measures: Making a Business Case for Software Process Improvement. Software Quality Control **8** (1999) 211-231

A Framework for Adopting Software Process Simulation in CMMI Organizations

He Zhang[1,2], Barbara Kitchenham[2], and Ross Jeffery[1,2]

[1] School of Computer Science and Engineering, UNSW
[2] National ICT Australia
{he.zhang,barbara.kitchenham,ross.jeffery}@nicta.com.au

Abstract. The Capability Maturity Model Integration (CMMI)[1] has become very influential as a basis for software process improvement. It is accepted that process maturity is associated with better project performance and organizational performance. Software process simulation is being applied to the management of software projects, product life cycles, and organizations. This paper argues that the successful adoption of one particular simulation paradigm to a large extent depends on an organization's capability maturity. We investigate four typical simulation paradigms and map them to their appropriate CMMI maturity levels. We believe that an understanding of these relationships helps researchers and practitioners in implementing and institutionalizing process simulation in software organizations.

Keywords: CMMI, process simulation and modeling, process improvement.

1 Introduction

Process simulation methods were fully introduced to software engineering by Abdel-Hamid's [1] and others' efforts in the late 1980s. However, in our experience of Australian software industry, these methods are seldom adopted in practice. One possible reason might be the lack of guidance for selecting and adopting the appropriate simulation and modeling paradigms in a specific organization's context. This paper aims to stimulate research in this important area.

CMM(I)-based process improvement has been discussed for many years in the community of software process simulation and modeling. For instance, Christie [2] argues that CMM-based process improvement can benefit from process simulation, and that simulation can help to tackle different questions on CMM levels. However, he did not distinguish the different simulation techniques in his discussion. Raffo *et al.* [3] further suggest that process simulation serves as organization's strategy for achieving a higher process capability and moving to higher CMM levels. However, there is no 'one-size-fits-all' simulation solution for all organization contexts, in particular organizations at different CMMI maturity levels. The selection of a suitable process simulation paradigm is the first necessary step to realize the value of simulation for software organizations. Unfortunately, there is a lack of the generic guidance on how to select the appropriate

[1] CMM and CMMI are service marks of SEI, Carnegie-Mellon University.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 320–331, 2007.

simulation paradigm(s). One possible problem is applying purely quantitative simulations for the organizations at lower maturity levels. In contrast to most previous discussions focusing on the simulation's positive contributions on CMM-based process improvement, this paper argues that the adverse effects can also occur. We propose a framework to support selection of the appropriate simulation paradigms by mapping selected process simulation techniques to their related CMMI levels. This provides general guideline for the adoption of process simulation in software organizations.

Section 2 discusses the scope of our proposed framework, and introduces the related concepts of process simulation and CMMI. We explain the framework and justify the mapping across the maturity levels in Section 3. It is followed by discussion of some associated issues (Section 4). Finally, Section 5 presents our conclusions and plans for future work.

2 Background and Motivation

When a software organization achieves a particular CMMI maturity level, it can be assessed as capable to adopt particular simulation paradigm(s); on the other hand, maturity levels are static points when introducing new simulations, and the adoption facilitates the process of improvement to the next higher level. The purpose of this framework is to help organizations maximize the benefits gained from process simulation; and use the appropriate paradigm(s) needed to achieve higher maturity levels.

2.1 Process Simulation Modeling

Scope. Kellner *et al.* present a wide variety of reasons for undertaking simulations of software process models [4]. Primarily, process simulation is an aid to decision making. They identified six categories of purposes: (1) strategic management; (2) planning; (3) control and operational management; (4) process improvement and technology adoption; (5) understanding; and (6) training and learning. We note that the last two objectives can benefit from all simulation paradigms no matter what maturity level the organization is on. For example, even a Level 1 organization can apply a role-playing simulation game, and gain insight from it. However, such application is not the case of simulation in support of the real process in a practical situation. The framework of this paper focuses on purposes (1) through (4).

Paradigms. We reviewed the software process simulation models published in ProSim[2] special issues since 1998, and selected two of the most popular simulation techniques for our framework: *System dynamics* (SD, 54%) and *Discrete-event simulation* (DES, 27%). Another reason for selecting the approaches is that they represent two different simulation approaches. The former is the widely applied *continuous simulation* paradigm which captures higher level project or product considerations and shows how feedback loops connect a variety of business characteristics. In contrast, *discrete simulation* is the modeling of systems in which the state variable changes only at a discrete set of points (events) in time [5]. It is

[2] International Workshop on Software Process Simulation Modeling.

excellent at capturing well-defined process tasks, incorporating, queuing and scheduling considerations.

However, both methods are purely quantitative approaches for modeling and simulating systems, and organizations at lower ends of CMMI lack the ability to obtain major benefits from these simulations (in-depth discussion in Section 3). Therefore, the framework must include two newly introduced simulation paradigms: *Qualitative simulation* and *Semi-quantitative simulation*. Table 1 presents their relationships.

Table 1. Selected simulation paradigms

	Continuous	Discrete
Quantitative	**System Dynamics**	**Discrete-event Simulation**
Qualitative	**Semi-quantitative Simulation**	
	Qualitative Simulation	

Qualitative simulation modeling reflects the systems in the real world at an abstract level. Fewer assumptions are required than for purely quantitative approaches. The outputs generated by Qualitative simulation are all the possible behaviors of the system, whose states are described by qualitative landmarks, instead of numeric values. Some initial ideas regarding the application of qualitative modeling to software engineering were discussed by Suarez *et al.* [6], however, the first major example of its use was Ramil and Smith's study of software evolution [7].

As an extension of Qualitative simulation, Semi-quantitative simulation focuses on the use of bounding intervals to represent partial quantitative knowledge [8]. This paradigm provides a seamless transition between purely qualitative and quantitative approaches. Our previous work introduced semi-quantitative approach by developing a qualitative model of the software staffing process with quantitative constraints [9].

2.2 CMMI

As the successor of CMM, CMMI describes the practices for software process change, and frameworks for measuring the compliance of organizations. CMMI selects only the most important topics for process improvement and then groups those topics into "areas". It represents ten years of lessons learned from many thousands of external and internal consultants, based on applying continuous improvement to CMM itself [10].

Representation. Unlike its predecessor, CMMI offers two representations, i.e. staged models for assessing organizational maturity and continuous models for measuring process capability. The main difference between *maturity levels* (MLs) and *capability levels* (CLs) is the representation they belong to and how they are applied. Table 2 shows the maturity levels of staged representation [11, 12]. We choose the staged representation for our framework for the following reasons:

- It provides a recommended path of improvement evolution (Fig. 1) for the entire organization based on the last decade's best practices.
- It allows comparisons across organizations by using appraised maturity levels.
- The single rating can be used as the indicator of the organization's overall maturity level, and provides an easy mapping to simulation paradigms.
- It provides a smooth migration from CMM to CMMI.

Table 2. CMMI Staged representation

Maturity Level	Staged Representation Maturity Levels
1	**Initial**
2	**Managed**
3	**Defined**
4	**Quantitative Managed**
5	**Optimizing**

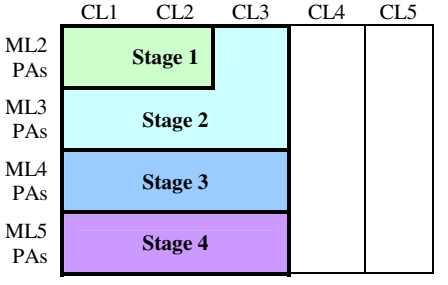
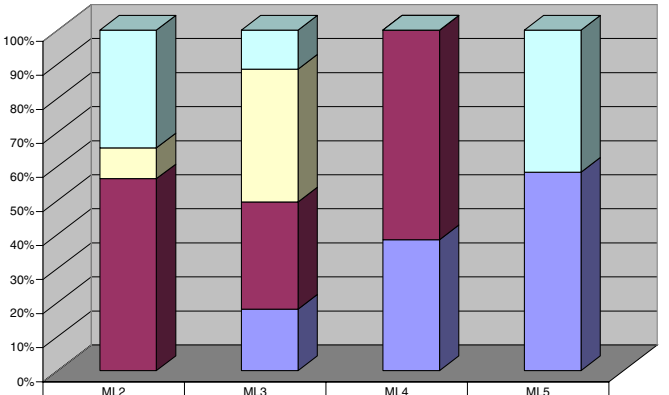


Fig. 1. Capability profile for maturity levels

Process Areas. CMMI contains 25 *process areas* (PAs) and 185 *specific practices* (SPs) grouped into four categories according to their scopes (Fig. 2). *Project Management* process areas consist of project management activities related to planning, monitoring, and controlling the project. *Process Management* process areas provide the organization with capability on cross-project activities related to defining, deploying, implementing, monitoring, appraising, measuring, and improving processes. *Engineering* process areas cover product development and maintenance activities shared across engineering disciplines. They define the product development processes rather than discipline-specific processes (such as software engineering). Since the *Support* process areas address processes that are used in the context of performing other process areas [11, 12], the first three process area groups are the main aspects considered in our framework at current stage.

Practices. The required component of the CMMI models is the "goal" that represents a desirable end state, and indicates that a certain degree of project and process control has been achieved. A *specific goal* (SG) is unique to a single process area; in contrast, a *generic goal* (GG) may apply across all of the process areas. Therefore, the proposed framework focuses on specific goals and specific practices, which represent the “expected” means of achieving the goal, and their different bias at maturity levels. We calculated the number of specific practices applied to each maturity level and categorize them into process area groups (Fig. 2). Although the allocated effort varies across the practices, and even for the same practice performed among different software organizations, this comparison generally illustrates the emphasis of improved process areas on each maturity level.



	ML2	ML3	ML4	ML5
Support	19	12	0	5
Engineering	5	41	0	0
Project Management	31	33	8	0
Process Management	0	19	5	7

Fig. 2. Allocation of specific practices across maturity levels

3 Mapping Framework

Simulation models may be mixed, both discrete and continuous. The choice of whether to use a discrete or continuous (or mixed) simulation model is a function of the characteristics of the system and the objective of study [5]. For a software organization, CMMI provides an assessment framework for the organization's capability (i.e. characteristics of the system), and the target of process improvements (which is one of possible objectives of study).

As CMMI depicts a progressive path to achieve continuous process improvement, the adoption of process simulation paradigms depends on the evolution of organization's capability. The mapping is implemented by analyzing the characteristics of software organization and the practices introduced at each maturity level, and comparing with the inherent capability of each simulation paradigm. We also provide simulation models of one well-defined software process as an example for each transition.

3.1 Overview

The introduction and adoption of the process simulation paradigm in an organization is also a process, rather than a single point of CMMI assessment. Thus simulations are normally introduced between two adjoining CMMI maturity levels (Fig. 3).

Along with the evolution of capability maturity, the organization can provide more precise process information with richer details. On the other hand, the simulation paradigm introduced at higher maturity level requires more specific, lower level, and quantitative information about specific software processes. This framework visualizes an evolution path of simulation paradigms for CMMI organizations, i.e. from qualitative to quantitative, from continuous to discrete, then to hybrid (see Section 3.6).

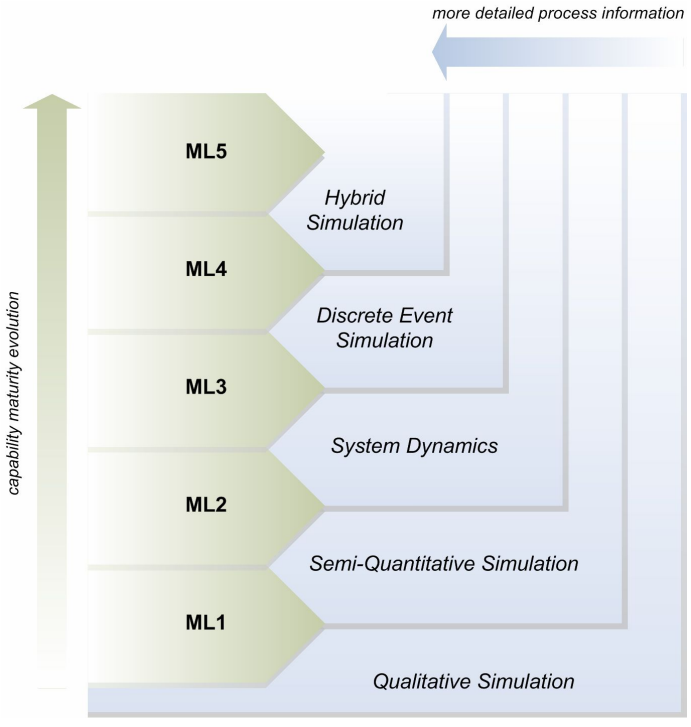


Fig. 3. Framework of adopting process simulation modeling in CMMI organizations

3.2 Initiating at ML1

At the entry level, software processes are performed in a chaotic and unstable organizational environment. "Maturity level 1 organizations are characterized by a tendency to over commit, abandon processes in the time crisis."[12] Given a poorly-defined process, significant uncertainty and high risk associated within such situation, project success and process performance can not be predicted quantitatively. Because of the large variance and contingency of the development behaviors, the organization is unable to repeat their past success.

Although it is the ad-hoc level and no stable processes are followed in organization, some qualitative disciplines still work (e.g. Brooks' Law and defect amplification across development phases). Qualitative assumptions can be abstracted from these disciplines, corresponding to general knowledge about the software development process. A *Qualitative Simulation* model can be then developed based on these qualitative assumptions.

Example. The software process focusing on staffing level is modeled and simulated as an example for each transition in this paper. Brooks' Law might be the most well-known statement about the software staffing problem. It argues "adding manpower to a late software project makes it later" [13], and has a negative impact on software development productivity. Our previous research developed a qualitative simulation

model to examine Brooks' law [14]. This model is built on ten basic qualitative assumptions of the software staffing process, such as "adding more people to a project results in a larger communication overheads", and "new employees' productivity is initially lower than experienced staff's productivity". The qualitative model generates 112 possible behaviors to describe the staffing process. Even without quantitative information, the simulation results can justify that under some scenarios adding more people helps the project complete earlier than the original schedule.

3.3 Transitioning from ML1 to ML2

In progressing to ML2, organizations start to apply the generic and specific practices. As illustrated in Fig. 2, over 55% specific practices implemented in this transition concentrate on adopting "basic" project management methodologies (process areas). On the other hand, no specific practice of process management is introduced until achieving ML2. Hence, the main improvements are expected on the project or management level, not the process level. For example, the estimates of attributes of the work products are established and maintained (PP-SP1.3); based on estimation rationale, project effort and the cost for work products are established (PP-SP1.4); project risks are identified and analyzed (PP-SP2.2); the actual values of the project parameters are monitored against the project plan (PMC-SP1.1); the project's progress, performance and issues are periodically reviewed (PMC-SP1.6), etc.

However, quantitative project management processes must be adopted progressively. It takes time not only to accumulate sufficient project history data, but to specify how measurement data will be obtained, stored, analyzed, and reported (MA-SP1.3/1.4). This implies there might be significant variance, even inconsistency, in the data collected during this transition. In addition, it can be noted in Fig. 1 that all adopted process areas at current stage are targeted at capability level 2, other than CL3 required for higher maturity levels. This implies only primary quantitative project management capability is expected at maturity level 2.

Obviously, the discrete paradigm is not appropriate for simulation in this transition, because of the absence of the specific practices of process management. In contrast, the continuous paradigms are more suitable for capturing the project or product characteristics at high level. However, in light of the lack of formal and complete history data from ML1, the estimates of project metrics are mainly based on project manager's personal experience and incomplete history information. Although quantitative simulation can cope with the uncertainty with stochastic methods, e.g. Monte Carlo simulation, unfortunately, the number of uncertain factors may be too many to handle in this way, and the statistical distributions are unknown or unstable at this stage. Therefore, blindly adopting a purely quantitative simulation within such context may result in over-optimistic or -pessimistic predictions, and may discourage the implementation of process simulation due to unreasonable expectations.

As the extension of Qualitative simulation, *Semi-quantitative Simulation* provides a seamless transition between qualitative and purely quantitative approaches. It can be introduced as a lens with a smooth zoom to match the organization's immature but continuously improved quantitative capability during this transition.

Example. The software staffing process model is extended with the quantitative information in [9]. Because the uncertainty is still high at level 2, and only incomplete historical project data is available, Semi-quantitative simulation assigns envelope functions to the relations in the model, and value ranges to the inputs and its initial state. When we apply stricter quantitative constraints, the simulation produces fewer but more precise behaviors for the specific staffing process. Given the primary and limited quantitative management capability (between ML1 and ML2), Semi-quantitative simulation is able to provide the possible behaviors of development process for decision-making while maintaining the integrity of the final solution.

3.4 Transitioning from ML2 to ML3

Once the organization achieves ML2, projects can be managed and a few successful project management practices can be repeated. The next transition to ML3 will produce the most distinct improvements across the maturity levels, because over 55% of all CMMI practices must be implemented successfully in order to reach ML3 (Fig. 2). The main adopted process areas in this transition are *Engineering* (39%) and *Project Management* (31.5%). The "advanced" *project management* practices (except "quantitative project management" processes) are introduced, such as to establish and maintain the project's defined process (IPPD-SP1.1); to define the parameters used to analyze and categorize risks and control risk management effort (RM-SP1.2).

System dynamics can be introduced during this transition for more precise management based on experience and knowledge. System dynamics is a dynamic feedback system, sometimes refined as a goal-seeking system. It is possible to study the interaction of control policies, exogenous events and feedback structures producing dynamic behavior, such as rise, drop or oscillation. System dynamics simulate the software process as a set of performance indicators. Most of them are active during the whole project or project phases. Although as one alternative solution beyond ML2, the Semi-quantitative simulation offers the capability of purely quantitative continuous simulation, here we prefer System dynamics for its wide application (the most popular simulation paradigm applied in software process modeling).

Meanwhile, the organization starts to adopt the "basic" *process management* practices (for ML3), such as to establish and maintain the description of the process needs and objectives for the organization (OPF-SP1.1); to establish and maintain the organization's set of standard processes (OPD-SP1.1); to deploy organizational process assets across the organization (OPF-SP1.1), etc. Thus, the organization can seldom benefit from Discrete event modeling until ML3 when these practices are well defined and implemented across the parts or the whole organization.

Example. Madachy developed a System dynamics model of the software staffing process to examine the Brooks' Law [15]. He simplified Abdel-Hamid and Madnick's model [1] by focusing on the assimilation procedure. The model is built using a set of specific numeric values, which were selected from the literature or historical data of company projects, to represent the relations in the model. Further, the process is simulated with the data from specific projects as inputs. His model generates single deterministic behavior through one simulation, and he analyzes the impact of different staffing policies by comparing the numeric values describing the project states through multiple runs.

3.5 Transitioning from ML3 to ML4

When software processes are well-defined at ML3, Discrete event simulation can be introduced (Fig. 2). A "defined process" clearly states: purpose, inputs, entry criteria, activities, roles, measures, verification steps, outputs, and exit criteria [12]. The discrete models are capable of capturing a well specified process, which is composed of the above process elements. Discrete event simulation is suitable to model a queuing system, which is observed by arrival rate, service time, queue capability and discipline [5]. The entities will be moved from one queue to another during simulation.

Discrete event simulation is a typical method employed in stochastic queuing models. All pre-defined rules, such as arrival rate and service times, will be sampled from the appropriate distributions. At ML3, the organization's measurement repository has been established and maintained (OPD-SP1.4), and the process asset library has been established and maintained (OPD-SP1.5), etc. These practices provide the condition for applying statistical methods and tailoring at the process level.

The maturity level 4 aims to achieve a "quantitatively managed process". A critical distinction between a well defined process and quantitatively managed process is the predictability of the process performance. The latter implies using appropriate statistical techniques to manage process performance so that the future performance can be predicted [11]. Discrete event modeling tries to answer "what if" questions. The model is run many times with different input variables, entity allocations and statistical distributions. The results are collected and examined to support the "quantitative project management" and improve the "organizational process performance", which contain the all (13) specific practices implemented at ML 4.

Example. Antoniol *et al.* develop three different queuing models, composed of nodes assessment, technical analysis, enactment and unit testing, to model a software maintenance process [16]. The stochastic discrete simulation was then used to compute the required team size (for the different nodes of each model) under the constraint to complete maintenance activities. In terms of the clearly defined process (which is required for ML3 and above), several simulations were carried out with changing team size (servers) for each node until all expected work packets were processed by the deadline. Project staffing levels were then refined to reach a compromise between personnel cost and waiting time.

3.6 Transitioning from ML4 to ML5

In our framework, typical simulation paradigms are introduced at ML1 through ML3 separately. Maturity level 4, where this transition starts, is characterized as a quantitative managed project and process. All four process simulation paradigms have been institutionalized in Level 4 organizations. They possess the competency to employ these simulation paradigms separately. At ML4, Hybrid simulation is proposed by combining multiple simulations and modeling techniques to help organization achieve continuous optimization.

Hybrid modeling means not only employing the different modeling paradigms concurrently, but developing an integrated model with modules created by different paradigms. For example, when pilot process and new technology are considered to

implement process improvement (OID-SP1.3), Qualitative or Semi-quantitative modeling may be employed for the specific module due to the limited knowledge about a specific process; a combination of System dynamics and Discrete event simulation can facilitate the causal analysis of selected defects and other problems (CAR-SP1.2) on project and process levels, and further help the evaluation of changes on process performance (CAR-SP2.2).

Example. As an example, Raffo and Setamanit develop a hybrid model that simulates the global software development (GSD) process with the component of staffing process [17]. The discrete event and system dynamic paradigms compliment each other and together enable the construction of models that capture both the dynamic nature of project variables and the complex sequences of discrete activities that take place. At a high level, their model has three major components: DES sub-model, SD sub-model, and Interaction Effect sub-model. The SD sub-model consists of a global SD sub-model and a site-specific sub-model, which include Human Resources (HR) modules. The modules deal with HR management, which involves hiring, training, assimilation, and transferring workforce. Whereas the DES sub-models simulate how tasks are allocated and specific activities are performed on site and global levels.

4 Discussion

This framework is not limited to organizations with CMMI certificates, but applies to any software organization that operates at a particular CMMI level. CMMI is a widely-accepted and easily-accessed maturity model. Our framework provides a general and approximate guideline for selecting and adopting process simulation. An organization can perform the self-assessment against the capability characteristics described at CMMI maturity levels, and then select the suitable simulation(s).

Each process simulation paradigm involved in our framework can also be applied at the maturity levels above its introduction level (Fig. 3). For instance, when a Level 5 organization plans to adopt a new technology or a new software process, Qualitative simulation may help to gain insight in the implication of the change. As another example, the success of a contemporary project might be better defined as a *cube* of metrics than a single point [18]. Since semi-quantitative approach has the inherent capability of coping with uncertainty in multi-dimensions, it can facilitate the decision-making under this condition even if for the organizations at higher levels [19].

Although Semi-quantitative simulation can predict process performance with value ranges and possible behaviors, it does not imply imprecision, it allows continuous refinement. The tolerance with its presentation guarantees the integrity of final solutions. When a software organization employs Semi-quantitative approach, its maturity level can be regarded as the capability to reduce the uncertainty by applying finer value ranges and more realistic envelope functions to specify its software practice.

Besides the consideration of an organization's maturity level, selection and adoption of process simulation paradigms also depend on other constraints, such as expertise with simulation and modeling tools, previous adoption experience. Meanwhile, process simulation should focus on the needs of an organization in the context of its business environment and the current needs of an organization and project. Transition

between paradigms is not mutually exclusive for simulation paradigms, prior techniques can be retained and optimized while introducing a new paradigm.

Along with the evolution of process capability, different process simulation paradigms are introduced into organization incrementally and separately. In staged representation of CMMI model, each maturity level forms a necessary foundation on which to build the next level, so trying to skip maturity levels is usually counterproductive [12]. Though organizations can introduce specific simulation paradigm at any time they choose (even before they are ready for advance to the recommended maturity level), similarly, skipping the adoption of simulation paradigm(s) for lower maturity level(s) is not recommended in our framework. For example, some organizations may try to collect the detailed process data for discrete event simulation, but they are likely to suffer from the inconsistency in processes and measurement definitions.

At present, this framework includes only four typical process simulation paradigms. However, its open structure provides a means to introduce other simulation paradigms and locate them at appropriate positions in the future.

5 Conclusion

This paper has proposed a framework by analyzing the organizational characteristics on CMMI maturity levels and requirements of the typical process simulation paradigms, and establishing a suitable mapping between them. This framework provides software organizations a primary guideline for selecting and adopting process simulation by assessing their CMMI maturity levels. This research can be extended by:

- Analyzing the maturity requirements of other simulation paradigms (e.g. state-based simulation) and including them at appropriate positions in the framework;
- Collecting more empirical evidence for validating and supporting our argument.

References

1. Abdel-Hamid, T.K. and S.E. Madnick, *Software Project Dynamics: An Integrated Approach*. 1991, Englewood Cliffs, N.J.: Prentice Hall.
2. Christie, A.M., *Simulation in Support of CMM-based Process Improvement.* Journal of Systems and Software, 1999. **46**(2/3).
3. Raffo, D.M., J.V. Vandeville, and R.H. Martin, *Software Process Simulation to Achieve Higher CMM Levels.* Journal of Systems and Software, 1999. **46**(2/3).
4. Kellner, M.I., R.J. Madachy, and D.M. Raffo, *Software Process Simulation Modeling: Why? What? How?* Journal of Systems and Software, 1999. **46**(2/3).
5. Banks, J. and J.S. Carson, *Discrete-Event System Simulation.* 1984, Englewood Cliffs, NJ: Prentice-Hall.
6. Suarez, A.J., et al., *Qualitative Simulation of Human Resources Subsystem in Software Development Projects*, in *16th International Workshop on Qualitative Reasoning*. 2002: Sitges, Spain.
7. Ramil, J.F. and N. Smith, *Qualitative Simulation of Models of Software Evolution.* Software Process: Improvement and Practice, 2002. **7**(3-4).
8. Kuipers, B., *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. 1994: MIT Press.

9. Zhang, H. and B. Kitchenham, *Semi-quantitative Simulation Modeling of Software Engineering Process*, in *International Software Process Workshop/International Workshop on Software Process Simulation and Modeling*. 2006, Springer: Shanghai.
10. Kasse, T., *Practical Insight into CMMI*. 2004: Artech House.
11. CMMI Product Team, *Capability Maturity Model Integration (CMMI-SE/SW/IPPD, v1.1), Continuous Representation*. 2002, Software Engineering Institute, Carnegie Mellon University: Pittsburgh, USA.
12. CMMI Product Team, *Capability Maturity Model Integration (CMMI-SE/SW/IPPD, v1.1), Staged Representation*. 2002, Software Engineering Institute, Carnegie Mellon University: Pittsburgh, USA.
13. Brooks, F.P., Jr., *The Mythical Man-Month: Essays on Software Engineering*. Anniversary Edition ed. 1995: Addison-Wesley.
14. Zhang, H., et al., *Qualitative Simulation Model for Software Engineering Process*, in *17th Australian Software Engineering Conference*. 2006, IEEE: Sydney.
15. Madachy, R.J., *Software Process Dynamics*. 2005: IEEE Computer Society Press.
16. Antoniol, G., G.A. DiLucca, and M. DiPenta, *Assessing Staffing Needs for a Software maintenance Project through Queuing Simulation.* IEEE Transactions on Software Engineering, 2004. **30**(1).
17. Raffo, D. and S. Setamanit, *A Simulation Model for Global Software Development Project*, in *International Workshop on Software Process Simulation and Modeling*. 2005: St. Louis, MO.
18. Kerzner, H., *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. 9th ed. 2006: John Wiley & Sons.
19. Zhang, H., B. Kitchenham, and R. Jeffery. *Planning Software Success with Semi-quantitative Reasoning*. in *18th Australian Software Engineering Conference*. 2007. Melbourne: IEEE.

Achieving Software Project Success: A Semi-quantitative Approach

He Zhang[1,2], Barbara Kitchenham[2], and Ross Jeffery[1,2]

[1] School of Computer Science and Engineering, UNSW
[2] National ICT Australia
{he.zhang,barbara.kitchenham,ross.jeffery}@nicta.com.au

Abstract. Software process modeling and simulation hold out the promise of improving project planning and control. However, purely quantitative approaches require a very detailed understanding of the software project and process, including reliable and precise project data. Contemporary project management defines the success of project as a cube, rather than the traditional single point, which allows the management of software project semi-quantitatively with uncertainty-tolerance. This paper introduces semi-quantitative simulation into software project planning and control, and develops a practical approach to enhance the confidence of project success under uncertainty and contingency. We illustrate its value and flexibility by an example implementation with a simplified software process model.

Keywords: project planning, project control, process simulation, process modeling, semi-quantitative simulation.

1 Introduction

Most current project planning methods use a target constrained by a single point in multi-dimensional space (i.e. cost, quality, delivery date etc.) to define project success. They handle uncertainty by using statistical techniques, such as Monte Carlo simulation. However, due to the ever increasing complexity and diversity of innovative projects, especially software projects, the definition of project success needs to evolve. A contemporary success definition needs to be based on a range of values in multi-dimensional space and requires project planning and control methods capable of dealing with uncertainty and contingency.

Our previous work introduced semi-quantitative simulation as a novel technique for software process modeling [1]. In this paper, we propose a new approach to software project planning and control to fit the evolved definition of project success, and introduce semi-quantitative simulation as the core paradigm to achieve this purpose. We also provide an example to illustrate the application of this approach in practice.

Section 2 explains the motivation behind our approach, and briefly introduces the concept of semi-quantitative modeling and simulation. We address the proposed project planning and controlling approach in detail in Section 3. It is followed by an example application with a semi-quantitative software process model in Section 4.

Q. Wang, D. Pfahl, and D.M. Raffo (Eds.): ICSP 2007, LNCS 4470, pp. 332–343, 2007.

We discuss various aspects of our approach in Section 5. Finally, Section 6 presents our conclusion and proposes future work.

2 Motivation and Paradigm

2.1 Project Success Definition

Historically, the success of a software project has been defined as getting the job done within the constraints of success metrics, such as time, cost, and quality. By using this standard definition, success could be visualized as a single point on a success factor grid [2]. However few projects, especially those requiring innovation (like software projects), can achieve such an exact target.

In practice, few software projects are ever completed without tradeoffs or changes to time, cost and quality. Software projects are sometimes considered successful when the overruns are held to thirty percent or when the user only rejects a quarter of result [3]. Hence, Kerzner argues in his "bible of project management" that project success might still occur without exactly hitting a single point target [4]. In this regard, the success of contemporary project might be better defined as a cube of project success metrics, rather than a single multi-dimensional point (Fig. 1), and the project is assessed as successful if it finishes at any point inside the cube.

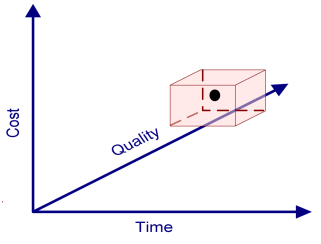


Fig. 1. Project success: point or cube?

Traditional quantitative techniques usually support single point predictions of project outcomes, rather than prediction of a range of possible outcomes. However, semi-quantitative simulation has the inherent capability of reasoning with multiple value ranges, and can facilitate decision-making based on project success factors represented as the acceptable ranges in multiple dimensions. This paper explains how semi-quantitative models can support flexible project planning and control.

2.2 Software Project Planning and Control

Comprehensive planning and control are two of the most important aspects of any project. In many case, development team simply failed to fulfill the original project goal. It is rather the fault of inflated and unreasonable expectations, and poor control.

Research into the success of IS projects has identified project planning and control as the second and third most important factors affecting project success [5]. It is also

estimated that the planning process of project management should require approximately 35% of the project manager's effort over the life of the project [6]. Project planning and control require the project manager to think and perform through the project, and remain focused on the final goal, i.e. project success, to be delivered.

In general terms, project planning is "to define and mature the project scope, develop management plan, and schedule the activities and resources"; while project control can be defined as "to compare actual performance with planned performance, analyze variances, assess trends, and evaluate possible alternatives" [7]. Our approach focuses on the quantitative management aspect of project planning and control.

2.3 Software Process Simulation Modeling

In the late 80's, Abdel-Hamid and Madnick (AHM) proposed the use of quantitative System Dynamics models to simulate the dynamic aspects of software projects [8]. Since then, other researchers have continued and extended their approach. Systems Dynamics models are intended to provide a better understanding of project behavior improving both project planning and project control.

Kellner *et al.* presented a wide variety of reasons for undertaking simulations of software process models [9]. Primarily, process simulation is an aid to decision making. They identified six categories of purposes: Strategic management, Planning, Control and operational management, Process improvement and technology adoption, Understanding, and Training and learning.

Unlike regression and mathematical models (e.g. COCOMO and SLIM), simulation models provide a systematic view of software process, and predict development performance with dynamic insights into the interdependencies among the elements of the process. Successful cases can be found in research papers published in ProSim.[1]

2.4 Semi-quantitative Simulation

Before explaining our approach, we briefly introduce semi-quantitative simulation, which is at the core of our approach. Semi-quantitative simulation is implemented in two stages: qualitative reasoning and quantitative constraint propagation. The qualitative model, which is implemented in QSIM [10], reflects system in the real world at an abstract level. Fewer assumptions are required than for quantitative model.

In conventional quantitative models (e.g. Systems dynamics), a system is represented as a set of ordinary differential equations (ODEs), which involves quantitative information. At a higher level, a qualitative differential equation (QDE) represents a large set of possible ODEs, e.g. each $M+$ function represents the set of all monotonically increasing functions. When only incomplete knowledge is available, we can replace ODEs with QDEs to represent the relationships and values of the variables qualitatively [11]. One or more QDEs are the input constraint model(s) to QSIM.

Qualitative reasoning starts from a given initial system state. The output generated by QSIM is a set of possible qualitative behaviors and each behavior consists of a sequence of states. Each state in a behavior describes an open temporal interval or a time point. These qualitative states present the system behavior from its initial state to its final state graphically.

[1] International Workshop on Software Process Simulation Modeling.

Semi-quantitative simulation focuses on the use of bounding intervals to represent partial quantitative knowledge. Q2 (Qualitative+Quantitative) is a basic semi-quantitative reasoner implemented as an extension to QSIM. Given interval bounds of landmarks and envelopes functions, its QDE defines a constraint-satisfaction problem (CSP). A solution to CSP is an assignment of an interval to each landmark consistent with the constraints. All possible qualitative behaviors are assigned to Q2, and then restricted to the behaviors that are consistent with the quantitative constraints.

3 Managing Software Project Semi-quantitatively

3.1 Project Planning

The approach proposed here includes an iterative refinement method for software project planning. However, the desired results may converge rapidly between the adjoining iterations by using semi-quantitative simulation, if a realistic solution existing. As illustrated in Fig. 2, this approach consists of five distinct steps: (1) defining project success criteria; (2) tailoring and updating process model; (3) creating project element-impact table; (4) simulating and fine tuning; (5) updating project plan.

Step 1: defining project success criteria. Unlike the following steps, the first step of our approach emphasizes the business aspect instead of the technical aspect of project planning. A variety of stakeholders may be involved in specifying the project success criteria, i.e. defining the success cube. The output of this step is the project success factor list, in other words, the metrics that are required to define the success for this project, their relative importance, plus the value ranges accepted, which are further visualized as project success cube.

Step 2: tailoring and updating project model. According to the nature of planned project and the organizational context, a prototype process model is selected from the literature or organization's model repository. This topic is too complicated to be included in this paper. The prototype model has to be tailored to fit the project's characteristics, and to be updated with its specific information.

Step 3: creating element-impact table. Not all elements can be changed in a process model. Only tunable elements are identified in this step, and any value change to these elements may induce changes in the project plan. These elements are further prioritized in order of their importance and contribution to the success factors in element-impact table, which also include their qualitative impact on success factors when they are changing. Table 2 gives an example of the element-impact table.

Step 4: simulating and fine tuning. All outputs from the above three steps are used as inputs to the semi-quantitative simulation (see below).

Step 5: updating project plan. When the simulation produces an acceptable prediction of the project outcomes, the project plan will be updated according to this result. In contrast, if there is a large deviation from the success criteria (such as "Impossible" state in Fig. 3), the management should consider canceling the project.

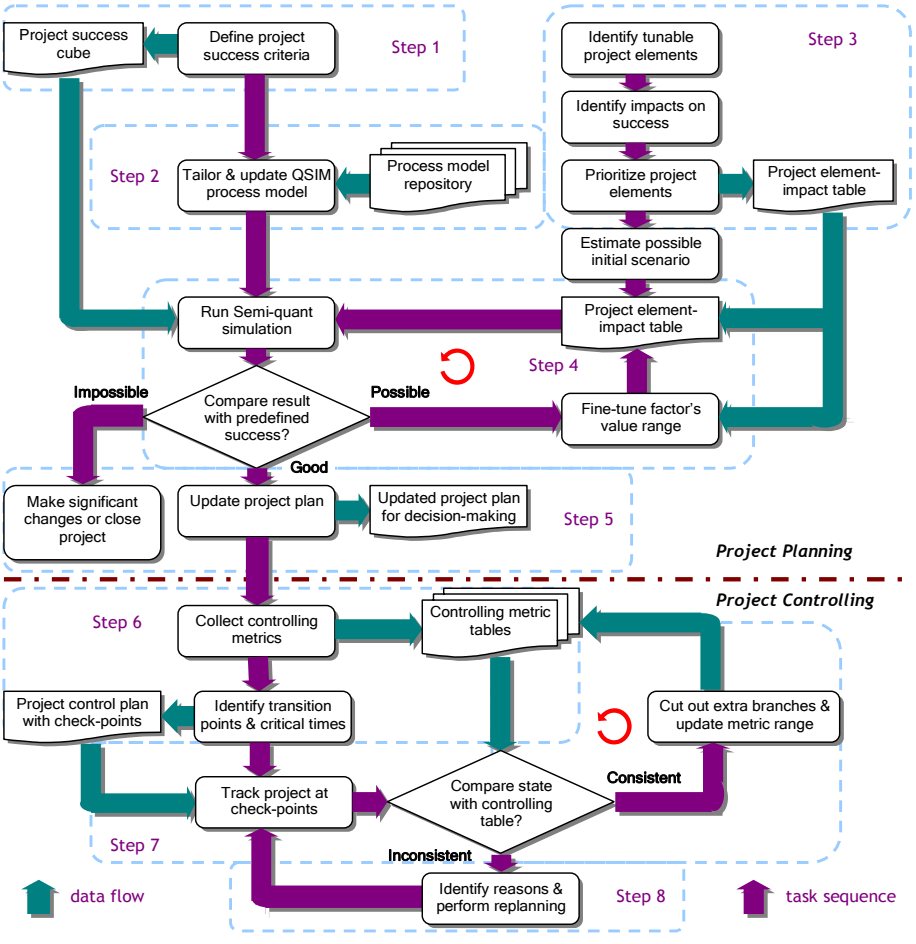


Fig. 2. Software project planning and controlling with semi-quantitative simulation

Simulation Iteration. The iteration procedure with semi-quantitative simulation can be regarded as a planning optimizing phase to find a fitted solution (the project plan) progressively for the predefined project success criteria. The generated results converge rapidly if the success criteria are realistic.

The process model selected from Step 2 is coded with the specific element values (ranges) and initial state of the project, and then is executed by QSIM, which generates all possible behaviors and predicts the project completion state, which is compared with the success criteria (from Step 1). Either "Impossible" or "Good" results cause an exit from the iteration procedure. Otherwise, if result is "Possible", the values of tunable elements need to be refined in terms of the element-impact table created in Step 3, and the next iteration is triggered with the updates.

Refinement Strategy. We present five types of project completion state (Fig. 3) contrasting with the predefined project success criteria. They are used as guidance to

indicate if the simulation iteration needs to continue with further refinement or stop. The strategies for other states, e.g. "Right-bottom", can be deduced similarly.

The "Included" state indicates the predicted project completion falls into the success cube. As the project success defined in Section 2, we can easily identify that it is a "Good" plan for project success.

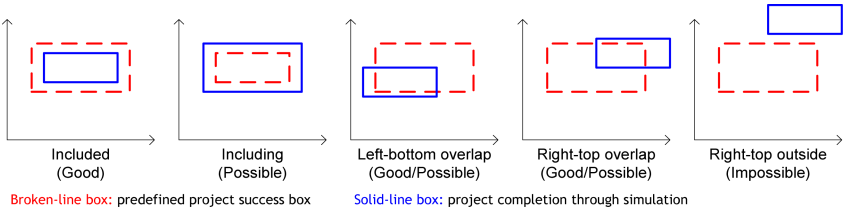


Fig. 3. Comparing simulation result vs. predefined success criteria

The counter part of "Included" state is "Including" state, which covers the success area with extra space. This state means the "Possible" success, i.e. the project can finish in success cube or outside. It needs to shrink with refinement.

Another state is that the project completion area locates at the "left-bottom" of the success cube, but with overlap. It may be translated to "Good" for some metrics, such as cost and schedule. But for some others, e.g. earned value and scope (functionality), the overlap only implies the "Possible" success, and then the iteration procedure has to continue. Similar discussion applies to the "right-top overlap" state. When the project completion area locates "outside" the success cube, i.e. no overlap existing, the project will be mostly evaluated as "Impossible". If no significant changes are available, the project is recommended to be canceled. The refinement strategy can be further extended and applied to multi-dimension or hyper-cube of success criteria.

3.2 Project Control

You control a project to the extent that you manage to ensure the minimum of surprises along the way. The best-controlled project is the one that best lives up to its prediction [3]. The semi-quantitative controlling approach can provide a flexible way to track and control project progress, and help the project manager observe whether the project is under control. The project control approach contains three major steps: (6) creating control metric tables; (7) tracking project at check-points; (8) identifying problems and replanning (Fig. 2).

Step 6: creating control metric tables. Semi-quantitative simulation generates all possible behaviors (behavior tree) for each scenario. The final project state cube is calculated as the union of the value ranges predicted by the branches. The behavior tree serves as the road map for the project. The distinction from the traditional methods is that it depicts the alternative routes. Fig. 5 is a simple behavior tree with five branches. The transition points are indicated as landmarks "Φ" in behavior tree.

Once reaching a "Good" solution for project planning, the control metric table should be created for each measurable variable based on its predictions of all behaviors. Table 3 is an example control table. The transition points and critical time points

need to be identified from the behavior tree, and added to project control plan as the check-points.

Step 7: tracking project at check-points. According the control plan, the performance indicators are measured and tracked at the check-points. The project can shift between the branches, and its progress state can be identified with the corresponding value ranges in control tables. If the progress is consistent with the estimated value ranges, it indicates the project under control, then the extra branches (inconsistent with actual project state) should be cut out, and the control table is updated with refined value ranges. Correspondingly, the project final state is refined with the remaining branches.

Step 8: identifying problems and replanning. When inconsistency is found against any branch at check-points, it alerts that the project might be out of control. Problems have to be identified and corrected, and replanning needs to be performed.

4 Illustrative Example

In this section, we present a simple application of the semi-quantitative planning and control approach, and show how the project management benefits from this novel approach. To avoid the excessive detail, we employ a simplified software process model focusing on the staffing process, which is described in [12], and apply the project success constraints in only two dimensions for demonstration.

4.1 Prototype Project

We select AHM's EXAMPLE project as a prototype project for demonstration. EXAMPLE is a middle-size project with 64 KDSI, and used COCOMO to calculate the workforce level [8]. The main attributes of EXAMPLE project are summarized in Table 1. As the originally planned, the project can be delivered on day 430.

Considering any contingency issues, such as leave or sickness, the initial project team size is defined with [4 5] developers. The progress of EXAMPLE project proceeds as planned until a request for change (RFC) by marketing department on day 240. They report that a competitor plans to release a similar software product in the near future, and argue that their own product must be released two months earlier than original schedule to remain competitive. After one-week's analysis and discussion across the organization, the management approves the RFC with the condition that the new total expenditure must be no more than 3000 man-days. The project manager is responsible for making the corresponding changes to the project plan.

4.2 Planning Procedure

This is a typical project replanning scenario, the project manager tries to find a "Good' solution using the semi-quantitative planning approach.

Step 1. The project manager updates the project plan on day 245 (one week after RFC). The changed project has to be completed two months (40 working days) earlier than the original schedule, in other words, the current project closure targets at day 390. The original estimated budget of the project is 2150 man-days. The project

success criteria are updated correspondingly. In *Step 2*, the experimental process model [12] is chosen for the simulation.

Step 3. By examining the elements of the experimental model, we identify four tunable elements: new workforce, productivity ratio, assimilation delay, and remaining project size (Table 2). The value of *"new workforce"* indicates how many new developers are introduced into the project. The available human resource is up to 12 developers for this project. The value constraints are further explained in [1].

It is noticeable that the first three elements are related to introducing more developers into the project. The fourth element, i.e. reducing remaining project size (functionality), is not desired for the clients, so it is ranked at the bottom. Considering the time for recruiting, the new staff can join the project team in three weeks, i.e. recruitment delay for 15 days. Correspondingly, the project completion is refined as [260 390]. Because the values of the second and third elements are highly dependent on the quality of the new staff, it is hard to refine the value ranges before the assimilation. Therefore, the project manager plans to start the simulation by adding extra workforce into the project without altering the uncertainty on the last two elements.

Table 1. Attributes of EXAMPLE project

Attributes	Values
project size	64KDSI
duration	430days
initial team size	[4 5]staff
maximum team size	16staff
average productivity	36DSI/man-day

Table 2. Project element table

Element	time	cost	Constraint
new workforce	[+/-]	[+]	[0 12] staff
productivity ratio	[-]	[-]	[0.4 0.6]
assimilation delay	[+]	[+]	[60 80] days
remaining size	[+]	[+]	

Iteration 1. We initially introduce [3 4] developers into the project to initiate the simulation process. The project can finish on day [339 423], and the completion cube is depicted in Fig. 4. Comparing the original completion time and project success cube, this decision slightly improves the product release schedule (but only guaranteed by 7 days), and improves the cost performance. However, the delivery date is still much behind the expected release date (day 390).

Iteration 2. One positive finding through Iteration 1 is that adding extra workforce may shorten the project duration. To amplify this positive effect, we introduce [11 12] developers. It generates 3 possible behaviors this time, which predict the project may finish on day [309 386], a bit earlier than requested release date. However, the completion cube indicates that the project cost may increase significantly and reach much higher than acceptable budget (Fig. 4). Although the financial performance looks terrible, it further verifies the positive contribution of extra workforce to schedule.

Iteration 3. With respect to the impacts identified in Table 2, we need to add fewer developers in this iteration to reduce the possible high expenditure caused in Iteration 2. We choose a modest number of developers, say [7 8] developers, for this simulation. Five possible behaviors are generated (Fig. 5), and indicate the project may finish at day [320 396], before or after the new members of staff are fully assimilated.

Both schedule and cost are slightly over the requested success cube. This means that the solution corresponds to the "right-top overlap" state in Fig. 3. Analyzing the impacts of increasing the workforce across iterations, we find that reducing the

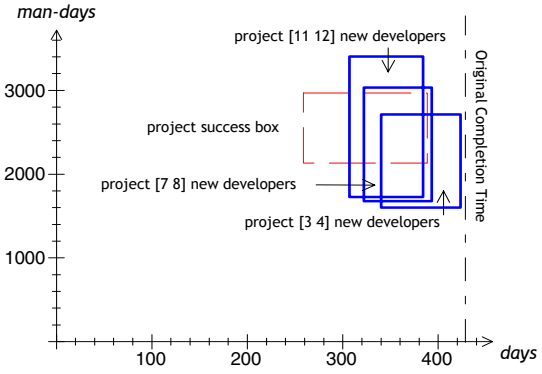


Fig. 4. Project completion cube through simulation

number of extra staff will incur a further delay of the project; and conversely, introducing more developers will result in higher cost, comparing with Iteration 3.

Step 5. After negotiating with the senior management and marketing department, they reach the agreement to update the project plan with new time frame of [320 396] days and budget of [1700 3068] man-days. Meanwhile, the management gives up the last option (in Table 2) of sacrificing software functionality or quality. Given this update of project success criteria, the project manager plans to recruit [7 8] developers into the team with confidence that the projects will be completed successfully.

4.3 Controlling Procedure

Fig. 5 is the project behavior tree generated for this simplified case by Iteration 3. It depicts five possible behaviors: three of them have one transition (*t1* when new workforce is introduced) during simulation, Behavior 5 ends exactly at the second transition point (assimilation ends at), only Behavior 3 passes two transitions (*t1*, *t2*). The behaviors are distinguished at the variables' trends (e.g. value going up or down).

Step 6. Based on the behavior tree, we develop the control metric table for each measurable variable to track its changes at check-points. The most important check-point is transition point *t2* that indicates the end of assimilation. Table 3 is an example control metric table for R_{SD} (software development rate) and S_C (completed size).

Step 7. When the project progresses to *t2*, the project state is compared to the controlling tables. Because only Behavior 3 goes through the second transition point, if we are aware of the end of assimilation and S_C falls into the range of [44 64] KDSI, we can predict the schedule might reach 396. Correspondingly, the behavior branch 1, 2, 4, and 5 can be cut out. On the other hand, if the project closes during the assimilation, it may happen in the time period [323 340].

Step 8. One unexpected situation might be that the assimilation finishes, but the project progresses to the outside of the range [44 64] KDSI. It means the project is out of control. The project manager has to identify and correct the assignable problems immediately. Replanning should be carried out to update the project end state.

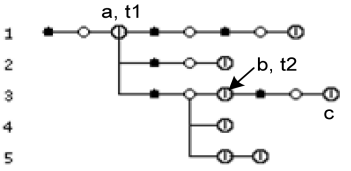


Fig. 5. Behavior tree of Iteration 3

Table 3. Control metric table for R_{SD}/S_C

#	R_{SD}@*t1*	R_{SD}@*t2* (S_C)	duration
1	[123 179]	[203 387] (-)	[325 340]
2	[141 179]	[203 387] (-)	[325 340]
3	[141 179]	[277 403] ([44 64])	[320 396]
4	[141 179]	[201 403] (-)	[323 340]
5	[141 179]	[277 403] (-)	[323 340]

5 Discussion

5.1 Approach Capability

Multi-dimension Reasoning. In the EXAMPLE project (Section 5), we employ two dimensions (project schedule and cost) to define the project success cube. The simplified scenario helps the understanding of our approach. However, in practice, management may consider more factors simultaneously, and needs to determine the tradeoffs between all of them for decision making. Semi-quantitative simulation provides this capability of reasoning alternative process behaviors in many dimensions.

Success Factors. In real software project, many success factors can be defined at the planning stage, and can be reasoned by using the multi-dimension capability of semi-quantitative simulation. We choose a simplified 2-D view of project success in example. However, normally, a variety of stakeholders are involved in the project planning process. Most of them, including the project manager, development team, project clients, and senior management, may possess very different perspectives on the expected project performance. For instance, the marketing department hopes to release a new software product earlier than its competitors. On the other hand, the development team estimates the required project duration based on their own experience. Therefore, both groups have to compromise with each other on the value ranges in dimension(s) of project success cube, accepting that delivery is impossible for developers before the lower value, and release is useless for clients after the higher value.

"Good" Solutions. Our approach is to find one "Good" solution that guarantees the project falls into the success cube. However, this solution is not a unique one fitting the predefined project success, but one of the possible "Good" solutions. Different iterations may produce slightly different solutions. Project managers have to identify the tradeoff (solution) among the success factors required by clients or executives, allowing for the resource available to the project. With regard to the definition of project success, there is no difference (better or worse) among the all possible "Good" solutions. Thus, the solution obtained through this approach can assist in the planning and control process by offering both flexibility and contingency tolerance.

5.2 Alternative Approaches

There are two broad categories of methods dealing with uncertainty: probabilistic and non-probabilistic. Semi-quantitative simulation falls into the second one.

Statistical Probability. With reference to the definition of project success, a software project will be evaluated as successful as long as it finishes at any point inside the success cube. One purpose of our approach is to guarantee that the project plan leads to the project success by allowing the definitions of success to include an element of uncertainty. Using semi-quantitative simulation, we can allow for the existence of many uncertain elements without needing to know their statistical distribution.

Monte Carlo Method and Sensitivity Analysis. Monte Carlo simulation and sensitivity analysis are both popular methods for project planning. Although they take many samples of the value range, unfortunately, they are still a finite set. Thus, they cannot guarantee the all possibilities fall into their solution. This problem turns to be more serious when more factors change simultaneously, which may result in missing some important behaviors. Meanwhile, the cost of using them increases dramatically with the combination of multiple dimensions in the possible variable space. However, the cost of semi-quantitative simulation does not depend on the size of the variable space. It is a function of the number of distinct qualitative behaviors predicted [11].

Another condition required for using Monte Carlo simulation is that we need to know the value distribution of variable on the range. If such information is unavailable, some distribution has to be assumed anyway. In contrast, semi-quantitative simulation can work without such assumption.

Fuzzy Logic. As another non-probabilistic method, Fuzzy logic describes the real world system with fuzzy set, which is a fuzzy subset of the universe of discourse. It applies a rough boundary to handle the uncertainty, and the mapping to fuzzy set is in an arbitrary way, linear or nonlinear. In contrast, semi-quantitative modeling describes the system boundary with real numeric values, which maintains the precision while coping with uncertainty. Each approach possesses its advantages and limitations. The selection between them depends on user's capability and requirements.

Semi-quantitative simulation performs reasoning by refinement: define a set of possible solution, and shrink it by cutting out the illogical behaviors. This approach guarantees integrity of the solution. In addition, semi-quantitative simulation produces not only the final states of project, but all possible process behaviors (routes) with the constraints of value ranges, which can be used for ongoing project control. This capability is unique to semi-quantitative modeling.

6 Conclusion

This paper has proposed a novel approach to project planning and control using semi-quantitative simulation, which matches a contemporary definition of project success. We also demonstrate how this approach works with a simplified example. The unique features and advantages of our approach are also discussed.

Semi-quantitative simulation is presented in this paper as a powerful technique for planning and controlling software project with uncertainty. Moreover, it offers a project manager the flexibility and confidence to cope with uncertainty and contingency during the software development, and guarantee the integrity of final project states. By contrast, the traditional approaches are only one-point sample of the set of solutions in the success cube. The future research on this topic will consider:

- Developing a planning and control tool with integration of steps, and visualizing project states through simulation;
- Automatically implementing iterative procedure and generating control table.

References

1. Zhang, H. and B. Kitchenham, *Semi-Quantitative Simulation Modeling of Software Engineering Process*, in *Software Process Workshop/International Workshop on Software Process Simulation and Modeling*. 2006, Springer: Shanghai.
2. Kerzner, H., *Using the Project Management Maturity Model: Strategic Planning for Project Management* 2nd ed. 2005: John Wiley & Sons.
3. DeMarco, T., *Controlling Software Projects: Management, Measurement & Estimation*. 1982, New York: Yourdon Press.
4. Kerzner, H., *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. 9th ed. 2006: John Wiley & Sons.
5. Rehessar, H., *Project Management Success Factors*. 1996, University of New South Wales.
6. Clark, T.A., *Project Management for Planners: A Practical Guide*. 2002: Planners Press, American Planning Association.
7. *A Guide to the Project Management Body of Knowledge*. 2004, Project Management Institute.
8. Abdel-Hamid, T.K. and S.E. Madnick, *Software Project Dynamics: An Integrated Approach*. 1991, Englewood Cliffs, N.J.: Prentice Hall.
9. Kellner, M.I., R.J. Madachy, and D.M. Raffo, *Software Process Simulation Modeling: Why? What? How?* Journal of Systems and Software, 1999. **46**(2/3).
10. *QSIM*, UT Qualitative Reasoning Software. http://www.cs.utexas.edu/users/qr/QR-software.html.
11. Kuipers, B., *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. 1994: MIT Press.
12. Zhang, H., et al., *Qualitative Simulation Model for Software Engineering Process*, in *Australian Software Engineering Conference*. 2006: Sydney.

Author Index