Danilo Caivano
Markku Oivo
Maria Teresa Baldassarre
Giuseppe Visaggio (Eds.)

Product-Focused Software Process Improvement

LNCS 6759

12th International Conference, PROFES 2011
Torre Canne, Italy, June 2011
Proceedings

Springer

Lecture Notes in Computer Science 6759

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

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Alfred Kobsa
University of California, Irvine, CA, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
TU Dortmund University, Germany

Madhu Sudan
Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

Danilo Caivano
Markku Oivo
Maria Teresa Baldassarre
Giuseppe Visaggio (Eds.)

Product-Focused Software Process Improvement

12th International Conference, PROFES 2011
Torre Canne, Italy, June 20-22, 2011
Proceedings

Springer

Volume Editors

Danilo Caivano
University of Bari, Department of Informatics
Via E. Orabona 4, 70126 Bari, Italy
E-mail: caivano@di.uniba.it

Markku Oivo
University of Oulu, Department of Information Processing Science
P.O. Box 3000, 90014 Oulu, Finland
E-mail: markku.oivo@oulu.fi

Maria Teresa Baldassarre
University of Bari, Department of Informatics
Via E. Orabona 4, 70126 Bari, Italy
E-mail: baldassarre@di.uniba.it

Giuseppe Visaggio
University of Bari, Department of Informatics
Via E. Orabona 4, 70126 Bari, Italy
E-mail: visaggio@di.uniba.it

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-21842-2 e-ISBN 978-3-642-21843-9
DOI 10.1007/978-3-642-21843-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011929368

CR Subject Classification (1998): D.2, K.6, J.1, H.3-4, C.2.4, J.3

LNCS Sublibrary: SL 2 – Programming and Software Engineering

© Springer-Verlag Berlin Heidelberg 2011
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

On behalf of the PROFES Organizing Committee we are proud to present the proceedings of the 12th International Conference on Product-Focused Software Process Improvement (PROFES 2011), held in Torre Canne, Italy. Since 1999 PROFES has grown in the software engineering community and has become a premium conference that brings together both academia and industry.

The roots of PROFES lie in the professional software process improvement motivated by product, process and service quality needs. The conference retains its high quality and focus on the most relevant research issues by addressing both perspectives, research and practice, from an academic and industrial point of view.

Today's software products and services are perceived as strategic assets for empowering business sectors at every level of the value chain, from strategic to operative. In this scenario, and considering the current global economic downturn, the challenge for developing software products and services consists in managing process diversity in order to reuse strategic software assets in various fields and environments quickly and cost effectively. This was the special theme for PROFES 2011.

In the last few years, many approaches and techniques have been proposed for managing diversity: experience bases for collecting and sharing knowledge and experiences; software development processes able to rearrange common assets in diverse products; process patterns as an instrument for filling the gap between process definition and the amount of customizations needed; estimation and calibration techniques that deal with the different processes in use; parametric and goal-oriented quality models; project management techniques able to fulfil the project goals in spite of project characteristics; cloud computing and service orientation for managing the diversity of hardware and software platforms. All these innovations provide exciting opportunities to make significant progress in understanding and facing real-world challenges.

This year's technical program featured invited talks, research papers, and experience reports on the most relevant topics in the focus area. We received 54 papers submitted from 22 nations, with each paper receiving at least three reviews. After a thorough evaluation, the Program Committee selected 24 technical full papers. The topics addressed in these papers indicate that the PROFES theme is a vibrant research area, but is also of high interest for industry as demonstrated by several papers that report on case studies or experience gained in industry.

We were proud to have two top keynote speakers: (1) Dennis Smith – Senior Member of the Technical Staff at Carnegie Mellon University's Software Engineering Institute and (2) David J. Kasik – Senior Technical Fellow, Visualization and Interactive Techniques, The Boeing Company. It also featured a

short-paper session, two workshops, Managing the Client Value Creation Process in Agile Projects (VALOIR) and Project and Knowledge Management Trends (PKMT), one tutorial, Establishing and Improving Project Management Using Assessment Models for Process Capability and Organizational Maturity, and a Doctoral Symposium.

We wish to thank the University of Bari, the University of Oulu, the competence center Driving Advances of ICT in South Italy-Net (DAISY-NET), the Project Management Institute Southern Italy Chapter (PMI-SIC) and Software Engineering Research and Practices s.r.l. (SER&Practices) – spin-off of the University of Bari – for supporting the conference. We are also grateful to the authors for their high-quality papers and the Program Committee for their hard work in reviewing the papers.

June 2011

Danilo Caivano
Markku Oivo
Maria Teresa Baldassarre
Giuseppe Visaggio

Organization

PROFES 2011 was organized by the University of Bari and University of Oulu.

General Chair

Giuseppe Visaggio	University of Bari and Daisy-Net, Italy

Program Co-chairs

Danilo Caivano	University of Bari and SER&Practices, Italy
Markku Oivo	University of Oulu, Finland

Organizing Chair

Maria Teresa Baldassarre	University of Bari and SER&Practices, Italy

Publicity Chair

Matias Vierimaa	VTT, Oulu, Finland

Publicity Co-chair Asia

Katsuro Inoue	Osaka University, Japan

Publicity Co-chair Australia

Emam Hossain	NICTA, Australia

Publicity Co-chair Europe

Luigi Buglione	ETS / Engineering Group, Italy

Publicity Co-chair South America

Guilherme H. Travassos	COPPE/UFRJ Federal University of Rio de Janeiro, Brazil

Short Papers and Poster Chairs

Guilherme H. Travassos	COPPE/UFRJ Federal University of Rio de Janeiro, Brazil
Per Runeson	Lund University, Sweden

Doctoral Symposium Chairs

Marcela Genero	University of Castilla la Mancha, Spain
Emilia Mendes	University of Auckland, New Zealand

Workshops and Tutorials Chairs

Alberto Sillitti	Free University of Bolzano, Italy
Felix Garcia	University of Castilla la Mancha, Spain

Web Chair

Giovanni Bruno	University of Bari, Italy

Program Committee

Zeiad Abdelnabi	Garyounis University - IT College, Libya
Silvia Abrahão	Universidad Politécnica de Valencia, Spain
Muhammad Ali Babar	ITU of Copenhagen, Denmark
Pasquale Ardimento	University of Bari, Italy
Maria Teresa Baldassarre	University of Bari, Italy
Stefan Biffl	Technical University of Vienna, Austria
Andreas Birk	SWPM - Software.Process.Management, Germany
Luigi Buglione	ETS / Engineering Group, Italy
Danilo Caivano	University of Bari, Italy
Gerardo Canfora	University of Sannio, Italy
Jeffrey Carver	Alabama University, USA
Marcus Ciolkowski	Fraunhofer Institute for Experimental Software Engineering, Germany
Dave Cliff	University of Bristol, UK
Reidar Conradi	Norwegian University of Science and Technology, Norway
Beniamino Di Martino	Second University of Naples, Italy
Torgeir Dingsøyr	SINTEF, Norway
Tore Dybå	SINTEF, Norway
Davide Falessi	University of Rome "Tor Vergata", Italy and Simula, Norway
Raimund Feldmann	Fraunhofer Center Maryland, USA
Rudolf Ferenc	University of Szeged (SZTE), Hungary
Alfredo Garro	University of Calabria, Italy
Paul Gruenbacher	Johannes Kepler University Linz, Austria
Jens Heidrich	Fraunhofer Institute for Experimental Software Engineering, Germany
Frank Houdek	Daimler AG, Germany
Martin Höst	Lund University, Sweden

Hajimu Iida	NAIST, Japan
Michel Jaring	Universtiy of Helsinki, Finland
Natalia Juristo	Universidad Politécnica de Madrid, Spain
Janne Järvinen	F-Secure, Finland
Yiannis Kanellopoulos	Software Improvement Group, Amsterdam, The Netherlands
Petri Kettunen	University of Helsinki, Finland
Pasi Kuvaja	University of Oulu Finland
Lech Madeyski	Wroclaw University of Technology, Poland
Kenichi Matsumoto	Nara Institute of Science and Technology, Japan
Makoto Matsushita	Osaka University, Japan
Maurizio Morisio	Politecnico di Torino, Italy
Mark Mueller	Robert Bosch GmbH, Germany
Juergen Muench	Fraunhofer IESE, Germany
Haruka Nakao	Japan Manned Space Systems Corporation, Japan
Risto Nevalainen	FiSMA ry, Finland
Mahmood Niazi	Keele University, UK
Linda Northrop	Software Engineering Institute, Carnegie Mellon University, USA
Markku Oivo	University of Oulu, Finland
Paolo Panaroni	INTECS, Italy
Dietmar Pfahl	Lund University, Sweden
Minna Pikkarainen	VTT, Finland
Teade Punter	Embedded Systems Institute (ESI), Netherlands
Austen Rainer	University of Hertfordshire, UK
Daniel Rodriguez	University of Alcalá, Spain
Barbara Russo	Free University of Bolzano-Bozen, Italy
Outi Salo	Nokia, Finland
Klaus Schmid	University of Hildesheim, Germany
Kurt Schneider	Leibniz Universität Hannover, Germany
Michael Stupperich	Daimler AG, Germany
Christos Tjortjis	Univ. of Ioannina, Greece and University of W. Macedonia, Greece
Guilherme Travassos	COPPE/UFRJ, Brazil
Markku Tukiainen	University of Joensuu, Finland
Mark Van Den Brand	Eindhoven University of Technology, The Netherlands
Rini Van Solingen	Delft University of Technology, Netherlands
Sira Vegas	Universidad Politecnica de Madrid, Spain
Aaron Visaggio	University of Sannio, Italy
Hironori Washizaki	National Institute of Informatics, Japan

Additional Reviewers

A

AcuÃa, Silvia T.

D

Dieste, Oscar

F

Fernandez, Adrian
Fgri, Tor Erlend
Ficco, Massimo
Fushida, Kyohei

G

González-Huerta, Javier

H

Haugset, Brge
Hegedűs, Péter

K

Kakuja-Tóth, Gabriella
Klabbers, Martijn

M

Massollar Da Silva, Jobson Luiz

N

Nicolau De França, Breno Bernard

O

Ohkura, Kimiharu

S

Scheinholtz, Lauri Ann

Y

Yoshida, Norihiro

Table of Contents

Keynote Addresses

Agile and Lean Practices

Cross-Model Quality Improvement

Global and Competitive Software Development

Managing Diversity

Product and Process Measurements

Product-Focused Software Process Improvement

Requirement Process Improvement

Software Process Improvement

The Impact of Emerging Software Paradigms on Software Quality and User Expectations

Dennis B. Smith

Carnegie Mellon University/ Software Engineering Institute
United States
dbs@sei.cmu.edu

This talk will discuss emerging approaches to software development and evolution that enable organizations to respond quickly to new business needs while maintaining their legacy applications. These approaches include:

- *Service oriented architecture (SOA)* which is a way of designing, developing, deploying, and managing systems where coarse-grained services represent reusable functionality, and service consumers compose applications or systems using the functionality provided by these services through standard interfaces. This approach enables the flexible composition and evolution of new services. Major barriers include lack of a long term strategy, lack of effective governance, unrealistic expectations, and inappropriate technical strategy.
- *Cloud computing* which is a "a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet [1]. This approach enables consumer organizations to have access to state of the practice hardware and software without making many of the upfront infrastructure investments. The main drivers for cloud computing adoption include scalability, elasticity, virtualization, cost, mobility, collaboration, and risk reduction. Major barriers include security, interoperability, control and performance.
- *Enterprise architecture planning and development* which develops a comprehensive plan for using business functionality across an enterprise and building applications from shared resources. Because it takes a perspective that crosses an entire enterprise, it enables the breaking down of barriers that take place within individual organizations. Major barriers include lack of long term commitment, and a focus on completeness rather than practicality.

The combination of these three approaches offers potentials for both success and failure. They can enable rapid responses to new business situations through the shared use of common resources, as well as the discipline to use a common plan for implementing enterprise wide priorities. However, the use of these approaches has often been over-hyped, resulting in failure. This talk uses lessons learned from current adoption efforts to identify the core concepts of these approaches, what their potentials are, as well as common misconceptions and risks.

Reference

1. Foster, I., Zhau, Y., Ioan, R., Lu, S.: Cloud Computing and Grid Computing: 360-Degree Compared. In: Grid Computing Environments Workshop (2008)

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, p. 1, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Acquiring Information from Diverse Industrial Processes Using Visual Analytics

David J. Kasik

Senior Technical Fellow, Visualization and Interactive Techniques
The Boeing Company, United States
david.j.kasik@boeing.com

Visual analytics is an emerging technology that provides another technique to derive better information from data. As a technology, visual analytics is a new approach that complements more traditional methods in business intelligence, information visualization, and data mining. The basic concept is to provide highly interactive visual techniques that let people explore multiple heterogeneous data sets simultaneously. The key aspect is interactivity, which lets people follow paths to seek answers to questions normally unasked. The visual aspect allows people to detect the expected and discover the unexpected.

The intelligence community in the United States and Europe began investing in and using visual analytics after the events of September 11, 2001. Boeing is a leader in exploring the application of visual analytics in industry. One of the primary attractions for visual analytics is the technology's applicability to data gathered from diverse processes. This talk will provide a glimpse of the highly complex business in a global aerospace company, an overview of visual analytics, Boeing's overall approach to visual analytics, and specific cases studies from multiple process domains, including bird strikes, industrial safety, and software system interrelationships.

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, p. 2, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Monitoring Bottlenecks in Agile and Lean Software Development Projects – A Method and Its Industrial Use

Miroslaw Staron[1] and Wilhelm Meding[2]

[1] Department of Computer Science and Engineering
University of Gothenburg
miroslaw.staron@ituniv.se
[2] Ericsson SW Research
Ericsson AB
wilhelm.meding@ericsson.com

Abstract. In the face of growing competition software projects have to deliver software products faster and with better quality – thus leaving little room for unnecessary activities or non-optimal capacity. To achieve the desired high speed of the projects and the optimal capacity,bottlenecks existing in the projects have to be monitored and effectively removed. The objective of this research is to show experiences from a mature software development organization working according to Lean and Agile software development principles. By conducting a formal case study at Ericsson we were able to elicit and automate measures required to monitor bottlenecks in software development workflow, evaluated in one of the projects. The project developed software for one of the telecom products and consisted of over 80 developers. The results of the case study include a measurement system with a number of measures/indicators which can indicate existence of bottlenecks in the flow of work in the project and a number of good practices helping other organizations to start monitoring bottlenecks in an effective way – in particular what to focus on when designing such a measurement system.

1 Introduction

Global software supply chain and competition contributes to adopting the principles of Lean in many software development organizations[1-3]. The organizations adopt the principles with hopes on achieving high development speed, reduction of unnecessary work (waste) and better alignment with customer needs (generate customer value). Such organizations often realize that the flows of software development are unlike flows of work in manufacturing and that the practice shows existence of bottlenecks unlike the ones in manufacturing. Complexity of large software development projects and the fact that each software development project is different from the previous ones cause the methods proven manufacturing to fail in Lean software development. In particular it is harder to identify, monitor and, above all, remove bottlenecks which hinder the efficiency of the software development projects as software is developed and not manufactured. In this paper we explore methods used at one of mature adopters of Lean software development– Ericsson AB– and observe

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 3–16, 2011.
© Springer-Verlag Berlin Heidelberg 2011

how they interpret and apply concepts related to bottlenecks. By conducting a formal case study we were able to observe and study the methods of monitoring bottlenecks at one of the units which develops large and complex telecommunication network products. The goal of the research project underlying our study was to identify and monitor bottlenecks that hinder efficient software development according to Lean principles. In particular the project addressed the following research question:

How can we identify and monitor bottlenecks in software development projects in order to prevent inefficiency?

Despite this problem being hard to tackle in general, the organization managed to simplify it by applying certain core concepts of Value Stream Mapping and Lean production systems: throughput and queue. Throughputwas defined as the number of units that are processes by a given phase or activity per unit of time, and queue as the number of units remaining to be developed/processed by a given phase or activity. In order to address this research question a set of measurement tools was combined into a dedicated measurement system which monitored the throughput and queue in a number of phases in the studied software project.

In short, our results show that (i) monitoring of bottlenecks can be effectively done using compound, yet abstract workflow models instead of commonly used process models, (ii) measures used to monitor the flow do not need to be very complex, and (iii) by changing the perception of well-known phenomena (e.g. using throughput instead of productivity) the development of the measurement system is straightforward and is not prone to spun conflicts. .

The remaining of the paper is structured as follows. Section 2 presents an overview of the work that has been done in this area to date. Section 3 presents the design of our research study. Section 4 presents the results of the study - the method and its realization at Ericsson. Finally section 5 presents the conclusions from our research.

2 Related Work

Petersen and Wohlin [4] used measurements of flow – cumulative flow diagrams – to identify bottlenecks. Their approach was based on identifying patterns in the flow diagrams and therefore point to the potential bottlenecks. The difference to our work is that we did not intend to measure flow since we recognized the fact that elements in the flow are not of equal size (something assumed by Petersen and Wohlin). We were also able to use measures which are automatically collected (as opposed to Petersen and Wohlin) which makes the method more efficient. Traces of measures related to flow, similar to Petersen and Wohlin, can also be found in [5].

Höst et al. [6] used discrete event simulation to explore bottlenecks in requirements engineering. Their approach was based on building a model of requirements engineering phase and using simulations to identify where bottlenecks appear. We used similar concepts as Höst et al. – in particular the concept of capacity – since our goal was also to use automated tools and measures.

Another example of using quantitative methods for identifying bottlenecks in a specific case is the work of Jasmine and Vasantha [7] who used graphs in order to identify reuse bottlenecks. Their approach is different from ours w.r.t. the use of mathematical tools behind the measures – matrices and vectors.

Theory of constraints [8, 9] was used in our work in order to initiate the discussions around the concepts of throughput and queue. In the future we intend to use this theory after we establish the criteria and thresholds for the measures of queue and throughput (planned for the future work) and in this way monitor bottlenecks in broad software product development (compared to the focus of one product only we intend to focus on the organization developing a number of products).

Ericsson's realization of the Lean principles combined with Agile development was not the only one recognized in literature. Perera and Fernando [10] presented another approach. In their work, Perera and Fernando, presented the difference between the traditional and Lean-Agile way of working. Based on our observations, the measures and their trends at Ericsson were similar to those observed by Perera and Fernando.

3 Case Study Design

In this section we shortly describe the design of our case study: industrial context, research questions, and units of analysis.

3.1 Context

The context of the case study was one of the software development organizations of Ericsson AB. The organization and the project within Ericsson, which we worked closely with, developed large products for the mobile telephony network. The size of the organization was several hundred engineers and the size of the projects can be up to 200 engineers. Projects were more and more often executed according to the principles of Agile software development and Lean production system referred to as *Streamlinedevelopment* (SD) within Ericsson [3]. In short, the principles of Streamline development postulated that software development was organized as a series of activities centered around the product and executed mainly in cross-functional teams responsible for the design, implementation and partially testing of their part of the product (e.g. a feature) [11]. This was found to be rather typical process design that addressed market pull in Lean development [12].

A noteworthy fact observed in our study was that in SD the releases of new software versions to customers were frequent and that there was always a release-ready version of the system: referred to as *Latest System Version* [3]. It is the defects existing (and known) in that version of the system that were of interest for the project and product management. Ideally the number of known defects in that version of the system should be 0, however, during the development (i.e. between releases) this number might vary as there might be defects which are discovered during the process of integrating new features (parts of code) into the latest system version. In practice there are almost always defects being fixed as integration and testing is continuous. However, at release times the main branch was frozen with the purpose of removing all defects – this removal was the responsibility of the *release project*, which was a project aimed at packaging the product and preparing it for the market availability.

In order to achieve the speed in releasing the software [2] and be able to quickly respond to changes (new features and requirements) the software development

organization needed to constantly operate at an optimal capacity[1]. This means that there should be no bottlenecks that hinder adding new features to software late in the process (i.e. agile response to change) and that such late changes do not postpone the releases of the software.

The capacity of this type of development was limited at any time by a given set of constraints which result in bottlenecks. Examples of constraints could be an amount of test equipment available, resources available for development of features, integration slots for the main branch, amount of resources available for defect removal after the internal testing processes.

The constraints were always present in the organization, but at certain points of time they were triggered and thus result in bottlenecks. For example if too many defects were discovered in the internal testing the resources available for defect removal might not be sufficient and therefore defect handling would become a bottleneck at a particular time.

The requirement from the studied organization for this research project was to establish a measurement system which would provide an overview of the phases in software development and monitor flow of work in the development process. Measured drops in the flow of work should warn the stakeholders about potential bottlenecks.

3.2 Research Questions and Units of Analysis

The research question presented in this section wasestablished together with Ericsson after a number of discussions with the organization in question. The main research question was:

> *How can we effectively identify and monitor bottlenecks in a Streamline software development project in order to assure efficiency of the process?*

The scope of the research project was deliberately limited to identifying and monitoring of bottlenecks excluding the aspects of how to handle the bottlenecks (i.e. how to remove constraints).

The unit of analysis was a single release of the product developed in the studied project. We decided to follow the project from the point of time when the previous release was ready (i.e. the release project took over) until the same point of time for the next release. The time between releases was a number of weeks, although due to confidentiality reasons we cannot provide the exact duration.

Our analyses consisted of both analyses of contemporary measurement systems used at Ericsson, documents (in particular documents related to the ways of working and requirements for quality of products), and analyses of interviews with stakeholders of the contemporary measurement systems as well as key persons from the project management team. The subjects of the interviews were: team leader of one of the development teams in the project, integration leader for the project, deputy project manager responsible for handling of defects during development, system test leader and network integration test leader. All of the subjects were senior software engineers with a number of years of experience in their fields. Three of them were also involved in our previous research studies.

[1] In this paper we can define the capacity as a total number of software features developed (i.e. analyzed, designed, implemented, tested and validated) in one release.

During the discussions within the studied organization we noticed that despite our initial goal to measure the flow and capacity in the organization, we should focus on a smaller and easier problem – monitoring throughput and queue. The simplification was dictated by the fact that the goal was to find bottlenecks, which can be found by observing falling/dropping throughput in phases after the bottleneck and growing queue before the phase where the bottleneck is located over time.

Our research process consisted of three parts:

1. Development of models of workflow in the software development projects organized according to the principles of Streamline development. The models simplified the view of the stakeholders about their ways of working, which in general could be different than the prescribed process view.
2. Development of measurement systems for measuring throughputs and queues for each phase in the workflow model.
3. Instantiation of the measurement systems for one project with the goal to evaluate the method.
4. As one could observe, we put effort on the actual ways of working and the perception of those by our stakeholders, rather than analyses of process descriptions.

The first two steps were done through interviews. For the development of models we asked the stakeholders to characterize their way of working in this phase. The interviews were also aimed to find measures and indicators for monitoring throughput and queue. In order to elicit these measures we asked the following two main questions:

- If you were supposed to notify the responsible stakeholder for the phase after yours about potential problems with the overflow of work – what kind of indicator would you like to use? – to elicit measures of throughput.
- If you were supposed to notify the responsible stakeholder for the phase before yours about potential "shortage" of work in your phase – what kind of indicator would you like to use? – to elicit measures of queue.

These questions were broken down into a number of aspects for discussion during the interviews.

4 Monitoring Throughput and Queue

In this section we present the results of the case study – description of how bottlenecks can be monitored using two main concepts: *throughput* and *queue*. We also present the results of the evaluation of our method in the project at Ericsson.

4.1 Definitions

In order to proceed towards the realization of measurement systems we needed to establish the basic vocabulary used in the project. It was interesting to observe that even the definition of the main concept – bottleneck – was different for different stakeholders in the project. The definitions shown below are taken from a number of sources and discussed with the organization.

- ***Bottleneck:*** A bottleneck is a phenomenon where the performance or capacity of an entire system [auth: software project] is limited by a single or limited number of components or resources.
- ***Flow:*** A flow is the motion characteristic of elements or functionality under development in a software project, see also [5].
- ***Throughput:*** Throughput is defined as the number of elements processed during a time period. For example number of test cases executed per week.
- ***Queue:*** Queue is defined as the number of elements to be processed (to flow) through a specific phase or a software project. For example number of test cases to be executed until the end of a given week.
- ***Capacity:*** Capacity is a total number of software features developed (i.e. analyzed, designed, implemented, tested and validated) in one release.

The general definitions of the concepts above are used to describe elements of software development process used at the studied unit of Ericsson.

4.2 Realization at Ericsson

As described in section 3.2, in order to identify and monitor bottlenecks we needed to establish a measurement system (see [13-15]). However, in order to establish this measurement system we needed to describe the process of software development in a model which allows monitoring of flows – e.g. similar to Value Stream Mapping[16] models using in Lean Development in other domains than software development. An example of such a model is presented later in the paper.

Figure 1 shows an overview model of process of developing software products in a similar way as describing production systems in Lean manufacturing. The boxes represent "stations" where a number of activities take place and the overview model presents those phases which we worked with during the study (the phases were defined by the stakeholders' responsibility areas). The process was drawn as a sequence, since the dependencies between stations are sequential; however, the whole Streamline development process at Ericsson was parallel. The teams developed new features in parallel; TR handling was done in parallel to development and integration, etc. The sequential dependencies meant that bottlenecks in one of these stations could influence stations after – i.e. that there are sequential causal relationships between these phases.

Each phase contained a number of stations, presented in detail in section 4.3. The model was created together with the stakeholders by asking them to characterize the work which they conduct and iterating the model a number of times in order to find the appropriate abstraction level for the description.

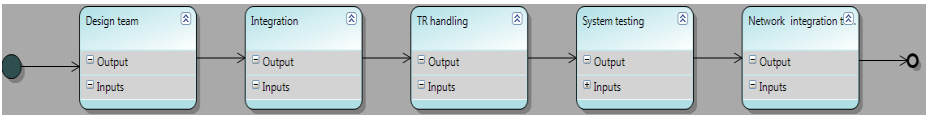


Fig. 1. Overview model of development process and related measurement systems

After describing the process model we needed a number of interviews to validate the model – in particular that it described the de-facto ways-of-working in the studied project. We also needed to perform a number of interviews with the goal to define the main concept of flow for each identified phase and throughput and queue for each phase.

4.3 Results

The results from our case study contain both the model of the workflow in the development connected to measurement systems (Figure 1) and the realization of the measurement systems visualized in a web page updated daily (Figure 6).

The model was created using the notation presented in our previous study [15] and dedicated for modeling measurement systems. The notation was complemented with Value Stream Mapping model of workflow in the development process. The model was intentionally created quite abstract as its purpose was to provide the stakeholders with the overview of the whole development flow. The overview was meant to inform the stakeholders what kind of main activities were executed at each phase. The measures and indicators were linked to the activities captured the notions of throughput and queue for each phase.

The measurement system for monitoring bottlenecks was realized by connecting together a number of measurement systems – one for each phase as it is presented in Figure 1.

The measures for throughput and queue identified during the interviews for the measurement systems are presented in Table 1.

Table 1. Measures for queue and throughput per phase

Phase	Throughput	Queue
Development team	# of function test cases developed per week	# of function test cases planned for the whole feature (but not yet developed)
Integration	# of features integrated per week	# of features planned for integration up to date in the integration plan (but not yet integrated)
Defect handling	# of defect reports entering the state “Closed” per week	# of defect reports that are not in state “Closed”
System test	no indicator was created for this phase[2]	# of test cases planned for execution up to a given week (but not executed yet)
Network integration test	# of test cases executed per week	# of test cases planned for execution up to a given week (but not executed yet)

[2] Due to the process changes in the organization we were not able to define this indicator. Potentially (or theoretically) it could be a similar indicator to the one for network integration test phase.

The measures summarized in the table were collected manually for the development team and automatically for all other phases. The reason for the manual collection was the fact that this information was not available in a form which could be automatically processed.

The rationale behind the measures, as a result of interviews, is presented in the following subsections.

4.3.1 Development Team

The measures defined for the development team were elicited in two interviews with a team leader and a senior system engineer working in the team. In the studied organization, the teams are usually formed by a number of specialists from a number of fields – e.g. designers, programmers, system engineers, function testers, object leaders – and are thus called cross-functional teams. Each team is responsible for the complete development of one feature at a time and is able to integrate the feature into the main code branch after it is tested.

The test cases are developed directly before execution (although they are planned in advanced) and in sequence. Therefore the most effective way to monitor the throughput of work in the team is to monitor the number of test cases developed and the most effective way of monitoring the queue was to monitor the changes in the

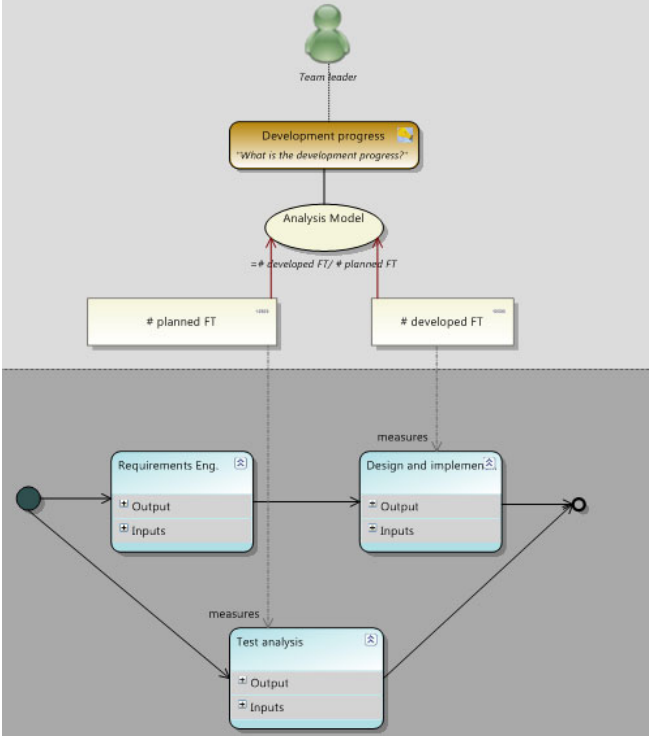


Fig. 2. Stations/activities, measures and indicators in the Development Team

number of test cases planned. The three stations that were identified for this phase are: (i) requirements engineering, (ii) design and implementation, and (iii) test analysis. The model of the flow and the related indicators and measures are presented in Figure 2. The figure shows also how we modeled the work flow in detail and how we linked the activities in the workflow with the measures and indicators for monitoring bottlenecks.

The lower part of the model is the flow model and the upper part is the model of the measurement system according to ISO/IEC 15939 [17] and our own modeling notation [15, 18]. In this flow model the test analysis is drawn in parallel to the requirements engineering and implementation. The reason for this is that there was no sequential dependency between these stations although the implementation and test analysis end simultaneously. The test analysis could influence the queue for this whole phase, i.e. the number of test cases to be developed. If there was a bottleneck in this phase, then the team would not be able to test and develop the test cases as planned.

4.3.2 Integration

When the teams were ready with the development of the feature assigned to them, the feature was integrated into the code at the main branch. This integration was a managed and planned process driven by two main principles: (i) the integration time slots should be used as efficient as possible (meaning that all integration slots should be occupied) and (ii) the integration of features should be spread during the whole project, not gathered in the end (i.e. to avoid big-bang integration).

The integration plan was the basis for the integration and was an electronic document frequently updated. The number of features in the integration plan are planned (but not yet integrated) up to a given week form the queue for this phase. The number of features integrated forms the throughput measure.

The rationale behind the queue was that if there were delays in the integration, it would grow since the features would not be integrated, but remain as planned in the plan. The queue would grow if the throughput drops in case there is a bottleneck in the integration phase since the teams would still deliver new code to be integrated.

If the queue dropped suddenly, it would indicate that there was a bottleneck in the phase before (development team) and therefore the integration process did not operate at the desired/planned capacity.

4.3.3 Defect Handling

During the function testing in the teams or in the integration process, a number of defects might have been reported. Before proceeding with the testing, these defects needed to be removed. The process of assessing the severity of the defect, assignment to the development team and verifying that the defect has been removed needed to finish. There were three main stations that were identified: (i) defect assessment, (ii) defect removal, and (iii) defect verification and closing.

Based on this process the measurement for the throughput was quite straightforward – it was the number of defects that were closed per week. The queue was then the number of defects that were still open and remained to be closed.

If there was a bottleneck in this phase then the queue would raise and it would eventually stop the whole development chain as no features would be ready for the delivery because of insufficient quality.

If there was a bottleneck in the previous phase then the number of open defect would drop significantly meaning that the test cases were not executed or that no new features were ready for integration.

4.3.4 System Test

System test (ST test) was the first phase where the source code integrated into the main branch was tested as a whole. The phase consisted of two main activities – test planning and ST testing (i.e. ST test execution), the measures for throughput and queue were therefore

- the number of test cases planned to be executed (and not yet executed) up to date – measure of queue.
- the number of test cases executed – the measure of throughput. Please note that this is an indicator proposed based on a similar indicator in the forthcoming phase. We were not able to verify this indicator since the internal testing processes were different.

If there was a bottleneck in this phase then the queue would raise while the throughput would drop. A growing queue also indicated an existence of a bottleneck in the ST test phase.

4.3.5 Network Integration Test

Network integration test phase (NIV testing) was the phase when the software products were tested in their target environment and at the target platforms. The function and system test cases were used to test the software in new configurations and therefore it mainly consisted of the test execution itself (defect handling was captured in a separate phase as described before). Test planning was then only limited to planning when different test cases should be executed and, according to the stakeholder, should not be distinguished from the test execution phases as both planning, re-planning and execution were conducted continuously and in parallel.

The two measures were:

- the number of test cases planned to be executed (and not yet executed) up to date – measure of queue.
- the number of test cases executed – the measure of throughput.

If there was a bottleneck in this phase then the throughput would drop as the number of test cases executed would decrease. An example of a constraint causing a bottleneck could be the access to the test equipment or the need to reconfigure the equipment more than planned.

4.4 Industrial Validation

The measurement system for bottlenecks was used in one project so far and resulted in identifying an efficient way of monitoring the flow of work in the project as presented in Figure 3[3]. The measurement system contributed to the identification of problems

[3] Please note that the numbers in the figure are only an example and do not show the real data from the organization due to the confidentiality agreements with the company.

with the last phase of the flow, which were forecasted based on the indicators from the previous phases. The throughput of the test process for the network integration test phase was shown to be insufficient to handle the growing number of features to be delivered. The release of the product was postponed as a result of this insufficient throughput. The future projects took another approach to testing in order to increase the throughput of that phase.

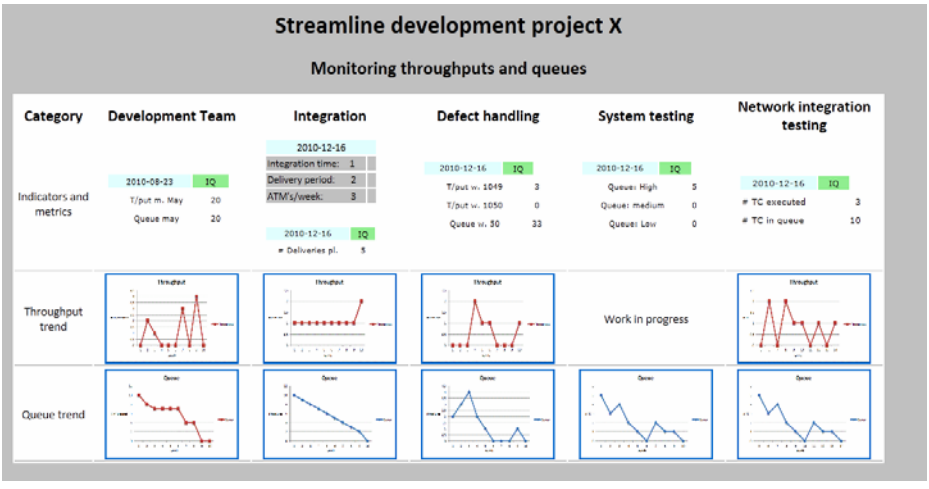


Fig. 3. Realization of the measurement system for monitoring bottlenecks

In Figure 3 we present how we realized the measurement system in practice. Behind the web page shown in the figure we used 5 measurement systems collecting the data from a number of systems, which together form an important tool for monitoring bottlenecks. Although we cannot show the real data, we illustrated dependencies between measures which we observed in the project – the most important one being the dependency between throughput and queues. We can also observe that the phase with the most stable flow of work is the integration phase, which is caused by the fact that the integrated code is usually divided into parts of equal size and that the builds usually take a constant period of time (as the merge, compile, etc. time does not significantly depend on the size of the integrated functionality).

In our further work we intend to investigate whether there are indeed bottlenecks in the flow of work and whether these are measured accurately by the measurement system. As one could observe from Figure 3, at this point of time there seem to be no bottlenecks in the flow.

4.5 Good Practices

While developing and deploying the measurement system we observed a number of good practices which helped the organization to be effective in monitoring bottlenecks.

1. **Monitor throughput, not productivity.** Although monitoring of productivity is tempting in a context like this (i.e. identifying bottlenecks), it was much harder to

achieve consensus on how the productivity should be measures and how it should be compared. As productivity in general is perceived related to humans, the consensus might even be impossible for larger organizations (e.g. productivity of designers and testers cannot be easily compared). However, since throughput is related to the number of items processed per time unit, the focus is on the items processed – hence consensus can be achieved more easily.

2. **Monitor capacity, not speed.** For a similar reason as above one should avoid measuring speed. Since speed is usually related to humans, the consensus is harder to obtain as no one wished to be "the slowest one". Capacity, on the other hand is a property of the organization and increasing speed is only one of the possible solutions (another one is to increase the number of person-hours in the project) and therefore it shifts the focus from individuals to collective responsibility of the project.
3. **Focus on measuring flow, not constraints.** When asked about potential bottlenecks in the organization, the stakeholders in our project could sometimes point to constraints which limited the flow. An example of such a constraint could be knowledge of particular technology. Although commonly referred to a bottleneck such a phenomenon is constraint that limits the capacity of the organization (or speed of individuals who has to learn instead of performing). In addition to that it is often not possible to measure such a constraint. However, if a throughput drops in one of the monitored phases, one could easily identify this constraint and decide to remove it – it is also more important to remove the constraint than to measure it.
4. **Do not forget about velocity of items flowing through the system.** When focusing on monitoring the flow the organization might forget that flow is not the only aspect important to monitor. The fact that the project or organization operates at the optimal capacity does not mean that the items are develop fast enough – it might still take a long time between the idea of the feature and its delivery to the customer. Therefore the organization also should measure whether the velocity of the flow is sufficient, since it is often important to be the first one to market with new features, not only the one with reasonable price.

The observed good practices seem to be rather generic and straightforward to use at other companies than Ericsson. They have shown themselves to be important when establishing and deploying the measurement systems.

4.4 Validity Discussion

In order to evaluate the validity of our study we have used the framework presented by Wohlin et al. [19].

The main threat to the external validity of our study is the fact that we have studied only one software development project. Although this threat is real, we still believe that the concepts of throughput and queue could be defined operatively for other projects.

The main threat to the internal validity is the fact that our study used the process description which was elicited in the study itself. This bears the risk that if we had another process description, then the results of our study would be different. However, since we explicitly asked about the measures during the interviews, the risk for it is minimized.

The main threat to the construct validity of our study is the fact that we have based the identification of our measures on interviews and not on statistical data analysis. In particular we have not used Principal Component Analysis to find related measures or time-series analysis. Since the period of the study was short and the measures were new we were not able to construct a valid statistical support.

5 Conclusions

In this paper we presented a new method for identifying and monitoring of bottlenecks in the flow of work in Lean software development. We conducted a case study at one of the software development units of Ericsson in order to develop the method together with the stakeholders in our project.

The results of our study show that in order to identify bottlenecks one can monitor throughputs and queues at a number of phases in software development. We also identified one bottleneck in the last phase of the development project, which provided the basic evidence that our method can effectively identify and monitor bottlenecks. Therefore we can conclude that monitoring of bottlenecks in software development projects can be done when we can use two crucial concepts - throughput and queue.

Our future work is planned to apply this method for other software development programs and to monitor bottlenecks there. We also plan to generalize this method and find "thresholds" for queues and throughputs that could be reused (something that requires collecting a significant amount of statistical data over longer period of time).

Acknowledgements

We would like to thank SoftwareArchitectureQualityCenter at Ericsson and IT University of Gothenburg for their support in the study. We would also like to thank all managers and stakeholders at EricssonAB for their contributions, insight and support for the project.

References

[1] Staron, M., et al.: A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation. Information and Software Technology 52, 1069–1079 (2010)
[2] Mehta, M., et al.: Providing value to customers in software development through lean principles. Software Process: Improvement and Practice 13, 101–109 (2008)
[3] Tomaszewski, P., et al.: From Traditional to Streamline Development - Opportunities and Challenges. Software Process Improvement and Practice 2007, 1–20 (2007)
[4] Petersen, K., Wohlin, C.: Measuring the flow in lean software development. Software: Practice and Experience, n/a–n/a(2010)
[5] Oppenheim, B.W.: Lean product development flow. Syst. Eng. 7, 2 (2004)
[6] Höst, M., et al.: Exploring bottlenecks in market-driven requirements management processes with discrete event simulation. Journal of Systems and Software 59, 323–332 (2001)

[7] Jasmine, K.S., Vasantha, R.: Identification Of Software Performance Bottleneck Components In Reuse based Software Products With The Application Of Acquaintanceship Graphs. In: International Conference on Software Engineering Advances, ICSEA 2007, p. 34 (2007)
[8] Dettmer, H.W.: Goldratt's theory of constraints: a systems approach to continuous improvement. ASQC Quality Press, Milwaukee (1997)
[9] Dettmer, H.W.: The logical thinking process: a systems approach to complex problem solving. ASQ Quality Press, Milwaukee (2007)
[10] Perera, G.I.U.S., Fernando, M.S.D.: Enhanced agile software development - hybrid paradigm with LEAN practice. In: International Conference on Industrial and Information Systems (ICIIS), pp. 239–244 (2007)
[11] Akg, A.E., et al.: Antecedents and consequences of team potency in software development projects. Inf. Manage. 44, 646–656 (2007)
[12] Liker, J.K.: The Toyota way: 14 management principles from the world's greatest manufacturer. McGraw-Hill, New York (2004)
[13] Staron, M., et al.: A Framework for Developing Measurement Systems and Its Industrial Evaluation. Information and Software Technology 51, 721–737 (2008)
[14] Staron, M., Meding, W.: Predicting Weekly Defect Inflow in Large Software Projects based on Project Planning and Test Status. Information and Software Technology, p. (available online) (2007)
[15] Staron, M., Meding, W.: Using Models to Develop Measurement Systems: A Method and Its Industrial Use. Presented at the Software Process and Product Measurement, Amsterdam, NL (2009)
[16] Dolcemascolo, D.: Improving the extended value stream: lean for the entire supply chain. Productivity Press, New York (2006)
[17] International Standard Organization and International Electrotechnical Commission, ISO/IEC 15939 Software engineering – Software measurement process, International Standard Organization/International Electrotechnical Commission, Geneva (2007)
[18] Meding, W., Staron, M.: The Role of Design and Implementation Models in Establishing Mature Measurement Programs. Presented at the Nordic Workshop on Model Driven Engineering, Tampere, Finland (2009)
[19] Wohlin, C., et al.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publisher, Boston (2000)

Applying Agile and Lean Practices in a Software Development Project into a CMMI Organization

Miguel Morales Trujillo[1], Hanna Oktaba[1], Francisco J. Pino[2,3], and María J. Orozco[4]

[1] Graduate Science and Engineering Computing, National Autonomous University of Mexico
Mexico City, Mexico
{migmor,hanna.oktaba}@ciencias.unam.mx
[2] IDIS Research Group
Electronic and Telecommunications Engineering Faculty, University of Cauca
Calle 5 No. 4-70, Popayan, Colombia
fjpino@unicauca.edu.co
[3] Alarcos Research Group
Institute of Information Technologies & Systems, University of Castilla-La Mancha
Paseo de la Universidad 4, Ciudad Real, Spain
francisco.pino@uclm.es
[4] Ultrasist
Miguel Ángel 28, Mexico City, Mexico
mjorozcom@ultrasist.com.mx

Abstract. This paper presents an approach based on a practical experience in applying agile and lean practices in a software development process performed into an organization evaluated CMMI level 5. As a result of a theoretical review on agile and lean practices, and the organization's needs, an integrated proposal between these practices and CMMI was found and was also put into practice. The work carried out by the organization using this proposal led to a successful integration experience in order to innovate, improve product quality, get clients' satisfaction, and the most important, show the feasibility of coexisting of CMMI and agile practices resulting in a significant improvement for the organization.

Keywords: Agile practices, Lean practices, SCRUM, CMMI, Software Development Process.

1 Introduction

In a software developer organization, the need to improve their capabilities is mandatory. One way to reach that goal is to adopt the best practices from a process reference model which guides the organizational improvement.

A world known reference model is the Capability Maturity Model Integrated (CMMI), it provides an approach to improvement in which processes and improvements to them are supported by the organization [1].

It is a strong belief that agile development methods and CMMI best practices are orthogonal with each other. The experience presented in this work is intended to

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 17–29, 2011.
© Springer-Verlag Berlin Heidelberg 2011

clarify and highlight the benefit from using both of them resulting into an improved business performance, as has been suggested by [1]. Furthermore, according to [2] the agile methods can support some success factors involved in improving organizational processes.

The main contribution of the practical experience discussed in this paper is to show how an organization assessed CMMI level 5 has integrated their software development process with agile and lean practices in order to inject agility, simplicity and refreshment into the processes, increasing the quality of the software product and the client's satisfaction. In this respect, this work reports a successful experience of melting agile practices into a CMMI level 5 organization. However it is important to highlight that it is not the purpose to present a new model or methodology, neither to create a new agile method. Clearly speaking, our approach is based on a selection of agile and lean practices suitable for a particular project in order to improve the process, but not to replace it.

The paper is organized as follows: in part 2 we present the background, part 3 shows the context of the organization and the project that we present as experience. The experience report is shown in part 4, and finally we conclude and mention the future work.

2 Background

Several solutions to improve software development processes are suggested, many of these suggestions have come from experienced practitioners, who have labeled their methods as *agile software development* [3].

Agile methods have been demonstrated to enhance customer value, improve organizational morale, and provide step improvement in product quality [4]. For that reason numerous organizations have chosen this particular path to improve; however the big problem is to find the right balance between agility and discipline.

According to [5], for a company in transition towards agile software development there is a need for practices and guidance for implementing and supporting an agile approach across the organization. Besides, taking into account the organizational context helps eliminate the suspicion about agility [6].

Another area of conflict for mature organizations will be the problem of considering how or to what extend agile processes will affect their ratings with respect to CMMI, ISO, or other process standards [7]. However, agile methods are in line with many CMMI level 5 concepts, mainly regarding continuous improving of processes performance, thus, instead of acting like a barrier, the agile methods impulse those CMMI concepts.

On the other hand, talking about their limitations, most agile methods do not support the degree of documentation and infrastructure required for lower-level certification; it might, in fact, make agile methods less effective [7].

From that point of view, the difficulty stands in filling the gap between the traditional development and the agile development, for that reason it is a favorable alternative to try to transform or remove particular aspects of the processes to make the most of each one. Lean software development [8] [9] promotes removing waste as one of

its principles. According to [10], extra or unnecessary functionality, slow internal communication, task switching and bureaucracy fall into the category of "waste". However, complexity science seems to show that waste can have various functions; in complex systems things that look like waste can actually be a source for stability and innovation [11].

In order to preserve stability into the processes, approaches like [7] suppose to help the integration of agile practices into the traditional process, firstly, by conducting an analysis of existing and proposed processes to identify mismatches in process requirements and expectations. Secondly, by taking into account the project's needs, there builds up a process with indispensable components, and specific responsibilities are defined to address with the agile approach, establishing the milestones to better fit an iterative approach.

Later on, the integration continues by implementing agile practices that support existing processes or new organizational priorities, like prioritizing requirements to keep on a schedule when new requirements emerge, or test-first and continuous integration to find problems beforehand.

An improvement seeking organization wants to assess the impact of process changes before introducing them to improve the way of working, however in development projects the change proposal is evaluated in a real situation [21]. A case of study is suitable for industrial evaluation of software engineering methods, according to [21], one way to arrange a case study, it's to make a comparison of results of using a new method against a company base-line, then is possible to compare both.

In this context, to develop the projects reported in this paper, which seek to incorporate agile and lead practices in the development process of a CMMI level 5 organization, the approach mentioned above was taken into account. This work reports the qualitative experience of transitioning from the traditional development to the agile development, by means of two projects carried out for this purpose, within an organization that has used CMMI as reference model to define its software development life cycle. Likewise it presents challenges faced by the project teams, as well as the benefits of using agile practices, and practical recommendations for those starting and facing similar challenges.

3 Context of the Software Development Project

This work was developed at Ultrasist [12], a Mexican organization certified with CMMI level 5 version 1.2 since March 2009. Ultrasist is oriented to develop applications through a work environment based on processes and international standards of quality. At the time of the experience presented, the organization was made up by 83 employees.

Innovation is a main objective of the organization, for that reason it created a weekly workshop called *Research Seminar*, where the members propose new ideas, methodologies, solutions and improvement opportunities in order to optimize the group of processes and performance of the organization.

One of the concerns at the workshop was introducing agile practices into the process with the objective of improving particular aspects. The existence of agile practices at the organization was not new, but they were merely focused on internal processes and activities, for example, as a result of this workshop, a technique based on Pair Programming and called eXtreme Analysis was developed. It is executed during the Analysis phase in order to improve the definition of the requirements, alongside with carrying out the Peer Review in every phase.

In this case, three factors were identified to be improved: customer's satisfaction, work team communication and, the most ambitious, reduction of the release time of a product.

At that moment a new project was required by a frequent client; thus, the High Management approved to search a set of practices that could be applied by the work team assigned to that project. The software products associated with those projects were related to receipt, control and administration of funds. For privacy reasons we will not mention the client's name and it will be simply referred to as "the client" and the projects involved as "PAL" (acronym for *Projecting Agile & Lean*).

After the ending of PAL1, the client asked for another product and we will call the corresponding project as PAL2. In general terms, when we mention PAL projects we will be referring to both PAL1 and PAL2 projects.

In order to speed up PAL projects, the organization assigned two resources (PAL team) from the Process Group, one of them being the first author of this paper. The PAL team was given the task to identify specific processes or activities candidates to be replaced or improved by agile and lean practices. Subsequently, those resources developed the necessary material, on one hand to apply the selected practices into the development process and, on the other hand, to train the work team to apply the required knowledge and achieve the desired objectives.

3.1 Project Description

The PAL projects involved the phases of Analysis, Design, Construction and Tests together with the resources specialized in each of those phases. The organization manages its resources in two levels: horizontal and vertical. The horizontal level consists in a group controlled by a specialist called *Technical Leader*, for example, the Specification Leader manages all the people with the role of Analyst, and the Test Leader is in charge of all the testers.

At the vertical level the main role is played by the Project Manager who is in charge of a team made up by people working on a particular project. Therefore, a person, who has issues associated with the technical part, should inform their Technical Leader; on the other hand, if he has issues associated with the management of the project, he should contact the Project Manager in charge of that project.

The nature of the projects required developing of all the Testing phases at the client's facilities. For that reason there have existed historically communication issues between the Test leader who stays at Ultrasist and the testers' team that has to be at the client's, as well as between the Construction leader and the programmers' team.

The work team assigned to the projects was composed by 7 members: 1 Project Manager, 2 Analysts, 1 Designer, 2 Programmers and 1 Tester, where it is important to observe that those are the official roles that they play in the organization normally.

However, for the purpose of the PAL projects their functions and responsibilities were not limited to those roles. The size of each project was given in use-case points, being 235 use-case points for PAL1 and, 270 for PAL2, respectfully. The estimated delivery dates were for PAL1 March 19th totalizing 35 estimated working days and, for PAL2 May 14th totalizing 40 working days, both developed during 2010. See Table.

Table 1. Estimated dates, use-case points and hours of each project

Project name	Contract Dates		Use-case points	Effort (in hours)
	Beginning	Closing		
PAL 1	February 1st	March 19th	235	603.50
PAL 2	March 22nd	May 14th	270	628.45

The client's satisfaction was a hard issue to deal with because of their high demand and low commitment, with almost none chances for communication after the contract is signed and ever-changing requirements on road.

4 Experience Report

To perform the introduction of agile and lean practices into the organization processes, the PAL team in the first place identified the specific processes or activities necessary to be replaced or improved by agile and lean practices.

Later it developed the necessary material to apply the selected practices into the development process and finally introduced the work team to the new practices. The PAL team work is described below to more detail.

4.1 Project Execution

In order to make clearer the structure and the execution, the project was divided into five steps. Each step having a specific purpose, the first four steps were developed only by the PAL team, and the last one developed in conjunction with the work team.

Step 0 had as its objective the selection of agile and lean concepts. Those concepts were selected mainly from SCRUM [13], Lean and Test Driven Development [14]. Some of the concepts selected were:

- Self directed and self organized team.
- Daily meetings are carried out to listen to the progress feedback and to troubleshoot impediments.
- Short iterations.
- Quantifiable measurements of progress.
- Design for change and respond to change.
- Bring decisions forward, Planning and responding.
- Design test first and test in short cycles.
- Fuzz the distinction between requirements and tests.

SCRUM was chosen as based on the approach described in [15], where an adaptation of SCRUM to a traditional software development process is presented.

Additionally, the SCRUM process flow, see Figure 1, provides guidance for efficient project management in a certain way that allows high flexibility and adaptability, and CMMI provides insight into what processes are needed to maintain discipline [16]. Thus, an accurate mix is obtained from both of them to start the improvement focused on the main objectives.

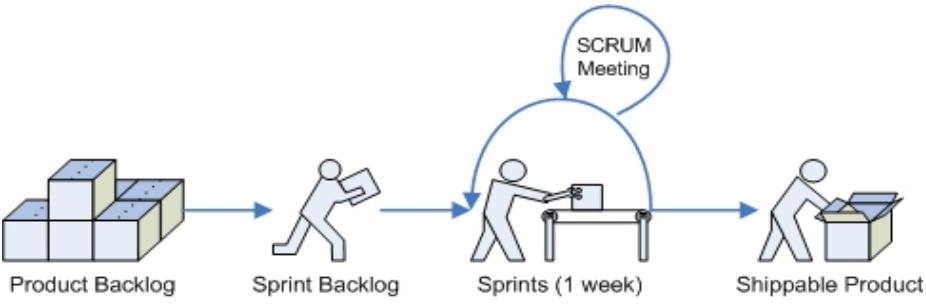


Fig. 1. SCRUM process flow

At Step 1, there was created a cookbook according to the organization processes and the agile and lean practices stemmed from the concepts selected in Step 0. Moreover, a couple of presentations and a guide were produced to introduce those selected items to the work team during the next steps. One presentation was based fully on SCRUM and its principles, benefits, requirements and practices for project management. The other presented the objectives of the organization that pursue agility and lean introduction.

The cookbook was divided into three parts, where the first one describes the actual software development process of the organization and the candidate practices to be replaced or improved; the next section shows the agile and lean practices selected, mainly for the project management and the test phase; and the last one presents the method of introduction and application of these new practices, including activities, roles customization and guidelines for tailoring the process. In Table 2 a fragment of the cookbook is shown.

The cookbook's contents focus on three major aspects: Planning: arrangement of the activities and scheduling tasks, Coordination & Communication: organizing and matching up the team, and Testing: improving the activities in the Test phase of the existing software development process. The agile and lean practices included came from [17] [18] [19], and can be summarized as follows:

- Parallelization of the Testing activities with the Design and Programming traditional activities.
- Planning and designing tests before coding.
- Merging the testing team within the development team at early stages.
- Avoiding giving a formal track to the discovery of defects.
- Making a best estimate of activities including the perspective of each member of the developing team.

Table 2. Fragment of the cookbook (left: old activity, right: proposed new activity)

Software Development Process	
A.7 Test Phase	
Previous Activities	PAL Activities
1. Distribute tasks to team members according to their roles.	**Avoid...Test department.** Sometimes the test department assigns the testers at the end of the iteration, this is not recommended.
2. Design the Test Cases.	**Avoid...Assuming testing means testing.** Testing does not start after coding is completed; specify the system behaviour by writing the test cases.
3. Check the Test Cases.	**Try...Simple testing classifications.** The test design should be guided by an expert, concentrating the efforts of the testing in business-critical requirements.
4. Fix the System Test Cases based on the Verification Report.	-
5. Validate System Test Cases.	**Try...Test the walls.** Sketch the design of the test cases on a whiteboard. This promotes collaboration and team review.
6. Fix and adapt, if needed, the System Test Cases based on the Validation Report.	**Avoid...Separating development and testing.** Tracking this kind of development involves merging the test team within the development team, another direct consequence of this action is a better planning and allocation of activities.
7. Execute tests into the test environment.	-
8. Register defects into the Defect Tracking System.	**Avoid...Using defect tracking systems during the iteration.** If it takes a lot of time and effort, a task on the Sprint Backlog is created.
9. Perform System Acceptance Testing.	-
10. Correct the defects found.	**Try… Zero tolerance on open defects.** This prevents from spending effort on tracking and prioritizing them, and from delays caused by waiting to fix them.
11. Verify and close defects.	**-**

The introduction of new practices was guided to accomplish the objectives covered by the previous activities. Each new practice responds to a necessity of changing the way of how to do it, not what to do, that is, the old set of practices is equivalent to the new one in terms of achieving the CMMI requirements and goals, because every new

practice proposed is associated with an old practice, consequently, both of them achieve the same goal and objective. Due to the fact that the old set of practices covered the CMMI requirements, the new practices do too.

Once the practices had been identified and validated, the PAL team started the creation of templates and their customization, being this Step 2. The templates are based on SCRUM products: the *Product Backlog*, the *Burn Down Chart* and the *Sprint Backlog* managed into an excel spreadsheet. It is worth mentioning that the election of a spreadsheet rests in two main factors: the first one is that it eases the customization in the first application, and the most important is that the work team does not have access to the web when they stay at the client's facilities.

The *Sprint Backlog* is the prioritized list of tasks to be completed during the sprint, in this case the template also considered who selected each task and their status. In the same template the hours estimated for each task and their daily progress were recorded. See Figure 2.

SPRINT	START	DURATION
1	5-mar-10	20

		M	T	W	Th	F	S
		5-mar	6-mar	7-mar	8-mar	9-mar	10-mar
	Pending tasks	4	3	3	2	2	0
	Pending working hours	19	12	12	6	5	0
Sprint Backlog							
Sprint ID	Tasks	Type	Status	Responsible	EFFORT (in hours)		
1	Requirement 1	Analysis	In progress	Work team member 1	8	3	4	2	1
2	Requirement 2	Design	Done	Work team member 2	6	5	4	0	0
3	Requirement 3	Construction	To be done	Work team member 3	4	4	4	4	4
4	Requirement 4	Test	Deleted	Work team member 2	1	0	0	0	0

Config | Sprint Backlog | Graphics | Product Backlog | Hours Board | Obstacles Board | SCRUM Roles

Fig. 2. Fragment of the Sprint Backlog customized template

Taking into account the status and the hours left from each task, the section *Graphics* had three charts, one showing the daily progress of tasks done, another screening the estimated hours left to the end of the sprint, alike *Burn Down Chart*, see Figure 3, and the last one presenting the hours left to the end of the sprint of each work team member.

The *Product Backlog* template was a list of each requirement, with the addition of three values: the name and number of tasks into which the requirement was decomposed, the estimated done date and the sum of estimated hours assigned to it.

Step 3 focused on presenting the principles of agile and lean practices to the work team. Also, a mapping between SCRUM roles and organization's roles, see Table 3, the cookbook and the templates was presented. For that purpose a couple of one-hour sessions with the work team were held. In those sessions the PAL team reinforced the organizational guidelines for tailoring the process.

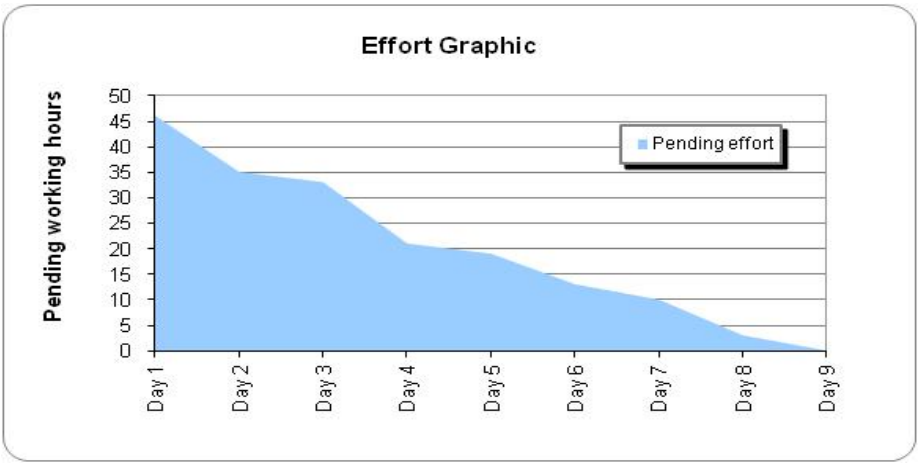


Fig. 3. Fragment of the Burn Down Chart

Table 3. Mapping between roles

Normally played role	SCRUM role
Technical Leader / PAL team	SCRUM Master
Analists Designers Programmers Testers Project Manager	Team
Client	Product Owner

To conclude, Step 4 started when the application of the practices and the data collection began. Taking into account that an often-overlooked difference between agile and traditional engineering processes is the way everyday business is conducted [7], during the first project, PAL1, the work team was helped every day to guide the SCRUM meeting, to introduce the new practices and to collect the data into the templates.

For the second project, PAL2, the work team applied the cookbook by themselves, but the PAL team attended the SCRUM meetings, monitoring and supporting the application of the cookbook.

It is worth mentioning that the effort in hours invested by the two members of the PAL team in the execution of the five steps was 186 hours, starting in the middle of January and finishing about the middle of May, the major part of hours concentrated in the first half of this period.

4.2 Data Collection

The data collection was developed using *Participant Observation*, a systematical and unobtrusive technique defined in [22], through SCRUM meetings and concentrated in the templates. A SCRUM meeting was held every morning, asking the team three

simple questions by the SCRUM Master: What have you done since yesterday's meeting? What are you planning to do today and how many hours will it take you? Do you have any problems preventing you from accomplishing your goal?

For practical reasons the SCRUM Master was played by the Technical leader supported by a member of the PAL team if needed. Having the answers to these questions, the PAL team collected hours, effort and task selection data.

In order to provide a self-contained spreadsheet, the tab *Config* controls specific values like: work team members' name, starting date, status names, phase names and number of sprints.

The tabs *Obstacles Board* and *Hours Board* were added, the first one to describe and manage the obstacles reported by the work team at the SCRUM meetings, such as their type, reporting date, solving date, impact and priority; and the second one, to control the hours assigned to the project by phases and by tasks. The use of those tabs was optional.

4.3 Results

At the end of the PAL projects the goals set initially were fully obtained. To recall, the objectives initially proposed were: increase the customer's satisfaction, improve the work team communication and the bonus of reducing product release time.

Some metrics were collected across the project, but some information had to be obtained by means of direct questions to the involved people. For that purpose, at the end of each PAL projects two Retrospective Meetings took place. The first one was held together with the PAL projects work team, the PAL team, the High Management and the Process Group. At the second one only the presence of the PAL projects work team and the PAL team was required.

These meetings are similar to the Sprint Retrospective meeting in SCRUM, and they had the purpose of reflecting on the past project, identifying what went well during the project, what could be improved, and mainly for our interest, detecting and perceiving the feelings of the team and the client.

Firstly we expand on the communication in the work team. This issue had the most important impact internally; the team members were integrated every day during the SCRUM meetings, managing to achieve direct communication of their needs and concerns and overcoming obstacles together.

There is no metric to show this improvement, but facts like simplicity and early feedback where mentioned constantly in the Review Meetings. In the work team's own words *"The solution to our problems (obstacles) is found at the end of the day, instead of at the end of the week!"*.

Besides, not only the communication at work team level was improved, but also the feeling of being heard and supported by the High Management level increased.

Secondly, the constant change of the requirements was managed considerably better. To be more explicit, Figure 4 shows a graphic of pending tasks with variables of working days and tasks number in similar projects, where a sudden appearance of new tasks in the middle of a project is clearly observed.

On the contrary, Figure 5 presents the number of pending tasks in PAL2 project, where the planning and estimation of tasks was enhanced, by taking into account new

practices in order to mitigate, since the beginning of the project, the impact of sudden unexpected tasks.

Thirdly, the reduction of the product release time was achieved, in PAL1 project the product was released 6 working days before planned, and in PAL2 5 days earlier. This fact brings about a considerable reduction of the effort invested and a higher motivation. In addition, according to the quality assurance metrics the PAL projects achieved lower defect densities than the historical metrics.

All together it resulted in the increase of the client's satisfaction, demonstrated by a phone call to the High Management praising the great job done and early signing of the delivery agreement.

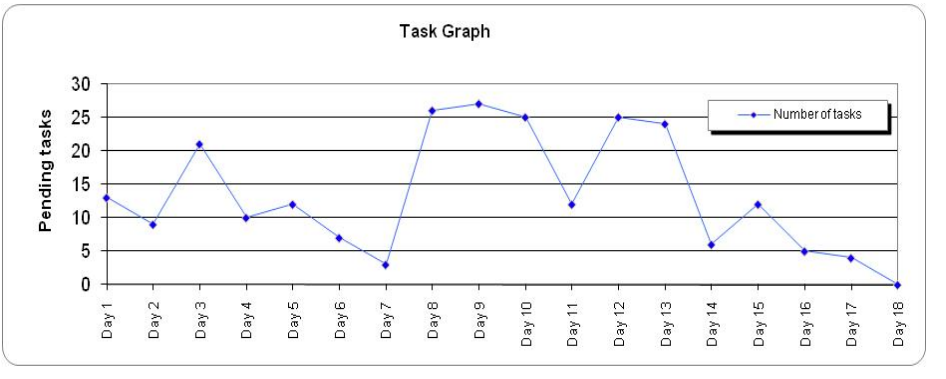


Fig. 4. Task graph of a similar project

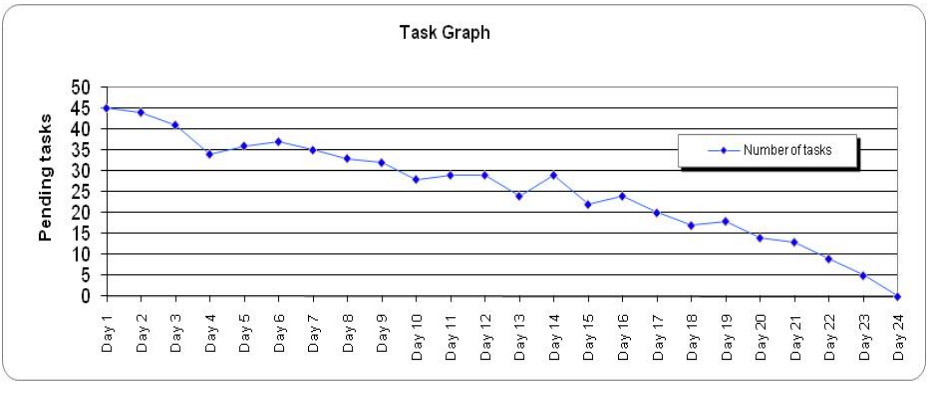


Fig. 5. Task graph of PAL2 project

5 Conclusions and Future Work

Nowadays, many organizations run agile software development projects alongside their traditional ones [20]. For this reason, the experience presented in this work is one more effort intended to clarify and highlight the benefits from using both.

We demonstrated the viability of merging CMMI best practices with agile and lean practices, in order to improve the capacity and maturity of the processes developed by the organization, always trying to preserve stability and helping the integration of agile and lean practices into the traditional process.

An analysis of existing and proposed practices was conducted to identify the project's needs and indispensable process components to be addressed by the agile approach, thus a balanced process was found to substitute the existing one.

As a result of this work, we provided a controlled and guided introduction of agile and lean practices into the organization processes, improving factors like:

- The client's satisfaction increased significantly; we even received a congratulation phone call.
- The work environment of the work team was enhanced, the team's professional and personal development was given a boost, the results obtained were better and the feeling of being heard and taken into account by the High Management increased and was motivating.
- The communication protocols and channels were improved and shortened, which motivated the team to be more collaborative and participating.
- The quality of the final product was improved.
- The estimation and selection of tasks were improved, obtaining better estimations and balanced workloads onto the work team members.

On the other hand, some aspects related to the change resistance were faced, in particular, the mistrust in agile practices caused by the work team members' unawareness. This skepticism was gradually solved by introducing the agile concepts in the workshops and clarifying doubts constantly, but mainly, the team was convinced by the results obtained day by day during the development of the projects.

Another problem was the feeling of "having more work to do", as the introduction of the new practices really required extra time and effort spent by the members involved. So, at the beginning, with the assistance of the PAL team that feeling was controlled, thereafter, the direct benefits obtained from that "more work" convinced the team that the outcome is worth the effort.

Finally through these projects, we highlight and foster the feasibility of integrating of agile and lean practices into traditional practices at an organization certified CMMI level 5. We encouraged success factors involved in improving organizational processes, work environment and product quality.

For future work, we consider the migration of the final customized templates to a tool that minimizes the effort and maximizes the portability. Refining and adapting the cookbook's contents in order to extend its scope to more processes are also considered as useful and beneficial for the organization.

Acknowledgements

This work has been possible thanks to Ultrasist's rising spirit of innovation and its desire to improve. Special thanks to the Process Group Leader: Claudia Alquicira Esquivel and the Process Specialist: Hugo García Figueroa. As well as to all the work team members involved in the PAL projects. We also acknowledge the assistance of

the following projects: PEGASO-MAGO (TIN2009-13718-C02-01, FEDER and MEC of Spain) and Agreement Unicauca-UCLM (4982-4901273). Francisco J. Pino acknowledges to the University of Cauca where he works as Assistant Professor.

References

1. Glazer, H., Dalton, J., Anderson, D., Konrad, M., Shrum, S.: CMMI or Agile: Why Not Embrace Both! CMU/SEI-2008-TN-003 (2008)
2. Pino, F., Pedreira, O., García, F., Rodríguez, M., Piattini, M.: Using Scrum to Guide the Execution of Software Process Improvement in Small Organizations. Journal of Systems and Software 83(10), 1662–1677 (2010)
3. Dyba, T., Dingsoyr, T.: Empirical studies of agile software development: A systematic review. Inform. Softw. Technol. (2008)
4. Cockburn, A., Highsmith, J.: Agile software development, the people factor. Computer 34(11), 131–133 (2001)
5. Boehm, B., Turner, R.: Balancing Agility and Discipline - A Guide for the Perplexed. Pearson Education, London (2004)
6. Lepmets, M., Nael, M.: Balancing scrum project management process. In: Sillitti, A., Martin, A., Wang, X., Whitworth, E. (eds.) XP 2010. Lecture Notes in Business Information Processing, vol. 48, pp. 391–392. Springer, Heidelberg (2010)
7. Boehm, B., Turner, R.: Management Challenges to Implementing Agile Processes in Traditional Development Organizations. IEEE Software, 30–39 (2005)
8. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit. Addison-Wesley, Reading (2003)
9. Lean Software Institute, `http://www.leansoftwareinstitute.com/`
10. Poppendieck, M., Poppendieck, T.: Implementing Lean Software Development: From Concept to Cash. Addison-Wesley, Reading (2006)
11. Appelo, J.: Complexity vs. Lean, the Big Showdown. In: Lean Software and Systems Conference 2010, Helsinki (2010)
12. Ultrasist, `http://www.ultrasist.com.mx/`
13. Schwaber, K., Beedle, M.: Agile Software Development with SCRUM. Prentice Hall, Englewood Cliffs (2001)
14. Beck, K.: Test-Driven Development by Example. Addison-Wesley, Reading (2003)
15. Dávila, M., Oktaba, H.: Especialización de MoProSoft basada en el Método Ágil Scrum (MPS-Scrum). In: SIS 2009 (2009) (in press)
16. Jakobsen, C., Johnson, K.: Mature Agile with a twist of CMMI. In: Agile 2008 Conference, pp. 212–217 (2008)
17. Larman, C., Vodde, B.: Practices for Scaling Lean & Agile Development. Addison-Wesley, Reading (2010)
18. Larman, C.: Agile & Iterative Development: A Manager's Guide. Addison-Wesley, Reading (2004)
19. Coplien, J., Bjornvig, G.: Lean Architecture. Wiley, Chichester (2000)
20. Salo, O., Abrahamsson, P.: Integrating Agile Software Development and Software Process Improvement: a Longitudinal Case Study. In: International Symposium on Empirical Software Engineering 2005, pp. 193–202 (2005)
21. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Dordrecht (2000)
22. Shull, F., Singer, J., Sjoberg, D.: Guide to Advanced Empirical Software Engineering. Springer, Heidelberg (2010)

Adopting Agile Practices in Teams with No Direct Programming Responsibility – A Case Study

Kirsi Korhonen

Nokia Siemens Networks,
Hatanpäänvaltatie 30, 33100 Tampere, Finland
kirsi.korhonen@nsn.com

Abstract. To respond to the need for flexibility in reacting to customer needs, the agile practices have been introduced in the software development organizations. In order to get full benefit of agile, it is proposed that the agile and lean practices should be adopted organization wide. Agile practices are mainly used by programmers, but in large companies, there can be different work roles and tasks which do not directly include programming of the product but are supporting the software development in the system level, such as system level testing and testing environment maintenance. In this study, the goal was to provide information on the progress of agile transformation in teams with no direct programming responsibility in a large-scale, distributed software development organization. A survey was done to collect data during the first year of agile adoption about agile practices utilisation within three different teams: the developers, the system level testers and people in test laboratory support. The results show that certain agile practices were adopted in the teams with no direct programming responsibility, and there were differences between the teams.

Keywords: agile software development, agile transformation, distributed development, large scale.

1 Introduction

Changing the software development method from plan-based to Agile is a challenge to any organization. Especially a large ([14], [19]) and distributed ([16], [12]) organization has been reported as one of the issues requiring further attention in agile adoption. In large organization the change may take time, and the lack of visibility to the changes may cause e.g. motivation decrease [13]. In order to get full benefits out of the agile, everyone in the organization should work in agile and lean way [21]. But the goal of agile transformation for the whole organization might be unclear as the agile practices are mostly used by software developers. So how should the agile practice adoption proceed among teams with no direct programming responsibility?

In this study, the goal was to explore how the adoption of agile practices progresses within team that do not have direct programming responsibility in a large, distributed software development organization. The effect was studied by comparing the progress of agile adoption of two teams with no programming responsibility with progress of a team that had direct programming responsibility. The main sources of

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 30–43, 2011.
© Springer-Verlag Berlin Heidelberg 2011

the data were two surveys conducted with six months interval after the transformation was started, and information collected from feedback discussions after both surveys.

The analysis showed that both the groups which were not directly involved in programming were adopting agile practices related to daily organization of the work (e.g. short iterations), and team work (e.g. Scrum). The team responsible for programming was most advanced in agile practice adoption, and in addition to the daily work and team practices, they adopted also the more technical, programming related practices (e.g. refactoring). The encountered problem areas varied between the teams, and were closely related to the work responsibility area of the team.

Based on the results it is proposed that while planning the agile transformation it is essential that an organization analyses the expected outcome of agile transformation within different teams in the organization, and defines the goals for the transformation accordingly. It does make a difference on what practices to adopt if the team has or has not direct programming responsibility, and it is good to share that expectation with the teams. Additionally workshops are recommended to go through the possible problem areas during the early stage of the agile transformation to clarify unclear issues, and to agree on the way forward. Team involvement is essential when planning the goals for the transformation, as it makes the change more concrete and helps to follow up the progress which in turn improves the motivation and supports people in their transformation.

The rest of this paper is organized as follows: Section 2 provides background information for agile transformation in general, and the research setup is described in Section 3. The empirical results are presented in Section 4 for the agile experience, for the adoption of agile practices in Section 5 and in Section 5 for the identified problem areas. The results are discussed in Section 6, and the paper is concluded with final remarks.

2 Background

2.1 Characteristics of Plan-Driven and Agile Development

Based on the summary on the advantages and disadvantages of the different development models done by Petersen and Wohlin [19], in plan-driven models the main cause for failure is related to poor requirement handling. According to their analysis, this has resulted as problems e.g. in integrating the overall system in the end and testing it [11]. Making any changes in later phases is very difficult, which has lead to a situation where customers' current needs are not addressed by the end of the project [9], and many of the features implemented are not used [10].

To respond more flexible to the customer needs, Agile methods propose to work in short (max 4 weeks), time-boxed iterations [24] [21], where the customer involvement is mandatory both in the beginning, and at the end. The purpose is to receive timely feedback both on the requirements and the end result of the time-boxed software development. This makes it easier to implement the changes, and improves the customer satisfaction [3], [26].

Though there are clear advantages in agile methods, there are also some issues as listed by Petersen and Wohlin [19]: architecture usually does not have enough attention [15], [22], and the agile development does not scale well [5]. Additionally, the

implementation of continuous testing is difficult due to different platforms and system dependencies [27], and testing can even become a bottleneck due to the need of testing and development ongoing at the same time [29].

2.2 Agile Transformation

Initial experiences with the adoption of agile practices can be tough, as problems become painfully visible and changes are required in the working habits and mindsets of individuals, teams and organizations alike. The changes do not take place overnight, and it is tempting to blame the agile methodology for all and any problems [4]. This might even result as abandoning the new practices that challenge the team most [2]. Especially in a large multisite organization there can be additional challenges caused by for example dependencies between teams on different sites [16], problems with communication [20,8], or misunderstandings on defect management practices [9].

Agile practices are reported to help improve product quality. For example results of the case studies at Microsoft and IBM [17] indicate that the TDD practice improved code quality. A set of agile practices was selected for a pilot at Ericsson [1] instead of trying to implement a completely agile process. These practices included pair programming, collective code ownership, the planning game and the on-site customer. At Primavera [23], basic agile practices, including cross-functional, self-managed teams and time-boxed iterations, were applied and the organization experienced increase in quality which was further improved with TDD implementation. However, these studies focus on the impact to developers work, and not on the impact of the agile adoption on other groups related to the software development.

There are structured approaches available to guide organizations in taking the agile practices into use. These include e.g. assessing organizational readiness for agile transformation [25], and identifying the set of agile practices and tools to be taken into use [25], [18], and [12]. These practices are presented in general level and not specifically to some group of people with certain responsibility. The main reason might be that the basic assumption in agile is that the scrum teams are self-organizing, co-located teams with all needed competencies available within the team [24]. In large organizations, where the system is big and there are several layers of integration needed, there are also other groups of people involved in the software development than just the code developer scrum teams.

Another survey made at Nokia Siemens Networks [28] summarizes that “basic” agile practices include short time-boxed iterations [24], product backlog [24], continuous integration [7], retrospectives [24] and self-organizing teams [18]. “Intermediate” agile practices add to basic practices refactoring [7], TDD, or acceptance test driven development (ATDD) [7], or tests written at the same time as code. More practices than that would be equated to a “fully agile” organization. In large organizations, getting to the fully agile level can take a few years [28, 2], and patience is required.

3 Research Design

This research was done as an industrial case study [6], and the main source of information were two surveys which were conducted in the organization. Additional information was collected from feedback sessions after the surveys.

3.1 Context

The organization in the study was large, with more than 150 experts working in the studied projects, distributed in several locations globally. Most of the people in the organization had several years of experience from traditional software development, and some had even previous experience from agile and lean software development gained from working in other parts of the company. The software development projects produced new versions of an existing product in the telecommunications domain.

The plan-driven approach used in the study organization before agile transformation reflects the main characteristics of the so called Waterfall-model [19]. There was a detailed plan done for the whole software development phase consisting of three main phases: specification, development and system level testing. The desired functionality of the software and the architecture were specified in high detail in the beginning. Any change afterwards required strict change process management. Programming and testing work in development phase were done by the people in the Developer team, and the system level testing was done in the end of the project by the SLT team. Support group provided laboratory support for both of these teams. The issues were known disadvantages to waterfall development model [19]: long cycle time to respond to customer needs and integration problems.

Based on the analysis of existing issues, a set of agile practices were selected and introduced organization wide. These practices included e.g. short time-boxed iterations, product backlogs, self-organizing teams and basic scrum team practices. The Scrum team practices included e.g. sprint planning sessions, demo at the end of the sprint, retrospectives and team level sprint backlogs.

3.2 Research Questions

The agile practices are mostly utilised by people directly involved with the software code development. On the other hand, in order to work to its full benefit, the whole organization needs to be agile [21]. When the agile adoption started, some of the selected practices were more targeted to developers, such as TDD, but there were also non-technical practices such as retrospectives and scrum, which could be taken into use even if the team does not have programming responsibility.

The primary goal in this study was to explore agile practise adaptation in teams which have no direct programming responsibility, and to compare the results with teams having the programming responsibility. The team with direct programming responsibility is called the Developers. The developers are responsible for implementation, programming and end to end functionality testing of their component in the software system. The teams with no direct programming responsibility are from system testing organization, and from an organization providing laboratory support for the other two organizations. The product in question has different levels of integration, and SLT people are verifying that the whole software system works as one entity once individual components have been integrated together. They also verify that the non-functional requirements (for example system level security or redundancy) have been fully implemented and are working according to expectations. People in the Support group are responsible to maintain the environments for the other two groups and thus support the software development and verification activities. These three

teams were selected as they work very closely together, and the assumption is that if one group changes their way of working, it has an impact on the other two as well.

The practices are divided into three groups: 1) Daily work practices, 2) Team practices and 3) Programming practices. Daily work practices are non-technical practices which set up the boundaries how the daily work of the team should be organized: time-boxed iterations, user stories and product backlog. Team practices are non-technical team level practices: retrospectives, Scrum and self-organized teams. Finally, the Programming practices are those more technical, code development related practices: continuous integration, tests written same time as code, refactoring, collective code ownership, pair-programming and TDD.

The first hypothesis of this study is that the teams with direct programming responsibility would proceed well in adopting the practices in all of these three groups of practices during the first year of agile transformation. The second hypothesis is that there is some progress in adopting the Daily work practices and team level practices also in the teams with no direct programming responsibility, but not so much progress is taking place with adopting the Programming practices.

To explore and compare the impact of the agile practice adoption between these three groups, the following research questions were defined: 1) how did the agile practices implementation proceed in the organization during the first year in the different groups, 2) was there a difference in adopting the practices between the teams who have and who do not have direct programming responsibility.

Third hypothesis was that the change in the working practices would have a clear impact on the daily work, and the biggest problems would be related to the early phase of the agile adoption indicating that practices are not familiar or not fully utilised. Due to the different responsibility areas of the teams, the encountered problems in the agile practice adoption might be of different nature. To explore this further, and to understand better the progress of agile adoption in the three study teams, a third research question was added: 3) what were the biggest problems in the teams during the agile transformation.

3.3 Data Collection and Analysis

While the agile transformation proceeded, the management wanted to collect feedback on the progress from the project personnel with a survey. The survey consisted of open-ended questions and closed-ended questions offering a number of defined response choices.

From the survey, four questions were analysed for this study. First, to analyse the progress of agile adoption within the teams, the following survey questions were analysed: 1. How long experience do you have with agile development? (Response choices: I don't have any experience, 1-3 months, 4-6 months, 6-12 months, 1-2 years, 3-4 years, more than 4 years.) 2. What Agile methods and practices are used in your team OR in your product currently (select all options that are in use to your knowledge): Short (max. 4 weeks) time-boxed iterations, Continuous Integration (min. daily build), Self-organized, co-located, cross-functional teams, Product backlog, TDD (Test Driven development), Tests written at the same time as code, Refactoring, Pair-programming, Collective code ownership, Scrum, User stories, Retrospectives.

Second, to find out the problems in the teams related to the agile practice adoption, two open-ended questions were added to the survey: 3. What are the biggest obstacles or challenges in your agile transformation? and 4. What do you think are the biggest wastes currently in your work? (waste = activity which does not bring value to customer or enhance our learning).

The survey was conducted twice, with 6 months interval, in two main locations of the unit, as the personnel in those locations had experience on both the traditional projects and the new way of working. There were no big changes in the personnel on these two locations during the study period. From the first survey, 96 responses were collected, and from the latter one 122. After both surveys, a feedback session was arranged in the organization to go through the results, and to collect further data on the possible reasons behind the results. The comments from the discussions were recorded, and were used together with the responses to open-ended questions to provide further insight to the results.

The survey responses were analysed separately for the Developer team, the system level testing (SLT) team and the Support team. Survey results were analysed in detail to draw comparisons between the teams and over the time. The results were summarized from open-ended questions and from the notes done during the feedback discussions with individuals and teams after the surveys. Due to the qualitative nature of the data no separate statistical analysis was done.

3.4 Threats to validity

Majority of the survey results are based on opinions, which as such do not provide clear facts on the progress and changes due to agile transformation. On the other hand, opinions of the people in the transformation cannot be neglected. Even if all metrics would provide evidence on the successful agile transformation, it is not really a success if the people in the organization do not see the change and are not satisfied with the results.

The impact of the agile transformation can differ from the results in this paper due to few reasons. One important issue to consider is the procedure the agile practices are adopted within the organization. The organization in this study started from practices, which are more focusing on the team practices instead of coding practices. If the first thing which is introduced to the organization is related to coding practices, TDD, as in [17], the results among the different groups might be different.

4 Agile Experience in the Organization

Before analysing the impact of practice adoption in the teams, a background study was made to analyse the previous agile experience level in the teams. The goal was to evaluate the starting point for the agile practice adoption, did the teams have previous experience on agile practices, or were the teams starting the adoption from the scratch. The agile experience is measured in months and the responses from the survey have been grouped into three categories: less than half a year experience, 6-12 months of experience, and over 12 months of experience. The experience is presented separately for the Developer team and SLT team in figure 1, and, respectively, for the Support team in figure 3.

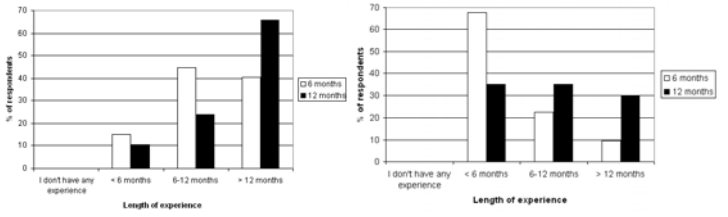


Fig. 1. Agile experience in the Developer team (chart on left) and SLT team (chart on right) 6 months (white bars) and 12 months (black bars) after starting the agile transformation

In the Developer group there were some people with previous agile experience before the agile transformation officially started, as over 40% of the respondents reported to have at least one year experience on agile practices after six months of the transformation start. Majority of the people though reported to have less than 12 months of experience, and there was no one with any agile experience at all. 12 months after starting the agile adoption, the highest number of responses is at over 12 months of experience, which is an expected result as the transformation had been ongoing half a year since the previous survey.

In the SLT group, there was no one claiming to have no agile experience at all. Majority of the people reported to have 6 months or less experience in agile at the six months' measurement point. The adoption of agile practices has proceeded in next six months, as at twelve moths' measurement point more people report to have 6-12 months of experience.

When comparing the results from people in the Support group with results in Developers group, the difference is even bigger (figure 2). At six months measurement point, 30% of respondents claim to have no experience of agile, and after 12 months, still nearly 20% felt they had no experience at all in agile. This result comes even if the people in the Support group are working directly with the people in Developer and SLT groups. But some experience is developing also within the Support group as there are more people at 12 months measurement point (23,5%) responding to have 6-12 months of experience than at 6 months measurement point (11.8%).

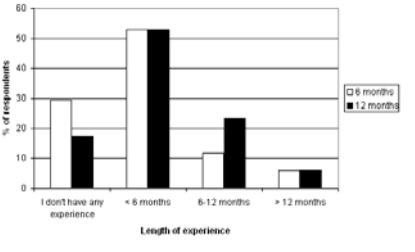


Fig. 2. Agile experience in the Support team 6 months (white bars) and 12 months (black bars) after starting the agile transformation

5 Adopting the Agile Practices

The responses about practices in use are analysed separately for the three practice groups: Daily work practices, Team practices and Programming practices. The results are presented for the Developers team in figure 3, SLT team in figure 4 and respectively, for the Support team in figure 5.

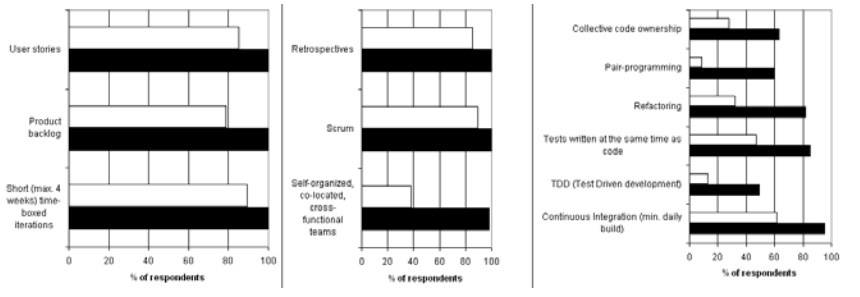


Fig. 3. Agile practices in use in the Developer team 6 months (white bars) and 12 months (black bars) after starting the agile transformation. First chart: Daily practices, second chart: Team practices and third chart: Programming practices.

As displayed in figure 3, during the first six months (white bars) after starting the agile adoption, all the Daily work practices, user stories (85%), product backlogs (79%) and iterations (89%), and most of Team practices, scrum (89%) and retrospectives (85%) - were taken widely in use in the Developers group. Adopting the Programming related practices was also progressing with e.g. continuous integration (61.7%) and tests written same time as code (46.8%). One key element, self-organizing teams, was in use according to less than 40% of the respondents, and further development with Programming related practices would be expected to get full benefit of being in agile.

In the SLT team (figure 4), the adoption of Daily work practices had progressed by taking the product backlog (59.4%%) and iterations (62.5%%) into use. User stories were not that much in use yet (31.2%). From team practices, the scrum and retrospectives were more widely in use, but again, the self-organizing teams was not reported to be as much in use (34.4%). As expected, Programming related practices were not in use.

During the next six months the situation had clearly advanced in both Developer and SLT teams. Time-boxed iterations, retrospectives, scrum teams, product backlog and user stories were reported to be used by all in the Developers group, and 98% in the SLT group. According to the responses, self-organizing teams were also in place after 12 months of agile transformation.

After 12 months of starting the agile transformation, the respondents in the Developer team utilized more practices which support directly the software development itself, such as refactoring, pair programming and test-driven development (TDD). These practices are also reported by the system level verification group (figure 4), but great utilization of these practices was not even expected as the practices supporting code development are not used in the system level testing as such. But these can be

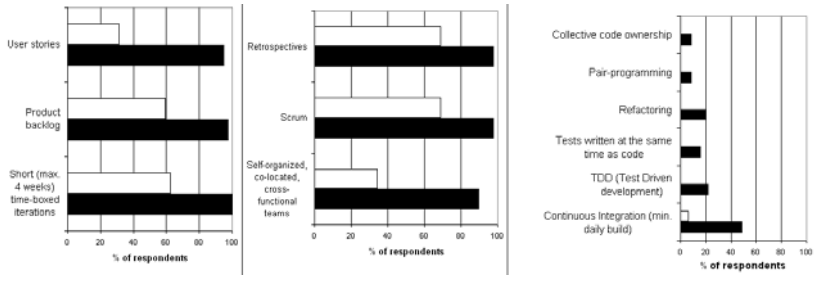


Fig. 4. Agile practices in use in the SLT team 6 months (white bars) and 12 months (black bars) after starting the agile transformation. First chart: Daily practices, second chart: Team practices and third chart: Programming practices.

used e.g. when preparing test automation. Getting continuous integration into use on the software development project level had improved significantly during the latter 6 months which is a clear sign that the organization was progressing to be more agile than just adopting the very basics.

Among the people in the Support team there were comments that agile practices do not fit to supportive work, because it is meant for software programming and the work in the Support team is too different from that. This is visible also from the results (figure 5), where the team level practices are being used quite well 6 months after starting the agile transformation, but the Daily work practices and Programming practices are reported to be in use only by a few numbers of users. Interesting detail is that the self-organizing teams is more in use in the Support team (58.8%) than in the other two teams after 6 months.

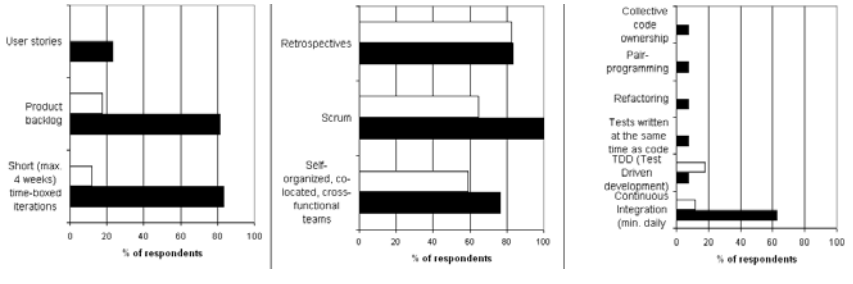


Fig. 5. Agile practices in use in the Support team 6 months (white bars) and 12 months (black bars) after starting the agile transformation. First chart: Daily practices, second chart: Team practices and third chart: Programming practices.

After 12 months, the agile practice adoption has made significant progress in the Support team. Scrum is in use by all respondents, and working in short, time-boxed iterations have been taken into use according 80% of the respondents compared to around 10% after 6 months. Same rapid development goes with the product backlog from 17.6% 6 months after to 81.2% 12 months after starting the agile transformation. So it seems that the support team was able to find their way of working with the agile, and they adopted daily work practices, except user stories, and all Team practices

6 Feedback on the Agile Transformation

In the survey there were questions included to evaluate what were the biggest obstacles or concerns due to the agile transformation in respondent's work. The main findings varied between the teams. The main findings were selected from the survey responses when the same topic was mentioned by several people, and the percentage indicates how many of the respondents mentioned the topic.

6.1 Developer Team

At the developer side, one concern was related to collective code ownership (8%). With collective code ownership, the code development would be on the whole team's responsibility, and some people were concerned that the code ownership would be lost resulting so that nobody would care about the good quality of the code. During the feedback sessions one respondent proposed that collective code ownership would still require some people to steer the code development and be responsible for that.

Second concern was related to insufficient specification, requirements and user stories (17%). This is clearly a result of change in the way of working. Previously, in plan-based software development there were detailed specifications done before the implementation was started. In agile mode, the high level plans were available, but the details were added later on, when the development was ongoing and the requirements at that moment were known. The idea in agile is to decide as late as possible so that the implementation can be aligned with the latest information on the required content. The survey responses reflect this change, as there were comments that the specification was not detailed enough, and requirements were insufficient. Also the big plan and long term visibility was not clear to everyone. One proposal was that guidelines on how to do the implementation would be still required, for example what third party components can be used. Additionally there were respondents who felt that the user stories were written in a too generic way, making it difficult to have realistic estimations on work effort needed.

6.2 System Level Testing Team

In the SLT group the main finding (reported by 28 %) was the increased amount of new meetings and need for status reporting. Implementing scrum practices does bring a set of new meetings - planning meetings in the beginning of the sprint, retrospectives, demos and daily scrum meetings, which in the beginning might feel overwhelming compared to the previous way of working. The point of these meetings is to plan and to solve impediments, not status reporting. But it may take time to find out how these can benefit the team in question.

Some respondents were concerned that the testing procedures in agile were not clear for them in the beginning (25%). During the feedback discussions it came up that some respondents saw that the length of the sprint created bigger pressure to get testing done in shorter time than before. Some testers felt that there were too many test cases to be run manually in this short time, and there was a proposal that test automation should be enhanced to speed up the testing further.

One concern among the testers was the parallel work, where code development and testing is done at the same time (15 %). Even though the SLT team installed the software as soon as it was available, by the time the installation was completed, the software was already "too old". During the installation time, the development teams had developed new code, implemented new functionality and fixed bugs in the software.

6.3 Support Team

Analysis of the responses from the Support group revealed that there were quite many who felt that the agile practices do not fit to the lab support kind of work (39% of the respondents). Some respondents were concerned on how should the practices be implemented when the support requests are coming in randomly, and the support work is usually needed immediately (17%).

Based on the feedback during the survey result discussion sessions with the Support group, there were two solutions identified, which helped the team to go forward with agile adoption. At first the people felt that using the general backlog tool was not really bringing any additional value. But later on they found that the tool they had been already using for maintaining the support requests in fact was already serving them as a backlog tool, and only minor enhancements were needed to the tool. Second solution was related to problem with sprint cycle. The sprint length made the sprint planning an impossible task as the support requests were pouring in daily with critical status. The sprint length was changed to be only one week for the support team, which helped the people in this group to plan their work for the sprint and keep the commitments.

7 Discussion

By implementing the agile practices, the organization started a big change process. According to the results, all three study teams: Developers, System level Testers and Support team, were adapting to agile way of working. Though the adoption proceeded in all the groups, there were differences. One year after starting the agile transformation, the Developer team was most advanced in all the practices with 100% utilization of the basic practices [28] and over 40% of the more technical. Based on this result the Developer team is above the Intermediate level of agile adoption. This is an expected result as the agile practices are well suited for the programming tasks.

SLV team implemented well the Team level and Daily work practices having almost 100% utilization of all the basic practices. In the support group, the scrum teams, time-boxed iterations, retrospectives and product backlog were seen most useful, and were reported to be in use by at least 80% of the team members.

Based on these results, the teams with no direct programming responsibility would be able to adopt the agile practices related to organizing the daily work and team practices. On the other hand, some practices especially related to programming, will not be useful at all thus there would be no progress to adopt those. Therefore the definition of when a group of people is fully agile could be tied to practices which are relevant or applicable for the team to adopt. In this case, the SLT and Support teams could be considered to be fully agile when they have in use all the Team practices and

Daily work practices as defined in this study. For the Developer team, being fully agile would mean adopting practices from all three groups: Daily work practices, Team practices and Programming practices.

The encountered problems in the teams were different and closely related to their specific responsibility area at work. The developers were concerned if the collective code ownership would result as quality drop, and the change in specifying the work was unclear. In System level testing team the main concern was the increased number of meetings. Additionally the time-boxed iterations made more pressure to run the testing. The biggest concern in the Support team was that how can they take agile practices into use when the practices are not meant for the kind of work they were doing.

Based on these results it is proposed that any organization planning to start the agile transformation would go through the different teams involved in the software development, and align the expectation of agile transformation based on whether or not the team has direct programming responsibility. This study has provided information on agile practices to be adopted within the different teams as well as some examples of possible obstacles and challenging areas for Developers, Testers and Support team. It is proposed that the possible areas of concern are separately discussed with the teams e.g. in workshops during the initial starting phase of the agile transformation to agree on the way of working and to help the teams proceed in their transformation more smoothly towards the team specific goals.

8 Conclusions and Future Work

The contribution of this paper lies in providing empirical evidence on how adopting agile methods proceeds in teams with no direct programming responsibility. In this paper the agile adoption is analyzed from the work responsibility area point of view for three different groups: developers, system level testers and lab support people. As expected, the developer team had adopted wider range of practices but the agile adoption was progressing also in the teams with no direct programming responsibility

Based on the results it is proposed that any organization planning to start agile adoption would agree on agile adoption goals for the individual teams with respect to whether the team has direct programming responsibility or not. These goals would clearly define what practices can be expected to be adopted by the team. Many of the practices can be adopted even if the team is not directly writing the code, and to support these teams in their agile transformation it is good to make the expectations for the practice adoption clear from the beginning.

In future research the survey results are studied in more detail to analyze whether there are further differences between these groups, for example on the perception on the change in code quality or impact of agile compared to traditional way of working.

Acknowledgements. The author thanks Kai Koskimies, Erik Hiltunen and Hannu Korhonen for comments, and Nokia Foundation for funding the manuscript writing.

References

1. Auvinen, J., Back, R., Heidenberg, J., Hirkman, P., Milovanov, L.: Improving the engineering process area at Ericsson with Agile practices. A Case Study. TUCS Technical report No 716 (October 2005)
2. Benefield, G.: Rolling out Agile in a Large Enterprise. In: Proceedings of the 41st Annual Hawaii International Conference on System Sciences, p. 462 (2008)
3. Ceschi, M., Sillitti, A., Succi, G., Panfilis, S.D.: Project management in plan-based and agile companies. IEEE Softw. 22(3), 21–27 (2005)
4. Cloke, G.: Get your agile freak on! Agile adoption at Yahoo!Music. In: Agile 2007: Proceedings of the AGILE 2007. IEEE omputer Society, 240–248 (2007)
5. Cohen, D., Lindvall, M., Costa, P.: An introduction to agile methods. In: Advances in Computers. Elsevier, Amsterdam (2004)
6. Yin, R.: Case Study Research: Design and Methods. Sage Publishing, Thousand Oaks (1994)
7. Crispin, L., Gregory, J.: Agile testing. A Practical guide for testers and agile teams. Addison-Wesley, Reading (2009)
8. Herbsleb, J.D., Mockus, A., Finholt, T.A., Grinter, R.E.: Distance, dependencies, and delay in a global collaboration. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 319–328. ACM Press, New York (2000)
9. Jarzombek, J.: The 5th annual jaws s3 proceedings (1999)
10. Johnson, J.: Keynote speech: build only the features you need. In: Proceedings of the 4th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2002) (2002)
11. Jones, C.: Patterns of software systems: failure and success. International Thomson Computer Press, Boston (1995)
12. Korhonen, K.: Migrating defect management from waterfall to agile software development in a large-scale multi-site organization: A case study. In: Abrahamsson, P., Marchesi, M., Maurer, F. (eds.) Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing, vol. 31, pp. 73–82. Springer, Heidelberg (2009)
13. Korhonen, K.: Exploring defect data, quality and engagement during agile transformation at a large multisite organization. In: Sillitti, A., Martin, A., Wang, X., Whitworth, E. (eds.) XP 2010. Lecture Notes in Business Information Processing, vol. 48, pp. 88–102. Springer, Heidelberg (2010)
14. Larsson, A.: Making sense of collaboration: the challenge of thinking together in global design teams. In: Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work, pp. 153–160. ACM Press, New York (2003)
15. McBreen, P.: Questioning extreme programming. Pearson Education, Boston (2003)
16. Misra, S., Kumar, U., Kumar, V., Grant, G.: The organizational changes required and the challenges involved in adopting agile methodologies in traditional software development organizations. In: Digital Information Management, pp. 25–28 (2006)
17. Nagappan, N., Maximilien, e.M., Bhat, T., Williams, L.: Realizing quality improvement through test driven development:results and experiences of four industrial teams. Empir. Software Eng. 13, 289–302 (2008)
18. Nerur, S., Mahapatra, R., Mangalaraj, G.: Challenges of migrating to agile methodologies. Communications of the ACM 48(5) (2005)
19. Petersen, K., Wohlin, C.: The effect of moving from a plan-driven to an incremental software development approach with agile practices. An industrial case study. Empir. Software Eng. 15, 654–693 (2010)

20. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J.: The impact of agile practices on communication in software development. Empir. Software Eng. 13, 303–337 (2008)
21. Poppendieck, M., Poppendieck, T.: Lean Software development. Addison-Wesley, Reading (2007)
22. Stephens, M., Rosenberg, D.: Extreme programming refactored:the case against XP. Apress, Berkeley (2003)
23. Schatz, B., Abdelshafi, I.: Primavera Gets Agile: A Succesful Transition to Agile Development. Software 22, 36–42 (2005)
24. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice-Hall, Englewood Cliffs (2002)
25. Sidky, A., Arthur, J.: A disciplined approach to adopting agile practices: the agile adoption framework. In: Innovations in Systems and Software Engineering, pp. 203–216. Springer, Heidelberg (2007)
26. Sillitti, A., Ceschi, M., Russo, B., Succi, G.: Managing uncertainty in requirements: a survey in documentation-driven and agile companies. In: Proceedings of the 11th IEEE International Symposium on Software Metrics (METRICS 2005), p. 17 (2005)
27. Svensson, H., Höst, M.: Introducing an agile process in a software maintenance and evolution organization. In: Proceedings of the 9th European Conference on Software Maintenance and Reengineering (CSMR 2005), pp. 256–264 (2005)
28. Vilkki, K.: Impacts of Agile Transformation. Flexi Newsletter (January 2009)
29. Wils, A., Van Baelen, S., Holvoet, T., De Vlaminck, K.: Agility in the avionics software world. In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 123–132. Springer, Heidelberg (2006)

Proposing an ISO/IEC 15504-2 Compliant Method for Process Capability/Maturity Models Customization

Jean Carlo Rossa Hauck[1,2], Christiane Gresse von Wangenheim[3], Fergal Mc Caffery[2], and Luigi Buglione[4]

[1] Graduate Program in Knowledge Engineering and Management - Federal University of Santa Catarina – Brazil
[2] Regulated Software Research Group & Lero - Dundalk Institute of Technology – Ireland
[3] Graduate Program in Computer Science (PPGCC) - Federal University of Santa Catarina – Brazil
[4] ETS/Engineering.IT - Rome - Italy
{jeanhauck,gresse}@gmail.com,
fergal.mccaffery@dkit.ie, luigi.buglione@eng.it

Abstract. The customization of software process capability/maturity models (SPCMMs) to specific domains/sectors or development methodologies represents one of the most discussed and applied trends in ICT organizations. Nonetheless, little research appears to have been performed on how theoretically sound and widely accepted SPCMMs should be developed to high quality. The aim of this paper is therefore to elicit the state-of-the-art regarding the processes adopted to develop such models and to propose a systematic approach to support the customization of SPCMMs. Such an approach is developed based on ISO/IEEE standard development processes integrating Knowledge Engineering techniques and experiences about how such models are currently developed in practice. Initial feedback from an expert panel indicates the usefulness and adequacy of the proposed method.

Keywords: Maturity Models, Standards, Knowledge Engineering, SPCMM, ISO/IEC 15504.

1 Introduction

Various Software Process Capability/Maturity Models (SPCMMs [1]) have been developed by the software engineering community, such as, CMMI-DEV [2] and ISO/IEC 15504 [3], and their use for software process improvement and assessment is well established in practice. These generic models have been customized to specific contexts [4] because diverse software development domains have specific process quality needs that should be addressed. Regulated software development domains have specific standards, such as in health care, which must be covered by the software development process in order to provide the necessary alignment to these domain-specific standards. Consequently, there is a current trend to the development of customizations of those generic process models for specific domains, such as SPICE4SPACE [5], OOSPICE [6] Automotive SPICE [7], etc. Despite this trend, most of the SPCMMs customization initiatives do not adopt a systematic approach for

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 44–58, 2011.
© Springer-Verlag Berlin Heidelberg 2011

the customization of those generic standards and models [8]. Furthermore, literature detailing how SPCMMs are developed / evolved / adapted is also extremely rare [9]. Standardization organizations, like ISO or IEEE, define high-level generic processes for developing and publishing standards. However, they do not describe how to customize existing models or provide detailed technical support for the specific development of SPCMMs. The contribution of this paper is the proposal of a method for customization of SPCMMs, based on an analysis of how existing customizations have been performed, integrating standard development procedures from a Knowledge Engineering viewpoint and aligned to the requirements of ISO/IEC 15504-2 for Process Reference Models (PRM) and Process Assessment Models (PAM).

In section 2, the requirements for SPCMMs are presented. Section 3 presents methods for SPCMMs development. In section 4, the method is proposed, and section 5 presents the first results from its pilot application. Conclusions are presented in section 6.

2 Requirements for SPCMMs

Different sets of requirements have been proposed for models expressing the capacity and/or maturity of processes. Becker et al. [10] propose seven criteria including: (i) comparison with existing models, (ii) iterative development, (iii) model evaluation, (iv) multi-methodological procedure, (v) identification of the relevance of problem, (vi) problem definition, (vii) published results and (viii) scientific documentation; based on the guidelines for design science. According to Matook & Indulska [9], reference models for software process quality must meet the following requirements: generality, flexibility, completeness, usability and comprehensibility.

Regarding their structure, generally speaking, software process capability maturity models (SPCMMs), have different characteristics. Lahrmann & Marx [11] propose a basic rationale of the structural characteristics of this type of model, as shown in Table 1.

Table 1. SPCMMs characteristics [11]

Criteria	Characteristics		
Dimensions	One-dimensional	Multidimensional	Hierarchic
Representation	Continuous	Staged	
Audience	Unique	Multiple	
Assessment approach	Qualitative	Quantitative	

The requirement that characterizes the structure of software process standards and reference models (to enable classification as SPCMMs) is the fact that they have at least two dimensions: the process and dimension of capability/maturity dimension.

In a more specific way than these generic requirements, the ISO/IEC 15504-2 standard [3] establishes specific requirements for the development dimensions of the process (PRM – Process Reference Model) and capacity (PAM – Process Assessment Model) of SPCMMs, which can be summarized as follows:

PRM

- **R1** - Declaration of the specific domain and community of interest, including aspects of consensus achievement;
- **R2** - Description of processes including: unique title, purpose and outcomes;
- **R3** - Presentation of the existent relationships between processes;

PAM

- **R4** - Statement of scope and coverage of the model;
- **R5** - Indication of the capability levels selected from a measurement framework for the processes, starting at level 1;
- **R6** - Mapping for the selected processes of the chosen PRM(s) ;
- **R7** - Details of performance indicators of the processes, mapped to the purposes and outcomes of selected the processes of PRM(s);
- **R8** - Detailed process attributes of measurement framework;
- **R9** - Objective evidence that the requirements are fulfilled.

The next section attempts to identify approaches that can possibly meet the existing proposed requirements for a SPCMM.

3 Existing Methods for SPCMMs Customization

This section presents three perspectives in an attempt to establish an overview of the development and customization of SPCMMs: proposals for approaches that support this type of development, process of standards development and main steps and techniques used in practice.

3.1 Existing Methods for the Development of Capability/Maturity Models

Although diverse software process capability/maturity model customizations have already taken place [8], research on how to perform such customizations in a systematic way is sparse. One of the few works in such direction was proposed by de Bruin et al. [14], introducing a six-step sequence for the development of Maturity Assessment Models. Although their work considers specific domain needs, it does not address in detail the customization of domain-specific best practices from generic models.

Mettler [15] performs a deeper analysis on the fundamentals of process maturity models, putting the main phases described in [14] under a design science research perspective. In this context, the phases are compared to a model user perspective of the maturity models, indicating a need for more formal methods and studies. Maier, Moultie & Clarkson [16] define a guide for development of Maturity Grids that consists of tools to assess the required abilities of an organization to deliver a product or a service. The purpose of the guide covers a wider range of models, not focusing on SPCMMs.

Salviano et al. proposed the generic framework PRO2PI [17] for the development of process capability/maturity models, based on the authors' previous experiences of developing diverse models with a 7-step process. However, no details are provided in relation to the research activities and techniques that would be required to provide support for the customization of SPCMMs. Matook and Indulska [9] proposed a QFD-based approach for reference modeling incorporating the voice-of-the-reference-model users, presenting a measure for the quality of such models. Becker et al. [10]

also proposed a general process for the development of Maturity Models that aim to cover a set of defined requirements. However, the work did not address the question of the evolution of the model after its publication.

As it can be seen, from the few existing approaches for developing models of maturity and/or process capability, none of them is specifically targeted to meeting the requirements of the ISO/IEC 15504-2 standard. Furthermore, they are not specifically targeted to SPCMMs customization.

3.2 Processes for the Development of Standards

Some SPCMMs have been developed in the form of standards, supported by some regulatory body or group of international standards [12]. Standards are, in general, developed according some principles [13]: (i) consensus: in the development of a standard wide range of interests are taken into account: manufacturers, vendors, users, governments, research organizations, etc; (ii) industry wide: standards must provide global solutions for industries and customers worldwide; (iii) voluntary: standardization is an activity based on voluntary involvement of all interests in the community. For instance, ISO standards are developed in a three-phase process [13]:

- **Phase 1:** in general, the perceived need for a new standard comes from an industry sector, which communicates this need for a national member body. This need is then evaluated and, once approved, the scope of the new standard is set, involving working groups composed of experts from different countries.
- **Phase 2:** this is the phase of consensus-building. After defining the scope, it begins the negotiation between group members to detail the contents of the standard.
- **Phase 3:** final approval and generation of draft standard is given at this phase, where it needs to receive the approval of at least two-thirds of all members of the group and 75% of those voting. After this process of ballot, the first version of the standard is published.

Since being first proposed for publication a standard goes through a series of 9 stages and 7 related sub-stages of development[1], from the *preliminary* till the *withdrawal* stage.

3.3 Development of SPCMMs in Practice

In order to complete the elicitation of the state-of-the-art in this context through the analysis of how SPCMMs are developed, a systematic literature review (SLR) was performed [12]. This review was performed to systematically investigate and synthesize the existing literature relating to the subject of software process capability/maturity models (SPCMMs), focusing on this research question: *How are software process capability/maturity models created?* Details on the SLR can be found in the Web Appendix of [12].

[1] www.iso.org/iso/standards_development/processes_and_procedures/stages_description/stages_table.htm

As a result of the SLR, 52 software process capability/maturity models were identified. Besides the evolution of new versions of existing models (such as, the evolution of the CMM/CMMI framework), there exists a clear trend toward the specialization of models to specific domains. Currently, a large variety of specific models exist for diverse domains, including, for example, small and medium enterprises, security engineering, knowledge management, automotive systems, XP (eXtreme Programming), etc[2].

Furthermore, it was observed that these models are developed using diverse approaches. Some models, typically the ones published as standards, have been developed by following a high-level process defined by the standardization organization. These processes involve the standards community in different stages and with varying degrees of participation [13]. However, in general, it was surprising to find very little information on how SPCMMs are currently developed. Only 21% of the papers found in the SLR [12] presented detailed information on the model development, 27% contained superficial model development information and 52% did not provide any substantial information on this aspect. The activities and techniques discovered in the detailed papers of the SLR were used within the method presented in section 4.

3.4 Discussion

The SLR demonstrated that a large variety of software process capability/maturity models have been developed and customized. However, in general there appears to be a lack of methodological support for the development and customization of such models. Therefore, in order to assist with the development and customization of models representing collections of best practices within a specific domain the processes used to develop and customize these models should be better understood and clearly presented. Access to standard processes for the development of such models could greatly assist the systematic development of such models and enable such models to be validated.

4 A Proposal for a Method for the Customization of SPCMMs

In order to promote the alignment of the customization of SPCMMs to the ISO/IEC 15504-2 requirements and to increase their quality, as well as their adoption rate in practice, a KE-based approach presented in this section was developed. The approach is based on an analysis of four elements: (i) standard development procedures; (ii) existing methods for the development of maturity models/grids; (iii) the way such customizations are currently performed; and (iv) KE techniques. From a KE viewpoint, the customization of such models relates to knowledge acquisition, collecting best practices of a specific domain by customizing generic SPCMMs to domain-specific models. A generic life cycle for KE includes [30]: (i) knowledge identification; (ii) knowledge specification and (iii) knowledge refinement. Currently, there exist several methodologies, frameworks and approaches that provide detailed support for the KE development life cycles, such as e.g., CommonKADS [31]. Furthermore, the usage and evolution of knowledge models is typically not covered

[2] Another list – regularly updated is available at this webpage: www.semq.eu/leng/proimpsw.htm#quinto

by SPCMMs developed to date [12]. In addition, KE techniques have so far, not yet been applied for the customization of generic SPCMMs knowledge to specific domains. The proposed method is structured in five phases:

- **Phase 1 - Knowledge Identification:** The main objective of phase 1 is to achieve familiarization with the target domain and a characterization of the context for which the SPCMM will be customized;
- **Phase 2 - Knowledge Specification**: During this central phase, a first version of the customized model is developed;
- **Phase 3 - Knowledge Refinement**: Within this phase, the draft model is validated, balloted and refined to develop a model approved by a majority of respective community;
- **Phase 4 - Knowledge Usage**: After its publication, the model is put into use and results of its usage are collected and analyzed;
- **Phase 5 - Knowledge Evolution**: It is necessary to provide methodological support for the continuous evolution of the model once the model has been implemented in the target domain.

Table 2. Techniques used in each method phase

Method Phase	Basic Technique(s)
1. Knowledge Identification	Ontology Development [32] [33] [34] Glossary Development [35] [36] Literature Review [37] [38] [39] Systematic Literature Review [40] Goal Question Metric [41] [42] Expert Selecting [43] [44] [45] Delphi [43] [44] [45] Focus Groups [46] [47]
2. Knowledge Specification	Delphi [43] [44] [45] Perspective-Based Reading [59] [60] [61] Checklist-based Reading [62][63] Semantic Mapping [20] [21] [22] [23] [24] Domain quality requirements elicitation [53] [54] [55] Focus groups [46] [47] Structured Interview [31] Nominal group [56] [57] Software Process Quality Function Deployment [54] [53] [58] Process Selection [34]
3. Knowledge Refinement	Expert Selecting [43] [44] [45] Delphi [43] [44] [45] Guidelines of Modeling [35] [36] Behavior Engineering [64] [65] [66] Interrater Agreement [67] Checklist-based Reading [62] [63]
4. Knowledge Usage	Goal Question Metric [41] [42] Practical Software and Systems Measurement [68] [69] [70]
5. Knowledge Evolution	Model change request management [71] [14] [72]

Each phase is composed by a set of activities that are not necessarily sequentially executed using different techniques identified as relevant both from literature and from real developing SPCMMs experiences. The activities of the various phases of the method are aligned standard 15504-2 [3], providing coverage to the requirements for PRM and PAM (see Annex 1). Table 2 shows the stages and techniques used in the method, including key references for each technique.

A detailed technical report describing the proposed method is available in [73].

5 First Results and Discussion

The proposed method for SPCMM customization has been developed in parallel with the customization of a SPCMM for the Software as a Service (SaaS) domain [74] and Medi SPICE [75]. So it has been applied as an Exploratory Case Study. Exploratory Case Study is a short case study, undertaken as a first step before a larger investigation. Its function is to develop the evaluation questions, measures, design and analytical strategy for a possible larger study. It is particularly useful when there is some considerable uncertainty about processes, goals and results are achieved due to the embryonic state of research [76]. Thus, for this evaluation, an exploratory case study was defined as the study design.

5.1 A Model for Software as a Service (SaaS) Domain

SaaS is a software solution offered as a service and is developed using SOA. As the SaaS scenario requires specific quality needs, such as, security, availability and service continuation, due to its characteristics of distributed software products as services, a customization of SPCMMs has been done. The SaaS SPCMM [74] has been developed by a group of researchers at the UFSC – Federal University of Santa Catarin (Brazil), involving experts from both the SaaS and SPI domains. The model was developed through adopting phases 1 to 3 of this method. To date, phases 4 and 5 have still not been performed.

During the development, the SaaS domain was characterized and the specialists were identified. Generic SPCMMs were also analyzed and identified as a basis for the customized model. SaaS experts were interviewed in order to analyze quality and performance needs. The results were validated in a second step through a survey. Then, SPI experts identified relevant processes and basic practices with respect to the identified quality and performance needs by mapping them. The result was a draft version of the process model.

5.2 Medi SPICE

In this second exploratory case study the method was applied during the development of the Medi SPICE [3] Process Reference Model (PRM). Medi SPICE is an international project involving the Regulated Software Research Group in Dundalk Institute of Technology, the SPICE User Group (developers of ISO/IEC 15504 and related software process domain models), representatives from international medical

[3] http://medispice.ning.com

device industry and representatives from the international standards community with the aim to develop a SPCMM containing software engineering best practices for the development and maintenance of medical device software [77].

Software development for medical devices has several characteristics that differentiate it from software development in other areas, especially as in order to market a medical device it is first essential to gain regulatory approval for the device within the particular region in which the device will be marketed. Due to these factors the software development activity in this area is heavily regulated by various bodies, through standards such as: AAMI / IEC 62304, FDA and European guidelines, ISO 14971, IEC 60601-1-4, ISO 13485, etc. Therefore, due to both the growth of software within the medical device industry and the revised definition of a medical device within the Medical device directive [78] there is now real need for Medi SPICE to assist software development organizations to put regulatory compliant software processes in place within the medical device industry.

The method was applied in the development of the Medi SPICE PRM during the period of January to December 2010. During this period phases 1 to 3 were also performed.

5.3 Observed Results

These experiences allowed us to identify strengths and weaknesses of initial versions of the proposed method in practice. One of its strengths is the involvement of specialists, although we also identified that in order to stimulate a wide adoption of the model, a much stronger involvement of the community is also required. Other strength is the methodological support which typically, for standard developments, is not available. We also observed several improvement opportunities:

- Support for a systematic mapping and harmonization of existing models;
- Better methodological support for consensus building among community representatives throughout the models development and not just elicitation of their knowledge;
- More systematic and formal support for the validation of the models.
- Integration of data-based input to the models if available in the specific domain in order to complete the expert's knowledge.

In addition, we are currently performing a systematic validation of the method through an expert panel. The main objective of this validation is to evaluate the method's ability to produce valid models (presenting generality, flexibility, completeness, usability and comprehensibility) and models aligned to the requirements of ISO/IEC 15504-2, from the point of view of specialists in maturity models in the context of an Expert Panel.

Experts discovered in the SLR [12] were invited to evaluate the method. To date, we have obtained responses from 12 SPI experts that have participated in the development of 17 different SPCMMs, with 55% having more than 10 years of experience in SPI.

A first preliminary analysis of the responses indicates that the method, in the opinion of 72% of the experts, has the potential ability to produce valid models

(presenting generality, flexibility, completeness, usability and comprehensibility). We also observed that 82% of the respondents felt that the method provides enough support for developing SPCMMs and adequately represents what is necessary to customize a SPCMM. All respondents felt that usage of the method could produce models aligned to the requirements of the ISO/IEC 15504-2 (for PRM and PAM).

6 Conclusions

In this paper, we outlined an approach for SPCMM customization by integrating a Knowledge Engineering (KE) perspective, customization experiences from literature and standard development processes. A first application of the proposed approach for the customization of a SaaS SPCMM provided a first indication that the approach can be useful for the customization of such models as well as enabling the identification strengths and weaknesses. Based on the feedback, we are currently evolving and refining the proposed approach as well as continuing its application in parallel for the customization of SPCMMs, such as, for medical devices as well as digital convergence.

Acknowledgements

This work was supported by the CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), both entities of the Brazilian government focused on scientific and technological development. The Medi SPICE part of this research is supported by the Science Foundation Ireland (SFI) Stokes Lectureship Programme, grant number 07/SK/I1299, the SFI Principal Investigator Programme, grant number 08/IN.1/I2030 and supported in part by Lero - the Irish Software Engineering Research Centre (http://www.lero.ie).

References

[1] Salviano, C.F., Figueiredo, A.M.C.M.: Unified Basic Concepts for Process Capability Models. In: 20th Int Conf on Sw. Eng. and Knowledge Eng (SEKE 2008), San Francisco, USA, pp. 173–178 (2008), http://pro2pi.wdfiles.com/local-files/publicacoes-sobre-a-metodologia/SalvianoandFigueiredo-2008-PRO2PI-SEKE-article.pdf

[2] CMMI Product Team. CMMI for Development (CMMI-DEV), Version 1.3, Technical Report, CMU/SEI-2010-TR-033, Software Engineering Institute (2010), http://www.sei.cmu.edu/cmmi/tools/cmmiv1-3/

[3] International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) ISO/IEC, ISO/IEC 15504: Information Technology Process Assessment - Part 1 to 5, International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), ISO/IEC International Standard (2005)

[4] Beecham, S., Hall, T., Rainer, A.: Building a Requirements Process Improvement Model, Faculty of Engineering and Information Sciences, University of Hertfordshire, Hertfordshire, Technical Report 378 (2003),

https://uhra.herts.ac.uk/dspace/bitstream/2299/986/1/S67.pdf
[5] Cass, A., Volcker, C.: SPICE for SPACE: A method of Process Assessment for Space Projects. In: SPICE 2000 Conference (2000)
[6] Torgersson, J., Dorling, A.: Assessing CBD - What's the Difference? In: 28th Euromicro Conference, Dortmund, Germany, pp. 332–341 (2002)
[7] Automotive SIG. The SPICE User Group Automotive Special Interest Group, Automotive SPICE Process Reference Model (2010), http://www.automotivespice.com
[8] Wangenheim, C.G., Hauck, J.C.R., Salviano, C.F., Wangenheim, A.: Systematic Literature Review of Software Process Capability/Maturity Models. In: Proceedings of the 10th SPICE Conference 2010, Pisa, Italy (2010)
[9] Matook, S., Indulska: Improving the quality of process reference models: A quality function deployment-based approach. Decision Support Systems 47, 60–71 (2009), http://espace.library.uq.edu.au/eserv/UQ:161303/Matook_Indulska_DDS_2009.pdf
[10] Becker, J., Knackstedt, R., Pöppelbuß, J.: Developing Maturity Models for IT Management – A Procedure Model and its Application. Business & Information Systems Engineering 1(3), 213–222 (2009), http://www.bise-journal.org/index.php,do=show/site=wi/sid=16424296734d2251f9628a7985168360/alloc=17/id=2429
[11] Lahrmann, G., Marx, F.: Systematization of maturity model extensions. In: Winter, R., Zhao, J.L., Aier, S. (eds.) DESRIST 2010. LNCS, vol. 6105, pp. 522–525. Springer, Heidelberg (2010)
[12] Wangenheim, C.G., Hauck, J.C.R., Mccaffery, F., Wangenheim, A.: Creating Software Process Capability/ Maturity Models. IEEE Software 27(4) (2010)
[13] International Organization for Standardization (ISO). How are Standards Developed, http://www.iso.org/iso/standards_development/processes_and_procedures/how_are_standards_developed.htm
[14] de Bruin, T., Rosemann, M., Freeze, R., Kulkarmi, U.: Understanding the Main Phases of Developing a Maturity Assessment Model. In: 16th Australasian Conference on Information Systems, Sydney (2005), http://www.followscience.com/library_uploads/ceba558bded879ccc0b45cd2c657e870/123/understanding_the_main_phases_of_developing_a_maturity_assessment_model.pdf
[15] Mettler, T.: A Design Science Research Perspective on Maturity Models in Information Systems, Universität St. Gallen, St. Gallen, Switzerland, Technical Report BE IWI/HNE/03 (2009), http://www.alexandria.unisg.ch/export/DL/67708.pdf
[16] Maier, A.M., Moultrie, J., Clarkson, P.J.: Developing maturity grids for assessing organisational capabilities: Practitioner guidance. In: 4th International Conference on Management Consulting, Academy of Management (MCD 2009), pp. 11–13 (2009), http://www.iff.ac.at/oe/full_papers/Maier%20Anja%20M._Moultrie%20James_Clarkson%20P.%20John.pdf
[17] Salviano, C.F., Zoucas, A., Silva, J.V.L., Alves, A.M., von Wangenheim, C.G., Thiry, M.: A Method Framework for Engineering Process Capability Models. In: 16th European Systems and Software Process Improvement and Innovation, Alcala, Spain, pp. 6.25–6.36 (2009)
[18] Lee, J.-H., Lee, D.H., Kang, S.: An overview of the business process maturity model (BPMM). In: Chang, K.C.-C., Wang, W., Chen, L., Ellis, C.A., Hsu, C.-H., Tsoi, A.C., Wang, H. (eds.) APWeb/WAIM 2007. LNCS, vol. 4537, pp. 384–395. Springer, Heidelberg (2007)
[19] Beecham, S., et al.: Defining a Requirements Process Improvement Model. Software Quality Journal 13(3), 247–279 (2005)

[20] Beecham, S., et al.: Using an Expert Panel to Validate a Requirements Process Improvement Model. Journal of Systems and Software 76(3), 251–275 (2005)
[21] Paulk, M.C., et al.:Capability Maturity Model for Software, Version 1.1, Technical Report, CMU/SEI-93-TR-024, Software Engineering Institute (February 1993), http://www.sei.cmu.edu/reports/93tr024.pdf
[22] Cass, A., et al.: SPiCE in Action - Experiences in Tailoring and Extension. In: 28th Euromicro Conference, Dortmund, Germany (2002)
[23] Bovee, M., et al.: A Framework for Assessing the Use of Third-Party Software Quality Assurance Standards to Meet FDA Medical Device Software Process Control Guidelines. IEEE Transactions on Engineering Management 48(4), 465–478 (2001)
[24] Niazi, M., et al.: A Maturity Model for the Implementation of Software Process Improvement: An Empirical Study. Journal of Systems and Software 74(2), 155–172 (2005)
[25] April, A., Coallier, F.: Trillium: A Model for the Assessment of Telecom Software System Development and Maintenance Capability. In: 2nd IEEE Software Engineering Standards Symposium, Montreal, Canada (1995)
[26] Burnstein, I., et al.: Developing a Testing Maturity Model, Part II, Crosstalk (September 1996)
[27] TMMI Foundation, TMMI - Test Maturity Model Integration, http://www.tmmifoundation.org/html/tmmiorg.html
[28] Kyung-A, Y., Seung-Hun, P., Doo-Hwan, B., Hoon-Seon, C., Jae-Cheon, J.: A Framework for the V&V Capability Assessment Focused on the Safety-Criticality. In: 13th IEEE International Workshop on Software Technology and Engineering Practice, Budapest, Hungary, pp. 17–24 (2005), http://spic.kaist.ac.kr/~selab/html/Publication/IntJournal/A%20framework%20for%20the%20V&V%20capability%20assessment%20focused%20on%20the%20safety-criticality.pdf
[29] McCaffery, F., Pikkarinan, M., Richardson, I.: AHAA – Agile, Hybrid Assessment Method for Automotive, Safety Critical SMEs. In: Proc: International Conference on Software Engineering (ICSE 2008), Leipzig, Germany (2008)
[30] Schreiber, A.T., Wielinga, B.J.: Knowledge Model Construction. In: 11th Workshop on Knowledge Acquisition, Modeling and Management, Voyager Inn, Banff, Alberta, Canada (1998), http://ksi.cpsc.ucalgary.ca/KAW/KAW98/schreiber/
[31] Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N., Van De Velde, W., Wielinga, B.: Knowledge Engineering and Management - The CommonKADS Methodology. The MIT Press, Cambridge (2000) ISBN 978-0262193009
[32] Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Technical Report, KSL 93-04, Knowledge Systems Laboratory, Stanford University, Palo Alto, CA, USA (1993), http://www-ksl.stanford.edu/knowledge-sharing/papers/
[33] Noy, N., McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880 (March 2001), http://www.ksl.stanford.edu/KSL_Abstracts/KSL-01-05.html
[34] Hsieh, S.H., Hsien-Tang, L., Nai-Wen, C., Kuang-Wu, C., Ken-Yu, L.: Enabling the development of base domain ontology through extraction of knowledge from engineering domain handbooks. In: Advanced Engineering Informatics (2010) (in press), http://dx.doi.org/10.1016/j.aei.2010.08.004
[35] Government of South Australia. Developing a Thesaurus Guideline Version 1.2, State Records of South Australia (2002), http://www.archives.sa.govol.au/files/management_guidelines_developingthesaurus.pdf

[36] Sikorski, M.: A Framework for developing the on-line HCI Glossary: Technical Report, Technical University of Gdansk (2002), http://www.org.id.tue.nl/IFIP-WG13.1/HCIglossary.PDF
[37] Mongan-Rallis, H.: Guidelines for writing a literature review, http://www.duluth.umn.edu/~hrallis/guides/researching/litreview.html
[38] Galvan, J.: Writing literature reviews: a guide for students of the behavioral sciences, 3rd edn. Pyrczak Publishing, Glendale (2006) ISBN 978-1884585661
[39] Cronin, P., Coughlan, M., Frances, R.: Undertaking a literature review: a step-by-step approach. British Journal of Nursing 17(1) (2008), http://lancashirecare.files.wordpress.com/2008/03/2008-undertaking-a-literature-review-a-step-by-step-approach.pdf
[40] Kitchenham, B.A.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Version 2.3 Technical report, University of Durham, Keele, UK (2007), http://www.elsevier.com/framework_products/promis_misc/infsof-systematic%20reviews.pdf
[41] Basili, V.R., Caldiera, G., Rombach, H.D.: The Goal Question Metric Approach, 2nd edn. Encyclopedia of Software Engineering. John Wiley & Sons, Chichester (1994), http://www.cs.umd.edu/~basili/publications/technical/T89.pdf
[42] Wangenheim, C.G., et al.: Software Measurement for Small and Medium Enterprises - A Brazilian-German view on extending the GQM method. In: 7th International Conference on Empirical Assessment ion Software Engineering, Keele, UK (2003)
[43] Okoli, C., Pawlowski, S.D.: The Delphi method as a research tool: an example, design considerations and applications. Inf. Manage (2004), http://chitu.okoli.org/images/stories/bios/pro/research/methods/OkoliPawlowski2004.pdf
[44] de Bruin, T., Rosemann, M.: Using the Delphi Technique to Identify BPM Capability Areas. In: 18th Australasian Conference on Information Systems, vol. 42, pp. 15–29 (2007), http://www.acis2007.usq.edu.au/assets/papers/106.pdf
[45] Powell, C.: The Delphi technique: myths and realities. Journal of Advanced Nursing 41(4), 376–382 (2003), http://rachel.org/files/document/The_Delphi_Technique_Myths_and_Realities.pdf
[46] Kontio, J., Lehtola, L., Bragge, J.: Using the Focus Group Method in Software Engineering: Obtaining Practitioner and User Experiences. In: International Symposium on Empirical Software Engineering, pp. 271–280 (2004), http://www.sbl.tkk.fi/jkontio/Papers/FocusGroup_ISESE_web.pdf
[47] Free Management Library. Basics of Conducting Focus Groups, http://managementhelp.org/evaluatn/focusgrp.htm
[48] Mutafelija, B., Stromberg, H.: Process Improvement with CMMI v1.2 and ISO Standards. Auerbach, Boca Raton (2008) ISBN 978-1420052831
[49] Thiry, M., Zoucas, A., Tristão, L.: Mapeando Modelos de Capacidade de Processo no Contexto de Avaliações Integradas de Processo de Software. In: II Workshop on Advanced Software Engineering, pp. 35–42 (2009)
[50] Wangenheim, C.G., Thiry, M.: Analyzing the Integration of ISO/IEC 15504 and CMMI-SE/SW, Universidade do Vale do Itajaí, São José, Technical Report LQPS001.05E (2005)
[51] Wangenheim, C.G., et al.: Best practice fusion of CMMI-DEV v1.2 (PP, PMC, SAM) and PMBOK 2008. Information and Software Technology 52(7), 749–757 (2010)
[52] Beel, J., Gipp, B.: Link Analysis in Mind Maps: A New Approach To Determine Document Relatedness. In: Proceedings of the 4th International Conference on Ubiquitous Information Management and Communication (ICUIMC 2010). ACM, New York (2010), http://www.sciplore.org/publications/2010-LAMM-preprint.pdf

[53] Cancian, M.H., Hauck, J.C.R., von Wangenheim, C.G., Rabelo, R.J.: Discovering software process and product quality criteria in software as a service. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 234–247. Springer, Heidelberg (2010), http://www.das.ufsc.br/~maiara/files/profes_maiara.pdf
[54] Richardson, I.: SPI Models: What Characteristics are Required for Small Software Development Companies. Software Quality Journal, 101–114 (2002), http://portal.acm.org/citation.cfm?id=650270
[55] Beecham, S., Hall, T., Rainer, A.: Defining a Requirements Process Improvement Model. Software Quality Journal 13(3), 247–279 (2005)
[56] Sample, J.A.: Nominal Group Technique: An Alternative to Brainstorming. Journal of Extension 22(2) (1984), http://www.joe.org/joe/1984march/iw2.php
[57] CDC. Gaining Consensus Among Stakeholders Through the Nominal Group Technique. Evaluation Briefs (7) (2006), http://www.cdc.gov/HealthyYouth/evaluation/pdf/brief7.pdf
[58] Matook, S., Indulska, M.: Improving the Quality of Process Reference Models: A Quality Function Deployment-Based Approach. Decision Support Systems 47 (2009), http://espace.library.uq.edu.au/eserv/UQ:161303/Matook_Indulska_DDS_2009.pdf
[59] Basili, V.R., et al.: The Empirical Investigation of Perspective-Based Reading. Empirical Software Engineering, Kluwer Academic Publisher 1, 133–164 (1996), http://www.cs.umd.edu/~basili/publications/journals/J63.pdf
[60] Shull, F., Rus, I., Basili, V.: How perspective-based reading can improve requirements inspections. Computer 33(7), 73–79 (2000)
[61] Robbins, B., Carver, J.: Cognitive factors in perspective-based reading (PBR): A protocol analysis study. In: 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), pp. 145–155 (2009)
[62] McMeekin, D.A., et al.: Checklist Based Reading's Influence on a Developer's Understanding. In: 19th Australian Conference on Software Engineering (2008), http://www.computer.org/portal/web/csdl/doi/10.1109/ASWEC.2008.7
[63] Fagan, M.E.: Design and code inspections to reduce errors in program development. IBM Systems Journal 15(3), 182–211 (1976), http://www.cs.umd.edu/class/spring2005/cmsc838p/VandV/fagan.pdf
[64] Behaviour Engineering, http://www.behaviorengineering.org/
[65] Dromey, R.G.: From Requirements to Design: Formalizing the Key Steps. In: 1st International Conference on Software Engineering and Formal Methods, Australia (2003)
[66] Tuffley, D., Rout, T.P.: Applying Behavior Engineering to Process Modeling. In: Proceedings of the Improving Systems and Software Engineering Conference, ISSEC (2009), http://www98.griffith.edu.au/dspace/handle/10072/31748
[67] Emam, K.E.: Benchmarking Kappa for Software Process Assessment Reliability Studies. Empirical Software Engineering 4(2), 113–133 (2004)
[68] Practical Software and Systems Measurement, http://www.psmsc.com
[69] DoD, Department of Defense & US Army. PSM - Practical Software and System Measurement, A foundation for Objective Project Management, Version 4.0b, Department of Defense & US Army (2003), http://www.psmsc.com
[70] ISO/IEC. ISO/IEC 15939:2007 Systems and software engineering - Measurement process. International Organization for Standardization (2007)
[71] CMMI Change Requests, http://www.sei.cmu.edu/cmmi/tools/cr/
[72] Enterprise Spice SPICE Change Request, http://www.enterprisespice.com/page/publication-1

[73] Hauck, J. C. R., Wangenheim, C. G., Wangenheim, A.: A Knowledge Engineering Based Method for SPCMMs customization, Technical Report RT_GQS_0_9, INCoD Software Quality Group (2010), http://www.inf.ufsc.br/~jeanhauck/method/RT_GQS_01_SPCMMs_Dev_Method_v_0_9.pdf

[74] Cancian, M.: Process Reference Model for SaaS. Technical Report. UFSC, Florianopolis/Brazil, http://www.gsigma.ufsc.br/~cancian/guide/

[75] McCaffery, F., Dorling, A.: Medi SPICE Development. Software Process Maintenance and Evolution: Improvement and Practice Journal 22(4), 255–268 (2010)

[76] GAO - United States General Accounting Office. Case Study Evaluations, Technical Report GAO/PEMD-91-10.1.9. Program Evaluation and Methodology Division (1990), http://www.gao.gov/special.pubs/10_1_9.pdf

[77] McCaffery, F., Dorling, A., Casey, V.: Medi SPICE: An Update. In: Proceedings of the 10th International SPICE Conference, Italy (2010)

[78] European Council. Council Directive 2007/47/EC (Amendment). Official Journal of The European Union, Luxembourg (2007), http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2007:247:0021:0055:en:PDF

Annex 1: Mapping to ISO/IEC 15504-2 [3] requirements of PRM and PAM to the method activities

Code	Requirement	A1.1	A1.2	A1.3	A1.4	A2.1	A2.2	A2.3	A2.4
Process Reference Model									
6.2.3.1-a	A Process Reference Model shall contain a declaration of the domain of the Process Reference Model	X						X	
6.2.3.1-b	a description, meeting the requirements of 6.2.4 of this International Standard, of the processes within the scope of the Process Reference Model			X				X	
6.2.3.1-c	a description of the relationship between the Process Reference Model and its intended context of use			X			X	X	
6.2.3.1-d	a description of the relationship between the processes defined within the Process Reference Model							X	
6.2.3.2	The Process Reference Model shall document the community of interest of the model and the actions taken to achieve consensus within that community of interest	X	X		X				
6.2.3.2-a	the relevant community of interest shall be characterized or specified	X			X			X	
6.2.3.2-b	the extent of achievement of consensus shall be documented	X			X			X	
6.2.3.2-c	if no actions are taken to achieve consensus, a statement to this effect shall be documented							X	
6.2.3.3	The processes defined within a Process Reference Model shall have unique process descriptions and identification						X	X	
6.2.4-a	a process shall be described in terms of its purpose and outcomes					X		X	
6.2.4-b	in any process description the set of process outcomes shall be necessary and sufficient to achieve the purpose of the process							X	
6.2.4-c	process descriptions shall be such that no aspects of the measurement framework as described in Clause 5 of this International Standard beyond level 1 are contained or implied					X		X	
Process Assessment Model									
6.3.2.1	A Process Assessment Model shall relate to at least one process from the specified Process Reference Model(s).					X			X
6.3.2.2	A Process Assessment Model shall address, for a given process, all, or a continuous subset, of the levels (starting at level 1) of the measurement framework for process capability for each of the processes within its scope.					X			X
6.3.2.3	A Process Assessment Model shall declare its scope of coverage			X					X
6.3.2.3-a	the selected Process Reference Model(s);			X		X			X
6.3.2.3-b	the selected processes taken from the Process Reference Model(s);			X		X			X
6.3.2.3-c	the capability levels selected from the measurement framework.					X			X
6.3.3	A Process Assessment Model shall be based on a set of indicators that explicitly addresses the purposes and outcomes, as defined in the selected Process Reference Model of all the processes within the scope of the Process Assessment Model; and that demonstrates the achievement of the process attributes within the capability level scope of the Process Assessment Model. The indicators focus attention on the implementation of the processes in the scope of the model.								X
6.3.4	The mapping shall be complete, clear and unambiguous.						X		X
6.3.4-a	the purposes and outcomes of the processes in the specified Process Reference Model;					X			X
6.3.4-b	the process attributes (including all of the results of achievements listed for each process attribute) in the measurement framework.					X			X
6.3.5	A Process Assessment Model shall provide a formal and verifiable mechanism for representing the results of an assessment as a set of process attribute ratings for each process selected from the specified Process Reference Model(s).					X			X
7.2	The party performing verification of conformity shall obtain objective evidence that the Process Reference Model fulfils the requirements set forth in 6.2 of this part of ISO/IEC 15504. Objective evidence of conformance shall be retained.							X	
7.3	The party performing verification shall obtain objective evidence that the Process Assessment Model fulfils the requirements set forth in 6.3 of this part of ISO/IEC 15504. Objective evidence of conformance shall be retained.								X

X - Indicates that the implementation of this activity generates, even partially, the coverage requirement in the final document

Homogenization, Comparison and Integration: A Harmonizing Strategy for the Unification of Multi-models in the Banking Sector

César Pardo[1,2], Francisco J. Pino[1,2], Félix García[2], Mario Piattini[2], Maria Teresa Baldassarre[3], and Sandra Lemus[2,4]

[1] IDIS Research Group
Electronic and Telecommunications Engineering Faculty,
University of Cauca, Calle 5 No. 4 - 70.
Kybele Consulting Colombia (Spinoff)
Popayán, Cauca, Colombia.
{cpardo,fjpino}@unicauca.edu.co
[2] Alarcos Research Group
Institute of Information Technologies & Systems,
University of Castilla-La Mancha, Paseo de la Universidad 4, Ciudad Real, España
{Felix.Garcia,Mario.Piattini}@uclm.es
[3] Department of Informatics, University of Bari.
SER&Practices, SPINOFF, Via E. Orabona 4, 70126, Bari, Italy
baldassarre@di.uniba.it
[4] Superintendencia de Bancos de Guatemala,
9 avenida 22-00 zona 1, Ciudad de Guatemala, Guatemala
slemus@sib.gob.gt

Abstract. Information Technologies (IT) play a crucial role in the development of the business processes in organizations. Acquiring the best technologies is quickly becoming as important as understanding and improving the business model of organizations. As a result, many (inter)national standards and models for IT Management, IT Government and IT Security have been developed. This situation allows organizations to choose and improve their processes, selecting the models that best suit their needs. Since several relationships between these models can be found, carrying out the harmonization of their similarities and differences will make it possible to reduce the time and effort involved in implementing them. In this paper, we present a harmonization strategy which has been defined to harmonize COBIT 4.1, Basel II, VAL IT, RISK IT, ISO 27002 and ITIL V3. This work intends to support organizations which are interested in knowing how to carry out the harmonization of these models. Furthermore, as a result of the execution of the harmonization strategy we have defined, a unified model for Banking, called ITGSM, is presented. It resolves the conflicts between the models mentioned above and provides a useful reference model to organizations that are planning to adopt them.

Keywords: Multi-model, Harmonization, IT Management, IT Government, IT Security, Homogenization, Comparison, Integration.

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 59–72, 2011.
© Springer-Verlag Berlin Heidelberg 2011

1 Introduction

There is a wide range of standards and models to support the process improvement of organizations. These benefit of international recognition and, according to [1], the models provide the descriptions and/or best practices for different spheres, such as software development, quality management, security, amongst others. In addition to these areas, many organizations are increasingly becoming interested in improving their business processes through IT management and IT security models such as COBIT, ITIL, ISO 27002 (also known as ISO 17799), amongst others. This is the case now more than ever.

Since Information Technologies (IT) are important for the development of almost all operations and activities of organizations, many organizations are showing an ever-growing interest in them. As in the case of other approaches, there is an abundance of good practices related to security, management and governance of IT. It has already been said that the good thing about standards is that there are so many of them [2], because when there many standards, organizations can use a particular one or choose several of them in a certain situation.

This situation allows organizations to select and complement their processes from the models which fit their contexts well, e.g. if a model focusing on Computer Security, like the NIST Handbook [3] is not suitable, an organization can go for the implementation of ISO 27002. Organizations can also implement more than one model, as well as improve more than one specific subject of their business processes, e.g. providing support for the risks of IT and managing IT investment, using models such as RISK IT [4] and VAL IT [5].

According to [2], experts and practitioners should use the better parts of existing standards as building blocks and be prepared to deconstruct standards. However, professional skills alone are not enough. The people involved need a map or guideline to tell them how to carry out the harmonization of multiple standards. This makes it possible to decrease the costs associated with the implementation of models, by not implementing each one separately [1]. For instance, the PO4 (Plan and Organize) practice of COBIT 4.1 in charge of the task "defines and implement process in an organization" is closely related to the VG2 (Value Governance) practice defined in VAL IT V2.0. The process engineers can take advantage of those relationships and incorporate them into a single management practice, which can fulfill the requirements of both models, thereby reducing the time and effort which would have been involved in implementing two practices. At present, some proposals related to the harmonization of multiple models have been defined [1]. They do not, however, provide formal toolkits such as techniques, methods, processes or methodologies, which make it easier to discover "how to" carry out the harmonization of multiple models.

Given the benefits that can be obtained from harmonization of multiple IT models, it is important to have information on how the practices of different models can be put in harmony with each other. In that sense, this article presents the harmonization strategy designed to harmonize the latest versions of COBIT 4.1, Basel II, VAL IT, RISK IT, ISO 27002 and ITIL V3, thus giving support to multiple regulations that the banking sector is subject to. The harmonization strategy is accomplished by means of three techniques: homogenization, mapping and integration, which have been joined together in a single strategy and give support to certain considerations: (i) resolving

the issues related to structural differences, (ii) mapping between multiple models and (iii) unifying process entities from a formal integration criterion.

This paper proceeds as follows. Section 2 presents related work. Section 3 gives an overview of the harmonization framework and harmonization strategy that has been designed. A summary of the execution of the harmonization strategy and the unified model obtained is presented in Section 4. Lastly, some relevant discussion is set out, along with the conclusions we have drawn and the future work we have planned.

2 Related Work

The literature presents some work that involves comparisons and mappings between different models. Among these pieces of work, those related to IT Management or IT Government are:

- In [6] a mapping between ITIL V3 and COBIT 4.1 is described.
- In [7] the relationships between COBIT 4.1, ITIL V3 and ISO/IEC 27002 are set out.

Similarly, the following studies have been conducted regarding the identification and analysis of the importance of different models in relation to the financial sector:

- In [8], a study aiming to identify to what extent Governance practices have been implemented in the financial sector in Romania is presented. It is a comparative study, with respect to the data presented by the IT Global Governance Status Report - 2006 of the ITGI. The models identified were: COBIT, BASEL II, BSC, ITIL and ISO 17799.
- In [9], we find an empirical study which made it possible to explore the importance and implementation of COBIT processes in Saudi organizations.
- In [10], a methodology, based on the COBIT processes, to bring about regulatory fulfillment of legislations like SOX and BASEL, is discussed.
- In [11], a study which investigates the way in which the companies of the financial sector of Belgium are applying IT Governance is presented; its aim is also to verify how this practice supports the alignment between IT and business.
- In [2], a study whose aim was to identify in a general way the family of standards that support IT Governance is described, along with their relationships and the value that each of these standards contributes to the management. The family of standards found was: COBIT, ITIL, ISO 13569, ISO 13335, MOF, ISO 17799, ISO 9001, BS 15000, COSO, PRINCE 2, PMBOK and PAS56.

As it can be seen from the work presented above, the most widely-used models in mapping and comparisons are: COBIT, BASEL II and ITIL. However, from the analysis of these studies it has been possible to find that the process entities (PEs) involved in the comparisons or mappings are of high-level abstraction, that is, they do not carry out the analysis of PEs, such as activities or tasks. Moreover, it has not been possible to find an integrated model which would harmonize multiple approaches to support IT Governance. Such a harmonization could reduce the effort and costs associated with the implementation of a new integrated model, which unifies the multiple regulations that the banking sector is subject to.

According to [1], we can note that there are few studies that provide solutions such as, amongst other elements: methodology, process, framework, activities, tasks, steps, for supporting the harmonization of multiple models. Moreover, most of them have been designed to give support to the harmonization of software models, and although some proposals do support a wider range of models, such as ITIL and COBIT, these are currently proposals within a formal method and validation. Taking that into account, in this article we provide the definition of a harmonization strategy obtained from the execution of a harmonization process. This is composed of three techniques or processes: homogenization, comparison and integration, which have made it possible to carry out: (i) An in-depth analysis of the models involved, (ii) Comparison and identification of relationships and differences between them and (iii) Integration and definition of an IT governance model for banking. The latest versions of models and standards such as COBIT [6], ITIL [12], RISK IT [4], VAL IT [5], ISO 27002 [13] (ISO 17799), and BASEL II [14] were used. These techniques have also been used in other harmonization projects that we have carried out (homogenizations of ISO 9001:2000, CMMI ISO/IEC 12207, CMMI-ACQ V1.2, COMPETISOFT, COBIT 4.0 and PMBOK [15] and comparisons of ISO 9001 and CMMI [16], CMMI-ACQ and ISO/IEC 12207:2008 [17], and between CMMI-ACQ and ISO/IEC 15504 [18]).

A detailed summary of the strategy followed to harmonize the models involved is presented in the next Section.

3 Configuration of the Harmonization Strategy

The management of the harmonization of models involved was supported by means of the execution of a *harmonization process*. This process provides a guideline that makes it easier to manage the tasks related to the definition and configuration of a suitable *harmonization strategy* for carrying out the harmonization of multiple models (see Figure 1). The goal is also to ensure the generation of a standard format for the documentation obtained. The harmonization strategy is made up of *techniques and/or methods,* which are configured according to the particular objectives and needs of the organization. Both the harmonization process and the harmonization strategy are elements of the *Harmonization Framework* defined to support the harmonization of multiple models. These and other elements that make up the Framework, together with their application in real case studies, are presented in [19]. A more detailed version of the harmonization process using SPEM 2.0 and edited with the EPF Composer can be seen in [20].

A harmonization strategy was obtained as the main work product of the execution of the harmonization process. It is made up of three techniques, which are part of the Harmonization Framework:

(i) Homogenization, to provide the tools which are suitable for setting in harmony the models involved, adding their information by means of a Common Schema or Common Structure of Process Entities (CSPE).
(ii) Comparison, to carry out the identification of differences and similarities between multiple models.
(iii) Integration, to give necessary support for combining and/or unifying the best practices of multiple models.

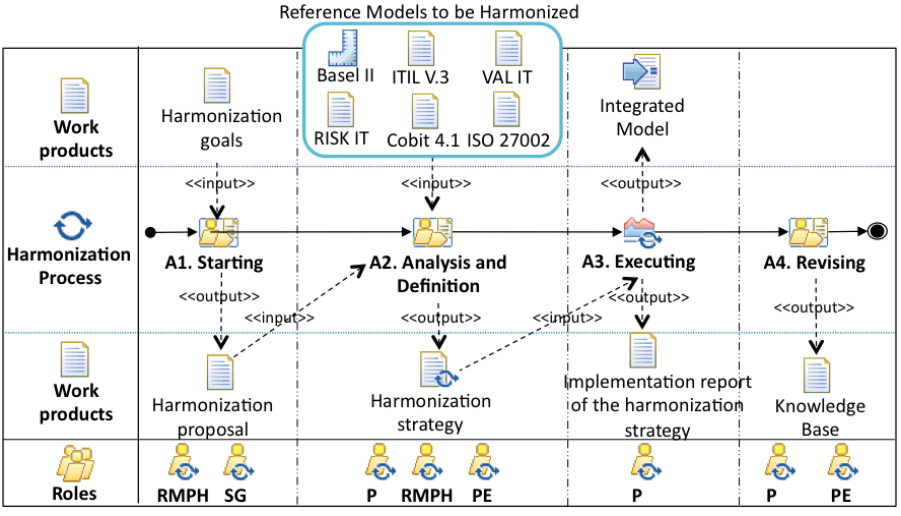


Fig. 1. Activity diagram of the harmonization process followed to obtain the harmonization strategy

The process carried out to perform the harmonization between COBIT, ITIL, RISK IT, VAL IT, ISO 27002, and BASEL II is described in the following lines. The purpose of this process was to provide a suitable harmonization strategy (or guideline) from the union and coupling of the three different techniques, aiming to guarantee the reliability of results obtained between them. Incorporating these techniques allowed us to carry out the step-by-step harmonization of the models involved. In order to organize and manage the people and activities throughout the strategy, this process establishes two roles: the performers and the reviewers, along with three stages:

- *Stage 1. Homogenization*. This stage involved the tasks: (i) acquisition of knowledge about the models involved, (ii) structure analysis and terminology, (iii) identification of requirements and (iv) correspondence.
- *Stage 2. Comparison.* This stage involved the tasks: (i) designing the mapping, (ii) carrying out the mapping, (iii) presenting the outcomes of the mapping and (iv) analyzing the results of the mapping.
- *Stage 3. Integration.* (i) designing the integration, (ii) establishing an integration criteria, (iii) carrying out the integration, (iv) analyzing the results of the integration and (v) presenting the integrated model.

Homogenization, comparison and integration are harmonization techniques which make up the Harmonization Framework. A detailed summary of these techniques can be seen in [12] and [19], respectively.

Figure 1 shows the activity diagram of the harmonization strategy described previously, which uses SPEM 2.0 notation and includes the main activities, tasks, roles and work products.

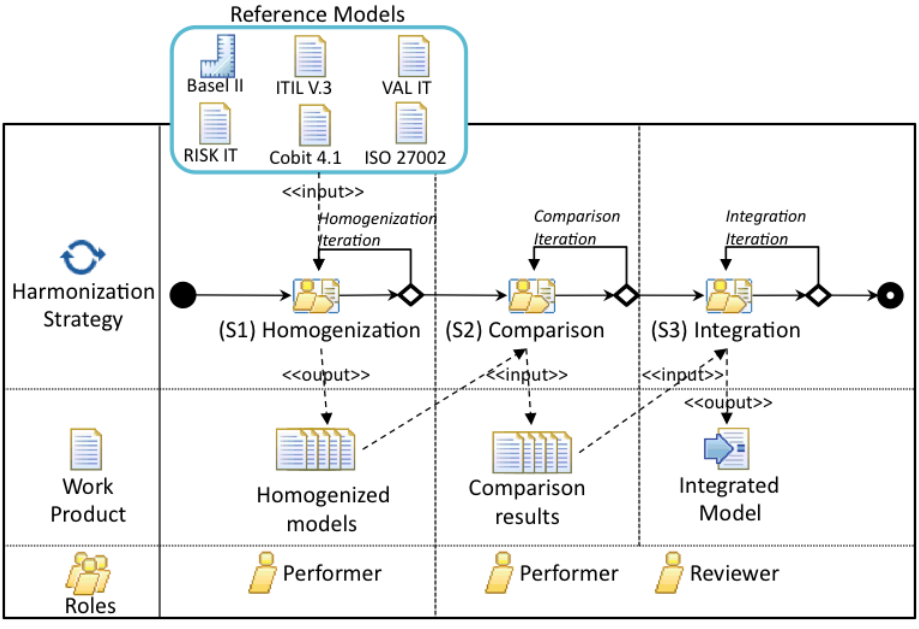


Fig. 2. Activity diagram of the harmonization strategy defined

4 Execution of the Harmonization Strategy

4.1 Homogenizing the Models

Before carrying out the execution of the harmonization strategy, an analysis of each model was performed with respect to some of their elements and/or attributes, e.g. approach, size (number of pages), organization developer and processes or practices that make up the models and obtain the reference document. Since each reference model defines its own structure of PEs, the *performer* carried out the homogenization of his/her structures through a CSEP template, which is defined and executed through a homogenization technique (see [15]). Homogenization of the models' structures made it easier to compare them, due to the fact that that they were structured under the same PEs during the execution of the comparison stage. We have performed the homogenization by means of an iterative and incremental procedure, in order to identify which specific practices of each model are supported by the CSEP template. This iterative and incremental approach has allowed us to manage the complexity where PEs of low-level abstraction are involved.

The homogenization of the PEs of each model allowed to prepare the models for the next stage. It also made it possible to carry out an initial comparison of the models at a high level of abstraction. This initial comparison permitted us to know if a model defines similar process entities or not, taking the entities process described in the CSEP as a basis. Table 1 shows an example of the CSEP and the homogenization of the PO1 process, which describes the best practices related to defining a strategic IT plan according to COBIT 4.1.

Table 1. Homogenization of the PO1 process defined in COBIT 4.1

Homogenization of PO1: Define a strategic IT plan		
Domain	Plan and organize	
Process	**ID**	PO1
	Name	Defines a strategic IT plan
	Purpose	To manage and drive all the IT resources in accordance with the particular strategy and priorities of the business.
	Objective	To improve the understanding of the main stakeholders as regards the opportunities and limitations of IT. In addition, to assess present performance and identify the capacity and requirements of human resources, as well as to clarify the level of research needed.
Activities		
PO1.1 IT Value Management: Work with the business, to guarantee that the IT investment portfolio of the firm contains programs with solid business cases. The task of accounting of profits and of cost-control is clearly assigned and monitored.		
PO1.2 Business-IT Alignment: Ensure that the goal of the business to which the IT is being applied is well understood. The business and IT strategies should be integrated, thereby creating a relationship between the goals of each, recognizing the opportunities, as well as the limitations in the present capacity. Broad-based communication should take place.		
PO1.3 Assessment of Current Capability and Performance: Assess the performance of the existing plans and of the information systems in terms of their contribution to the business objectives, their functionality, stability, complexity and costs, as well as their strengths and weaknesses.		
PO1.4 IT Strategic Plan: Create a strategic plan which, in cooperation with the relevant stakeholders, defines how IT will contribute to the firm's strategic objectives (goals), while also setting out costs and related risks. The strategic plan of the IT should include an estimate of the investment/operation, the sources of funding, the strategy to obtain as well as the legal and regulatory requirements.		
PO1.5 IT Tactical Plans: Produce a portfolio of tactical IT plans which are a by-product of the IT strategic plan. These tactical plans should describe the initiatives and requirements of resources demanded by the IT, as well as how the use of resources will be monitored, along with the profits gained.		
PO1.6 IT Portfolio Management: Actively administer, in conjunction with the business, the portfolio of IT investment programmes which is needed in order to achieve business goals.		

4.2 Comparing the Models

Since the homogenization allowed us to harmonize the models at the level of their structures and PEs, the *performer,* along with the *reviewer,* carried out the comparison of the models at the level of two PEs: *processes* and *activities*. This stage also followed an iterative and incremental approach. We say "iterative and incremental", because the comparison was carried out completely on one BASEL principle and COBIT processes, and then on the other BASEL principles. The result of this comparison was taken as a basis for carrying out the comparisons with the processes of other models involved. In each iteration, the comparisons of the descriptions of each PE were performed through a *semantic analysis*. Semantic analysis allowed us to identify the common features, differences and relationships at a low level of abstraction between the compared PEs. This consisted of studying the relationships between the descriptions of PEs that were being

compared. Figure 3 shows the tasks diagram of the comparison iterations, displaying the comparison iterations, work products, outcomes, role, directionality of the comparisons and quantity of relationships identified.

From a first comparison between BASEL II and COBIT 4.1, 44 relationships (or 18 related processes of COBIT) were found; these relationships were compared with the other models involved. The results found in each comparison iteration can be summarized as follows:

- 44 relationships (or 18 processes) found between: BASEL II (10 principles) and COBIT (34 processes),
- 35 relationships found between: COBIT (18 processes) and VAL IT (34 processes),
- 33 relationships found between: COBIT (18 processes) and RISK IT (9 processes),
- 108 relationships found between: COBIT (18 processes) and ISO 27002 (39 processes),
- and 112 relationships found between: COBIT (18 processes) and ITIL V3 (37 processes).

Given the space limits, it is not possible to show all comparisons performed; a detailed summary of the relationships found during the comparison between Basel II and COBIT 4.1 is set out in Annex 1.

4.3 Integrating the Models

On the basis of the results obtained in the comparison stage, the work group, comprised of the *performer* and *reviewer,* carried out the integration of relationships found between BASEL II and COBIT and after that, the integration of the relationships found in the other comparison iterations. That being so, the integrated model is based on the integration of the set of comparisons of the models involved; its process entities structure is comprised of 44 COBIT processes related to the operational risk principles defined in BASEL II. The integration of relationships (or related processes) found was carried out by (i) analyzing the results obtained in the comparison iterations and (ii) analyzing and identifying the activities needed to fulfill each purpose described in the 44 processes.

In each iteration, the integration was performed at the level of the PEs compared (processes and activities) in them. In addition, this stage followed an iterative and incremental approach. That made it possible to carry out the systematic management of the PEs involved and reduce the complexity coming from the integration of the descriptions of each model. Figure 4 shows the tasks diagram of the integration iterations performed.

In an effort to make the integration of descriptions between PEs easier, a set of rules or integration criteria was defined. It allowed us to know how to merge the descriptions in certain situations, e.g. (i) when the description of the process/activity which has less detail is supported and contained within the description of the procedure/activity that has greater detail and (ii) when the description of the procedure/activity with greater or less detail is not contained in the other process/activity. To apply these suitably, previous knowledge of the models involved and experience in the supervision of the banking sector of the *performer* were fundamental.

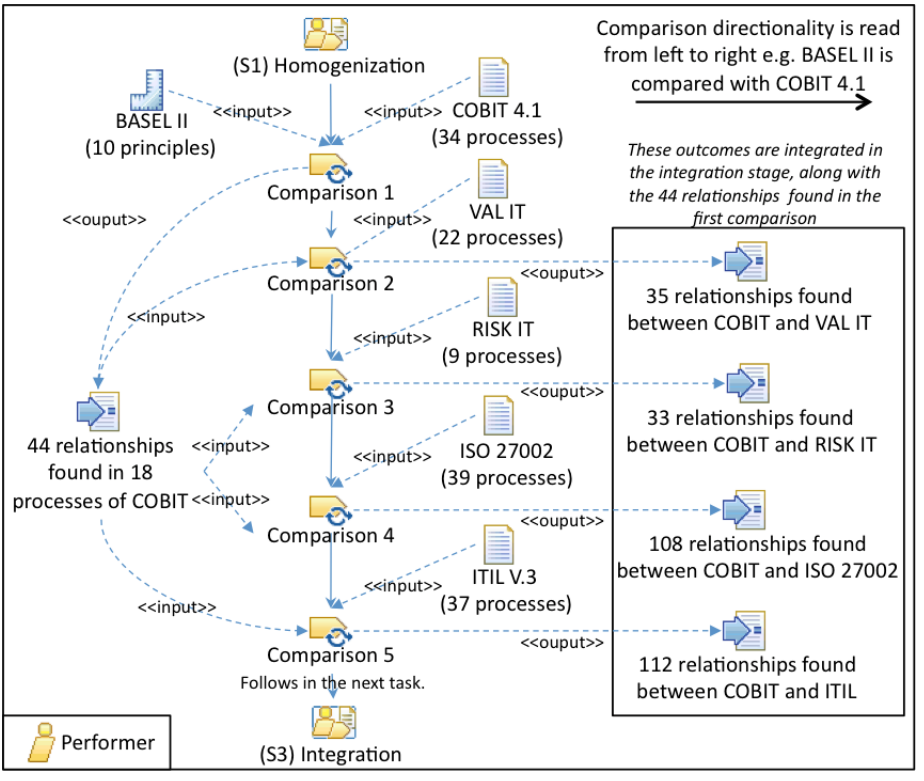


Fig. 3. Activity diagram of the comparison iterations

As a result of the execution of the harmonization strategy presented, a unified IT Governance Model for Banking, called ITGSM, has been obtained. It complies with operational risk principles established by BASEL II. In this way, ITGSM also consolidates the governance model proposed from the perspectives of: (i) Investment management IT - VAL IT, (ii) IT Risk Management - IT RISK, (iii) Management of information security - ISO 27002 and (iv) Life cycle management services - ITIL V3. Figure 5 shows the approaches, models and relationships that make up ITGSM. ITGSM defines 22 processes initially, which support the various approaches unified. We cannot show the unified model in its complete form here, due to limits on space. In Table 2, however, we present an extract, giving an overview of the structure of ITGSM. A detailed summary of ITGSM is presented in [21].

Although ITGSM's structure is oriented in one direction or another, according to the particular approach of each model, that itself makes it easy to maintain it if international bodies provide new versions of the integrated models. It will also be easy to adopt its practices and reflect the changes in the organizations' processes.

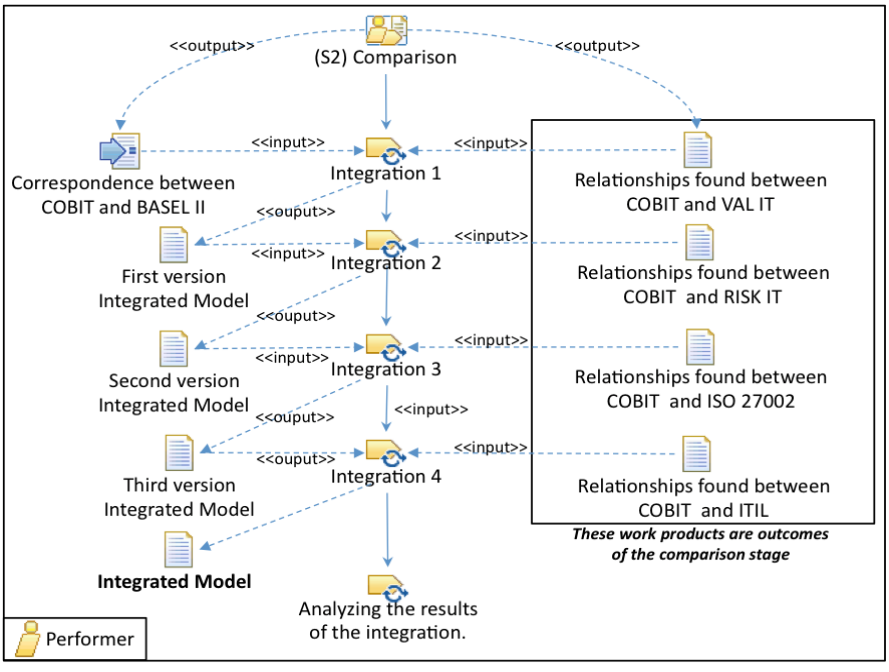


Fig. 4. Activity diagram of the integration iterations

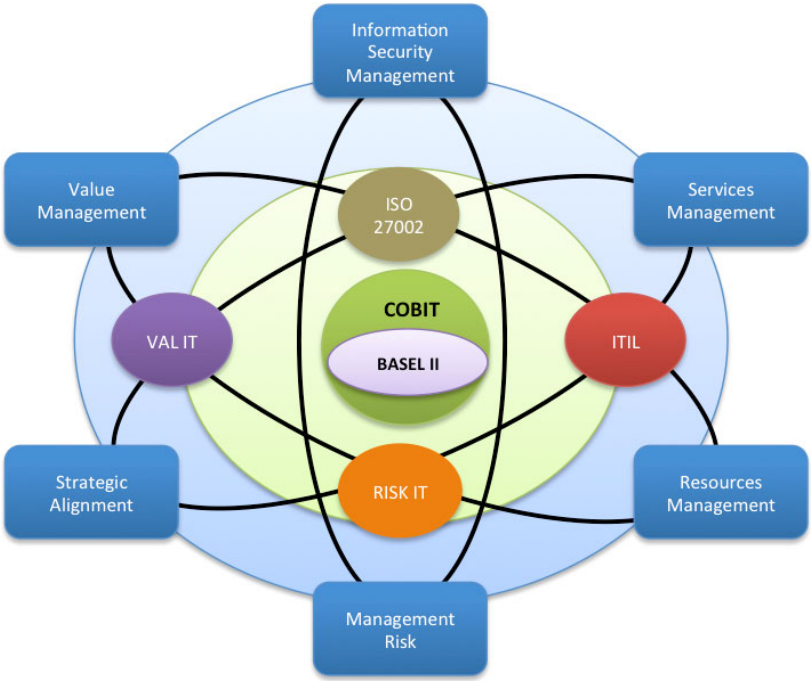


Fig. 5. IT Governance Model for Banking - ITGSM

5 Discussion and Conclusions

This paper has presented the harmonization strategy which has been designed to define an IT Governance Model for Banking, called ITGSM. The harmonization strategy has helped to organize and manage the work performed to obtain ITGSM through the configuration of three techniques: homogenization, comparison and integration, which have been joined in such a way as to take into account the harmonization objectives and specific needs of this research project. The systematic configuration of these techniques has allowed us to know "what to do", as well as "how to harmonize" BASEL II, VAL IT, RISK IT, ISO 27002, ITIL V3 and COBIT 4.1. To increase the reliability of results, this harmonization strategy used a reviewer who was present in each stage, who also helped to validate the results obtained and to resolve disagreements with the performer. The harmonization strategy has also made it possible to overcome some issues which arise when multiple models are integrated; these are:

(i). *There are different structures*. Since a harmonization strategy described a homogenization stage, it was possible to harmonize the structural differences between the integrated models and prepare them for the execution of other stages or techniques, in this case, to run comparison and integration techniques. Having the same structure of PEs between models involved has allowed us to carry out the comparisons and integrations between similar *process entities*. It has not been necessary to define rules of comparison to address the issues concerning structural differences.

Table 2. Structure of ITGSM (Extract)

PR	Process	**Activities**				
		IT governance	Management the IT investment	Specific risk management IT	Management of information security	Management the service lifecycle
PR 1	P1	PO4.2, PO4.3	VG1.4, VG1.5, PM1	RG2.1	Clauses 5.1, 6.1, 6.2, 8.1.1, 8.2.1, 15.1, 15.2	SD2.4.2, SS6.1, SO3.2.4
	P2	PO9.1, PO9.2	IM1.2	RG1.5, RG1.7, RG1.8, RG3.3, RG3.4, RE1.1, RR1.3, RR3.4	Clauses 4.1, 5.1, 13.1, 14.1.1	SS9.5
	P3	ME4.2	VG1, VG2.1, VG5	RG1, RG2	Clauses 5.1, 6.1.2, 10.1	SD3.10
PR 2	P	ME2.1, ME3.2	--NA--	RG1, RE2, RR1.2, RR1.3	Clauses 5.1, 6.1.8, 15.2, 15.3	--NA--
	P5	ME3.1, ME3.3	--NA--	--NA--	Clause 15.1	--NA--
PR3	P6	PO1.2, PO1.4	VG1.5, VG2.1, VG4, PM1, PM6	RG1, RG2	--NA--	SS2.1, SS2.3, SS3.3, SS4.1, SS4.2, SS4.4, SS5.5
	P7	PO4.1, PO4.8	VG2.4, VG2.6	RG1.2, RG2.4, RE1.1, RE3.1	Clauses 6.1, 6.2.1, 7.2, 8.1, 8.2, 8.3, 9.1, 9.2, 10.1.2	SS2.6, ST4, SO4
BASEL		**COBIT 4.1**	**VAL IT**	**RISK IT**	**ISO 27002**	**ITIL V3**

(1) BASEL II: PR= Principle, (2) COBIT 4.1: PO= Plan and Organize, ME= Monitor and Evaluate, DS= Deliver and Support, (3) VAL IT: PM= Portfolio Management, IM= Investment Management, VG= Value Governance. (4) RISK IT: RG= Risk Governance, RE= Risk Evaluation, RR= Risk Response, (5) ITIL V3: SD= Service Design, SS= Service Assets, SO= Service Operations, CSI=Continual Service Improvement, ST=Service Transition, (6) NA= Not Applicable

(ii). *The confusion caused by many-to-many comparisons.* The comparison technique used has allowed us to make the correspondences and identify the relationships clearly and concisely. We adjust the mapping template of [18] to establish the comparisons performed to define the integrated model.

(iii). *Complexity of harmonization.* Since six models at a low level of abstraction had to be integrated, using an iterative and incremental approach has made it possible to: (i) manage and reduce the complexity and scope iterations of each stage of the harmonization strategy that has been designed, (ii) carry out supervision by the reviewer in each iteration, (iii) obtain feedback quickly, (iv) measure the progress in short periods of time and (v) integrate the results obtained in each iteration continuously. The total effort spent during the execution of the harmonization strategy was 9880 minutes. This effort was expended between the performer and reviewers.

This work intends to support and guide organizations in homogenizing, comparing and integrating multiple models. Currently, we are working on the definition of a widespread harmonization strategy (WHS), which will offer companies a generic strategy to harmonize multiple models, regardless of their approach. It will be defined from the harmonization strategy presented in this article. It will likewise be implemented in other case studies, to validate its generality and adaptation according to needs in other contexts.

As a result of the execution of the harmonization strategy defined, ITGSM currently defines 22 processes and gives support to five different approaches: (i) IT Governance, (ii) investment management IT, (iii) IT risk Management, (iv) management of information security and (v) life cycle management services. In that sense, IGTSM can be useful for organizations which are planning to adopt practices concerning IT Management, IT Governance and Information Security Management, even if the organization does not have a BASEL certification. In the future, we will evaluate this model empirically in a case study, in an attempt to confirm its efficiency and assess the way in which information technology supports business and operational risk management.

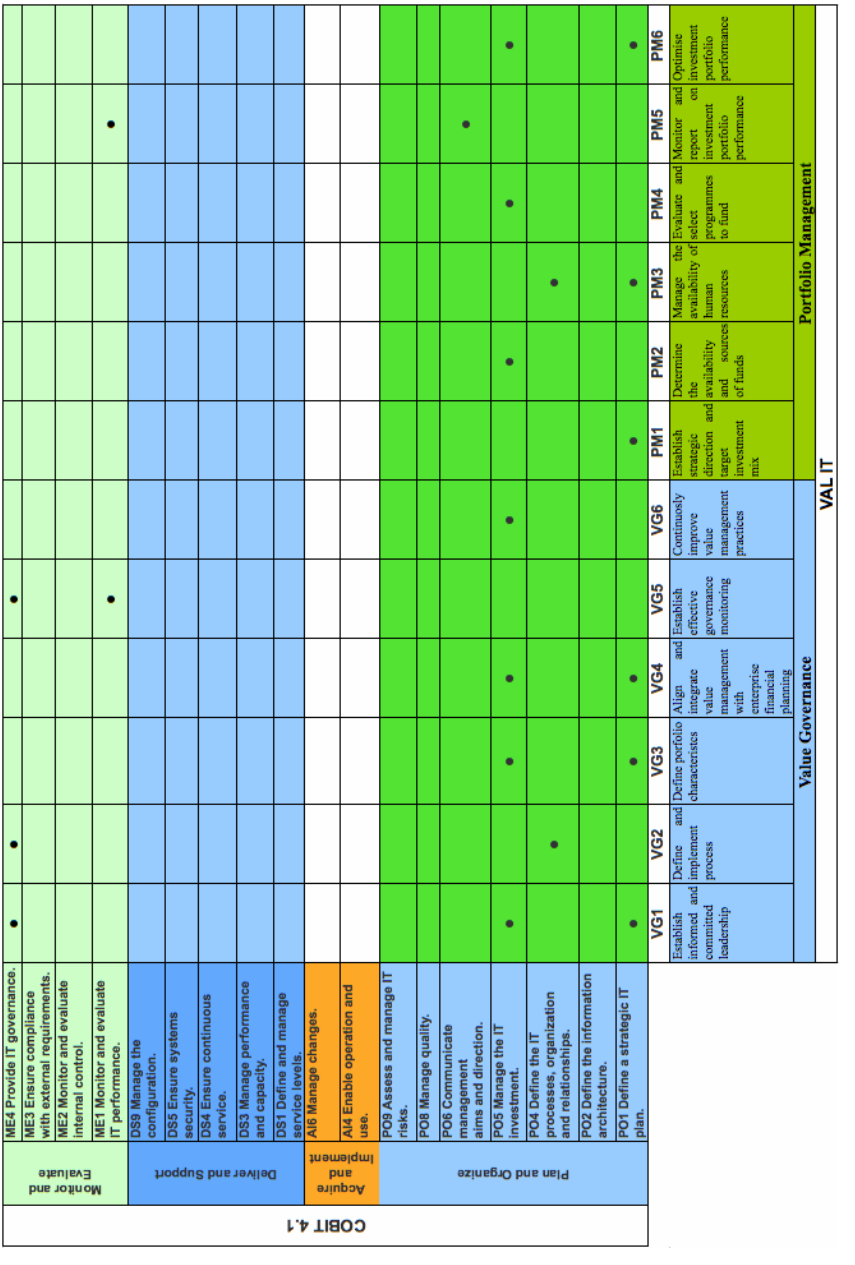
Acknowledgments. This work has been funded by the projects: ARMONÍAS (JCCM of Spain, PII2I09-0223-7948) and PEGASO/MAGO (MICINN and FEDER of Spain, TIN2009-13718-C02-01). Acknowledgements by Francisco J. Pino to the University of Cauca where he works as Associate Professor.

References

1. Pardo, C., Pino, F.J., García, F., Piattini, M., Baldassarre, M.T.: Trends in Harmonization of Multiple Reference Models. In: Evaluation of Novel Approaches to Software Engineering. LNCS, Springer, Heidelberg (2011) (in press) (Special edition best papers ENASE 2010, extended and updated paper)
2. Oud, E.J.: The Value to IT of Using International Standards. Information Systems Control Journal 3 (2005)
3. NIST: National Institute of Standards and technology, NIST (2011), `http://csrc.nist.gov/`

4. ITGI: Risk IT: Framework for Management of IT Related Business Risks. IT Governance Institute (2009), http://www.isaca.org/
5. ITGI: VAL IT Framework 2.0. IT Governance Institute, EEUU (2008)
6. ITGI: COBIT Mapping: Mapping of ITIL V3 with COBIT 4.1. Technical report, IT Governance Institute (ITGI) and Office of Government Commerce, OGC (2008)
7. ITGI: Aligning Cobit 4.1, ITIL V3 and ISO/IEC 27002 for Business Benefit. Technical report, IT Governance Institute (ITGI) and Office of Government Commerce, OGC (2008)
8. Gheorghe, M., Nastase, P., Boldeanu, D., Ofelia, A.: IT governance in Romania: A case study. Technical report, International Trade and Finance Association (2008)
9. Abu-Musa, A.: Exploring the importance and implementation of COBIT processes in Saudi organizations: An empirical study. Information Management & Computer Security 17, 73–95 (2009)
10. Kulkarni, B.: Banking Industry Regulatory Challenges: Moving From Regulation-based to process based Compliance. In: LNCS, pp. 4–8 (2009)
11. Haes, S.D., Grembergen, W.V.: An Exploratory Study into IT Governance Implementations and its Impact on Business/IT Alignment. Inf. Sys. Manag. 26, 123–137 (2009)
12. ITIL: Information Technology Infrastructure Library V3 (2010), http://www.itil-officialsite.com/
13. ISO: Information technology -security techniques- code of practice for information security management - ISO 27002:2005. International Organization for Standardization (2005), http://www.iso.org/
14. BIS: International Convergence of Capital Measurement and Capital Standards - Basel II. Bank for International Settlements (2004), http://www.bis.org
15. Pardo, C., Pino, F., García, F., Piattini, M.: Homogenization of Models to Support multimodel processes in Improvement Environments. In: 4th International Conference on Software and Data Technologies, Sofía, pp. 151–156 (2009)
16. Baldassarre, M.T., Caivano, D., Pino, F.J., Piattini, M., Visaggio, G.: A strategy for painless harmonization of quality standards: A real case. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 395–408. Springer, Heidelberg (2010)
17. Pino, F.J., Baldassarre, M.T., Piattini, M., Visaggio, G., Caivano, D.: Mapping software acquisition practices from ISO 12207 and CMMI. In: Maciaszek, L.A., González-Pérez, C., Jablonski, S. (eds.) ENASE 2008/2009. Communications in Computer and Information Science, vol. 69, pp. 234–247. Springer, Heidelberg (2010)
18. Pino, F., Balssarre, M.T., Piattini, M., Visaggio, G.: Harmonizing maturity levels from CMMI-DEV and ISO/IEC 15504. Journal of Software Maintenance and Evolution: Research and Practice 22, 279–296 (2009)
19. Pardo, C., Pino, F.J., García, F., Piattini, M., Baldassarre, M.T.: A Process for Driving the Harmonization of Models. In: The 11th International Conference on Product Focused Software Development and Process Improvement (PROFES 2010). Second Proceeding: Short Papers, Doctoral Symposium and Workshops, Limerick, pp. 53–56 (2010)
20. ARMONÍAS: A Process for Driving Multi-models Harmonization, ARMONÍAS Project (2009), http://alarcos.esi.uclm.es/armonias/
21. Lemus, S.M., Pino, F.J., Piattini, M.: Towards a Model for Information Technology Governance applicable to the Banking Sector. In: V International Congress on IT Governance and Service Management (ITGSM 2010), pp. 1–6. Alcalá de Henares (2010)

Annex 1: Extract of Mapping between COBIT 4.1 and VAL IT



Supporting Audits and Assessments in Multi-model Environments

Andre L. Ferreira[1], Ricardo J. Machado[1], and Mark C. Paulk[2]

[1] Departamento de Sistemas de Informação
Universidade do Minho, Campus de Azurém 4800 - 058 Guimarães , Portugal
{andre.ferreira,rmac}@dsi.uminho.pt
[2] Institute for Software Research
Carnegie Mellon University, Pittsburgh, PA, USA
mcp@cs.cmu.edu

Abstract. Software development organizations are adopting multiple improvement technologies to guide improvement efforts. A recent trend is the simultaneous adoption of CMMI and ISO models into a single environment originating multi-model process solutions. Some of these models address similar areas of concern and share similar quality goals. Reusing organizational implemented practices is an opportunity to establishing compliance with multiple models and reduce implementation costs.

Audits and assessments can take advantage of practices reuse if information characterizing similarities between quality goals of different models is maintained. This paper proposes a conceptual model to support management of quality goals information in support of multi-model audits and assessments.
An example is described of applying the proposed model in supporting the generation of data collection checklists to perform, in a single effort, a multi-model audit process. The conceptual model is being applied in a Portuguese software house with a multi-model process solution compliant with several improvement technologies.

Keywords: Multimodel Environments, Software Process Audits, Process Assessment, Software Process Improvement.

1 Introduction

Software organizations are adopting several improvement technologies to improve their overall performance. Improvement technologies is used in this paper as shorthand for reference models, quality standards, best practices or any type of practice based improvement technology. Adoption of these improvement technologies is driven by several reasons, namely: market pressure, the need to comply with regulations and performance improvement. A multi-model process solution results from adopting several improvement technologies into a single organizational environment. A recurrent combination is the simultaneous adoption of the best practice model CMMI-DEV [1] and the ISO9001 standard [2] into a single environment, originating a multi-model process solution.

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 73–87, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Some improvement technologies often address similar domains of concern defining similar expected practices and outcomes (hereafter referred as quality requirements). In a recent study the authors concluded that an high level of shared scope exists between improvement technologies frequently adopted in the software domain [3]. When this is the case, implementing practices to assure compliance with shared quality requirements is an opportunity to reduce costs of implementing multiple improvement technologies (hereafter also referred as quality models).

A related concern is that quality models change and evolve. As result organizations inevitably need to change their practices by adopting new or dropping obsolete practices. Evolution is a natural result of new releases of quality models, changes in regulatory requirements or even the decision to address new markets, which require market specific practices. In a managerial prescriptive, assuring traceability between implemented practices and quality requirements is a good practice, it assures that changes are traced back to implemented practices and manage the potential impact. However, assuring this type of alignment and, at the same time, take advantage of shared scope between adopted quality requirements is not straightforward. It requires identifying similarities between quality requirements and then manage how implemented practices related to quality requirements. An ad-hoc approach is prone to inefficiencies that can jeopardize compliance objectives and cost effective implementations of multiple models.

Thus, an approach is needed to help improvement groups in assuring traceability between multiple quality models and manage their change and evolution. Organizations in these scenarios are left with the challenge of assuring traceability of implemented practices with quality requirements from different quality models and manage information related to shared scope between quality requirements.

Siviy et.al. introduced the concept of harmonization in response to some challenges identified in multi-model environments [4]. A harmonization framework is outlined describing the general steps needed to choose, compose and implement multiple quality models. Specifically, one of the opportunities identified in harmonizing multiple quality models is to optimize costs in audits and assessments for operational units and projects. This paper addresses this issue considering that, if quality models share scope of concern, implemented practices and resulting outcomes can be used to establish compliance with multiple quality models. Audits and assessments will benefit from systematizing this information with the purpose of, in a single effort, collecting evidence to evaluate compliance to multiple quality models implementation.

The paper is organized as follows: Section 2 discuses related work in the area of harmonization and multi-model comparison and composition. Section 3 analyses background information related to assessment, audits and appraisals, and describes relevant considerations in the process of mapping models. Section 4 proposes a conceptual model to systemize information relevant to support multi-model audits and assessments. Section 5 describes an example of applying the proposed model and Section 6 concludes and outlines future work.

2 Related Work

Multi-model environments challenges and opportunities were documented by Siviy et.al. in [4]. Competition between improvement initiatives using several improvement technologies, when implemented separately, becomes costly and benefits are eroded

when compared to benefits of single efforts. Harmonization is introduced as a general approach to align different model into a single environment by means of a harmonization framework composed of four steps: 1) alignment of organizational and improvement objectives and identify improvement technologies, 2) categorize improvement technologies strategically, 3) design the improvement solution and 4) implement the multi-model solution and measure results.

One recurrent technique applied in multi-model scenarios is the model mapping technique. It is used to compare quality models with the purpose of finding associations between quality models by using a mapping function. Model mappings can be used, in the harmonization context, in support of selection and composition of quality models. The semantic associated to the mapping function determines the type of model comparison and composition. In recent literature, the most recurrent type of comparison, involves comparing model in terms of purpose and expected outcomes, also denominated *what/what* comparisons and named as *degree of relation* or *support* in [5] or *mapping confidence* in [6] and characterizes the amount of shared scope between models. Examples of these mapping exercises are provided by Pino et.al in [7-9].

In [10], Siviy *et.al*. identify tactical combinations between Six Sigma and CMMI-Dev 1.2 (hereafter shortened to CMMI). Elements of each considered improvement approach are compared and mapped to identify possible tactical combinations to drive process improvement. The semantics of the mapping function is now centered in identifying synergies between elements of process improvement initiatives, also denominated *what/how* combinations. The focus is on finding similarities in addressable scope by identifying synergies between improvement approaches.

Audits and assessments in multi-model environments will benefit from identifying similarities concerning purpose and expected outcomes of considered quality models. The model mapping technique can be used to obtain this type of information. To our knowledge, previous research has not considered how information on identified shared scope between quality requirements can be operationalized to support multi-model audits and assessments. That is the subject of this research work.

3 Multi-model Audits and Assessments

When organizations adopt a quality model, practices and requirements are interpreted according to organizational specific context and needs. Organizational specific practices are implemented aligned with adopted quality models. If a more formal approach is used to define organizational practices, e.g., to satisfy CMMI maturity level 3 goals, practices definitions need to be formalized using process models and/or process modeling languages, e.g., SPEM (Software & Systems Process Engineering Metamodel Specification version 2.0) [11] specification provides relevant process concepts for process definition. The result is an OSSP (Organizational Set of Standard Processes) that provides a collection of process and practices definitions to be enacted by the organization. From this set, project or organizational units define specific processes considering tailoring guidelines if applicable.

When organizations adopt several quality models, the sequence of model adoption becomes an issue that must be considered. One may choose a first model for adoption,

carry out an implementation of the chosen model and then choose a second model for implementation, following the implementation of the first model. Other possibility is to choose more than one model and plan a joint implementation. When adopting more than one quality model, harmonizing is beneficial to improve efficiency of joint implementations [4]. Harmonizing focuses on finding possible similarities and synergies of chosen models to facilitate and improve efficiency of joint implementations.

Multi-model environments will benefit from an explicit step for harmonizing models before practices are incorporated in the organizational environment. Fig. 1 depicts a high level interpretation (not exhaustive on concerns related to harmonization) of the harmonization framework introduced by Siviy *et.al.* in [12].

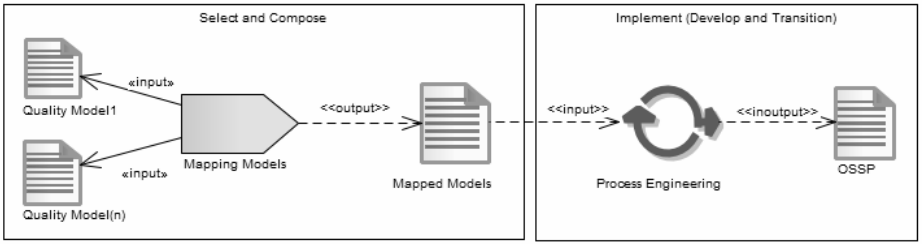


Fig. 1. High level process supporting harmonization

In the *Select and Compose* phase a *Mapping Models* task produces a mapping table by receiving as input different quality models. Quality requirements from considered quality models are compared and a mapping table is produced. The mapping table can be used as input to an engineering process to support development of a multi-model process solution. In *the Implement (Develop and Transition)* step, similarities and difference are identified and used to design practices and associated outputs alighted with harmonized quality requirements.

In designing our approach to support multi-model audits and assessments we considered the model mapping technique and the organizational scenario where an OSSP exists, providing detailed definitions on how the organization processes should be performed. This means the sequence in Fig. 1 has completed at least one iteration. The motivation to consider such scenario is twofold: first, although model mappings are subjective in nature, we considered publically available mappings by Mutafelija and Stromberg [13] [14] has a good example in identifying shared scope between ISO and CMMI quality models, due to their level of detail and completeness of comparison. The second reason is the fact research is being carried out in the context of a Portuguese software house with a multi-model process solution. Critical Software S.A. that recently achieved a CMMI maturity level 5 rating and complies with standards like ISO9001, Aerospace Standards 9100 and 9006 and ISO12207[15]. Our approach is based on the following assumptions: a mapping between quality models that considers shared scope as the semantic associated to the mapping function can provide a first level guidance on identifying possible reuse points for joint audits and/or assessments. Further, if an OSSP provides the necessary detail on how practices should be performed and these practices are aligned with one or more quality models, OSSP elements can be reused to improve efficiency of data collection tasks facilitating the implementation of audits and/assessments on a single effort.

The following subsections detail concerns related to model mappings and some considerations regarding audits, assessments and appraisals that will support the design of our conceptual model.

3.1 Model Mappings Considerations and Implications

One purpose of a model mapping exercise is to find similarities and differences between a pair of improvement technologies. A mapping involves pairs of models and a mapping function that relates entities of both models to deliver a mapping result. A mapping result is a set of relations categorized by the mapping function between every entity of one model to every entity of the second model. When mapping models, differences in structure need be considered to produce the mapping e.g., CMMI defines specific practices within process areas, ISO12207 uses activities and tasks and ISO 9001 uses shall statements. We are not considering a specific mapping between models so we abstract these structural differences and refer to them as quality requirements.

When executing a mapping with objective of providing support to joint audits and assessments, the following considerations assume central relevance:

1) A quality requirement from a quality model can share a "*scope of concern*" with one or more quality requirements from other models. The degree of the sharing or similarity can be characterized quantitatively or qualitative, e.g., a CMMI practice can share, with different degrees of similarity, scope with several ISO9001 shall statements. In practice, the mapping defines how much of one quality requirement when implemented can be re-used to support the implementation of a mapped quality requirement
2) The degree of similarity of scope between quality requirements of different models is not reflexive (*à priori)* – e.g., stating that a CMMI practice is related in a certain degree to an ISO 9001 shall statement is not the same as stating the mentioned ISO9001 shall statement is related to the CMMI practice in the same degree. This fact has been also mentioned in [7].

The first consideration assumes that a relation can be established between quality requirements to characterize the degree of shared scope. Whatever the scale used for characterizing the degree of relationship, the semantic associated should be how related are intended purpose and expected outcomes of compared quality requirements (product or service), e.g., the contents of the output can be used as evidence to demonstrate, partially or totally, the fulfillment of the compared quality requirement.

Fig. 2 depicts this relation where a quality requirement can be related to multiple quality requirements from different origins and each relation is characterized by a coverage value that translates the aforementioned semantic. The mapping between quality requirements defines dependencies between quality requirements, allowing identifying possible reuse points for evidence collection.

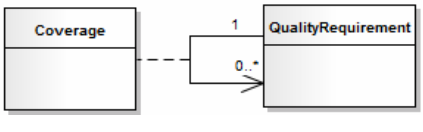


Fig. 2. Coverage between quality requirements

The second consideration states the self-association *(Coverage) in* Fig. 2 is not reflexive. This is a direct result of the type of semantic associated with the considered mapping. When relating a quality requirement to a second quality requirement from a different model and analyzing the degree of similarity of the intended output, one needs to consider that one quality requirement is fully implemented and compare how it relates to the mapped quality requirement, e.g., when comparing a CMMI specific practice purpose and expected output, one may assert that, if fully implemented, it can be used as evidence to satisfy an ISO 9001 shall statement. In this case CMMI assumes the role of reference model and ISO 9001 as the mapped model. The degree of similarity is characterized as the amount of reuse of the output of the CMMI implemented practice is expected to provide to satisfy the compared ISO 9001 shall statement.

When comparing quality requirements to identify shared scope the following scenarios may occur: in Fig. 3, the first Venn diagram from the left shows how a mapped requirement (transparent circle) can be partially (70 out of 100) covered by using a subset of the outcome of a reference quality requirement (grey circle).

The second from the left represents an example where full coverage is attained but the reference requirement can be said more extent in the scope it defines. The third diagram represents an example where the comparison can be considered reflexive; the scopes are similar and the outcomes are similar. Therefore, association between two requirements cannot be considered bi-directional *à priori* and is defined as unidirectional in Fig. 2. In the fourth Venn diagram no scope is shared between quality requirements.

When defining mappings to support a joint audit and/or assessments, choosing a quality model that provides the most detailed and most alighted requirements with organizational business needs may be considered a logical decision, e.g., a software company may consider CMMI as the reference model and ISO 9001 and ISO12207 as secondary models. Thus, the mapping should be established using as reference model CMMI and ISO9001 and ISO12207 as mapped models.

Fig. 3. Quality requirements mappings

3.2 Tracing Quality Requirements to Implemented Practices

Audits and assessments require objective evidence to establish conformance of implemented practices with reference standards, regulations, plans, specifications and capability frameworks and other relevant reference guidelines. Objective evidence is any result or byproduct of implementation or institutionalization of practices. Objective evidence is mentioned in ISO1028 IEEE Standard for Software Reviews and Audits [16], Standard CMMI Appraisal Method for Process Improvement [17] and ISO15504-2 - Process assessment [18] to represent any relevant work product that may be used to evaluate conformance.

In the previous section we discussed that quality requirements provide guidance for defining and implementing needed organizational practices. We also considered that quality requirements from different models may be compared by relating their expected outcomes and characterize then according their degree of similarity. This section discusses how implemented practices can be linked back to quality requirements for the purpose of supporting audits and assessments in multi-model environments.

According to IEEE Standard 1028 [18] the purpose of a software audit is to provide an independent evaluation of conformance of software products and processes to applicable regulations, standards, guidelines, plans, specifications, and procedures. Concerning evidence collection for evaluation purposes, the standard makes reference to interviews, examination of documents and witnessing processes as means to gather objective evidence of non-conformance or exemplary conformance. Audit observations are documented based on these objective evidence and are classified as major or minor. It does not provide any detail on how and where objective evidence should be looked for.

According to ISO 15504 [16] process assessments have two primary contexts for their use: process improvement and process capability determination. Process assessments aim to find strengths, weaknesses and risks inherent to processes providing drivers for improvement of processes. Process capability is determined by analyzing organizational processes against a capability profile. A capability profile is based on a measurement framework that defines a set of attributes that characterize the capability of a process to fulfill its goals.

Three entities are relevant in performing process assessments:

- A *measurement framework* provides the capability profile and is used to derive a capability rating.
- *A process reference model* or models e.g., CMMI or ISO 12207, provide the necessary process descriptions that will be used as frame of reference for organizational practices capability determination.
- An *assessment model* defines elements to relate processes of the process reference model(s) chosen as reference and the measurement framework process attributes to produce a capability rating. According to ISO 15504-5 - An exemplar Process Assessment Model, elements of the assessment model can be indicators of performance and capability.

A process assessment model forms a basis for the collection of evidence and rating of process capability. It requires to establish a mapping between organizational processes to be assessed and the process reference model(s) process definitions [16]. ISO15504-2 refers to the suitability of a process model as a function of the degree of focus of assessment model indicators on observable aspects of process enactment and the assessment model degree of alignment with relevant process reference model [19].

An *appraisal* is defined as an examination of one or more processes using as reference an appraisal reference model as a basis for determining, as a minimum, strengths and weaknesses [17]. It can be considered a type of assessment if it is performed internally by the organization. One underpin of the SCAMPI (Standard CMMI Appraisal Method for Process Improvement) appraisal method is the link between CMMI process goals and implemented organizational practices. Goals are

satisfied by gathering objective evidence of each generic and specific practice implementation. Objective evidence is expected to be from different types e.g., oral affirmations must be collected to support objective evidence concerning practices implementation. The SCAMPI method defines the concept of practice implementation indicator to support the evaluation of practices implementation. A practice is considered implemented when direct artifacts, indirect artifacts and affirmation are gathered that provide substantiate evidence of practices implementation. Direct and indirect artifacts can be documents and affirmations are oral or written statements that result from interviews presentation or demonstrations.

Based on the analysis of audits and assessments approaches the concept of indicator and the concept objective evidence assume a central importance. Indicators are an abstract representation to group objective evidence of organizational practices implementation and establish the association between performed processes and measurement attributes, if a capability assessment is to be performed. Also, affirmations are obtained manly from interviews and are required to substantiate and provide objective evidence of implemented practices. Based on this highlighted concepts, the next section elaborates on a model that relates relevant entities in support of multi-model audits and assessments.

4 Multi-model Audits and Assessments

As discussed in the previous sections, quality requirements of different model can be related by the amount of shared scope. Purpose and expected outcomes are compared to characterize their degree of similarity. Quality requirements also provide motivation and guidance to define organizational practices. Those are interpreted considering organizational context and needs to define the most adequate set of practices in achieving desired business goals.

In the context of multi-model environments, performing an evaluation of areas of concern related to different quality models in a single audit or assessment can reduce costs and improve efficiency of audits and assessments. e.g., in a single exercise evaluate process compliance to ISO12207 and CMMI by reusing collected evidence. In order to reuse collected evidence one needs to identify which artifacts are shared among different quality requirements. This is possible by defining maps between organizational practices and quality requirements, allowing to list artifacts relevant to a specific quality requirement implementation. By considering coverage associations between quality requirements it allows identify which artifacts can be shared among related quality requirements.

The meta-model in Fig. 4 introduces relevant entities and how these relate to each other in support of audits and assessments in multi-model environment. A *QualityRequirement* is associated to zero or more *QualityRequirement* entities of different origin, e.g., one can map an ISO9001 shall statement to several CMMI specific practices. The association between quality requirements is set by *Coverage* association, defining the degree of shared scope between *QualityRequirement* instances. As an example, in a mapping between ISO9001 and CMMI , an ISO shall statement, *Establish QMS,* maps to 29 specific practices of CMMI with different coverage values, defined by a scale of comparison that can assume values of 0,30,60,100.

Considering that a formal definition of organizational practices is provided, e.g., an OSSP describing with detail performed practices and expected artifacts, one can use this information and establish an association between quality requirements and artifacts defined in the OSSP. An *Artifact* refers to any tangible process element used to describe or maintain process related information, e.g., an artifact can be a Work Product Definition, Task Definition and other process constructs if e.g., SPEM specification is used as a modeling language to define an OSSP.

An *Indicator* is used to group relevant process related artifacts defined in the OSSP, which are expected to provide objective evidence of quality requirements implementation. This step requires that a mapping between artifacts and related quality requirements is established, e.g., in support of specific practices of CMMI process areas, a set of relevant work products and task descriptions are identified that are expected to provide evidence of practice implementation, when these are enacted by project or organizational units.

With mappings established between OSSP artifacts and quality requirements with coverage associations between quality requirements defined, artifacts used as evidence for a quality requirement implementation can be reused also as evidence for mapped quality requirements, e.g., artifacts associated with CMMI specific practices implementation can be reused to provide objective evidence of ISO9001 shall statements which are mapped to CMMI specific practices.

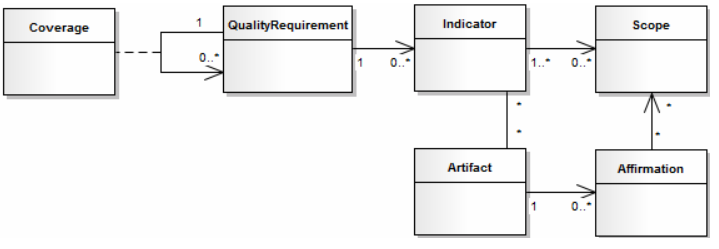


Fig. 4. Traceability between quality requirements and implemented practices

Both audit and assessment standards make reference to the need of having supporting oral of performed practices from practice implementers. The element *Affirmation* is associated to *Artifact* to emphasize that artifacts require oral or written statements as supporting objective evidence. An affirmation is a type of objective evidence to confirm artifact related evidence. An *Affirmation* instance is expected mainly as result of interviews when assessments and audits are performed.

The proposed model includes the concept of *Scope* to make explicit the notion that audit and assessment may consider different scopes. A *Scope* instance has always an associated *Indicator* instance which is always associated to a *Quality Requirement* instance. By choosing relevant quality requirements from different quality models, associated indicators are automatically identifiable, whether these are obtained directly by the *Indicator/QualityRequirement* association or indirectly by the Coverage association defined between *QualityRequirement* instances.

An *Affirmation* instance can be associated to multiple scopes. This allows defining different affirmation instances related to a same artifact, providing flexibility in defining different elements to support collection of oral or written statements for different scope scenarios.

Fig. 5 depicts an example of a generic joint multi-model audit scenario based in the perspective of the model proposed in Fig. 4. The two left columns depict six quality requirements (QR) considered for a desired scope (not shown in the diagram). The first column represents mapped quality requirements and the second column represents quality requirements from a reference model. Coverage (C) associations are established between selected quality requirements. The quality requirements in the second column have associated indicators (I) defined. Each indicator results from identifying relevant artifacts (A) that are expected to provide objective evidence of implemented practices. Indicators are represented in the third column with associated artifacts. It is possible to verify that different indicators may reuse artifacts as evidence for different quality requirements implementation. This is possible as it depends how practices and expected outcomes are implemented in the organizational environment.

From the mappings defined between quality requirements it is possible to identify QR(1), QR(2) and QR(3) from the mapped model have coverage associations defined to QR(4) and QR(5) of the reference model, respectively. Five coverage associations are defined with values of C(100), C(100), C(100) and C(60). The notation C(X) is used to describe the *Coverage* association instead of an actual object to simplify the object diagram. QR(1) and QR(2) can reuse artifacts from indicators I1 and I2 of QR(4) and QR(5) respectively. QR(3) has only a portion of reused scope with QR(5) and requires an indicator (I3) that identifies the set of artifacts to assure full compliance coverage of QR(3).

The fourth column represents affirmations instances associated to artifacts for each indicator. Questions are a possible type of affirmations that can be defined and maintained by internal quality teams or process improvement groups, to use in obtaining required oral or written statements as support of objective evidence, e.g., obtain statements if a work product or activity description is implemented as expected.

In support of software audits, the model in Fig. 4 can be used to define multiple audit scenarios, e.g., in performing project or process audits one may define different types of audits and chose different scopes for each type. The scope of the audit is defined by identifying relevant indicators which are associated to quality requirements from multiple quality models. Question can be maintained as instances of affirmation which can be associated to multiple different scopes

In the specific context of assessments and using as example the assessment model proposed in ISO 15504-5 [19], an assessment model indicator is refined into performance and capability indicators. We opted to not include this level of refinement in the conceptual model by considering that it depends on the method defined for the assessment model. By considering solely the concept of indicator we leave the possibility of extending the concept of indicator to support possible different assessment methods, e.g., by considering different measurement frameworks and associated capability indicators.

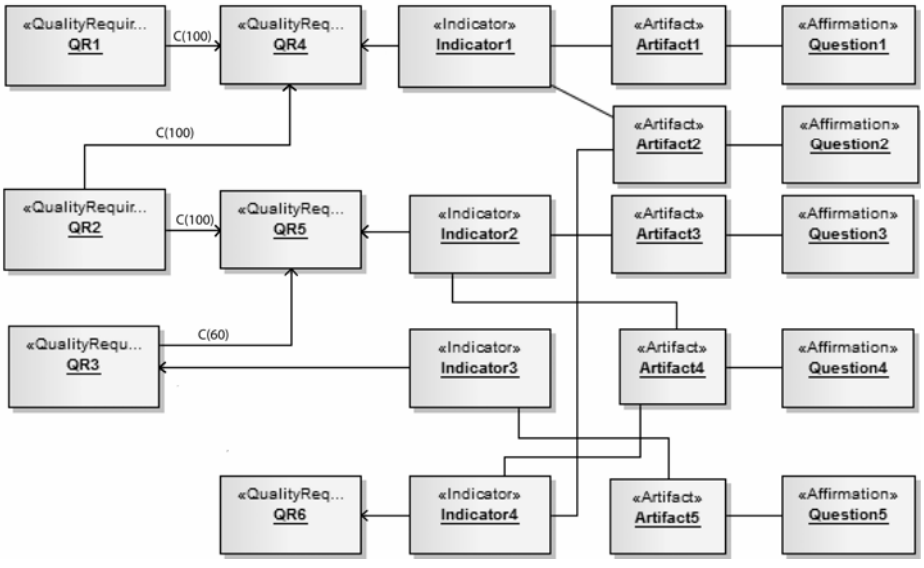


Fig. 5. Generic joint audit or assessment scenario

5 Multi-model Process Audit Example

This section details the use of the conceptual model presented in the previous section in support of a joint audit exercise. An organizational scenario is provided to describe how it can be applied. ISO12207, ISO9001 and CMMI are considered to exemplify a multi-model process environment and highlight the benefits of considering models similarities in support of multi-model audits.

The model presented can be used in the context of QM (Quality Management) activities and in the scope of Audit Process activities. In the scope of a QM process one expects to identify possible similarities between quality models and then proceed to identifying which OSSP process assets can be used in the process of determining compliance with considered quality models. Fig. 6 depicts example QM related tasks of *Model Mapping* and *OSSP and QR mapping* defined using SPEM 2.0 notation. The output of the *Model Mapping* task are mapping tables where quality requirements from different quality models are mapped and a coverage function is used to characterize their relationship, considering their similarity in terms of purpose and outcomes. To perform this task one needs to determine one of the quality models as the reference model. By choosing a reference quality model the direction of the relationship for the mapping function is determined. As an example CMMI will be considered as the reference model and ISO12207 and ISO9001 the mapped models. The mappings between ISO12207 and ISO9001 to CMMI in [13, 14] provide output examples of the *Model Mapping* task for this type of scenario.

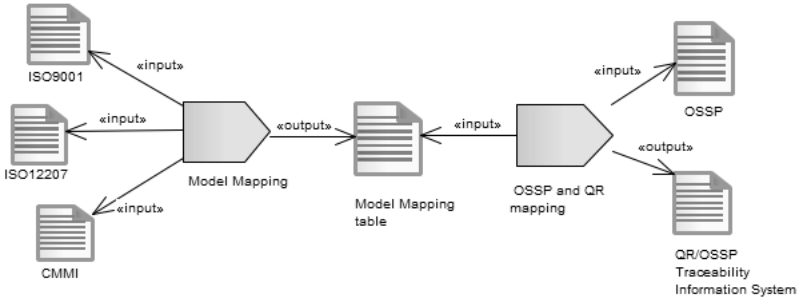


Fig. 6. Quality Management process

The resulting mapping tables are used as input to *OSSP and QR mapping* task along with process definition related information. The expected output is an information system based on the conceptual model described in the previous section.

The *OSSP and QR mapping* task includes the following steps:

1) The mapping tables are used to create *QualityRequirement* and *Coverage* instances. First, all specific practices of CMMI, ISO12207 activity and tasks and ISO9001 shall statements originate a *QualityRequirement* instance. The mappings resulting from the previous task are used to define *Coverage* instances between *QualityRequirement* instances.
2) For all *QualityRequirement* instances of the model considered as reference, an *Indicator* instance is defined by identifying relevant *Artifacts* in the OSSP, e.g., if SPEM is used as process modeling language to define the OSSP, SPEM constructs like *Task Definition*, *WorkProduct Definition*, *Activity,* among others, can be used as instances of type *Artifact* to define indicators for each specific practice of CMMI.
3) For all remaining *QualityRequirement* instances, not belonging to the reference model, an analysis is required to evaluate if *QualityRequirement* related instances (defined by a *Coverage* instance) do include all relevant artifacts in the OSSP that can be useful in supporting desired compliance. This is a vital point in the process of mapping quality requirements with OSSP process entities. It allows reusing most of information regarding mapped quality requirements and takes full advantage of shared scope between adopted models. If mapped quality requirements do not provide full coverage, additional indicator instances need to be defined identifying missing relevant OSSP artifacts, e.g., if an ISO12207 activity is fully covered by related CMMI specific practices it can reuse artifacts identified by the indicators associated to CMMI specific practices and still require extra artifacts to support full compliance for the activity considered.

Fig. 7 depicts an example (not exhaustive) of a QR/OSSP Traceability Information System, describing associations on shared scope between quality requirements and OSSP process related entities. The first and second columns represent quality requirements from ISO9001 and ISO12207 respectively, along with their coverage association with CMMI practices, which are represented in the third column.

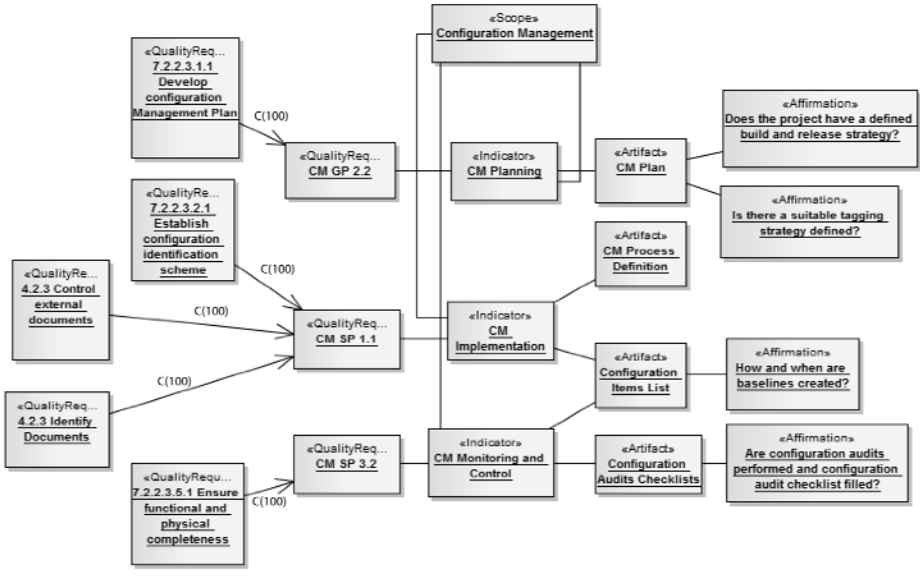


Fig. 7. QR/OSSP Traceability Information System example

The QR/OSSP Traceability Information System can be used to support the joint model audit process. As an example, an audit scope can be defined to include, among others, evaluation of CM (Configuration Management) process area of CMMI and CM process from ISO12207. Checking the mapping table in [14] used to originate *QR and the QR/OSSP Traceability Information System*, (see Fig. 6) it is possible to check that all ISO 12207 CM activities and tasks have full coverage by CMMI CM specific practices. Indicators defined to support CMMI CM practices data collection could be reused to guide data collection for compliance with ISO12207 CM process. Within a scope set to include CM practices from CMMI and ISO12207 an audit checklist template for data collection can be easily defined in an audit planning task. The checklist is defined by selecting the *Configuration Management* Scope, which lists the audit questions and expected artifacts that could provide evidence of practices implementation (see Fig. 7).

Further, to optimize the audit process, questions can be added by internal auditors concerning a specific scope to help gathering objective evidence on performed practices. These questions become instances of type *Affirmation*, which become associated to artifacts of the OSSP and the scope defined for a specific type of audit or assessment. This association allows defining different questions for different scopes relative to a same artifact. By maintaining information regarding *Scope* and *Affirmation* instances it becomes simple to manage data collection checklists in support of multi-model audits and/or assessments.

This section provided a small example on how joint audits can be performed reusing most of the effort of evidence collection. In [3] a quantitative evaluation is performed based on the mappings considered for this example that allowing to conclude that ISO9001 and ISO12207 share with CMMI 83% and 74% of their scope

respectively. This provides a measure on the amount of effort that could be optimized when performing full compliance evaluations for multiple quality models. Our conceptual model is designed to dispose information concerning shared scope between quality requirements and how, in the presence of OSSP formal definition, process related artifacts are being used in support of QRs implementation. By maintaining information aligned with the proposed conceptual model, one can improve quality management activities in multi-model environments by:

- Precisely identify which OSSP related artifacts are involved in quality requirements implementation.
- Define indicators that operationalize the information related to shared scope between quality requirements from different quality models.
- Identify which quality requirements are affected by possible changes in process related artifacts.
- Identify which organizational practices can be dismissed by dropping specific quality requirements
- In the context of a project enactment system based on OSSP definitions, project audits and capability assessments can be partially automated. This is possible by monitoring process enactment of by collecting objective evidence of performed practices.

6 Conclusions

In this paper a conceptual model to support management of information related to quality requirements of multiple improvement technologies was presented. The main goal is to support audits and assessments of multiple improvement technologies in a single effort. The underlining motivation is that different improvement technologies often share scope of concern providing the opportunity to reuse evidences of organizational practices implementation in evaluating compliance to multiple quality models.

The proposed conceptual model is based on the model mapping technique and in the concept of performance indicator. The mapping is used to compare quality requirements from different improvement technologies and evaluate their degree of similarity concerning purpose and expected outcomes. The concept of indicator is used to group organizational process entities involved in quality requirements implementation, which can be used to guide the collection of objective evidence of their execution.

The model aims to help improve internal quality management capability in managing multi-model internal audits and assessments. An external multi-model certification scheme is impossible as certification bodies yet do not acknowledge certifications from other certification bodies.

A small example of using the concepts introduced in this paper is provided to exemplify how the model can be useful in providing support to a joint process audit considering ISO9001, ISO12207 and CMMI. The conceptual model is currently being implemented in a Portuguese software house, Critical Software S.A., with the objective to improving the efficiency of conducting project and organizational audits. A series of experiments are in progress to further validate the proposed model.

References

1. Chissis, M.B., Konrad, M.: CMMI for Development, V1.2. Addison-Wesley, Reading (2006)
2. ISO, ISO 9001:2000 Quality Management Systems - Requirements (2000)
3. Ferreira, A.L., Machado, R.J., Paulk, M.C.: Quantitative Analysis of Best Practices Models in the Software Domain. In: 2010 17th Asia Pacific in Software Engineering Conference, APSEC (2010)
4. Siviy, J., et al.: Maximizing your Process Improvement ROI through Harmonization (2008), http://www.sei.cmu.edu/process/research/prime.cfm(accessed February 2011)
5. Baldassarre, M., et al.: A Strategy for Painless Harmonization of Quality Standards: A Real Case. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) Product-Focused Software Process Improvement, pp. 395–408. Springer, Heidelberg (2010)
6. Mutafelija, B., Stronberg, H.: Systematic Process Improvement Using ISO 9001:2000 and CMMI. Artech House, Boston (2003)
7. Pino, F.J., Baldassarre, M.T., Piattini, M., Visaggio, G., Caivano, D.: Mapping Software Acquisition Practices from ISO 12207 and CMMI. In: Maciaszek, L.A., González-Pérez, C., Jablonski, S. (eds.) ENASE 2008/2009. Communications in Computer and Information Science, vol. 69, pp. 234–247. Springer, Heidelberg (2010)
8. Pino, F.J., et al.: Harmonizing maturity levels from CMMI-DEV and ISO/IEC 15504. Software Process: Improvement and Practice (2009)
9. Baldassarre, M.T., et al.: Comparing ISO/IEC 12207 and CMMI-DEV: Towards a mapping of ISO/IEC 15504-7. In: ICSE Workshop on Software Quality, WOSQ 2009 (2009)
10. Siviy, J., Penn, M., Stoddard, R.W.: CMMI and Six Sigma Partners in Process Improvement. Addison-Wesley, Reading (2008)
11. Object Management Group, Software & Systems Process Engineering Metamodel Specification (SPEM) (2008)
12. SEI, Siviy, J., et al.: The Value of Harmonizing Multiple Improvement Technologies (2008)
13. Mutafelija, B., Stromberg, H.: ISO 9001:2000 to CMMI v1.2 Map (2009), http://www.sei.cmu.edu/cmmi/casestudies/mappings/cmmi12-iso.cfm (accessed February 2011)
14. Mutafelija, B., Stromberg, H.: ISO 12207:2008 to CMMI v1.2 Map, SEI, Editor (2009)
15. ISO/IEC 12207 - Systems and software engineering - Software life cycle processes. ISO/IEC and IEEE (2008)
16. ISO, ISO/IEC 15504-2 Software engineering — Process assessment — Part 2: Performing an assessment in ISO/IEC(2004)
17. SEI, Standard CMMI Appraisal Method for Process Improvement (SCAMPISM) A, Version 1.2: Method Definition Document (2006)
18. IEEE, IEEE Standard for Software Reviews and Audits, in IEEE STD 1028-2008 (2008)
19. ISO, ISO/IEC 15504-5 Information Technology — Process assessment — Part 5: An Exemplar Process Assessment Model (2006)

Scrum Practices in Global Software Development: A Research Framework

Emam Hossain, Paul L. Bannerman, and D. Ross Jeffery

NICTA, Australian Technology Park, Sydney, Australia
UNSW, The University of New South Wales, Sydney, Australia
{Emam.Hossain,Paul.Bannerman,Ross.Jeffery}@nicta.com.au

Abstract. Project stakeholder distribution in Global Software Development (GSD) is characterized by temporal, geographical and socio-cultural distance, which creates challenges for communication, coordination and control. Practitioners constantly seek strategies, practices and tools to counter the challenges of GSD. There is increasing interest in using Scrum in GSD even though it originally assumed collocation. However, empirically, little is known about how Scrum practices respond to the challenges of GSD. This paper develops a research framework from the literature as a basis for future research and practice. The framework maps current knowledge and views on how Scrum practices can be used to mitigate commonly recognized challenges in GSD. This research is useful as a reference guide for practitioners who are seeking to understand how Scrum practices can be used effectively in GSD, and for researchers as a research framework to validate and extend current knowledge.

1 Introduction

Global Software Development (GSD) is a major trend in software engineering. Rapid advances in computer networks, telecommunications, internet technologies and collaboration tools have provided enough infrastructures to enable firms to take advantage of high-skilled, low-cost offshore resources in developing and maintaining new software. It is claimed that GSD can reduce time to market, increase productivity, improve quality, and provide cost efficiencies for software developing organizations [28]. However, as well as expected benefits, GSD challenges have also been identified [29]. In particular, GSD is usually characterized by engagements with different national and organizational cultures in different geographic locations and time zones, using various traditional and IT-enabled means to collaborate. Such arrangements may encounter major difficulties in team communication, coordination, control, infrastructure compatibility, conflicting expectations, achieving shared understanding and building trust [8]. Therefore, GSD practitioners need to employ suitable context-specific mechanisms to mitigate these problems. Various solutions have been proposed, such as dividing work into separate well-specified independent modules to minimize communications between sites [30]. However, in practice, practitioners still experience significant issues and continue to look for more effective mechanisms to mitigate inherent GSD challenges and risks [31].

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 88–102, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Agile methods are gaining popularity in software development contexts characterized by high uncertainty. They promise handling requirements changes throughout the development lifecycle; promote extensive collaboration between customers and developers; and support early and frequent delivery of products [32]. However, an issue for GSD is that the design of Agile methods assumes collocated development teams whereas GSD assumes distributed teams. Notwithstanding this, practitioners have begun to adapt Agile practices to GSD to leverage the advantages of both approaches [33]. Among the current Agile methods, Scrum practices are increasingly being considered and trialled in GSD projects. Initial empirical findings suggest that using Scrum practices in GSD can improve communication, trust, motivation and product quality [1]. Also, industry experience suggests that Scrum practices may promote communication, collaboration, frequent product delivery, and reduce some GSD challenges and risks [8, 24].

We acknowledge that GSD challenges apply to any software development method used in GSD projects, including non-Agile methods, and that Scrum may share some mitigation mechanisms, including complementary enabling tools, with other methods. However, our focus in this study is solely on Scrum practices, not other methods or the relative capabilities of Scrum versus other software development methods.

Little empirical research has been done on how Scrum practices can mitigate the challenges presented by the distributed nature of GSD projects [34]. A small number of case studies discuss the benefits and difficulties of using Scrum practices in GSD [1, 2], but most reports in the literature record only industrial experiences. Therefore, a knowledge gap exists. There is a need for more empirical studies on how Scrum is being used in GSD and how Scrum practices might be applied and with what tool support to overcome the challenges encountered in GSD.

To begin to address this gap, we propose a research framework that maps current literature-based prescriptions on how Scrum practices might mitigate commonly recognised GSD challenges. The framework draws on views from 27 primary papers that were identified in a Systematic Literature Reviews (SLR), following the guidelines prescribed in [35].

Since most current software engineering literature on the topic comprises industry experience reports, the proposed framework offers two main contributions: first, it provides a consolidated representation of current views on how Scrum practices might be applied to overcome the recognized challenges of GSD, and; second, it provides an integrated set of initial propositions about the potential role of Scrum practices in GSD that can be validated in subsequent empirical studies and used to guide practice.

The remainder of this paper is organized as follows. Section 2 overviews the literature and describes the framework development. Section 3 describes the research framework. Section 4 then discusses the contribution and future research before concluding in the last section.

2 Research Background

This section overviews the literature on GSD challenges and Scrum practices in GSD as background to describing the development of the proposed research framework.

2.1 GSD Challenges

There is a growing body of literature that focuses on challenges in GSD. Communication, coordination and control challenges are recognised to arise due to geographic, temporal and socio-cultural distances encountered in GSD [33]. A recent literature review also identified group awareness, knowledge management, coordination, collaboration, project and process management, process support, and risk management as likely issues in distributed software development [28].

Temporal distance may create communication issues such as reduced hours of collaboration, difficulties in holding synchronous meetings, and response delays [40]. As a result, GSD coordination processes may also be significantly affected [33]. *Geographic distance* makes communication difficult because of the reduced ability to hold face-to-face meetings [32]. Lack of face-to-face meetings reduces informal contact and this can lead to a lack of critical task awareness, lack of "teamness" and reduced trust [32, 37]. Therefore, a fundamental GSD problem is that many of the mechanisms that function to coordinate work in a collocated setting are absent or disrupted [12]. *Socio-cultural distance* may create issues relating to inconsistent work practices, different perceptions of authority, and lack of mechanisms for creating shared understanding and avoiding misunderstandings and reduced cooperation [1, 32, 33, 38, 44].

Several GSD issues frameworks and models already exist in the literature [24, 33, 34, 39, 40]. For example, one maps communication, coordination and control process issues against temporal, geographical and socio-cultural distance dimensions [33]; others present risk frameworks for the use of Agile practices in GSD [24, 31]; and another presents a model of how remote communication and knowledge management, cultural diversity and time differences negatively impact requirements engineering in GSD [39]. Our proposed framework integrates the GSD issues classification scheme of [33] as one dimension and uses Scrum practices [1] as the other dimension.

2.2 Scrum Practices in GSD

Scrum is an iterative, time-boxed, incremental project management method based on a simple "inspect and adapt" framework [29]. A major reason for the success of Scrum is the physical collocation of development team members [32]. However, the literature also reports instances of success in using Scrum practices in GSD [34]. For example, a recent study found that using Scrum practices in GSD improves communication, trust, motivation and quality [1]. Some reports also claim that some Scrum practices can mitigate some of the recognized GSD challenges [8, 24]. For example, based on experience, one report claims that Scrum practices such as daily scrum, scrum of scrums, sprint planning and retrospective meetings engage distributed team members in collaboration, help visualization of hidden problems, develop trust and increase team spirit [23]. Similarly, [8] claims that sprint planning provides shared visualization of project activities and increases "teamness". Furthermore, [21] suggests that daily scrum meetings bring transparency and encourage informal communication among distributed stakeholders; sprints provide frequent offsite work monitoring opportunities; sprint planning meetings provide shared understanding of common goals and improve task awareness, and; sprint 'demos' bring transparency to stakeholders and prevent problems early.

Our proposed research framework draws on the Scrum practice set identified in one of the few empirical studies of the use of Scrum in GSD, namely [1].

2.3 Research Framework Development

In sum, a review of the developing literature on the use of Scrum practices in GSD indicates encouraging results [34]. However, there is a paucity of empirical studies in the literature. Most current studies are industry experience reports so there is a need for further empirical research.

An initial way forward is to construct a research framework from the current reports as a basis on which to empirically validate and build on current knowledge and experience. The framework is intended to integrate current literature-based results and views about how Scrum practices have been found to be effective (or are expected to contribute) in mitigating the challenges that arise in GSD. We propose such a framework and approach in the remainder of this paper.

The framework, in Table 4, shows GSD challenges as rows and Scrum practices as columns. The rows list challenges facing software development in global contexts identified in [33] and [40]. Twelve GSD challenges were identified from [33] and [40] in three broad categories (communication, coordination, and control), each relating to a particular source characteristic (temporal, geographical or socio-cultural distance). These challenges are summarized in Table 1, adapted from [33] (and each challenge is further described in [40]). For example, *reduced opportunities for synchronous communication* are attributed to communication and feedback delays due to offset time zones between locations (refer to [40] for the remainder).

Table 1. Summary of GSD challenges

Process Category	Challenge	Source Category
Communication	Reduced opportunities for synchronous communication	Temporal distance
	Face to face meetings difficult	Geographical distance
	Cultural misunderstandings	Socio-cultural distance
Coordination	Increased coordination costs	Temporal distance
	Reduced informal contact can lead to lack of critical task awareness	Geographical distance
	Inconsistent work practices can impinge on effective coordination	Socio-cultural distance
	Reduced cooperation arising from misunderstanding	Socio-cultural distance
Control	Management of project artefacts may be subject to delays	Temporal distance
	Difficult to convey vision and strategy	Geographical distance
	Perceived threat from training low-cost 'rivals'	Geographical distance
	Different perceptions of authority can undermine morale	Socio-cultural distance
	Managers must adapt to local regulations	Socio-cultural distance

The columns in the reference framework in Table 4 list seven Scrum practices identified in an empirical study of the use of Scrum in GSD [1]. These are: sprint planning meeting, sprints, daily scrum, scrum of scrums, sprint demo (or review), retrospective meeting, and backlog (refer to [1] for a description of these practices).

The table cells are populated from our literature review, with the literature-based challenge mitigation mechanisms summarized in categories. The categories, described in Tables 2 and 3, were derived as follows. The initial SLR [34] and a subsequent

search using the same protocol found 27 primary papers [1-27]. These papers were examined for empirical evidence, unsupported theory, and/or observations from experience indicating how Scrum practices might mitigate GSD challenges. The search resulted in 202 scrum-related mechanisms being identified. Due to the large number, the mechanisms were then examined to find common themes, which enabled them to be clustered into eight categories. The categories are described in Table 2. The primary papers that identified one or more mechanisms within a category are marked with an X in Table 3 (a spreadsheet of the full detailed list of practices, reference sources and categorizations is available from the first author upon request).

The search protocol, paper selection and practice extraction and categorisation were enacted, validated and cross-checked by two researchers.

Table 2. GSD Challenge Mitigation Mechanism Categories

Category	ID	Description
Synchronized work hours	GSD_P1	Increase overlapping working hours between sites to enable synchronous communication for meetings; for example, adjust working hours at sites to create some overlap or participate in meetings from home
ICT-mediated synchronous communication	GSD_P2	Practices that enable synchronous formal or informal communication between teams; for example, use individual or conference phone calls, teleconference, video conference, web conference, or application
ICT-mediated asynchronous communication	GSD_P3	Practices that enable asynchronous communication between team members; for example, email, Instant Messaging, or Wiki
Visit	GSD_P4	Face-to-face meeting made possible by travelling between sites. Two main kinds: seeding visits to build relationships, and; maintaining visits to sustain relationships
Frequent (or Improved) communication	GSD_P5	Enable frequent formal and informal communication among team members through tools and/or face-to-face meetings
Iteration	GSD_P6	Activities that involve cyclical repetition enable multiple incremental opportunities to monitor progress and resolve issues
Review	GSD_P7	Formal or informal activities that enable reflection on prior activities, assessment of completed work, and the opportunity for stakeholders to provide feedback to the teams
Planning	GSD_P8	Activities that establish the scope of work, resourcing, scheduling, and the processes to be employed

The categories described in Table 2 reflect a range of approaches found in the literature to overcome challenges arising from the temporal, geographical, and socio-cultural distances encountered in GSD. For example, mechanisms to address *temporal distance* include adjusting working hours [37], increased use of asynchronous collaboration tools [39] and use of bridging across sites [40]. To address *geographical distance*, mechanisms include using synchronous and asynchronous communication tools [39], visits (travel) [40] and modularization of work [38]. To address *socio-cultural distance*, mechanisms include liaisons [40], increased use of asynchronous tools [39], and planned rotations (visits) [39]. While these mechanisms appear to be generic (that is, they are equally applicable to any development method), the underlying principles of Agile methods imply that Scrum practice may leverage additional benefits from their use. For example, the last four categories (GSD_P5 to GSD_P8), represent mechanisms that reflect inherent characteristics of many Scrum practices themselves (that it, the Scrum practices themselves may mitigate the GSD challenge).

Table 3. GSD challenge mitigation mechanism categories referenced in the literature

Mechanism Categories / Paper Reference	GSD_P1	GSD_P2	GSD_P3	GSD_P4	GSD_P5	GSD_P6	GSD_P7	GSD_P8
[1]	X	X	X		X	X	X	X
[2]	X		X		X		X	
[3]	X	X	X	X	X		X	X
[4]	X	X				X	X	X
[5]	X	X				X	X	X
[6]	X	X	X	X	X	X	X	X
[7]					X			
[8]		X	X		X			X
[9]	X		X			X	X	
[10]			X				X	X
[11]		X					X	X
[12]	X	X			X	X		
[13]	X	X	X					X
[14]	X	X	X				X	X
[15]		X		X	X	X	X	X
[16]	X	X	X		X		X	X
[17]	X	X			X	X	X	
[18]					X		X	X
[19]	X		X		X	X	X	
[20]		X		X	X	X	X	
[21]	X	X	X	X	X	X	X	X
[22]					X		X	
[23]			X		X		X	X
[24]		X			X		X	
[25]	X	X	X	X	X		X	X
[26]	X	X			X			
[27]	X	X						

The findings on how Scrum practices might contribute to overcoming each GSD challenge are described in the next section. Empty cells in Table 4 indicate that no mitigation mechanism was found relating to that combination of variables in the literature. The framework contributes an integrated summary of the literature-based findings and expectations of how common Scrum practices might mitigate (minimize or reduce to zero) the distance-based challenges that can arise in developing software in dispersed global locations.

The research framework provides value as an independent research outcome, as a reference guide to inform current practice, and as a basis upon which to conduct further empirical research (starting with validation of the literature-based views contained within it).

3 Research Framework

This section describes the proposed research framework.

3.1 Communication Challenges

According to the literature, temporal, geographical and socio-cultural distance may impact GSD communication processes by creating three main challenges: reduced

opportunities for synchronous communication; face-to-face meeting [are] difficult, and; cultural misunderstandings (Table 1). The mechanisms that may mitigate each challenge, summarised by category in Table 4, are discussed following.

***Reduced opportunities for synchronous communication (due to temporal distance)*:** The effect of time zone offsets can be so great that there is little or no opportunity for direct contact between dispersed team members. The literature suggests that Scrum practices can mitigate this challenge by *synchronizing work hours* (GSD_P1) and/or *ICT-mediated asynchronous communication* (GSD_P3). For example, Scrum team members may participate in daily scrums in common working hours or outside normal hours by adjusting working hours, or by emailing answers to the three questions before the meeting (What did I do yesterday? What will I do today? What impediments are in my way?), and reviewing minutes emailed back after the meeting. Similarly, for most (if not all) of the other meeting practices, team members from each site may synchronize work hours to achieve a suitable meeting time and/or use various asynchronous communication tools (such as wiki or email).

***Face-to-face meetings difficult (due to geographical distance)*:** Due to developers being located in different countries, it can be difficult to hold face-to-face meetings. The literature suggests that this challenge can be met by using *ICT-mediated synchronous communication* (GSD_P2) to hold meetings online. It also suggests, in the case of sprint planning, that *ICT-mediated asynchronous communication* (GSD_P3), and/or *visits* (GSD_P4) can be used, and; in the case of sprints, that the challenge can be mitigated by *visits* (GSD_P4) to another site. Distributed teams may use a wide range of ICT-mediated synchronous communication tools to support their Scrum practices, such as: teleconference; videoconference; web-conferencing; audio/video Skype; Microsoft Office Communicator with Live Meeting; and/or desktop sharing and Microsoft Net Meeting for application sharing. Similarly, ICT-mediated asynchronous communication tools such as email, Internet Relay Chat (IRC), and Wiki may be used to support meetings. Distributed team members may also physically visit a counterpart site to plan and/or conduct a sprint as a collocated team; or, in some cases, when new knowledge needs to be transferred, some offshore team members may visit the onshore site to work for a few sprints.

***Cultural misunderstandings (due to socio-cultural distance)*:** With developers located in different countries, misunderstandings can occur due to language and cultural differences. The literature suggests that Scrum practices may reduce cultural misunderstandings by using: *ICT-mediated asynchronous communication* (GSD_P3); *visits* (GSD_P4), and/or; *frequent (or improved) communication* (GSD_P5). For example, distributed Scrum team members could post answers to the three Scrum questions into a wiki (an ICT-mediated asynchronous communication tool) which can help to reduce misunderstandings. Similarly, visits by distributed team members to other sites for sprints can also increase familiarization and understanding between teams, and; frequent participation by distributed teams in daily scrums and sprint planning meetings can provide many communication opportunities to break down cultural barriers and increase cultural awareness.

Table 4. Research Framework: GSD Challenge Mitigation Mechanisms by Categories

	Scrum practice / GSD challenges	Sprint Planning	Sprint	Daily Scrum	Scrum of Scrums	Sprint Review	Retro-spective	Backlog
Communication	*Temporal:* Reduced opportunities for synchronous communication	GSD_P1, GSD_P3		GSD_P1, GSD_P3	GSD_P1	GSD_P1, GSD_P3	GSD_P1, GSD_P3	
	Geographical: Face to face meetings difficult	GSD_P2, GSD_P3, GSD_P4	GSD_P4	GSD_P2	GSD_P2	GSD_P2	GSD_P2	
	Cultural: Cultural misunderstandings	GSD_P5	GSD_P4	GSD_P3, GSD_P5				
Coordination	*Temporal:* Increased coordination costs	GSD_P1, GSD_P3		GSD_P1, GSD_P3	GSD_P1	GSD_P1, GSD_P3	GSD_P1, GSD_P3	
	Geographical: Reduced informal contact can lead to lack of critical task awareness	GSD_P2, GSD_P8	GSD_P4, GSD_P6	GSD_P3, GSD_P5, GSD_P7, GSD_P8	GSD_P5, GSD_P7, GSD_P8	GSD_P2, GSD_P7	GSD_P3, GSD_P7	GSD_P3, GSD_P7, GSD_P8
	Cultural: Inconsistent work practices can impinge on effective coordination		GSD_P6	GSD_P5				
	Cultural: Reduced cooperation arising from misunderstandings	GSD_P3, GSD_P5, GSD_P8	GSD_P4, GSD_P6	GSD_P5	GSD_P5	GSD_P5, GSD_P7	GSD_P5, GSD_P7	
Control	*Temporal:* Management of artifacts may be subject to delays							
	Geographical: Difficult to convey vision and strategy		GSD_P4, GSD_P8	GSD_P5, GSD_P8		GSD_P7		
	Geographical: Perceived threat from training low cost 'rivals'							
	Cultural: Different perceptions of authority can undermine morale	GSD_P5		GSD_P5		GSD_P5		
	Cultural: Managers must adapt to local regulations							

3.2 Coordination Challenges

According to the literature, temporal, geographical and socio-cultural distance may impact GSD coordination by creating four main challenges: increased coordination costs; reduced informal contact, leading to a lack of critical task awareness; inconsistent work practices that can impinge on effective coordination, and; reduced cooperation arising from misunderstandings (Table 1). The mechanisms that may mitigate each challenge, summarised by category in Table 4, are discussed following.

***Increased coordination costs (due to temporal distance)*:** The effect of time zone differences can be so great that project coordination complexity and costs increase. As for the communication challenge, the literature suggests that Scrum practices can be used to mitigate coordination costs by: *synchronizing work hours* (GSD_P1), and/or; using *ICT-mediated asynchronous communication* (GSD_P3). For example, distributed team members may adjust work hours to participate in Scrum meetings, thereby

improving project coordination by enabling progress to be reviewed, work to be planned and open issues resolved between sites in a cost effective manner. Costs may also be reduced by distributing relevant material before and/or after meetings via asynchronous means (such as email, audio and video recordings, and wikis).

***Reduced informal contact can lead to lack of critical task awareness (due to geographical distance)*:** Due to geographical dispersion, lack of close interaction between developers may reduce team awareness. The literature suggests that Scrum practices can mitigate this challenge by: *ICT-mediated synchronous* (GSD_P2) and/or *asynchronous communication* (GSD_P3); *frequent (or improved) communication* (GSD_P5); *iteration* (GSD_P6); *review* (GSD_P7), and/or; *planning* (GSD_P8). For example, distributed team members may use synchronous communication tools such as video-conferencing to increase their sense of presence in meetings with other team members. They may also use ICT-mediated asynchronous communication in support of meetings to increase task awareness. For example, for retrospectives, one approach is to implement a transition backlog into which distributed team members can record and save ad-hoc improvement ideas. A significant benefit of the Scrum model is that all practices offer the opportunity to review what is required and/or what is being done, which increases task awareness. Similarly, the iterative nature of sprints and work planning also provide opportunities for, and encourage, informal interactions as well as formal contacts that can improve task awareness within the distributed teams.

***Inconsistent work practices can impinge on effective coordination (due to socio-cultural distance)*:** Due to developers being located in different countries, there may be differences in national culture, language, motivation and work ethics that can impede effective project coordination. The literature suggests that daily scrum and sprint practices can mitigate this challenge through *frequent (or improved) communication* (GSD_P5) and/or *iteration* (GSD_P6), respectively. For example, regular participation of distributed team members in daily scrum meetings can help to maintain consistent work practices between sites. Similarly, iterations of sprints reinforce practice consistency and improve coordination between distributed sites.

***Reduced cooperation arising from misunderstandings (due to socio-cultural distance)*:** Similarly, team member cooperation might be reduced due to cultural and language differences creating misunderstandings. The literature suggests that Scrum practices can mitigate this challenge by: *ICT-mediated asynchronous communication* (GSD_P3); *visits* (GSD_P4); *frequent (or improved) communication* (GSD_P5); *iteration* (GSD_P6); *review* (GSD_P7), and/or; *planning* (GSD_P8). For example, sprint planning meetings may be recorded by a web-conferencing tool and played back by the offshore team to reinforce its understanding. Distributed team members may also travel to other sites to participate in a few sprints as a collocated team to develop their understanding and increase cooperation and team coordination. Also, participation of distributed team members in Scrum meetings and reviews can provide frequent opportunities for communication to reduce misunderstandings between team members and improve cooperation and coordination. Similarly, the iteration of sprints and related meetings enables misunderstandings to be identified and resolved early, thereby reinforcing ongoing cooperation and improving project coordination.

3.3 Control Challenges

Finally, according to the literature, temporal, geographical and socio-cultural distance may impact GSD control processes by creating five main challenges: management of project artefacts may be subject to delays; difficulty in conveying vision and strategy; perceived threat from training low cost rivals; different perceptions of authority that undermine morale, and; managers must adapt to local regulations (from Table 1). The mechanisms that may mitigate each challenge, summarised by category in Table 4, are discussed following.

Management of project artefacts may be subject to delays (due to temporal distance): When a project involves members from different sites, enforcing process and artefact standards can be particularly important in maintaining consistency and interoperability. There was no evidence in the literature, however, of the use of Scrum practices to mitigate this specific GSD challenge.

Difficult to convey vision and strategy (due to geographical distance): Due to stakeholders being located in different countries, it can be difficult for onshore-based managers to convey the project vision and strategy to offshore sites. The literature suggests that Scrum practices can variously mitigate this challenge by: *visit* (GSD_P4); *frequent (or improved) communication* (GSD_P5); *review* (GSD_P7), and/or; *planning* (GSD_P8). For example, daily scrums enable team members to communicate frequently, enabling the project's vision and strategy to be reinforced within the project. In sprint reviews, offshore team members are able to ask product- and project-related questions of the onshore team and, in some cases, the product owner or customer, thereby increasing their understanding of the project's mission. Also, distributed team members may visit the onshore site and/or participate in a Sprint "zero" (project kickoff Sprint), which can develop an understanding of the product vision and strategy and project goals.

Perceived threat from training low cost 'rivals' (due to geographical distance): Employees in higher cost economies can feel that their jobs are under threat from resources sourced from lower cost economies. However, there was no evidence in the literature of the use of Scrum practices to mitigate this specific GSD challenge.

Different perceptions of authority can undermine morale (due to socio-cultural distance): Perception of authority in a team environment can vary between cultures. For example, in some cultures (e.g. Irish), developers may require their superiors to earn their respect and in others (e.g. US culture) give more unquestionable respect to figures of authority [40]. The literature suggests that Scrum practices can mitigate this challenge by *frequent (or improved) communication* (GSD_P5). For example, participation of onshore and offshore teams in daily scrums, sprint planning and review meetings can help members develop a sense of individual worth and value as a team member. These meetings can also clearly establish the boundaries of responsibility for the sprint within and outside the Scrum team.

Managers must adapt to local regulations (due to socio-cultural distance): When working in a global setting, onshore-based managers must be aware of the limitations that local regulations can bring to the project. There was no evidence in the literature, however, of the use of Scrum practices to mitigate this specific GSD challenge.

4 Discussion

Industry experience suggests that use of Scrum practices can provide communication, coordination and control benefits in GSD. As a result, there is increasing interest in using Scrum practices for GSD. However, empirical research on Scrum practices in GSD is scarce. Therefore, a knowledge gap exists on the use of Scrum in GSD in research and practice. Based on findings and experiences reported in the literature, we have proposed a research framework as a basis for developing further understanding. The framework integrates current results and views on how Scrum practices mitigate the challenges of distributed and global software development. The mechanisms were found to fit into eight categories. The framework contributes a research outcome in itself, a reference guide for current practice, and a basis for future empirical validation and investigation for research.

Examining the table overall, the literature appears to suggest that Scrum practices have no distinctive advantage over other development methods in mitigating temporal distance-based challenges (since adjusting working hours for meetings and using ICT-mediated asynchronous communications tools are available to any method). However, while communication tools (which are common mechanisms) are also prescribed to mitigate geographical challenges, these other challenges appear to also be mitigated by mechanisms such as frequent communication, iteration, planning and review that are distinctively inherent in Scrum practices.

4.1 Limitations

The proposal has some limitations. First, the framework is a theoretical contribution that remains to be empirically validated. Since few of the source articles from the literature are empirical studies (indeed, most are industry experience reports), the framework represents mostly theoretical propositions that need to be empirically validated. This work is proceeding in related research.

Second, it may be possible, in some instances, that the researchers misinterpreted an author's intent. For example, the researchers may have incorrectly interpreted an effect of a Scrum practice on a GSD challenge that is inferred in a paper rather than explicitly stated. The use of two researchers reviewing the papers reduced the likelihood of this occurring. Furthermore, subsequent use of the framework in empirical testing will validate its legitimacy from the perspectives of both the researchers who developed it and the authors of the papers on which it was based.

Third, the challenges and mechanisms contained in the framework are not exhaustive so may not be complete. For example, relevant studies may also exist in the systems engineering and/or information systems literatures, which were not included in this study. Also, this is an evolving field in software engineering; other studies may have emerged since the literature review was completed. The framework, however, is easily extendible to take on new challenges and mitigation mechanisms as new research emerges.

Finally, the framework implicitly assumes a generic GSD context so it may obscure project-specific variations in mechanisms. GSD takes many forms, depending on contextual factors, which may heavily influence how and which Scrum practices are used [42]. For example, contextual variables such as team size, collaboration

mode, number of distributed sites, degree of overlap in time zones, and degree of socio-cultural-economic compatibility will influence the configuration of a particular GSD project. Our expectation is that the framework will evolve to encompass a base set of challenges and mitigation mechanisms that will be relevant in most GSD projects using Scrum practices.

4.2 Implications

This review and the framework it has produced raise implications for research and practice. First, for research, questions arise from some initial observations of the framework:

1. Three challenges (in the Control section of Table 4) are not mitigated by any mechanisms in the literature. Is this because of gaps in the literature? Are these challenges invalid? or Is Scrum unable to respond to these challenges at all?
2. Two other control challenges are mitigated by only a few mechanisms. Does this mean that these challenges are significant threats to the use of Scrum in GSD?
3. Six of the nine rows that contain mitigating mechanism categories, include one or more mitigation mechanism from categories GSD_P5 to GSD_P8. This implies that for these GSD challenges, the nature of the Scrum practice itself is sufficient to mitigate the challenge. For example, the literature suggests that the challenge *"Inconsistent work practices can impinge on effective coordination"* can be mitigated by the fact of holding daily (frequent) scrums and/or iterative sprints. Does this mean that Scrum practices are particularly effective in mitigating these challenges?
4. Maintaining backlogs appears to have a limited role in reducing GSD challenges. Alternatively, does this represent a gap in the practice and research literature?

These observations point to areas for future research.

Second, at a higher level of thinking, the framework reinforces the contingency theory view of development methodologies that there is no one "ideal" method for every context [43]. Every development context is different, requiring an adapted method and, to be practically useful, every method needs to be adaptable to different development contexts. Scrum was originally conceived for Agile development in collocated teams. However, with some support and enablement of various tools and mechanisms, Scrum may also be directly or indirectly effective in developing software in globally distributed team environments.

Third (and more concretely), the framework raises other questions for ongoing software engineering process research. For example, given that it may be possible to use Scrum in GSD, is this approach more effective than other development methods? What comparative set of GSD challenges does Scrum mitigate in contrast to other methods? What (if any) differences exist in the toolsets used (or needed)? What can each approach learn from the other? Does it matter which approach is used?

Finally, for practice, the framework provides practitioners with an initial understanding of how Scrum practices may be used effectively in mitigating some of the challenges in developing software collaboratively across borders. It can serve as a reference framework of current knowledge and experience to guide practices in current and future projects.

4.3 Future Work

Related and future research will conduct multiple case studies in real life industry settings to validate and extend the framework. Due to the scale of the framework, these studies may focus separately on challenge-based sections of the framework. Based on these studies and other emergent findings in the literature, we will modify and extend the framework as a reference map for research and practice.

For example, work is underway on qualitative analysis of four case studies of GSD projects using Scrum practices from three internationally known corporations involving industrial, telecommunications and software engineering applications. This study is focusing on validating the prescriptions for the four *coordination* challenges in Table 4. Data was analysed and coded using NVivo. First the data was examined for evidence of the four coordination challenges (which was found in each case) and then for evidence of how Scrum practices were used to mitigate each challenge (mitigation mechanisms were found for each practice/challenge combination; that is, for each cell in this section of the table, thus potentially extending the framework). Currently, the case-based findings are being compared to the literature-based prescriptions.

5 Conclusion

Software engineering cannot escape the trend towards globalisation that faces many business endeavours today. To effectively and efficiently develop and maintain high quality products collaboratively across borders, traditional software development and project management processes need to be re-examined and re-thought. Agile methods such as Scrum need to be sufficiently 'agile' to adapt to new usage domains for them to continue to be relevant and to survive. GSD needs adaptable processes to overcome the significant challenges that can arise in this environment. Practice is likely to continue to inform research as software developers try out different ways of succeeding in GSD projects. As presented in this paper, the literature indicates that there is a substantial need for research to "catch up" and support the needs of practice. In addition to offering an integrated view of the current literature to support practitioners, the literature-based analysis and resulting framework presented in this paper offers some forward directions for this research.

References

1. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Distributed agile development: Using Scrum in a large project. Software Process Improvement and Practice 13(6), 527–544 (2008)
2. Sutherland, J., Viktorov, A., Blount, J., Puntikov, N.: Distributed Scrum: Agile project management with outsourced development teams. In: Proceedings of HICSS-40, p. 274 (2007)
3. Sutherland, J., Schoonheim, G., Rijk, M.: Fully distributed Scrum: Replacing local productivity and quality with offshore teams. In: Proceedings of HICSS-42, pp. 1–8 (2009)
4. Cho, J.: Distributed Scrum for large-scale and mission-critical projects. In: Proceedings of AMCIS 2007 (2007)

5. Williams, W., Stout, M.: Colossal, scattered, and chaotic (planning with a large distributed team). In: Proceedings of Agile 2008, pp. 356–361 (2008)
6. Drummond, B., Unson, J.F.: Yahoo! Distributed Agile: Notes from the world over. In: Proceedings of Agile 2008, pp. 315–321 (2008)
7. Cristal, M., Wildt, D., Prikladnicki, R.: Usage of Scrum practices within a global company. In: Proceedings of ICGSE 2008, pp. 22–226 (2008)
8. Holmstrom, H., Fitzgerald, B., Agerfalk, P.J., Conchuir, E.O.: Agile practices reduce distance in global software development. Information Systems Management, 7–26 (Summer 2006)
9. Vax, M., Michaud, S.: Distributed Agile: Growing a practice together. In: Proceedings of Agile 2008, pp. 310–314 (2008)
10. Smits, H.: Implementing Scrum in a distributed software development organization. In: Proceedings of the Conference on Agile 2007, pp. 371–375 (2007)
11. Jensen, B., Zilmer, A.: Cross-continent development using Scrum and XP. In: Proceedings XP 2003, pp. 146–153 (2003)
12. Kussmaul, C., Jack, R., Sponsler, B.: Outsourcing and offshoring with agility: A case study. In: Proceedings of XP/Agile Universe, pp. 147–154 (2004)
13. Sureshchandra, K., Shrinivasavadhani, J.: Adopting Agile in distributed development. In: Proceedings of ICGSE 2008, pp. 217–221 (2008)
14. Danait, A.: Agile offshore techniques: A case study. In: Proceedings of Agile Development, pp. 214–217 (2005)
15. Summers, M.: Insights into an Agile adventure with offshore partners. In: Proceedings of Agile 2008, pp. 333–338 (2008)
16. Therrien, E.: Overcoming the challenges of building a distributed agile organization. In: Proceedings of Agile 2008, pp. 368–372 (2008)
17. Berczuk, S.: Back to basics: The role of Agile principles in success with a distributed Scrum team. In: Proceedings of Agile 2007, pp. 382–388 (2007)
18. Karsten, P., Cannizzo, F.: The creation of a distributed Agile team. In: Proceedings of XP 2007, pp. 235–239 (2007)
19. Cottmeyer, M.: The good and bad of Agile offshore development. In: Proceedings Agile 2008, pp. 362–367 (2008)
20. Paasivaara, M., Lassenius, C.: Could global software development benefit from Agile method? In: Proceedings of ICGSE 2008, pp. 109–113 (2006)
21. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Distributed Agile development: Using Scrum in a large project. Proceedings of ICGSE 2009, 195–204 (2009)
22. Bondi, A.B., Ros, J.P.: Experience with training a remotely located performance test team in a quasi-Agile global environment. In: Proceedings of ICGSE 2009, pp. 254–261 (2009)
23. Hansen, M.T., Baggesen, H.: From CMMI and isolation to Scrum, Agile, Lean and collaboration. In: Proceedings of Agile 2009, pp. 283–288 (2009)
24. Hossain, E., Babar, M.A., Verner, J.: How can agile practices minimize global software development co-ordination risks? In: O'Connor, R.V., Baddoo, N., Cuadrago Gallego, J., Rejas Muslera, R., Smolander, K., Messnarz, R. (eds.) EuroSPI 2009. Communications in Computer and Information Science, vol. 42, pp. 81–92. Springer, Heidelberg (2009)
25. Lee, S., Yong, H.: Distributed agile: project management in a global environment. Empirical Software Engineering 15(2), 204–217 (2010)
26. Sutherland, J., Schoonheim, G., Kumar, N., Pandey, V., Vishal, S.: Fully Distributed Scrum: Linear Scalability of Production between San Francisco and India. In: Proceedings of the Agile Conference 2009, pp. 277–282 (2009)

27. Therrien, I., Lebel, E.: From Anarchy to Sustainable Development: Scrum in Less Than Ideal Conditions. In: Proceedings of the Agile Conference 2009, pp. 289–294 (2009)
28. Jimenez, M., Piattini, M., Vizcaino, A.: Challenges and improvements in distributed software development: A systematic review. Advances in Software Engineering, Article ID 710971, 1-14 (2009)
29. Herbsleb, J., Moitra, D.: Global software development. IEEE Software 18(2), 16–20 (2001)
30. Herbsleb, J., Grinter, R.: Architectures, coordination, and distance: Conway's law and beyond. IEEE Software 16(5), 63–70 (1999)
31. Hossain, E., Babar, A.M., Paik, H., Verner, J.: Risk identification and mitigation processes for using Scrum in global software development: A conceptual framework. In: Proceeding of the Asia Pacific Software Engineering Conference, APSEC 2009, pp. 457–464 (2009)
32. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods: review and analysis. Technical Report # 408, VTT Publications, Espoo (2002)
33. Ågerfalk, P.J., Fitzgerald, B.: Flexible and distributed software processes: old petunias in new bowls? Communication of the ACM 49(10), 27–34 (2006)
34. Hossain, E., Babar, A.M., Paik, H.: Using Scrum in global software development: A systematic literature review. In: Proceedings of ICGSE 2009, pp. 175–184 (2009)
35. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report, EBSE-2007-01 (2007)
36. Herbsleb, J.D., Mockus, A., Finholt, T.A., Grinter, R.E.: Distance, dependencies, and delay in a global collaboration. In: Proceeding of CSCW 2000, pp. 319–327 (2000)
37. Moe, N.B., Šmite, D.: Understanding a lack of trust in global software teams: A multiple-case study. Software Process Improvement and Practice 13(3), 217–231 (2008)
38. Carmel, E.: Global software teams: Collaborating across borders and time zones. Prentice-Hall, NJ (2009)
39. Damian, D., Zowghi, D.: Requirements engineering challenges in multi-site software development organizations. Requirements Engineering Journal 8(1), 149–160 (2003)
40. Ågerfalk, P.J., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B., O'Conchuir, E.: A framework for considering opportunities and threats in distributed software development. In: International Workshop on Distributed Software Development 2005, pp. 47–61 (2005)
41. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice Hall, Upper Saddle River (2001)
42. Hossain, E., Ali Babar, M., Verner, J.: Towards a Framework for Using Agile Approaches in Global Software Development. In: Bomarius, F., Oivo, M., Jaring, P., Abrahamsson, P. (eds.) PROFES 2009. Lecture Notes in Business Information Processing, vol. 32, pp. 126–140. Springer, Heidelberg (2009)
43. Avison, D., Fitzgerald, G.: Information Systems Development: Methodologies, Techniques and Tools, 4th edn. McGraw-Hill, Maidenhead (2006)
44. Krishna, S., Sahay, S., Walsham, G.: Managing cross-cultural issues in global software outsourcing. Communication of the ACM 47(4), 44–47 (2004)

Towards the Competitive Software Development

Andrzej Zalewski and Szymon Kijas[1]

[1] Warsaw University of Technology,
Institute of Automatic Control and Computational Engineering
{a.zalewski,s.kijas}@elka.pw.edu.pl

Abstract. The concept of competitive software development is founded on the observation that the system's owner and development companies have not only common interests, but also conflicting interests as well. This is particularly true in the case of large-scale software systems. Competitive development is an answer to the syndrome of large-scale software systems evolution and maintenance being monopolised by the companies that originally developed these systems. Competitive development is founded on the idea that the entire system is divided into smaller units, which are independently developed by different companies, i.e. no co-operation is assumed between various development organisations and there is no communication between them. However, strong and efficient co-ordination has to be performed on behalf of the system's owner in order to make such an approach successful. These assumptions are radically different to the typical collaboration assumption for agile development. This prevents one software company from making a system owner entirely dependent on its services. We show that such demonopolisation can save large sums of money, making the prices of software development considerably lower than they would be in the case of a single software development company. Our experiments show that, if efficiently co-ordinated, such a distributed, competitive development requires a similar effort to traditional approaches.

Keywords: Software process, software process improvement, empirical studies.

1 Introduction

This paper has been inspired by the work done for Public Procurement Office in Poland, supervising purchases for public or publicly-owned institutions. One of its responsibilities is to ensure that public money is spent on services rendered and goods supplied by competing contractors. Large-scale software systems are a very specific subject of public procurement.

These are the largest and the most complex systems, tailored to the specific requirements of a commissioning organisation. Their evolution and maintenance becomes usually monopolised by the company that originally developed the system. This enables them to act in the condition of limited competition, allowing them to expand their profit margins above the market average, which is exactly the situation that every company is striving for.

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 103–112, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Such a monopolisation syndrome occurs whenever any part of the system becomes too large, i.e. too complex to freely hand over its development to some other development company (competitor). This is studied in detail in part 2. We argue that, to prevent monopolisation, an antimonopoly function should be included in the development, maintenance and evolution processes for large-scale software systems.

This approach is the core concept presented in this paper – part 3. Its aim is to enable competing software companies to work independently on the decoupled parts (macro components) of the same software system, while assuring necessary coordination. No co-operation is assumed between different software development companies. The process has been validated experimentally, as described in part 4. The impact of such a novel approach on the various aspects of software development and its cost have been discussed in parts 5 and 6 respectively. A summary of results and ideas for further research are given in part 7.

2 The Software Monopolisation Syndrome

From a business point of view, software products are supposed to overcome market competitors, attract clients, gain a certain market share, and retain and attract clients to buy newer and newer versions or other software from a given company. At the same time, they should be hard to substitute by market competitors. What makes a software product successful depends on lots of factors such as the target group of the product, the software properties, market conditions etc. etc.

Large-scale software systems are a very specific kind of software product in this context. They are built at the commission of a single entity and tailored to its specific requirements. There is usually no other organisation that could use that system. Their extreme complexity makes them very difficult to substitute by a system provided by another software company. They are also difficult to evolve by making changes and extensions. This effect has theoretically been described in [1], [2], [3] and attributed to the high level of coupling between system components.

As a result, such systems are typically perceived as a kind of a natural monopoly (similar to water supply and sewage system) in which maintenance and further development can only be carried out by the company that originally developed the system. The lack of pressure from competition results, as with any other monopoly, in the expansion of profit margins and increased cost of software maintenance and evolution.

The tendency of large IT systems to become monopolised comes from:

- the very nature of software systems i.e. their complexity, enhanced by the architectures dominating modern business applications designs,
- the inefficiency of system documentation and the means of knowledge management,
- the lack of motivation to minimise the monopoly,
- legal / regulatory / policy issues (e.g. intellectual property rights, special purpose systems like military ones).

We now delve deeper into details of the first three reasons, leaving behind the legal ones. To modify any IT system the developers need to know its construction in sufficient detail. Monopolisation is generally caused by the inability to obtain or transfer design knowledge on complex software systems efficiently in terms of time and cost.

This, in turn, is a result of the limitations of software documentation [4] and design knowledge management methods and tools [5], which, despite many years of research, have not yet been overcome.

If the given organisation does not have a team of its own shadowing the activities of the developing company, knowledge on the system and software design can only be obtained from system documentation, configuration and source code. If a system is kept small enough, the owner can always (given sufficient intellectual property rights to the source code and documentation) change the current developer with negligible disturbances to the system's operation. In such a case, knowledge on the system construction can be obtained by the new developers within a short time and at minimal cost.

However, when a system becomes too big, design knowledge cannot be obtained within a reasonable time and money by the new developers. This means that disturbances in the system operation cannot be avoided when the developing company is replaced with another one, or the necessary preparations would take a very long time (tens of months) and cost a lot (reports indicate 50% or even more of the entire maintenance cost [4]) before the change takes place. Even if the owner succeeds, the system is monopolised again – this time by the new developers. Another important factor is that, in the case of large-scale software systems, such a step is generally very risky in business and technical terms. Thus there is no real incentive for the owner to undertake such a challenge. If the system becomes too large, the owner cannot avoid the monopolisation of maintenance and evolution.

The problems of over-dependence on a certain system, and its developers, have not passed unnoticed. It currently takes two extreme forms:

- A single subsystem becomes too complex – this is the basic version of the syndrome presented above – companies try to resolve the problem by acquiring systems from various vendors (often overlapping existing functionality!) and integrating them with EAI solutions such as Enterprise Service Bus or Process Management Systems;
- An integration system (e.g. ESB) becomes too complex – this is the opposite situation encountered by companies that exaggerated with the extent and complexity of an integration solution – it is clear that integration solutions can easily be monopolised by the companies that developed them as the system becomes too complex and its owner grows too dependent on its operation.

3 Demonopolising Software Development

To draw up a remedy for the monopolisation syndrome we need to look more into the technical details of the monopolisation problem. The extreme cases of the monopolisation syndrome indicate that the source of the problem is located in a common component of a system on which large parts of the software depend. It was an integration solution in one of the extreme cases. However, in most cases, such a component is usually a database on which most of the components of a given software system depend. Most of the modern business systems use a layered architecture, which is imposed by all the most popular implementation technologies, like J2EE or .NET. Hence, modern business software is built on the foundation of a common database. This means that a change made to a single component can possibly impact all the other components, as illustrated in fig. 1.

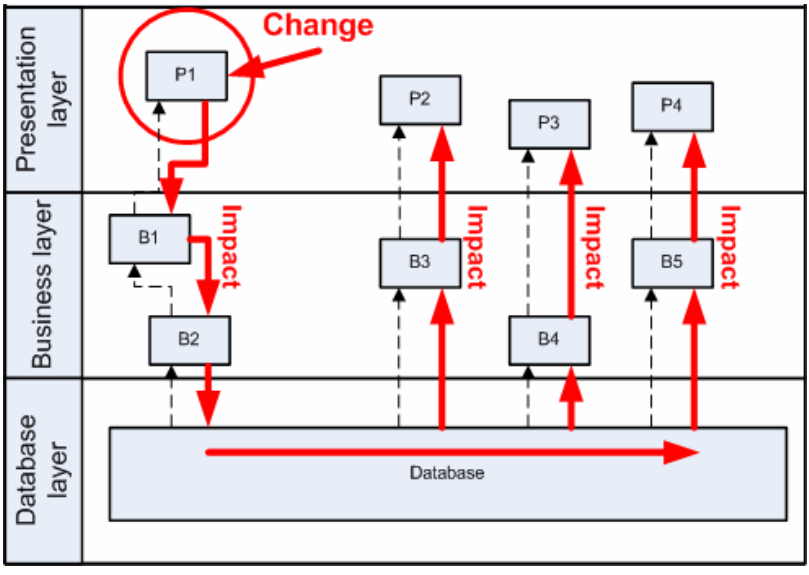


Fig. 1. Coupling in a three-tier software system

To eliminate, or at least to minimise, the possibility of monopolisation, we need to decouple strongly coupled components. It means that software should be split into parts that plug into an existing database without altering its structures common to a greater number of software units (components, etc.). Such macro components (applications, subsystems) could be developed independently of the work done by the developers of the other subsystems. Obviously changes to common database structures cannot be entirely avoided, but a careful consideration of the database can minimise such needs and their impact. Developing organisations can develop data structures of their own, but they should be treated as private unless the owner decides to include them in a common database schema.

This task has to be allocated to a skilful team, known as a "co-ordinating team", which should work on the system owner's behalf to ensure conditions in which various development teams can work independently without excessive interference. The organisation of competitive development has been illustrated in fig. 2. Its interesting property is that the teams can use different development processes.

The co-ordination team's role comprises:

- Common database management – the development and maintenance of the database structures common to larger parts of the system, negotiating changes to database structures with the software development companies and assessing their impact, providing information on the common database structures;
- Development organisation and co-ordination – this is part of the co-ordination effort needed to define and manage system decomposition, to place all the independently parts together, to create software releases, and to co-ordinate independent work;

- Integration management – the implementation of integration solutions necessary to integrate subsystems resulting from splitting subsystems that have grown too large, or from purchases of ready-to-use configurable systems (such as ERP);
- Complexity control – this means that whenever any part of the system becomes too complex, it is split into manageable, independently developable parts (this is a typical antimonopoly action similar to those undertaken by state competition protection agencies), or when co-ordination becomes too complex then separate systems could be merged.

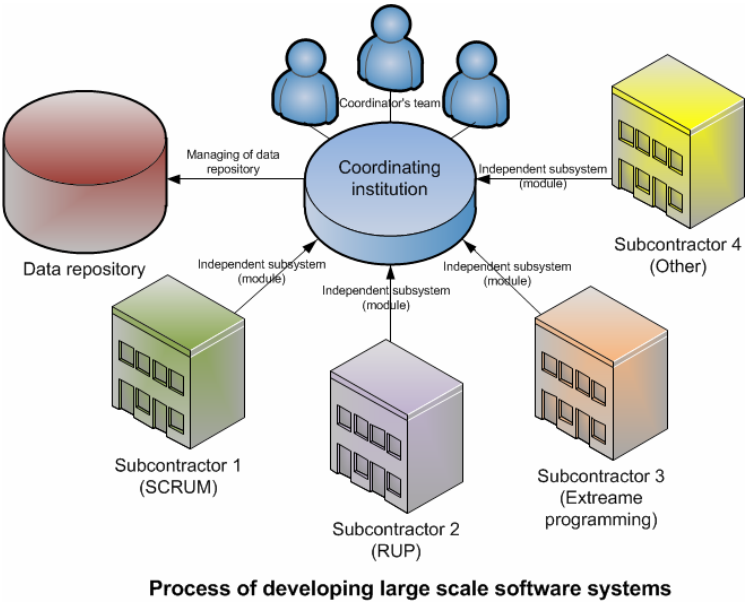


Fig. 2. Competitive development

4 Experimental Assessment

An experiment was performed to show that separated teams working independently under central co-ordination will not increase the amount of work done by the programmers compared to a single team working on the same project. The experiment was performed on the modular system for managing studies at our faculty. This system comprised three modules using the same database:

- **Module one:** Student module – an application used by the students for semestral declaration, course registration, browsing results, etc.
- **Module two:** Teacher module – web application used by teachers for subject descriptions, grading etc.
- **Module three:** Dean and dean's office module – GUI application used for configuration, approval for course registration, student promotion, final exam procedures support etc.

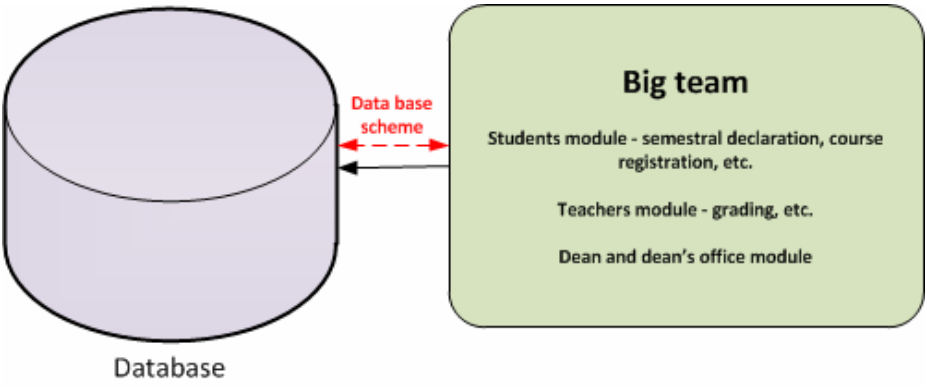


Fig. 3. Scheme of system developed by a single big team

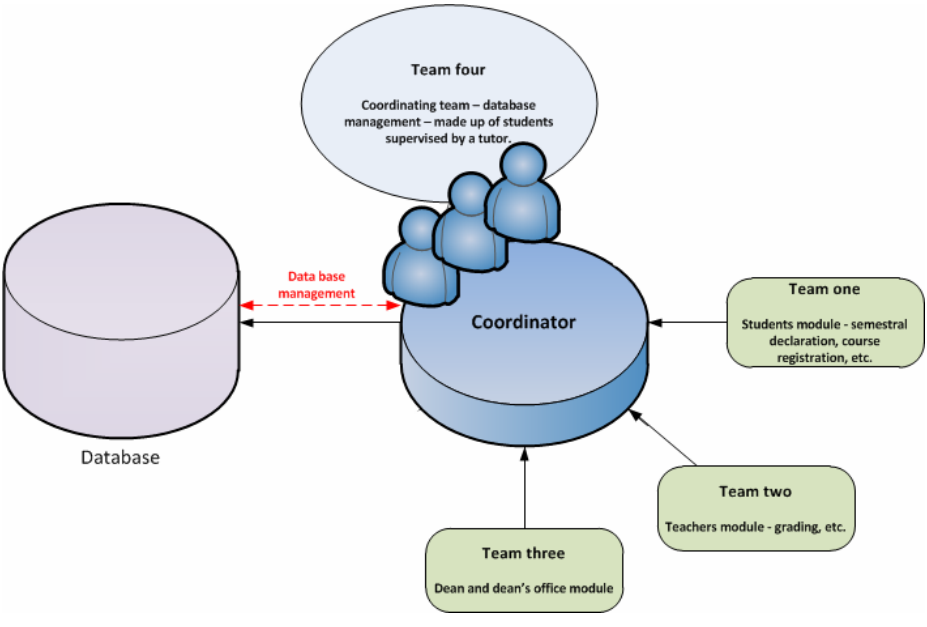


Fig. 4. Scheme of system developed by four small teams

These modules share database structures and their functionality partially overlaps.

Two teams of eight students were chosen for every time the experiment was conducted. One team was working together to develop the required functionality (fig. 3) without a predefined internal structure, but with a certain amount of co-ordination ensured by an early draft of a common data model. The other team (fig. 4) was divided into small teams consisting of two people each. Three of the sub-teams were assigned to the development of selected modules (one independent module for each group) and one became the co-ordinating group (responsible for co-ordinating the work and managing the common database structure).

All of the teams used Extreme Programming. The experiment was supposed to last for three working days (24 hours per person - which makes 192 man-hours altogether). The development times of individual modules and the entire system were measured and compared.

Table 1. Results of the experiment

First measurement				
	Module 1	**Module 2**	**Module 3**	**Total**
One large development team	49 man-hours	55 man-hours	98 man-hours	202 man-hours
Developing in small teams	55 man-hours	58 man-hours	107 man-hours	220 man-hours
Second measurement				
	Module 1	**Module 2**	**Module 3**	**Total**
One large development team	51 man-hours	49 man-hours	97 man-hours	197 man-hours
Developing in small teams	43 man-hours	51 man-hours	95 man-hours	189 man-month

The development effort measured during the experiments is presented in table 1. The analysis of the results and of monitoring the teams' activities revealed the following observations:

- General observation:
  - The level of effort required to develop the whole functionality turned out to reach similar levels in both approaches. This indicates that competitive development should not increase effort compared to conventional approaches (agile, RUP, waterfall).
- Observations concerning the single team exercise:
  - Weak co-ordination of work on the database structure was a large problem, due to many clashes between changes introduced by different people.
  - Team members trying to reuse code developed by the others lost a lot of time on debugging immature solutions of their colleagues. In many cases it required more time than developing the whole thing from scratch.
- Observations concerning the independent teams exercise:
  - Independently working teams turned out to be less effective in the first run of the experiment, because of the poor quality of co-ordination and database structure developed by the co-ordination team. This improved considerably during the second run. This indicates that the efficiency of competitive development depends strongly on the quality of co-ordination, and in particular on careful database design;
  - Team separation eliminated clashes on the changes to database structures, as well as error propagation resulting from the reuse of poorly verified software.

5 Discussion: The Impact of Competitive Development

The idea of a number of teams working together on the same project under some sort of co-ordination is obviously nothing new. It can easily be related to current achievements of global software development [7], [8], which raises issues of efficient communication and co-ordination between geographically separated teams.

However, the idea of competitive development is based on quite opposite assumptions: instead of striving to resolve the challenge of multi-party communication between separate teams [6], it assumes that this should not take place, as all the conflicts should be resolved by the co-ordinator. This obviously makes sense only for strongly cohesive and large enough modules like applications or even subsystems, otherwise the co-ordinator has the heaviest task and has to make the greatest effort.

The proposed approach assumes that the interests of the organisation commissioning the system and the interests of the developing companies are partially conflicting and only partially consistent. This is far removed from the agile approaches, which assume close co-operation between the parties and implicit sharing of the same interests. This synergy does not necessarily take place in the case of large-scale software systems.

Competitive development apparently ignores the advantages of reusing code, though this applies only to the reuse of code created during systems development, which in many cases turns out to be error prone. At the same time, each of the separate development teams can reuse proven COTS or libraries.

6 The Savings from Software Demonopolisation

Competitive development is designed for the development and evolution of large scale software systems. Modern IT systems evolve perpetually throughout their whole lifetime, never reaching a stable state. Market surveys show that the cost of system maintenance and evolution incurred during the entire life cycle is often twice or even three times as high as the initial cost of system construction. Table 2 contains a sample of large-scale software systems developed for public organisations in Poland over the past 10 years. They belong to the group of the largest systems developed in our country and across Central and Eastern Europe, and are all affected by the software monopolisation syndrome.

Table 2. A Sample of Large-scale Software Systems Developed for Public Organisations in Poland

Name of the system or application domain	Approximate Development and Maintenance Costs Incurred so far (in USD)
Comprehensive System for Social Security	$1.2 bln
Integrated Administration and Control System (IACS) /Agricultural Aid Programmes Administration and Supervision (IACS)	$0.6 – $1 bln
Cars and driving licences register	$80 mln (development only)
Energy Market	$50 mln
EU Aid Management (SIMIK)	$6 mln

Different internet sources, like [9], indicate that increased levels of competition can make prices in groceries cheaper even by about 10%, for electronic devices it can even be 30% off [10]. As the profit margins assumed in the software industry are rather high, due to the high level of risk, we expect that it should be possible to save at least 20% of the development cost if competition is properly ensured. In the case of the largest systems mentioned in table. 1, i.e. systems Nos 1 and 2, it means that at least $20 mln could be saved each year on maintaining each of these systems (assuming an annual maintenance cost of $100 mln for each of these systems).

7 Conclusion

Large-scale software systems are a kind of natural monopoly (like water supply or sewage system) in the genre of IT. Recent history shows that some natural monopolies were broken by splitting up large organisations into a number of smaller ones, or through the introduction of some sophisticated co-ordination of common resources usage as in the case of the electric energy market. The concept of competitive development is assumed to be the first step toward breaking monopolies on the development of large scale software systems.

Competitive development supports development process diversity, as different processes can co-exist in the development of the whole system. This approach is radically different to the existing ones, as it assumes conflicting interests rather than consistent interests of the parties taking part in the development of large-scale software systems.

The further development of this concept should include mechanisms of efficient co-ordination of independent developments, demonopolisation activities undertaken by the co-ordinator, as well as the techniques of efficient management of common databases. Obviously, further validation on a greater number of development cases is needed and is planned for the future.

Acknowledgement. This research has been supported by a grant from the Ministry of Science and Higher Education of the Republic of Poland under grant No 5321/B/T02/2010/39.

References

1. Simon, H.A.: The Architecture of Complexity. Proceedings of the American Philosophical Society 106(6), 467–482 (1962)
2. Ethiraj, S.K., Levinthal, D., Roy, R.R.: The Dual Role of Modularity: Innovation and Imitation. Management Science 54(5), 939–955 (2008)
3. Ethiraj, S.K., Levinthal, D.: Modularity and Innovation in Complex Systems. Management Science 50(2), 159–173 (2004)
4. Corbi, T.A.: Program understanding: Challenge for the 1990s. IBM Systems Journal 28(2), 294–306 (1989)
5. Ali Babar, M., et al.: Architecture knowledge management. Theory and Practice. Springer, Heidelberg (2009)
6. Lanubile, F., Ebert, C., Prikladnicki, R., Vizcaino, A.: Collaboration Tools for Global Software Engineering. IEEE Software 27(2), 52–55 (2010)

7. Damian, D., Moitra, D.: Global Software Development: How Far Have We Come? IEEE Software 23(5), 17–19 (2006)
8. Boden, A., Nett, B., Wulf, V.: Operational and Strategic Learning in Global Software Development. IEEE Software 27(6), 58–65 (2010)
9. Jarman, M.: Grocery competition brings shopper savings, http://www.azcentral.com/community/westvalley/articles/2010/10/12/20101012arizona-food-prices-decline.html
10. Tough Competition Brings Down Smart Phone Prices, http://www.iwabs.com/content/tough-competition-brings-down-smart-phone-prices

Defect Detection Effectiveness and Product Quality in Global Software Development

Tihana Galinac Grbac[1] and Darko Huljenić[2]

[1] Faculty of Engineering, University of Rijeka, Vukovarska 58, HR-51000 Rijeka, Croatia
tihana.galinac@riteh.hr
[2] Ericsson Nikola Tesla, Krapinska 45, HR-10000 Zagreb, Croatia
darko.huljenic@ericsson.com

Abstract. Global software development (GSD) has become a common practice in the software development industry. The main challenge organizations have to overcome is to minimize the effect of organizational diversity on the effectiveness of their GSD collaboration. The objective of this study is to understand the differences in the defect detection effectiveness among different organizations involved into the same GSD project, and how these differences, if any, are reflected on the delivered product quality. The case study is undertaken in a GSD project at Ericsson corporation involving nine organizations that are commonly developing a software product for telecommunication exchanges. Comparing the effectiveness of defect detection on the sample of 216 software units developed by nine organizations, it turns out that there is statistically significant difference between defect detection effectiveness among organizations. Moreover, the defect density serves better as a measure of defect detection effectiveness than as a measure of the product quality.

Keywords: Global software development, defect detection effectiveness, defect density, software quality.

1 Introduction

Global Software Development (GSD) is becoming a common practice in the modern software industry, involving software development teams from the organizations that are distributed around the globe to develop a common software product. The main drivers for globalization of software development is in cost benefits, entrance into global market and access to a large multi-skilled resource pool. On the other hand, the main challenge the organizations working in the GSD environment have to overcome is to minimize the influence of its diversity onto GSD project success. The software development project success is highly dependent on the software project team effectiveness in executing the project processes to achieve the project goals. The main identified barriers for teams in GSD environment are coordination, communication and control [11].

The researchers and practitioners effort have been focused on defining general processes and providing guidelines to overcome these barriers. Besides having

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 113–127, 2011.
© Springer-Verlag Berlin Heidelberg 2011

implemented the general processes that are highly supported with collaboration tools and following guidelines for distributed work, the main concern is whether the impact of the organizational distribution is significant or not. The objective of this study is to explore differences in the defect detection effectiveness among different organizations involved into the same GSD project and how these differences, if any, are reflected on the delivered product quality.

The rest of the paper is organized as follows. In Section 2 the metrics used for evaluating the defect detection process is introduced. In Section 3 the related work is reviewed. Section 4 describes the research context and methods used in the case study. The results of the study are presented in Section 5. Finally, in Section 6 the results are discussed, and we conclude the paper in Section 7.

2 Metrics

There exists a variety of metrics defined for measuring software quality [9], [10], [13]. Still, the most dominant metric that is used in the empirical studies for evaluation of software quality is the number of detected defects [3], [5]. The size of software, on which the number of defects in reported, is used for the comparison purposes. The defect density is defined in [12] as the ratio of the number of defects found and the software size involved into defect detection. The aim of defect detection activities is to deliver software product with zero remaining defect density. Therefore, the defect density is also considered as a measure for defect detection effectiveness [17].

The defect detection effectiveness may be used as a process control measure. Comparing this measure with the average from the history projects or with the project goal, one could bring decision about additional investment into defect detection process, as suggested for example in [8], [7] for the purpose of software inspection. The problem with defect density as a defect detection effectiveness measure lies in the fact that it is hard to distinguish if the number of defects identified by defect detection process is due to bad design, coding or good defect detection process. A number of defect injection and defect detection factors have been identified in [21]. Nevertheless, a number of studies have used defect density measure for the purpose of comparison of defect detection technique effectiveness in a given environment. The defect seeding technique is commonly used in these cases to increase the reliability of the study. Furthermore, the defect density considered "in time", that is as a function of invested testing effort, is used in reliability growth modelling [14]. The control of defect detection process is based on the trend of the defect density curve.

In this study we are neither interested into reasons for the amount of defects detected, nor to control the defect detection process. Instead we are analyzing the differences in the effectiveness of completed defect detection activities among distributed organizations that were supposed to apply the same general defect detection process for GSD project. Therefore, in this study the defect density was used as measure to determine differences in defect detection effectiveness among organizations involved into GSD and influence of the organizational diversity on the defect detection effectiveness and the product quality.

3 Related Work

One of the main motivator for the organizational movement into global software development is in its cost benefits. This implies access to larger and well–skilled resource pool at lower cost [4]. On the other hand, dislocation of development teams has been identified as the main challenge for effective communication, coordination and control that may have significant impact on the project performance and software quality.

A number of studies have questioned the performance ability of GSD along with the resulted software product quality. The empirical evaluation of GSD effect is hard to isolate from other effects, and requires well defined and consistent measurement system in all organizations involved into GSD. This is hard to achieve in practice resulting with limited related work in this area.

In [3] the post–release defects and software unit attributes are compared between software units developed by distributed versus collocated teams. Comparison of mean values for the number of post–release defects detected per software unit, has resulted with slightly more, but statistically significant, defects in GSD. Moreover, if the number of developers is included into analysis, the significance of the conclusion that larger number of post–release defects is affected by GSD becomes even smaller. Another analysis performed in [3], compared software unit attributes, such as code churn, complexity, dependency, and test coverage, measured per software unit. It turns out that there is no significant difference between software units developed in GSD and collocated environment.

A number of metrics have been evaluated as possible predictors of software unit defect–proneness. The aim of these is to control and predict software quality, and thus, better direct future verification efforts. In [15] the organizational complexity is proposed as a predictor of software quality, where the software quality is measured as the amount of post–release defects. The metrics scheme for organizational complexity qualification, as proposed in [15], consists of eight measures that are representing organizational distance of the developers, the number of developers involved into software unit development, the developers level of multi–tasking, and the amount of organizational involvement into software unit change. Using the proposed metrics scheme on the data obtained from the Windows Vista development resulted with the conclusion that such organizational complexity measure is a statistically significant predictor of the defect–prone software units. Moreover, the results showed that the suggested metrics scheme is better predictor for post–release defects than traditionally used metrics such as static code attributes, coverage, and pre–release defects.

Another approach to evaluating the impact of GSD on the software quality is by using the process maturity levels as defined in [6]. The study performed in [5] analyzed the impact of the process maturity level on the software quality, where the software quality is measured as the number of defects reported during the integration and system testing for each software unit. The analysis resulted with the conclusion that the process maturity level and the level of distribution have significant impact on the quality of software units. Moreover, the process

maturity impact on software quality becomes more significant as development becomes more distributed.

The code size, level of distribution and pre–release defect density are used to evaluate the experiences working within GSD in [2]. Comparing the pre–release defect densities for several completed projects working in GSD and the average defect density reported for US industry (2.6 defects per 1000 Lines of Code) the paper concludes that GSD has not increased the defect density.

4 A Case Study

In this section we define and explain the context of the research study, research questions, methods and strategy used for the data collection and analysis.

4.1 Context of the Study

The context of the study is the defect detection process used in globally distributed software development (GSD) within Ericsson corporation.

The software product is developed for telecommunication exchanges, in particular, the Mobile Switching Center (MSC) node that is one of the network nodes within next generation core telecommunication network. The software product is developed in the product line [1] fashion, in which the product releases are following the Core Network evolution strategy as prescribed by [19]. A product release is the outcome of a GSD project. The smallest self contained administrative unit of the software product is called a software unit. The size of a software unit is small enough so it can be understood and managed by humans. The software product we considered in this paper consisted of 216 software units that were impacted by some modifications. The total volume of impacted software units was $1.5\,M$ volume of code with included modification of over $400\,k$ volume of code. The volume of code is a measure obtained as a sum of the kilo programming statements, the number of kilo Assembler instructions multiplied by factor 0.3, and the amount of data storage. Each software unit is in responsibility of a software developer as suggested in [16], and according to its affiliation the software units are assigned to organizations involved into GSD project.

The software development process is an evolved version of the V–model with incremental delivery strategy and feature based development. Waterfall sequence of development phases such as analysis, design, coding, testing is kept on the feature level. Moreover, the classical project management model based on Tollgate concept [18] is used but with rather flexible tollgate criteria. For example, a requirement for passing a project Tollgate between two consecutive project phases could be that at least 60% of the features is already in the later phase. The process evolves from project to project, as best practices and lessons learned are incorporated into the new revision of software development process that is used in further software development projects. Since the process is used in GSD for years, it is very well supported by a standard toolbox, which is also updated with all process improvements. Collaboration tools are also incorporated into the standard toolbox, and the majority of existing software development tools are adapted for collaboration purposes.

The software development is a case of global software development where multiple partner organizations are involved into Software Design and Software Verification projects. The software design project is performed by nine Software Design (SD) organizations that are globally distributed and develops together the same software product (software for the MSC node). The software development work is divided among these SD organizations following the product ownership criteria, which follow the software product architecture, as suggested in [16]. All SD organizations participate in the same software development process phases, such as analysis, design, coding, and part of testing that are tailored for GSD.

All defect detection activities performed by SD organizations will be called early defect detection (EDD). The EDD includes software code inspections, code reviews, basic test, and those function tests that are performed in the simulated environment, simulating the rest of the software product. In the study, we compare the EDD effectiveness of different SD organizations that participate in the same software design project. They are supposed to follow the same early defect detection process prescribed by the GSD project.

The software verification project is performed by the Network Integration and Verification (NIV) organizations that are also globally distributed. However, unlike SD organizations, their work is divided mostly with respect to the specific market/customer. Since our study is concentrated only at two customer references, that are related to the basic node system test and basic integration and verification test, only one NIV organization is involved.

All defect detection activities performed by the NIV organization will be referred to as late defect detection (LDD). The LDD activities are performed on the software units delivered from the SD organizations. The whole software product is under test, without any simulations within the software product. The processes followed by the NIV organization are also standardized within their organization and they evolve with projects. In the study we compare the LDD effectiveness between software units developed by different SD organizations. The NIV organization is completely independent from SD organization, with separated resources. Nevertheless, as a best practice it is customary to borrow resources, but only as the support personnel during the hand–over processes. The results of the NIV organization, when analyzed between different SD organizations, could be a useful indicator of the impact of GSD organizations distribution on the delivered product quality. The LDD effectiveness could be also a measure of product quality.

4.2 Research Hypotheses

The main objective of our research study was to understand how the organizational diversity influences the early defect detection effectiveness, the late defect detection effectiveness and the software product quality. As already explained in Section 2, the defect density is used as the metric for this investigation. Therefore we selected the following research hypotheses:

HA_0 : The mean values of defect density in early defect detection for samples of different SD organizations within the same GSD project are equal.

HB_0 : The mean value of defect density in late defect detection for samples coming from different SD organizations within the same GSD project are equal.

The alternative hypotheses are that the mean values considered in the corresponding null hypothesis are not all equal.

4.3 Data Collection

The selected software design and software verification projects for which data collection is performed are one of the history projects where all software development activities are finalized. All data were collected per software unit as presented in Table 1 and are associated to the relevant SD organizations.

For each software unit, the main data collection questions, that are related to the research hypotheses, are as follows:

- What is the volume and modified volume of the software unit?
- What is the number of defects detected in the software unit during early defect detection process?
- What is the number of defects detected in the software unit during late defect detection process?
- Which SD organization is responsible for the software unit?

The random variables associated to these questions are listed in Table 1.

The measurements for all random variables in Table 1 except the one counting the defects in LDD were collected from the Quality Report documents that are stored in the project repository through the standard Ericsson Clear Case tool. This is a mandatory document required at TollGate3, at which the software product is handed–over from the software design project to the software verification project. All software units that were included into Quality Report documents, and that have reported modification volume, were included into this data collection procedure.

In order to verify the reliability of the collected data for random variable counting the defects in EDD, the data were additionally collected from the corporate Change Notification database, which is Ericsson's official database for reporting defects detected during early defect detection process. The software units in which data inconsistency is identified were removed from the sample.

Table 1. Measured random variables

Name	Description
SWU Size	Volume of respective SWU
SWU Modification Size	Modified volume of respective SWU
SWU Defects in EDD	Count of defects detected in EDD per SWU
SWU Defects in LDD	Count of defects detected in LDD per SWU

The measurements for the random variable that counts defects in LDD were collected from the corporate Modification Handling database. For all software units in which the modification is made in the considered software design project (according to the Quality Reports mentioned above), the number of defects reported in the database were counted. Note that we considered only defects that were analyzed and answered with correction, and ignore all enhancements and market adaptations. Moreover, our analysis was limited only on defects reported by the NIV project with two customer references, that are related to the node system test and basic configuration of the network test. Analyzing only the customer that is common for all SD organizations, we secure the same treatment in late defect detection process for all observed SD organizations. Note that, due to the product line development, the same software unit can be present in several product releases, as well as a number of node configurations intended to serve in several markets and customer sites. So, the number of different customers that can be involved into modification handling vary among the software units, and it is not necessarily that only one customer is involved into modification handling process of all SWUs that were treated by SD organizations within selected GSD project.

4.4 Threats to Validity

Since we perform an empirical case study, it suffers from a number of threats to validity. According to [20] there are four different aspects of validity: internal, external, conclusion, and construct validity.

In our study we identified the following threats to validity. The threat to external validity is the fact that the study is performed only within one GSD project and that the study was not performed on the random sample of software units. However, the company where the study is performed is ISO certified and six participating organizations are on the CMM level 2 and three on the CMM level 3. Hence, it is quite possible that the findings generalize to such organizations.

The stated conclusions are based on the data collection from the well defined tools that are used for a long time by the organization's personnel. Eventual bias caused by data collection in the considered critical case is eliminated by using two sources of data collection as already explained in Section 4.3. Moreover, large sample of the data collected increases the reliability of the study.

The construct validity examines whether the selected measures are representative for the considered research questions. Effectiveness is a common measure used for the evaluation of defect detection techniques. In our case, we are comparing the effectiveness of early defect detection, performed by different organizations that are supposed to follow the same process. We used them as a measure of organizational impact. On the other hand, the defect detection process is not so strict and may involve various defect detection techniques depending on the defined defect detection strategy. So, the differences in the early defect detection process may be caused by differences in the chosen detection strategy, and not by the organizational diversity. In the case of the study, the defect detection strategy in early phases is defined independently for each SWU by the

development team. Thus, taking a large sample of software units solves this issue. Also, the effect of the software unit difficulty is assumed to be minimized by takeing a sample of software units large enough per organization.

5 Results

In this section, the results of the statistical analysis of the collected data regarding the defect detection process are explained. In the first subsection the descriptive statistics is presented for all collected and derived variables used in the study. In the rest of the section we deal only with the defect density.

5.1 Descriptive Statistics

The main concern of the statistical analysis is the random variable measuring the defect density as defined in Section 2. We consider separately the defect density of early and late defect detection denoted, respectively, by X_0 and X_1. For both random variables X_0 and X_1, as explained in Section 4.3, we have collected nine samples, one for each of nine SD organizations involved in the software design and maintenance projects. These SD organizations and associated samples are denoted by upper case letters A to I.

Table 2. Descriptive statistics for random variable X_0

SD org.	N	Mean	Std. dev.	95% C.I.	Min.	Median	Max.
A	13	11.740	12.893	[3.949, 19.531]	1.177	7.485	45.281
B	9	10.441	7.616	[4.587, 16.295]	3.333	6.536	25.779
C	33	17.279	19.793	[10.261, 24.298]	0.799	8.972	82.192
D	17	16.211	14.125	[8.949, 23.473]	2.515	12.176	47.393
E	34	8.088	6.512	[5.816, 10.360]	0.528	5.855	28.221
F	5	10.275	5.231	[3.780, 16.771]	5.199	8.523	18.692
G	45	10.478	12.485	[6.727, 14.229]	0.128	5.583	62.500
H	11	18.083	24.104	[1.890, 34.276]	1.916	9.332	86.752
I	49	22.957	27.955	[14.927, 30.987]	1.268	12.976	151.163

The descriptive statistics of the defect density random variables X_0 and X_1 measured per SWUs in each of nine SD organizations are summarized in Table 2 for X_0 and in Table 3 for X_1. In the tables the column labels "N", "Mean", "Std. dev.", "95% C.I.", "Max.", "Median", and "Min" stand for the number of observations, mean value, standard deviation, 95% confidence interval, maximum, median, and minimum of the sample, respectively.

Observe that the sample measured in SD organization F consists of only five SWU. Since this sample size is insufficient for a significant statistical analysis, we removed it from further analysis. In the remaining samples we removed the outliers. The descriptive statistics given in the tables is for the samples in which the outliers are removed.

Table 3. Descriptive statistics for random variable X_1

SD org.	N	Mean	Std. dev.	95% C.I.	Min.	Median	Max.
A	13	14.547	13.838	[6.185, 22.909]	0.657	9.543	41.147
B	9	16.218	17.496	[2.769, 29.666]	1.669	9.757	57.143
C	33	8.855	8.377	[5.885, 11.825]	1.268	6.247	37.500
D	17	10.228	12.960	[3.564, 16.891]	0.846	6.827	47.847
E	34	18.241	22.179	[10.502, 25.979]	2.253	11.389	114.755
F	5	8.186	4.065	[3.138, 13.233]	4.422	6.490	14.521
G	45	12.960	19.269	[7.171, 18.748]	0.385	4.785	85.386
H	11	17.017	18.275	[4.740, 29.294]	2.874	12.273	65.476
I	49	17.399	21.857	[11.121, 23.677]	0.856	12.622	142.857

5.2 Normality Tests

The parametric hypothesis tests are more robust and reliable than non–parametric tests. However, a general assumption for all parametric tests is that the analyzed data samples are normally distributed. Hence, in order to justify the use of parametric tests for our research hypotheses, we need to check whether the samples follow the normal distribution.

Table 4. Normality test for transformed random variable X_0

SD org.	Kolmogorov–Smirnov		Shapiro–Wilk	
	d statistic	p–value	W statistic	p–value
A	0.165	> 0.20	0.961	0.768
B	0.178	> 0.20	0.915	0.355
C	0.129	> 0.20	0.954	0.173
D	0.096	> 0.20	0.958	0.600
E	0.096	> 0.20	0.963	0.302
F	0.165	> 0.20	0.983	0.950
G	0.078	> 0.20	0.974	0.390
H	0.154	> 0.20	0.968	0.869
I	0.091	> 0.20	0.982	0.632

The standard tests for normality are the Kolmogorov–Smirnov and Shapiro–Wilk test. We apply both tests to the nine samples for each of the defect density random variables X_0 and X_1. However, it turns out that none of the samples follows the normal distribution. This result was expected, since several authors investigating the defect density in software defect detection process already reported that the defect detection random variable follows the log–normal distribution. Hence, we transformed all the samples by applying the natural logarithm, and applied the Kolmogorov–Smirnov and Shapiro–Wilk test to the transformed

samples. The results of the normality tests on the transformed samples of random variables X_0 and X_1 are given in Table 4 and Table 5, respectively. From the tables we read that all the tests are significant (p–value > 0.05), which means that all the transformed samples follow the normal distribution.

Table 5. Normality test for transformed random variable X_1

SD org.	Kolmogorov–Smirnov		Shapiro–Wilk	
	d statistic	p–value	W statistic	p–value
A	0.153	> 0.20	0.919	0.240
B	0.145	> 0.20	0.969	0.886
C	0.085	> 0.20	0.964	0.331
D	0.115	> 0.20	0.973	0.870
E	0.112	> 0.20	0.946	0.094
F	0.215	> 0.20	0.962	0.819
G	0.098	> 0.20	0.968	0.253
H	0.182	> 0.20	0.941	0.532
I	0.096	> 0.20	0.964	0.144

As a consequence of these conclusions, we are free to apply parametric statistics for the further analysis.

5.3 Hypothesis Testing

The research hypotheses are stated in Section 4.2. In order to test these research hypothesis we applied one–way ANOVA on transformed random variables X_0 and X_1. The assumptions for applying ANOVA is that the dependent variables are normally distributed and that the groups have approximately equal variance on the dependent variable. The normality of the transformed samples was confirmed in Section 5.2. To verify the homogeneity of variances of the transformed samples we use Levene's and the Brown–Forsythe test. The results of these tests are given in Table 6. It turns out that for X_0 both tests show that the assumption of homogeneity of variances between samples should be rejected (p–value < 0.05). On the other hand, for X_1 both test confirm the homogeneity of variances. Thus, the assumptions for applying ANOVA are satisfied only for random variable X_1.

Consider first the random variable X_0. Since the assumptions of ANOVA are not satisfied, we performed the non–parametric tests for equality of means to test the research hypothesis HA_0. These tests are the Kruskal–Wallis ANOVA and median test. The results are given in the first row of Table 7. The conclusions of the two tests are not consistent. The Kruskal–Wallis ANOVA would imply that HA_0 should be rejected (p–value < 0.05), while the median test implies that it should not (p–value 0.05). Since the p–value for the median test equals $p = 0.055$, which is very close to the critical value 0.05, we conclude that we should reject the hypothesis HA_0.

Table 6. Homogeneity of variance tests

Transformed Var.	Levene		Brown–Forsythe	
	F statistic	p–value	F statistic	p–value
X_0	3.158	0.003	2.829	0.008
X_1	1.861	0.078	1.592	0.140

Afterwards, we also performed ANOVA, and the results are given in the first row of Table 8. The column labels "SS", "df", "MS", "F", and "p" stand for the sum of squares, degrees of freedom, mean squares ($\frac{SS}{df}$), F statistic, and p–value, respectively. The results show that the equality of means hypothesis should be rejected (p–value < 0.05). Although this conclusion should not be taken seriously, since the assumptions for ANOVA are violated, it provides more evidence in favor of our conclusion that the hypothesis HA_0 should be rejected.

Table 7. Non–parametric tests

Transformed var.	Kruskal–Wallis ANOVA		Median Test	
	H statistic	p–value	χ^2 statistic	p–value
X_0	20.170	0.005	13.803	0.055
X_1	12.786	0.078	10.309	0.172

Consider now the random variable X_1. In that case the assumptions of ANOVA for the eight considered samples are satisfied. Hence, we apply ANOVA, and the results are given in the second row of Table 8. The conclusion is that the equality of means hypothesis HB_0 could not be rejected (p–value > 0.05). Thus, we may assume that the means of the eight samples for the late defect detection density are equal.

Table 8. ANOVA tests

Transformed var.	SS	df	MS	F	p
X_0	25.157	7	3.594	3.069	0.004
X_1	16.337	7	2.334	1.921	0.067

In order to confirm this finding, we additionally performed non–parametric tests. The results are presented in the second row of Table 7. They also show that the hypothesis HB_0 could not be rejected (p–value > 0.05). This confirms our conclusion that the means of X_1 samples are equal.

6 Discussion

The study is performed on samples of software units that are grouped, by organizational responsibility, into nine software design organizations that were responsible for the software unit design. For software units, the effectiveness of defect detection process has been measured in two consecutive phases, early and late defect detection. The early defect detection of a particular software unit is performed by the same organization that performed the software design. The late defect detection is performed by the verification organization for the complete software system, which is composed of all the analyzed software units. The main outcome of the defect detection process is a number of defects that are identified and reported per each software unit. So, the effectiveness of defect detection process is measured per software unit as defect density, that is the number of identified defects per software unit's volume of code.

In the analysis, one of the research goals was exploring the difference in the early defect detection effectiveness among the software design organizations involved into the same global software development project. The results of the analysis indicate that there is a significant difference in the effectiveness of early defect detection among the software design organizations. Note that the same organization was responsible for design and early defect detection on software unit. One may conclude that software design organizations are not equally effective in early defect detection process, but this conclusion might be misleading. The differences in early defect detection effectiveness may be a result of the differences in organizational defect injection process as result of less experience, more complex part of functionality for implementation or other factor identified in [21], although that is also performed by the same organizations.

The other research goal was to understand the influence of the software design organizational distribution on the late defect detection effectiveness and the product quality. The research hypothesis was testing the significance of differences in late defect detection effectiveness for the groups of software units with aforementioned software design responsibilities. Note that late defect detection is performed for all software units by a single verification organization, that is different from the software design organizations. Surprisingly, the result of the analysis was that there is no significant difference in the effectiveness of late defect detection between the software units that were grouped according to the software design responsibility. One may conclude that late defect detection was equally effective for all groups of software units, that were developed and early verified by different software design organizations. In other words, the diversity in early defect detection effectiveness is not statistically significantly reflected in diversity in late defect detection effectiveness.

From these results, we may argue that the defect density is a good effectiveness measure in this context. The finding that late defect detection was equally effective for all software units no matter of effectiveness achieved in early defect detection may be followed by the conclusion that there is no relationship between early and late defect detection effectiveness. Moreover, the diversity of

early defect detection performed within software design organization does not influence the later defect detection effectiveness.

These conclusions should be taken with caution and should be analyzed in the light of the previous work. For example, in a number of studies the early defect detection is identified as a good predictor of late defect detection and remaining defect density in the system. On the other hand, in [9] and [10], it is empirically identified that units with many defects identified in the early defect detection phases have small number of defects in late defect detection phases, and units with majority of defects in late defect detection phases have been the units with small number of defects in early defect detection phases. From these conclusions we may expect that variation in the early defect density is repeated for late defect density that does not happen in our case.

Note that in this study the organizational influence participating in the GSD on the effectiveness of defect detection process and product quality is evaluated. For a proper valuation of the quality produced by development organization the evaluation of outliers is also an interesting indication. Generally, when evaluating organizational influence on the product quality the analysis of the number of faults should not be excluded although one should be aware that the number of faults depends on software size and the relation is not linear.

7 Conclusion and Future Work

The importance of the study presented in this paper lies in its empirically based evaluation of the influence of software development distribution on the global software development. Many challenges have been identified by researchers and practitioners in the global software development, but still a limited empirical evidence is presented. This study is a step in that direction.

In the context of this study, the diversity of early defect detection effectiveness has not influenced the late defect detection effectiveness. On the other hand, the early defect detection was performed by different organizations in the GSD project, while the late defect detection is performed by the same verification organization. This suggests that the defect density is better as the measure of the organizational effectiveness than as the measure of product quality. However, the presented study should be replicated in different contexts so that such conclusion could be generalized.

Another direction for future work is to explore defect detection effectiveness diversity among different GSD projects in the same context of the study. The results of these analysis could bring conclusions on the impact of GSD process evolution on diversity of defect detection effectiveness.

Finally, to get clear picture of the defect detection effectiveness impact on product quality, the defect density over the software unit life–cycle and failure density on the customer site should be also analyzed. Furthermore, the influence of organizational diversity on the severity of defects is also an issue to explore.

References

1. Ajila, S., Dumitrescu, R.: Experimental Use of Code Delta, Code Churn, and Rate of Change to Understand Software Product Line Evolution. J. Syst. Softw. 80(1), 74–91 (2007)
2. Battin, R.D., Crocker, R., Kreidler, J., Subramanian, K.: Leveraging Resources in Global Software Development. IEEE Softw. 18(2), 70–77 (2001)
3. Bird, C., Nagappan, N., Devanbu, P., Gall, H., Murphy, B.: Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. In: 31st International Conference on Software Engineering ICSE 2009, pp. 518–528. IEEE Computer Society, Washington DC (2009)
4. Carmel, E., Agarwal, R.: Tactical Approaches for Alleviating Distance in Global Software Development. IEEE Softw. 18(2), 22–29 (2001)
5. Cataldo, M., Nambiar, S.: On the Relationship between Process Maturity and Geographic Distribution: an Empirical Analysis of their Impact on Software Quality. In: 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering ESEC/FSE 2009, pp. 101–110. ACM, New York (2009)
6. Chrissis, M., Konrad, M., Shrum, S.: CMMI: Guide for Process Integration and Product Improvement. Addison-Wesley, Boston (2004)
7. Ebenau, R.G., Strauss, S.H.: Software Inspection Process. McGraw Hill, Workingham (1994)
8. Fagan, M.E.: Design and Code Inspections to Reduce Errors in Program Development. IBM Syst. J. 15(3), 575–607 (1976)
9. Fenton, N.E., Ohlsson, N.: Quantitative Analysis of Faults and Failures in a Complex Software System. IEEE Trans. Softw. Eng. 26(8), 797–814 (2000)
10. Fenton, N., Neil, M.: A Critique of Software Defect Prediction Models. IEEE Trans. Softw. Eng. 25(5), 675–689 (1999)
11. Herbsleb, J.D.: Global Software Engineering: the Future of Socio–technical Coordination. In: 2007 Future of Software Engineering FOSE 2007, pp. 188–198. IEEE Computer Society, Washington DC (2007)
12. Institute of Electrical and Electronics Engineers (IEEE): Software Verification and Validation. IEEE Standard 1012–2004, Software Engineering Standards Committee of the IEEE Computer Society (2005)
13. International Organization for Standardization and International Electrotechnical Commission (ISO/IEC): Software Engineering – Product Quality – Part 1: Quality model. ISO/IEC Standard 9126–1, Geneva (1997)
14. Musa, J.D., Iannino, A., Okumoto, K.: Software Reliability Measurement, Prediction, Application. McGraw-Hill, New York (1987)
15. Nagappan, N., Murphy, B., Basili, V.: The Influence of Organizational Structure on Software Quality: an Empirical Case Study. In: 30th International Conference on Software Engineering ICSE 2008, pp. 521–530. ACM, New York (2008)
16. Parnas, D.L.: On the Criteria to be Used in Decomposing Systems into Modules. Commun. ACM 15(12), 1053–1058 (1972)
17. Pfleeger, S.L.: Software Engineering, Theory and Practice. Prentice-Hall, New York (2001)
18. Project Management Institute (PMI): A Guide to the Project Management Body of Knowledge (PMBOK Guide). PMI, Newtown Square (2004)

19. Third Generation Partnership Project (3GPP): Technical Performance Objectives. 3GPP, Technical Specification Group Core Network (2005)
20. Wohlin, C., Höst, M., Henningson, K.: Empirical Research Methods in Software Engineering. In: Conradi, R., Wang, A.I. (eds.) ESERNET 2003. LNCS, vol. 2765, pp. 7–23. Springer, Heidelberg (2003)
21. Jacobs, J., van Moll, J., Kusters, R., Trienekens, J., Brombacher, A.: Identification of factors that influence defect injection and detection in development of software intensive products. Inf. Softw. Technol. 49(7), 774–789 (2007)

Managing Process Diversity by Applying Rationale Management in Variant Rich Processes

Tomás Martínez-Ruiz, Félix García, and Mario Piattini

Alarcos Research Group, Institute of Information Systems and Technologies,
Escuela Superior de Informática, University of Castilla-La Mancha,
Paseo de la Universidad 4, 13071 Ciudad Real, Spain
{tomas.martinez,felix.garcia,mario.piattini}@uclm.es
http://alarcos.esi.uclm.es

Abstract. Process diversity arises as software processes are influenced by organization, project and other contextual factors. Managing this diversity consists of considering how these factors actually modify the process. Variant rich processes offer support for process tailoring, but they do not currently link these changes with the business factors motivating them. The lack of decision traceability signifies that variant rich processes are not suitable for addressing process diversity. This article aims to fill this gap by applying rationale management to supporting decision-making when tailoring processes. Rationale management has become one of the main assets in variant rich process tailoring, since it handles how context-related factors are transformed into real variations in the tailoring process, as a consequence of well-reasoned and traceable steps. An application study shows how rationale provides useful mechanisms with which to tailor a process according to its context of enactment.

1 Introduction

Software processes are implemented in organizations whose characteristics, teams and people affect how the software process will actually be instantiated [1]. Software processes therefore diverge over time according to the projects or teams to which they are applied [2], while they are strongly based on international process standards [3]. Software process development organizations must therefore handle the diversity of processes by means of considering how they vary from the standard process, and more specifically, what exact variations standard processes experience to allow them to be adapted according each context.

Variability in software processes supports tailoring, and these therefore meet diversity in their enactment context. In previous works we have developed the Variant Rich Process Paradigm (VRP), which supports software process variability by following the Product Line Engineering (SPLEs) and Aspect Oriented Software Engineering (AOSE) approaches [4, 5]. The paradigm has been implemented in vSPEM notation [6, 7], which enhances SPEM 2.0 to support the modelling of Variant-Rich Processes. vSPEM is also the core notation of SPRINTT, a framework to support the institutionalization of software processes [8]. This framework can help users to define and institutionalize process variations and therefore supports the management of process

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 128–142, 2011.
© Springer-Verlag Berlin Heidelberg 2011

diversity, but the framework lacks support to link these variations with the real causes in the context motivating the change. This lack actually signifies two main disadvantages: variations are neither traceable nor justifiable, and knowledge cannot be extracted from them, since the decisions about variation in a process are human-based activities.

Human based decisions have been supported in software engineering by means of Rationale Management [9]. Rationale management deals with managing all the reasoning that goes into determining the design of an artifact [9]. In fact, stored and retrieved information about decisions during the creation of a software product is useful in other different phases of this development or maintenance. This information may also be useful in making similar decisions in similar projects. In fact, design Rationale has been successfully implemented in software engineering approaches such as product lines [10], and is also used in software process evolution [11]. As a result of both successful implementations, it may be applied to variant rich processes.

This paper tackles the inclusion of Rationale Management in the previously developed Variant Rich Process Paradigm and in the vSPEM notation. Rationale is considered as the cornerstone that connects the process' context, the variant rich processes and all the tailored processes created from them. As it contextualizes variations, it also supports the combination of context and project performance knowledge in institutionalizing processes with SPRINTT.

This paper is organized as follows. Section 2 summarizes the state of the art with regard to rationale and knowledge storing in software engineering. Section 3 shows how rationale can contribute to process institutionalization in the SPRINTT framework. The approach for managing rationale in variant rich processes is presented in Section 4. Section 5 shows the application of rationale in tailoring a software process in a company. Finally, conclusions and future work are presented in Section 6.

2 State of the Art

Rationale management is widely defined in [9]. Schneider [12] proposes to use it in a manner which is as embedded as possible in the actually tasks with the aim of minimizing effort, and considers the benefits to be higher than the effort involved in creating it. In addition, Lee [13] describes the six main issues concerned with managing rationale, including the different storing layers and elements. As a result, it has been used in software engineering [14], and has been specially applied in deciding variations of product lines. Knodel and Muthig [10] propose a five-step approach for documenting variations. However, previous to these works, there were some approaches for managing traceability in product lines [15, 16]. All of them have the same main objective: linking variations with the actual causes motivating them, and offering the best support with regard to deciding variations, but they do not provide mechanisms with which to support variation knowledge reuse.

On the other hand, some approaches apply rationale to software processes. Ocampo et al. [11] propose using rationale for process evolution. Nkwoka et al. [17] also use rationale in an environment for process improvement. Both approaches focus

on enhancing processes by means of rationale, so all of them evolve or improve processes by using well-defined variations from variant rich processes.

With regard to the management of Process Diversity, it is important to note that this is not a new topic in software processes, as it has already been applied to support development and maintenance activities. In fact, Sutton et al. suggested it in 1996 when they discussed the programming of software processes, since programmed processes diverge each other [18]. In the field of process tailoring some works can be considered such as Simidchieva et al. [19], Barreto et al. [20]. These works tackle diversity but they lack of specific support for linking variations with actual causes and learning from variations generated knowledge. Caivano et al. propose the use of patterns in making decisions about the processes and solving diversity [21], which strongly link modifications with the causes that originate them. Henninger [22] uses the experience factory approach from Basili [23] to design a process diversity knowledge storing system in which adaptations are strongly performed by means of rules.

In summary, the related works are mainly focused on applying variability mechanisms for process tailoring or applying rationale to processes, but they do not tackle both approaches in an integrated manner. Moreover, they do not consider that process tailoring could benefit from rationale management just as much as products do, since product and process tailoring are similarly managed. The work presented herein aims to enhance the VRP tailoring approach with rationale for driving variations with the intention of obtaining advantages such as traceability, justification of variations or knowledge reuse, and of therefore optimizing variations when tailoring variant rich processes.

3 Rationale in the SPRINTT Institutionalization Framework

Institutionalization constitutes the superlative degree of process tailoring. It implies process adaptation from the organization's standard processes. The SPRINTT environment was designed to provide support in process institutionalization [8], as it allows feedback in a continuous four step cycle (tailoring, execution, analysis and standardization), and thus helps to determine which variations are better than others and promote their re-use: they are *validated*.

A comparison of processes with cars may help the reader to understand the rationale needed. Hire car companies do not classify cars according to the elements of which they are composed (engine, battery…) or by names, since this would not be logical However, if we wished to make a long journey, a car hire company would offer us a private car, if we had a large family they would give us a mono van, and if we were moving house they would offer us a removal van. This does not mean that any of these models is always better than the others, but all of them fit particular usage circumstances better than the others.

Similarly, variants in a variant rich process need to be stored according to their context. SPRINTT proposes to validate the variations by comparing tailored process definition with execution logs, but a good variation in a process may be clearly bad in the case of readapting the same variant rich process in order to fit the characteristics of another totally different context. Rationale management therefore plays a principal role in linking and justifying variations with their application context, and then providing validated and *contextualized* variants. The context knowledge is therefore

stored with any other project or process knowledge. Moreover, tailoring new processes based on the knowledge base also takes into account the fact that *the variations were well enacted in processes tailored to similar contexts.*

Fig. 1 describes the main elements in the SPRINTT cycle, and how these interact with rationale in order to define the contextualized variants. If we start from *the process adaptation* step, both rationale and the tailored process are created by taking characteristics from the enactment context, and they are both stored. Process is enacted in the corresponding project (*process enactment* step), within its context, and is also stored. The post-mortem *analysis* step retrieves tailored processes and projects, and decides the validated variations, i.e. those whose changes were correct. At this point, the validated variants are those which are well enacted. As these are identified, they could be realized again, but *when* and *how* can they be satisfactorily reused?

The *standardization* step takes the validated variations and contextualizes them by using the previously stored rationale (including context and justification). That is, it specifies the context in which the validated variations were used, and supports the reuse of the variations in similar contexts.

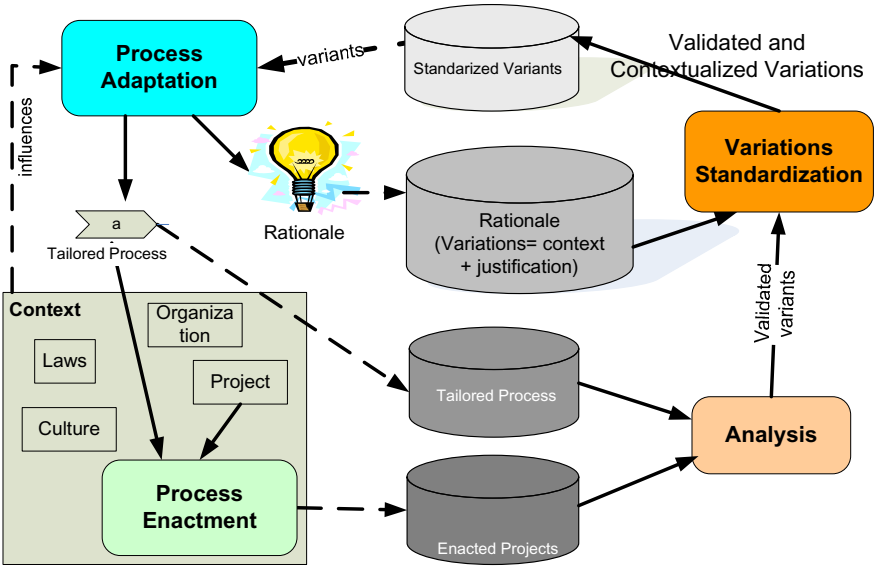


Fig. 1. SPRINTT Institutionalization Cycle with Rationale Management

Tailoring support in the sprint environment is based on controlled and scoped variations. In previous works we have developed the vSPEM notation, which adds the new package *Variations* to SPEM, supporting on-point and crosscutting variations in software processes. On-point variations are defined by means of variation points and variants of the process composing elements by applying the Software Product Lines (SPLEs). Crosscutting variations are defined by applying the Aspect Oriented Software Engineering (AOSE) approach, and they support the encapsulation of several of these on-point variations in process aspects, providing the capability of crosscutting tailoring software processes.

4 Rationale in Tailoring Variant Rich Processes

Rationale is the basis by which to optimally tailor software process from the viewpoint of variant-rich processes. vSPEM supports it through its inclusion in a new package, the *VRichProcess,* which is focused on tailoring software processes by means of variant rich processes (based on the variability elements in the *Variation* package) with rationale to handle how to carry out variations.

Fig. 2 shows the new elements defined in the *VRichProcess* package and their relationships. The elements, which inherit from the *Variations* package, are represented with white boxes. The new elements allow us to describe the context in which a tailored process is applied, which is used to tailor it (elements filled in grey), and to manage the rationale from the tailoring in order to determine how to transform the characteristics of a context into the optimal variations in the variant rich process (elements marked with bold border). These elements apply the storage layers defined by Lee [13], and their intention is to capture information as it is produced, from the problem description (project and variant rich process) to the solution (tailored process to that project), that is from the elements on the left to those on the right of the model shown in Fig. 2. They also elicit the main assets in consistently building the solution.

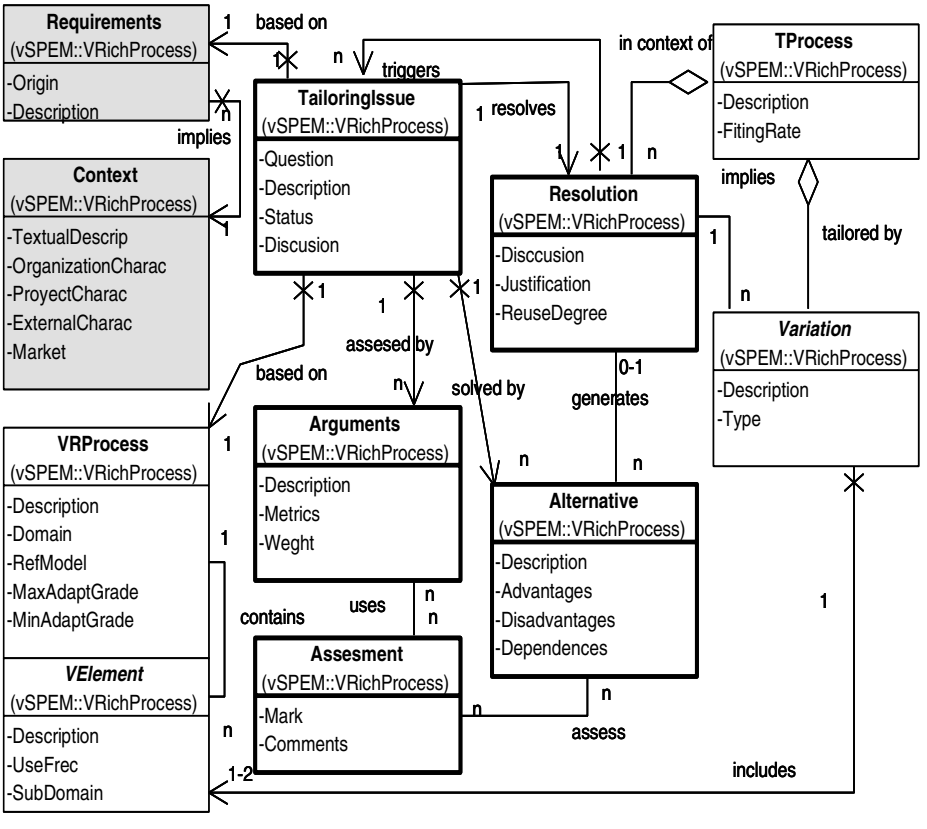


Fig. 2. Elements supporting rationale in the vSPEM notation

The white and non-marked classes in Fig. 2 inherit from homonymous classes in the *Variations* package and they support the process variability definition and their usage in tailoring processes. They also add some new attributes to support rationale, as Tables 1 and 2 show.

Table 1. Description of the elements defining variability in processes

Element	Description
VRProcess	Represents a process with variability (variants, variation points, process aspects). Attributes: *Domain,* which defines the domain the variant rich process is designed in (*real time software, systems, management, embodied, Artificial Intelligence*, etc); *Description* allows the broad definition of the variant rich process; *RefModel* specifies the reference models the variant rich process is based on, if any; *MaxAdaptGrade* and *MinAdaptationGrade* indicate the maximum or minimum adaptation grade of the variant rich process. The former corresponds with the maximum number of variation points defined into the variant-rich process plus the maximum number of process aspects that could be used in the VRP; the latter corresponds with the mandatory number of variation points included in the variant-rich process.
Velement	Variable element in a variant rich process. There are three types: variation points and variants (for on-point variations) and process aspects (for crosscutting variations). It includes three attributes: the S*ubDomain* it is designed for, if any; a long *description*; and a ratio of frequency of use, *UseFrec*.

Table 2. Description of the elements defining tailored processes

Element	Description
Variation	Describes the variations (both on-point and crosscutting). Attributes: The *Type* differentiating the on-point and crosscutting variations; a detailed *description* of the '*how to*' of the variation.
Tailored Process	Represents the tailored process created from the variant rich process with the characteristics of the context. It includes a detailed *Description*, and a *FittingGrade* signifying how well the tailored process fits the context.

The *VRichProcess* package include the *Context* and *Description* elements (filled in grey) with which to include knowledge about the *context* in which the processes are enacted, and supports in a high abstraction level of the goals involved in tailoring any process (Table 3).

Specific support for Rationale is provided by means of five elements detailed in Table 4. These elements are consistent with the explicit representation of the decision sub-layer proposed in [13]. They are also the *core* of the *VRichProcess* package (marked with a bold border).

In summary, the rationale elements included in the *VRichProcess* package specify the way in which the tailoring knowledge is created and used in the process institutionalization cycle (see Fig. 1). Three main objectives are fulfilled by means of these

Table 3. Description of the elements in the Context description area

Element	Description
Require-ment	Process tailoring requirements which the variant rich process need to achieve in order to fit the context as well as possible. Attributes: a *description* and *origin,* which indicates whether the requirement is owing to the project, the organization or external characteristics.
Context	Represents the context in which the process will be executed. Attributes: *TextualDescription* of the organization. *OrganizationChar* specifies any characteristic of the organization affecting the process enactment (such as the segment they work, the number of employees…); *ExternalChars* defines the context in which the organization is, but which cannot be controlled or changed (laws, culture etc); *ProjectChars* includes the project plan; *Market* indicates the type of market the organization uses to develop software (healthcare, army…).

Table 4. Description of the elements in the Rationale set

Element	Description
Tailoring Issue	Represents a tailoring issue of the variant rich process, which is a change in the variant rich process in order for the tailored process to meet one specific requirement. Attributes the *question* should satisfy; a complete *description*; the status (open or closed) of the tailoring issue; and any *discussion* needed to decide about the tailoring issue.
Alternatives	Proposals for resolving the tailoring issue. Attributes: a long Description; the set of *Advantages* and *Disadvantages*; and a list of the *Dependences* it could generate. Since variations are not independently executed, variability mechanisms also consider dependences between them [6, 7], and these dependences are therefore included in the documentation of all the alternatives. In fact, the dependences of an alternative are an important point when it is assessed.
Arguments	Criteria for deciding how good the alternatives are. Attributes: a long *Description*; a *Metric* for assessing and comparing the alternatives; and the *Weight*, defining the different significance of the arguments.
Assessment	Represents the assessment of each argument in each alternative. It includes two attributes: the *Mark* (how well the alternative to that argument is satisfied) and the *Comments* about the assessment.
Resolution	Selected alternative to resolve the tailoring issue by means of some variations. Attributes: the *discussion* about analyzing alternatives, and the *justification* of why the alternative must be selected; *ReuseDegree,* indicating from 0 to 1 to what extent the resolution is based on previous ones. When the alternative becomes a resolution, its dependences become new triggered tailoring issues to ensure consistence.

mechanisms: traceability from end to end (context and variations, respectively); justifiability, since through rationale it would be possible to ensure that each variation is

created as the best solution after assessing all the alternatives, making the variant rich process fit the process context; knowledge reusing since the approach allows the knowledge to be extracted from the variations and managed.

5 Application Study

An application study is being conducted on the use of the vSPEM notation (including the rationale management) as the core mechanisms for applying the MEDUSAS methodology [24] in a given project. MEDUSAS is an environment containing the processes, which aims to support the evaluation and improvement of the Usability, Security and Maintainability of software products. To achieve its purpose, MEDUSAS is composed of a methodology to guide organizations in carrying out the quality assurance process and metrics, indicators and guidelines to support the evaluation process.

The methodology includes the quality assessing and support processes, clearly defining *who*, *what* and *how* quality is assessed. It contains three processes, the *Quality Assessment Process*, which is the main process and is composed of four phases (*Planning* to establish the assessment contract and specify the assessment plan; *Specification* to determine the scope of the quality assessment project, and what to assess from the products; *Execution* which includes all the assessment activities; and *Ending* which is focused on realizing the assessment report); and the *Quality Assessment Management*, and the *Infrastructure Management* processes, which support the first process.

There are two ways in which to apply MEDUSAS: by considering the methodology as another asset in the client organization and guaranteeing alignment between its processes and those from the development cycle (on-site mode), or by applying MEDUSAS to ensure the quality of the artefacts that a third organization is developing for the client (external assessment mode). Table 5 shows a description of the variant rich process of MEDUSAS developed according to the rationale structure presented in Section 4. This VRP provides mechanisms for tailoring MEDUSAS methodology according to the different contexts in which it could be used. The defined variations are focused on two main points: the first is to provide the methodology with the profile, in accordance with the assessment type (internal or external); this is managed by means of two process aspects dealing with crosscutting variations; each of them varies some activities and tasks in all three processes, in accordance with the type of implementation. The second includes tailoring support depending on whether certain roles take part in the project, the standards they use, and the list of evaluating work products, among other aspects. All these are on-point variations and are defined by means of variation points and variants. Table 6 lists some of the variation elements included in the variant rich process, These are variants, variation points and process aspects. However, a detailed description of them is not provided here owing to space limitation (a definition of these concepts appears in [6, 7]). This study presents the first occasion on which the variants were used to tailor a process by applying the rationale steps.

Table 5. Description of the MEDUSAS Variant-rich process

Description	*Presented above*
Domain	Quality Assessment
Reference Model	ISO/IEC 29119, ISO/IEC 25020, ISO/IEC 25000
MaxAdapGrade	17 (15 *variation points* +2 *process aspects*)
MinAdaptGrade	8

5.1 Description of the Tailoring Context

The MEDUSAS Variant Rich Process is applied in the context of a Spanish *organization* focused on ensuring quality in the field of software testing, mainly to other enterprises or public institutions. It works in *nearshoring* mode since its offices are in Ciudad Real (Spain). It must apply some Spanish laws, such as the *Personal Data Protection Law* (LOPD), *or the Information Society Services Law* (LSSI).

The project in which the MEDUSAS will be applied is that of ensuring the quality of the software that a third organization is developing for a *client*. Table 7 resumes the information about the context, according to the context structure Section 4 presents. Based on the previously described context, some requirements for process tailoring appear, and these are listed in Table 8.

Table 6. Set of variability elements in the MEDUSAS variant rich process

Id	Description	Use Freq	Subdom.
1	Process Aspect "On-site".	First Use	n/a
2	Process Aspect "External".	First Use	n/a
3-5	Role variation point in some activities.	First Use	n/a
6-8	Role variants concerning the "*Client Team*", "*Quality Responsible*" and "*External Team*" roles.	First Use	n/a
9	Work product variation point related to the "*Deciding Assessment Model*" activity.	First Use	n/a
10	Work product variant "Standard X".	First Use	n/a
11,12	Activity variation points in the *Specification* Phase.	First Use	n/a
13,14	Variants of some optional activities.	First Use	n/a

Table 7. Characterization of the context in which the Variant Rich Process is applied

Textual Desc.	*Presented above*
Organization Characteristics	It works ensuring software quality in the field of software testing. It includes computer experts from the UCLM.
Project Characteristics	It focuses on assessing the quality of the software that a *third organization* is developing for the *client.* It is realized from the organization's offices (externalized mode).
External Char.	Projects must meet Spanish laws such as *LOPD* and *LSSI*.
Market	External Quality Assessment

Table 8. Set of requirements for a process meeting the organization's needs

Id	Origin	Description
1	Project	It must be adapted to be applied in an externalized context.
2	Organization	The complexity of the methodology must be simplified.
3	Organization	The number of roles involved must be minimized.
4	Project	The number of activities and tasks must be reduced, if possible.
5	Project	Any adaptation must affect the quality of the methodology.
6	Project & external	The project and external laws force the standard for quality assessment X to be used.

5.2 Rationale in Tailoring MEDUSAS to the *Organization*

If we consider the requirement the context of the *Organization* presents, then the variant rich process of MEDUSAS needs to be tailored. The matching of the tailoring requirements (Table 6) with the variability elements in the variant rich process (Table 8) generates certain *Tailoring Issues*, as Table 9 details. These tailoring issues propose the use of some of the variability elements in the MEDUSAS VRP in order to fit the requirements to the specific context that the executing project demands.

Satisfying the tailoring issues presented in the Table 9 generates some alternatives, as Table 10 lists; the resolution of the tailoring issue is obtained from these. Table 10 does not show certain details regarding the application of the variability elements owing to confidentiality constraints, but it depicts the different options that could be used when tailoring the software process. These alternatives are assessed according to the arguments presented in Table 11.

Table 9. Tailoring Issues for tailoring the variant rich process to the *Organization*

Id	Req.	Question	Description	Status	Discussion
1	1	The possibility of using the process aspects to adapt the methodology according to the execution mode.	The VRP contains some aspects to….	Open	It might be interesting to…
2	2, 3, 5	The possibility of reducing those optative roles	Some roles are optative...	Open	It might be a good idea…
3	5, 6	The possibility of assessment using the standard X	The standard used may…	Open	…
4	2, 4, 5	The possibility of reducing optative tasks and activities	Removing the optative activities…	Open	…

Table 10. Description of the alternatives for solving the previous tailoring issues

Id	Description	Adv. /Disad	Dep.
1.a	Uses the process aspect "*On-site*".	Confidential	None
1.b	Uses the process aspect "*external*".	"	"
1.c	Does not use the process aspects.	"	"
2.a	Considers the role, "*Client Team*".	"	"
2.b	Does not consider the role "*Client Team*".	"	"
2.c	Considers the role, "*Quality Responsible*".	"	"
2.d	Does not consider the role "*Quality Responsible*".	"	"
2.e	Considers the role, "*External Team*".	"	"
2.f	Does not consider the role "*External Team*".	"	"
3.a	Considers using the standard "X".	"	"
3.b	Considers using the standard "Y".	"	"
3.n	Considers not using standards.	"	"
4.a	Considers carrying out some optative tasks	"	"
4.b	Considers not using these optative tasks.	"	"

Alternatives are assessed by means of arguments, resulting in a matrix. Table 12 summarizes this matrix, showing alternatives in rows and arguments in columns. Because of confidentiality constraints, we show an estimation of how good they are. As a result, we assess them by using linguistic tags: *high*, *medium* and *low*.

As a result of the evaluation of alternatives, some are selected to solve each tailoring issue. These alternatives, and the tailoring issue they solve, are described in Table 13. Owing to space constraints, any of the presented resolutions triggers a new *Tailoring Issue*.

Table 11. Description of the Arguments for assessing the Alternatives

Id	Description	Metrics	Weight
a	The variations do not remove functionality.	The functionality is reallocated throughout the process	0,75
b	The Assessment's Quality and Confiability are not affected.	Variations do not affect the assessment quality	1
c	The process is complete.	No mandatory variation points are empty.	1
d	Alternatives fit the assessment client's necessities as much as possible.	The characteristics of the alternative meet the context requirements.	0,5

Table 12. Overview of the assessment of the alternatives by means of the arguments

	a	b	c	d		a	b	c	d
1.a	low	low	high	low	**3.a**	**med.**	**high**	**high**	**med.**
1.b	**med.**	**high**	**high**	**med.**	3.b	low	low	high	low
1.c	low	low	low	low	3...				
2.a	med.	high	high	low	3.n	low	low	high	low
2.b	**med.**	**high**	**high**	**med.**	4.a	med.	low	high	low
2.c	med.	high	high	low	**4.b**	**med.**	**high**	**high**	**med.**
2.d	**med.**	**high**	**high**	**med.**	**4.c**	**med.**	**high**	**high**	**low**
2.e	**med.**	**high**	**high**	**med.**	4.d	med.	low	high	med.
2.f	low	low	high	med.					

Table 13. Resolutions created after assessing the alternatives

Tailoring Issue	Alternatives Selected	Justification
1	1.b	The process aspect meets the context requirements
2	2.b, 2.d, 2.e	The "Client Team" and "Quality Responsible" optative roles can be deleted, but the "External Team" cannot, because it ensures the quality…
3	3.a	The variant rich process allows precisely the standard the context requires to be used.
4	4.b, 4.c	One optative activity is not included, but the other is included, as it ensures the quality of the assessments.

5.3 Tailored Process from the MEDUSAS Variant Rich Process

As a consequence of the Tailoring issues described in Table 9, and the solution described in Table 13, some variations are realized. The *Organization's* tailored MEDUSAS process effectively considers the enactment context. The principal asset is that the "*External* aspect" is used, signifying that the methodology has been explicitly configured to carry out external validations, which is the main characteristic of the enactment context. In addition, some roles, such as the "*External Team*" have been configured according to factors such as the quality assurance level, or the human resources available in the *Organization*. Both of them clearly influence the enactment, and not taking them into account may compromise the entire project execution. Some other adaptations regarding the standards used, or skipping certain activities, influence whether the resulting process is or is not in accordance with external laws and the budget.

All of the above signifies that the tailored process fits the enactment context to a high degree. It is therefore possible to ensure that the enactment consistence is higher than if the process to be enacted was the standard (and strict) version of MEDUSAS. The variations are listed in Table 14 and configure the process described in Table 15.

Table 14. Variations realized as a consequence of the resolutions

Resolution	Variable Elements	Type	Description
1	2	Crosscut.	The External Aspect is activated.
2	5,8	On-point	The role variant "*External Team*" is placed in the role variation point, in the corresponding activity.
3	8,10	On-point	The work product variant "Standard X" is placed in the work product variation point related to the "*Deciding Assessment Model*" activity.
4	12,14	On-point	The optative activity variant is placed in the corresponding activity variation point.

Table 15. Tailored process configured by using the MEDUSAS VRP and the context

Textual Description	*Described above.*
Fitting Rate	High.

5.4 Lessons Learned from the Proposal of Application Study

Tables 6 to 16 show an excerpt of the use of rationale in tailoring a software process. These details meet the three main objectives that rationale management supports. Firstly, the excerpt shows easy tracing from the inputs in the process tailoring (the variant rich process and the context) to the resulting tailored process. Secondly, after eliciting alternatives and comparing them, it justifies each variation as the best solution for solving each tailoring issue. Thirdly, rationale allows knowledge to be stored and permits us to learn from what the variations realize based on the context and the variant rich processes.

The application study also provides us with some other advantages we had not previously considered. Firstly, as rationale elements are built based on others, rationale offers the sequence steps with which to consistently tailor a software process. These steps really define a process tailoring activity. The sequence of steps also provides another advantage in that they are well defined and could be easily included in a tool giving support to process tailoring. This could transparently manage rationale from the user viewpoint.

In addition, since elements supporting rationale are well defined and well related, the tailoring steps may be easily recomputed in the case of input change. New variations are then obtained after any improvement or update is made in the context or the variant rich process. To sum up, we have shown how rationale management optimizes variant rich process variations, and software processes therefore meet process diversity challenges.

6 Conclusions and Future Work

Process diversity is as real in software processes as they enact and it is as important as using a capable process. Managing process diversity is therefore essential in attempting to perform good process enactments, and this therefore implies ensuring that processes meet the characteristics of their context. That implies tailoring the process by considering several context factors. Variability has typically been used for the consistent and easy tailoring of software processes, but totally exploiting variability mechanisms is not possible unless they are suitably managed. Tailoring not only implies realizing certain variations in the software process, but also needs to ensure that the variations make the process fit its context; these variations actually come up to and meet the tailoring objectives. Moreover, this knowledge needs to be used again in tailoring other processes and in improving the process tailoring activity.

Designing a process variability notation, such as vSPEM or the Variant Rich Process (VRP) paradigm, supports half of the problem of tailoring processes. But this notation is not fully usable since it does not relate tailoring causes to tailoring results, from end to end. Rationale Management can be used to solve this problem since it supports relationships between the process context and the actual variations, it allows each variation to be justified not only as a good variation, but ensures that all the variations are the best ones for tailoring the whole process. It also implies that knowledge about the tailoring is stored when it is produced, and retrieved again if it is necessary. Applying rationale in process tailoring implies executing several well-defined steps and documenting them. This extra work might initially be viewed as a disadvantage; however it forces structured software process tailoring in the same way that software processes structure software development. The advantages of applying this rationale therefore compensate for the extra work that it involves.

Rationale management has been merged with previous variability mechanisms in the VRP paradigm and notation. This also qualitatively improves the process institutionalization framework, as tailored processes are linked with the problem they actually solve. The application study shows that rationale management supports the achievement of all the advantages that were assumed, and offers other advantages. In addition, it structures and makes the whole tailoring activity consistent by dealing with improving this activity itself.

Future work is focused on validating the vSPEM language, considering that rationale is now the cornerstone of the variability mechanisms, and the most important asset in tailoring processes. This will provide us with feedback in order to improve our approach. We are carrying out some other case studies in the context of real variant rich processes tailoring and we hope to develop a tool to support the vSPEM language, as was mentioned in the section concerning the lessons learned. This tool will helps us to carry out the case studies automatically, thus reducing the effort needed in documentation and assisting during the decision making process.

Acknowledgments

This work is partially supported by the Program FPU of the Ministerio de Educación, and by the INGENIOSO (JCCM, PEII11-0025-9533), MEDUSAS (CDTI (MICINN), IDI-20090557), PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01), and ALTAMIRA (JCCM, Fondo Social Europeo, PII2I09-0106-2463) projects.

References

1. Lindvall, M., Rus, I.: Process Diversity in Software Development. IEEE Software, 14–18 (2000)
2. Siebel, N.T., Cook, S., Satpathy, M., Rodríguez, D.: Latitudinal and Longi-tudinal process diversity. Journal of Software Maintenance and Evolution: Research and Practice 9, 9–25 (2003)
3. Deck, M.: Managing Process Diversity While Improving Your Practices. IEEE Software, 21–27 (2001)
4. Filman, R.E., Elrad, T., Clarke, S., Aksit, M.: Aspect-Oriented Software Development. Addison-Wesley, Boston (2004)
5. Clements, P., Northrop, L.: Software Product Lines. Practices and Patterns. Addison-Wesley, Boston (2002)
6. Martínez-Ruiz, T., García, F., Piattini, M.: Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. In: Lee, R. (ed.) SERA. SCI, vol. 150, pp. 115–130. Springer, Praga (2008)
7. Martínez-Ruiz, T., García, F., Piattini, M.: Framework to the Standardization and Institutionalization of Software Processes. (2008)
8. Martínez-Ruiz, T., García, F., Piattini, M.: Process Institutionalization using Software Process Lines. In: Cordeiro, J., et al. (eds.) ICEIS 2009. ISAS, pp. 359–362 (2009)
9. Dutoit, A.H., McCall, R., Mistrík, I., Paech, B.: Rationale Management in Software Engineering: Concepts and Techniques. In: Dutoit, A.H., et al. (eds.) Ra-tionale Management in Software Engineering, pp. 1–45. Springer, Heidelberg (2006)
10. Knodel, J., Muthig, D.: The Role of Rationale in the Design of Product Line Architectures – A Case Study from Industry. In: Dutoit, A.H., et al. (eds.) Rationale Management in Software Engineering, pp. 297–312. Springer, Heidelberg (2006)
11. Ocampo, A., Münch, J.: Rationale Modeling for Software Process Evolution. Software Process Improvement and Practice 14, 85–105 (2008)
12. Schneider, K.: Rationale as a By-Product. Rationale Management in Software Engineering, 91–110 (2006)
13. Lee, J.: Design Rationale Systems: Understanding the Issues. IEEE Expert 12, 78–86 (1997)

14. Dutoit, A.H., mcCall, R., Mistrík, I., Paech, B.: Rationale Management in Software Engineering. Springer, Heidelberg (2006)
15. Mohan, K., Ramesh, B.: Ontology-based Support for Variability Management in Product and Service Families. In: Proc. of the HICSS 2003 (2003)
16. Kumar Thurimella, A., Wolf, T.: Issue-based Variability Modelling. In: GREW 2007, pp. 11–22 (2007)
17. Nkwoca, A., Hall, J., Raspanotti, L.: Design Rationale Capture for Process Improvement int he Globalised Enterprise: An Industrial Study. Faculty of Mathematics and Computing. The Open University, Milton Keynes (2010)
18. Sutton, S., Osterweil, L.J.: PDP: Programming a Programmable Design Process. In: 8th Int. Workshop on Software Specification and Desing, pp. 186–190 (1996)
19. Simidchieva, B.I., Clarke, L.A., Osterweil, L.J.: Representing Process Variation with a Process Family. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2007. LNCS, vol. 4470, pp. 121–133. Springer, Heidelberg (2007)
20. Silva Barreto, A., Murta, L., Rocha, A.R.: Software Process Definition: a Reuse-Based Approach. In: XXXIV Conferencia Latinoamericana de Informática, Santa Fe, Argentina, pp. 409–418 (2008)
21. Caivano, D., Visaggio, C.A.: Process Diversity and how Practitioners Can Manage It. Novatica V (2004)
22. Henninger, S.: An Environment Supporting Software Process Standardization and Innovation
23. Basili, V., Caldiera, G., Rombach, D.: The Experience Factory. In: Marciniak, J. (ed.) Encyclopedia of Software Engineering, pp. 469–476. John Wiley, Chichester (1994)
24. Torre, D., Blasco, B., Genero, M., Piattini, M.: CQA-ENV: An integrated environment for the continuous quality assessment of software artifacts. In: The 8th International Conference on Software Methodologies, Tools and Techniques (SoMeT), Praga, vol. 199, pp. 148–164 (2009)

Usage of Open Source in Commercial Software Product Development – Findings from a Focus Group Meeting

Martin Höst, Alma Oručević-Alagić, and Per Runeson

Department of Computer Science, Lund University
P.O. Box 118, SE-211 00 Lund, Sweden
{martin.host,alma.orucevic-alagic,per.runeson}@cs.lth.se
http://serg.cs.lth.se/

Abstract. Open source components can be used as one type of software component in development of commercial software. In development using this type of component, potential open source components must first be identified, then specific components must be selected, and after that selected components should maybe be adapted before they are included in the developed product. A company using open source components must also decide how they should participate in open source project from which they use software. These steps have been investigated in a focus group meeting with representatives from industry. Findings, in the form of recommendations to engineers in the field are summarized for all the mentioned phases. The findings have been compared to published literature, and no major differences or conflicting facts have been found.

Keywords: open source, industrial, off-the-shelf components.

1 Introduction

Open source software denotes software that is available with source code free of charge, according to an open source license [1]. Depending on the license type, there are possibilities to include open source components in products in the same way as other components are included. That is, in a large software development projects, open source software can be used as one type of component as an alternative to components developed in-house or components obtained from external companies.

There are companies that have experience from using well known open source projects. Munga et al. [2], for example, investigate business models for companies involved in open source development in two case studies (Red Hat and IBM) and concludes that "the key to their success was investing resources into the open source development community, while using this foundation to build stable, reliable and integrated solutions that were attractive to enterprise customers". This type of development, using open source software, is of interest for several companies. If open source components are used in product development there are

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 143–155, 2011.
© Springer-Verlag Berlin Heidelberg 2011

a number steps that the company needs to go through, and there are a number of questions that need to be solved for each step.

First potential components must be identified, which can be done in several ways. That is the company must decide how to identify components. Then, when potential components have been identified, it must be decided which component to use. In this decision there are several factors to consider, and the company must decide how to make this decision. Using the components there may be reasons to change them, which gives rise to a number of questions on how this should be done and to what extent this can be recommended. A company working with open source components must also decide to what extent to get involved in the community of an open source project.

There is some research available in this area [3], although there is still a need to collect and summarize experience from companies working in this way. In this paper, findings are presented from a workshop, in the form of a focus group meeting, where these topics were analyzed by industry representatives.

The outline of this paper is as follows. In Section 2 the methodology of the research is presented, and in Section 3 the results are presented. The results are compared to results presented in the literature in Section 4, and in Section 5 the main conclusions are presented.

2 Methodology

2.1 Focus Group

The workshop was run as a focus group meeting [4,5]. At the workshop, participants informally presented their experience from development with open source software, for example from using open source components in their product development, or from participating in open source communities. The intention was to give all participants both an insight into how others in similar situations work with these issues, and to get feedback on one's own work from other organizations. The result of a similar type of workshop was presented in [6].

Invitations to the workshop were sent to the network of the researchers. This includes earlier participants at a seminar on "research on open source in industry" where rather many (≈ 50) people attended, and mailing lists to companies in the region. This means that the participants cannot be seen as a representative sample of a population and generalizations cannot be made in traditional statistical terms. Instead analysis must be made according to a qualitative method, e.g. as described by Fink [7, p. 61-78]. This is further discussed in Section 2.4.

2.2 Objectives and Discussion Questions

The main research questions for the study were:

– How should open source components for inclusion in products be selected? Is there a need to modify selected components, and if so, how should this be done?

– To what extent is code given back to the open source community, and what are the reasons behind doing so?

Discussion questions could be derived from the objectives in different ways. One possibility would be to let the participants focus on a specific project and discuss issues based on that. The advantage of this would be that it would probably be easy for the participants to know what actually happened since it concerns a specific project. The difficulties with this approach are that there is a risk that participants have valuable experience from more than one project and therefore cannot express all experiences they have since they should focus on one specific project. There is also a risk that data becomes more sensitive if it is about a specific project. Another alternative is to ask about more general experience from the participant and let them express this in the form of advice to someone working in the area. That is, the participants use all the experience and knowledge they have, without limiting it to a specific project or presenting details about projects, customers, etc. This was the approach that was taken in this research.

Based on the objectives of workshop, the following discussion questions were phrased:

1. How should one identify components that are useful, and how should one select which component to use?
2. How should one modify the selected component and include it in ones product?
3. How should one take care of updates from the community?
4. How should one handle own modifications/changes? What are the reasons for giving back code (or not giving back code)?

In order to get a good discussion, where as many relevant aspects as possible were covered, it was monitored in the following way. For each discussion question, the participants were given some time to individually formulate an answer, or several answers, on a Post-it note. When individual answers had been formulated each participant presented their answer to the others, and the notes were posted on the wall. During the discussions, the researchers also took notes.

2.3 Analysis Procedure

The main data that was available for analysis were the notes formulated by the participants ("P-notes" below) and the notes taken by the researchers ("R-notes" below). The analysis was carried out in a number of steps, which are summarized in Figure 1 and explained below.

First all P-notes were transcribed into electronic form. In this step one note was transformed into one line of text. However, in some cases the participants wrote lists with more than one note at each piece of paper. In these cases this was clearly marked in the transcript. When interpreting the notes, the researcher were helped by the fact that the participants had presented the notes at the meeting earlier.

The R-notes were derived by dividing a longer text into single notes. After this the P-notes and the R-notes were on the same form.

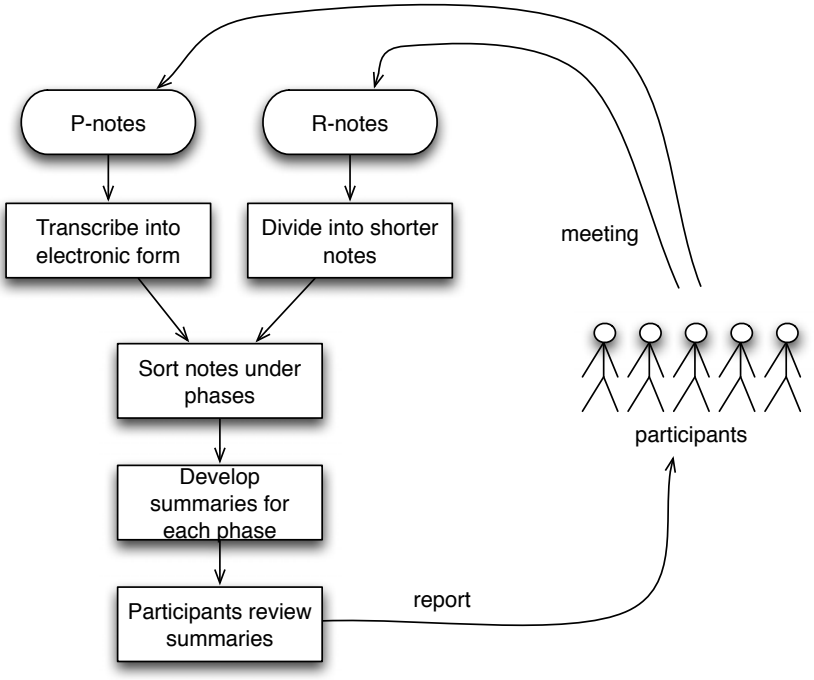


Fig. 1. Main analysis steps

After this a set of phases were defined, based on the lifecycle phases in software development. These phases were based on the areas covered by the questions, but not exactly the same. Then, all notes could be sorted under the phases in which they are relevant.

Next, all notes were grouped in related themes within phases, and based on these summaries were developed. This means that one presentation summary was developed for each phase. The final version of these summaries are presented in Section 3.

Based on this, a report was developed with the summaries. The participants were given the possibility to review and adapt the summaries in the report. This resulted only in minor changes.

This procedure results in a summary, as presented in Section 3. The results were given back to the participants in the form of a technical report. This result is also compared to the literature in Section 4 of this article.

2.4 Validity

Since the collected data is analyzed qualitatively, the validity can be analyzed in the same way as in a typical case study, which in many cases also is analyzed

qualitatively. Validity can for example be analyzed with respect to construct validity, internal validity, external validity, and reliability [4,8].

Construct validity reflects to what extent the factors that are studied really represent what the researcher have in mind and what is investigated according to the research questions.

In this study we believe that the terms (like "open source", "component", etc.) that are used are commonly used terms and that the risk of not meaning the same thing is low. It was also the case that the participants formulated much of the notes themselves, which means that they used terms that they fully understood. Besides this, the researchers participated in the whole meeting, which means that it was possible for them to obtain clarifications when it was needed. Also, the report with the same material as in Chapter 3 of this paper was reviewed by the participants.

Internal validity is of concern when causal relations are examined. In this study no causal relations are investigated.

External validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case.

The study was conducted with a limited set of participants from a limited set of organizations. This means, of course, that the results cannot automatically be generalized to other organizations. Instead it must be up to the reader to judge if it is reasonable to believe that the results are relevant also for another organization or project. The results are compared and validated to other literature and the type of results is not intended to be specific for a certain type of results.

It should also be noticed that the findings from the focus group are based on the opinions of the participants. There may be a risk that the opinions are very specific for one participant or for the organization he/she represents. The nature of a focus group meeting helps avoiding this problem. According to Robson there is a natural quality control and participants tend to provide checks and react to extreme views that they do not agree with, and group dynamics help in focusing on the most important topics [4, Box 9.5].

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers.

In order to obtain higher validity with respect to this, more than one researcher were involved in the design and the analysis of the study. Also, as mentioned above, the report with the same material as in Chapter 3 of this paper was reviewed by the participants.

Another aspect that is relevant to this is how the questions were asked and what type of data the participants are asked to provide. In order to avoid problems with confidentiality, the participants were asked to formulate answers more as advice to someone who is working in the area than as concrete experiences from specific (and named) projects. We believe that this makes it easier to provide data for this type of participants.

3 Results from Focus Group Meeting

3.1 Participants

At the workshop the following participants and organizations participated:

A. Four researchers in Software Engineering from Lund University, i.e. the authors of this paper and one more person
B. One researcher in Software Engineering from another university
C. Two persons from a company developing software and hardware for embedded systems.
D. One person from a company developing software and functionality based on an embedded system
E. One person from an organization developing software and hardware for embedded systems with more than 10 years tradition of using open source software
F. One person from an organization with the objective of supporting organizations in the region to improve in research, innovation and entrepreneurship in mobile communications

That is, in total 10 persons participated, including the authors of this paper.

3.2 Identification

Previously, companies were used to choose between making components themselves or to buying them. Now the choice is between making or buying, or using an open source component. That is, there is one more type of component to take into account in the identification process. It should also be pointed out that it is a strategic decision in terms of whether the product you are developing should be seen as a closed product with open source components or as an open source product.

When components are identified it is important that this is based on a need in the development and that it maps to the product requirements. When it comes to the criteria that are used when identifying components, they should preferably be identified in advance.

In the search process, the advice is to start with well-known components and investigate if they fulfill the requirements. There is also a lot of knowledge available among the members in the communities, so if there are engineers in the organization that are active in the community, they should be consulted. A further advice is to encourage engineers to participate in communities, in order to gain this kind of experience. However, the advice to consult engineers in the organization is not depending on that they are members of the communities. A general knowledge and awareness of existing communities is also valuable.

The next step is to search in open source forums like sourceforge and with general search engines like google. The advice here is to use technical terms for searching (algorithm, protocols, standards), instead of trying to express what you try to solve. For example, it is harder to find information on "architectural framework" than on specific techniques for this.

3.3 Selection

The more general advises concerning the selection process is to, again, use predefined criteria and recommendations from colleagues. It is also possible to conduct a basic SWOT-analysis in the analysis phase.

A more general aspect that is important to take into account is if any of the identified components can be seen as an "ad hoc standard", meaning that they are used in many products of that kind and if it will increase interoperability and the ease communication with other components. One criterion that is important in this selection concerns the legal aspects. It is necessary to understand the constraints posed by already included components and, of course, other aspects of the licenses.

Other more technical criteria that are important include programming language, code quality, security, and maintainability and quality of documentation. It is necessary to understand how much effort is required to include the component in the architecture and it is necessary to understand how the currently used tool chain fits with the component. A set of test cases is one example of an artifact that is positive if it is available in the project.

A very important factor concerns the maturity of the community. It is necessary to investigate if the community is stable and if here is a "backing organization" taking a long-term responsibility. It is also important to understand what type of participants in the community that are active. The roadmap of the open source project is important to understand in order to take a decision that is favorable for the future of the project.

3.4 Modification

First it should be emphasized that there are disadvantages of making changes to an own version of the components. The disadvantages are that the maintenance costs increase when updates to new versions of the components are made, and it is not possible to count on extensive support for specific updates from the community. So, a common recommendation is to do this only if it is really necessary.

There are some reasons why modifications must be made. Especially adaptation to specific hardware is needed, but also optimizations of different kind. When these changes are made it is in many cases favorable to give back to the community as discussed in the next section but if this is not possible an alternative is to develop "glue software" and in that way keeping the API unchanged.

If changes should be made it is necessary to invest effort in getting a deep knowledge of the source code and architecture, even if a complete set of documentation is not available.

3.5 Giving Back Code

It is, as discussed in the previous section, in many cases an advantage to commit changes to the open source project instead of working with an own forked version. In this way it is easier to include updates of the open source component. In order to manage this it is in many cases an advantage to become an active member of the community, and maybe also take a leading role in it. When modifying an

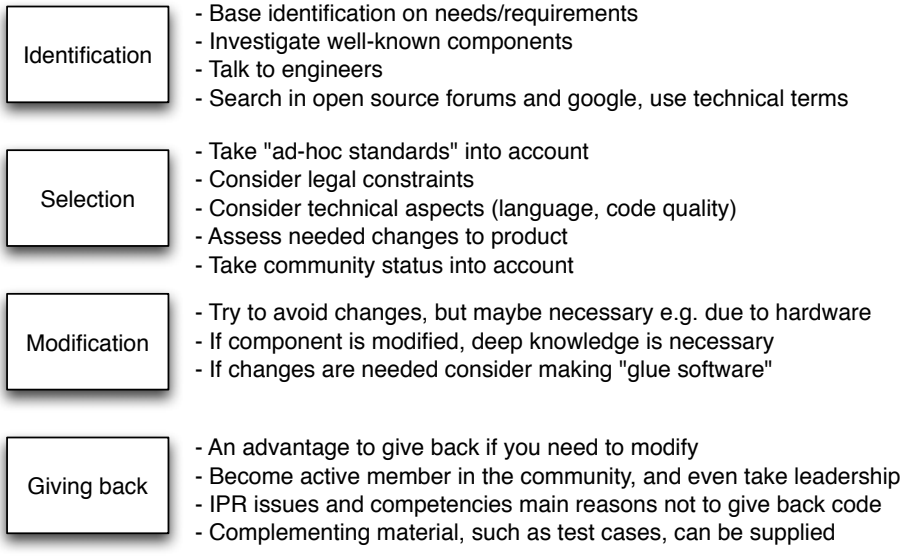


Fig. 2. Main findings from workshop

open source component it is, of course, an advantage if ones own changes can be aligned with the future development of the open source component.

However, there are some reasons not to give back changes too. The most important reason is probably that you want to protect essential IPR's and core competences in the organization. That is, key competence must in some situations be hidden from competitors. It should, however, be noticed that there may be requirements from the license to give back code. Also, after some time, all software will be seen as commodity, which means that this kind of decision must be reconsidered after a while. Another reason not to make changes public is that possible security holes can be made public. In some cases it is easier to get a change accepted if test cases are supplied.

3.6 Summary of Results

The main findings from the workshop, in terms of recommendations for the four phases, are summarized in Figure 2.

4 Comparison to Literature

The area of open source in product development has been investigated in the literature, and it is therefore possible to compare the results in the workshop to the results reported in literature.

The two first authors conducted a systematic review of open source in commercial organizations [3]. In that work it was found that presented research in the area could be divided into four main areas:

1. Company participation in open source development communities
2. Business models with open source in commercial organizations
3. Open source as part of component based software engineering
4. Using the open source process within a company

Of these different topics, the first one about company participation in open source projects, and the third one about open source as one type of components are the most relevant for this study.

4.1 Company Participation in Open Source Development Communities

There is company participation in many open source projects. For example, Bonaccorsi et al. [9] report that in a third of the most active projects on SourceForge there is or was some kind of company participation. Companies can participate as project coordinator, collaborator in code development, and by providing code etc. This can be compared to the related result of this study, where it was argued by the participants that it is important to become an active member of the community and even take leadership.

Hauge et al. [10] identify one additional role, which is more concerned with integration of open source components. The need for this role can also be confirmed in this study, since the participants acknowledge the need to use the components without changing them too much. That is, it can be seen as it is better to integrate components than to change them.

For example Hauge et al. [10] emphasize that in case software should be provided to a community it is important to provide enough documentation and information to get the community members going. This was not really discussed at the meeting, but the recommendation to get involved in the community and maybe even to take a lead in the open source project do also point at the importance of taking part in developing the community of a project.

From the data presented by Lundell et al. [11] and Robles et al. [12] it is clear that that a rather large part of the open source code has been provided by commercial organizations, and that those commercial organizations play crucial roles in open source projects. This is especially clear in the larger and more active projects. This can, of course, not be confirmed in a statistical way by a small sample like in this study, but it can be noted that the involvement in open source projects was seen as important by the participants. The amount of participation can be summarized as follows: It is suggested that a significant number of the companies marginally participate in open source community, although the participation has increased especially in SME, compared to earlier conducted studies. Of the companies that use open source projects, 75% can be said to have "symbiotic relationship" with the OS community [11]. This can be compared to the investigation presented by Robles et al. [12] that show that 6-7% of the code in Linux Debian GNU distribution over the period 1998-2004 has been contributed by corporations.

One risk that is identified by companies is that people working in the organization would reveal too much information to the outside of the organization

if they work with an open source community. However, the revealing behavior of this kind of software engineers was investigated by Henkel [13] and it was found that even if the engineers identified with the community they were significantly less identified and ideological about open source than the control group of non-commercial developers. The conclusion from that research is that there are indicators of commercially harmful behavior in this kind of development. In the focus group it was found that intellectual property is important, which points in the same direction as the results presented by Henkel [13].

4.2 Open Source as Part of Component Based Software Engineering

The report from Arhippainen [14] presents a case study conducted at Nokia on the usage of the OTS components. The report presents an analysis on usage of third party components in general, and discusses advantages of using proprietary over open source components and vice versa. The results of the presented case study focus to some extent on development of "glue software", which is also in line with the results of the focus group meeting, that is, if changes are needed do this through glue software.

Li et al. [15] present the results of a set of surveys in the form of 10 facts about development with off-the-shelf (OTS) components, of which open source components is one type. Various aspects, and more details, have also been presented in other articles by the authors. The results of this study are compared to the 10 facts below. Note that the study by Li et al. has a broader scope than this study, e.g. since it investigates all sorts of OTS and not only open source, which is one reason why all identified facts have not been investigated in this study.

1. "*Companies use traditional processes enriched with OTS-specific activities to integrate OTS components*": In this study we did not investigate the development in detail, although there was nothing in this study that argued against the fact.
2. "*Integrators select OTS components informally. They rarely use formal selection procedures*": The aspects that were discussed at the focus group meeting were rather "high level", like "take 'ad-hoc standards' into account" and taking legal aspects into account. Research-based formal processes for selection were not mention. Even if this does not mean that this kind of methods is not used, in many cases more ad-hoc methods are probably used. It is probably important to be able to take many different factors of different nature into account when selecting components.
3. "*There's no specific development process phase in which integrators select OTS components. Selecting components in early phases has both benefits and challenges*": In the study presented in this paper we did not investigate this aspect in detail. However there was nothing in the study that argued against this fact. It could also be noted that the participants could use separate phases of working with open source components, like identification, selection, modification, and giving back code, in the discussion. This means that it is possible to divide the work with open source components in different phases.

4. "*Estimators use personal experience when estimating the effort required to integrate components, and they're usually inaccurate. Stakeholder-related factors will dramatically affect estimates' accuracy.*": In the study presented in this paper we did not investigate this aspect in detail. However there was nothing in the study that argued against this fact. However, it was seen as advantage to have good knowledge of the open-source project, which of course affects the possibility to estimate effort for adaption of components.
5. "*OTS components only rarely have a negative effect on the overall system?s quality.*": Since this concerns more the result of using open source components than how to work with them, this was not discussed during the meeting.
6. "*Integrators usually use OSS components in the same way as commercial components – that is, without modification*": OSS was not compared to COTS at the meeting. However, the participants recommended not to make changes if it is not absolutely necessary. That is, the result of the meeting supports this fact.
7. "*Although problems with OTS components are rare, the cost of locating and debugging defects in OTS-based systems is substantial*": The participants pointed out the importance of involving themselves in the open source projects in order to be informed of all aspects of the project. It was not discussed, but the knowledge that is gained through this can be useful in this type of debugging.
8. "*The relationship with the OTS component provider involves much more than defect fixing during the maintenance phase*": The participants pointed out the importance of involving themselves in the open source projects, and maybe even taking the leadership of open source projects. That is, the result of the meeting clearly supports this fact.
9. "*Involving clients in OTS component decisions is rare and sometimes infeasible*": This was not discussed at the meeting.
10. "*Knowledge that goes beyond OTS components' functional features must be managed*": The participants pointed out the importance of involving themselves in the open source projects in order to be informed of all aspects of the project, not only technical aspects. That is, the result of the meeting supports this fact.

In general it can be concluded that the facts presented by Li et al. [15] are in line with this study. No data in this study pointed against the facts, and some facts, like facts 2, 8, and 9, were directly supported by this study.

5 Conclusions

We believe that many of the recommendations from the participants are important to take into account in research and in process improvement in other companies. The most important findings from the workshop are summarized below. The findings are in line with presented research in literature as described in Section 4, although the details and formulations are specific to the results of this study.

In the identification phase it is important to take the needs and the requirements into account, and to investigate well-known components. It is also advised to discuss the needs with engineers in the organization, since they can have knowledge of different components and communities. When forums are searched, an advice is to use technical terms in the search string. When selecting which components to use it is important to, besides taking technical aspects, like programming language, into account, also consider legal constraints and "ad-hoc standards". It is important to investigate the status of the community of a project, and the future of the project, which for example depends on the community. In general it can be said that changing components should be avoided if possible. If it is possible to make adaptations with "glue-code" this is in many cases better since less effort will be required in the future when components are updated by the community. However, there are situations when it is necessary to make changes in the components.

Even if there may be issues with property rights, it is in many cases an advantage to provide code to the community if changes have been made. In general it can be said that it is advised to become an active member in open source projects.

The findings from the focus group meeting were compared to published literature, and no conflicting facts were found.

Together with further research on the subject it will be possible to formulate guidelines for software project managers on how to work with open source software.

Acknowledgments

The authors would like to thank the participants for participating in the study.

This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering, (http://ease.cs.lth.se).

References

1. Feller, J., Fitzgerald, B.: Understanding Open Source Software Development. Addison-Wesley, Reading (2002)
2. Munga, N., Fogwill, T., Williams, Q.: The adoption of open source software in business models: A Red Hat and IBM case study. In: Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, pp. 112–121 (2009)
3. Höst, M., Oručević-Alagić, A.: A systematic review of research on open source software in commercial software product development. In: Proceedings of Evaluation and Assessment in Software Engineering, EASE (2010)
4. Robson, C.: Real World Reserach, 2nd edn. Blackwell Publishing, Malden (2002)
5. Kontio, J., Bragge, J., Lehtola, L.: The focus group method as an empirical tool in software engineering. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering, Springer, Heidelberg (2008)

6. Engström, E., Runeson, P.: A qualitative survey of regression testing practices. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 3–16. Springer, Heidelberg (2010)
7. Fink, A.: The Survey Handbook, 2nd edn. Sage, Thousand Oaks (2002)
8. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14, 131–164 (2009)
9. Bonaccorsi, A., Lorenzi, D., Merito, M., Rossi, C.: Business firms' engagement in community projects. empirical evidence and further developments of the research. In: First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS 2007: ICSE Workshops 2007) (2007)
10. Hauge, Ø., Sørensen, C.F., Røsdal, A.: Surveying industrial roles in open source software development. In: International Conference on Open Source Systems (OSS), pp. 259–264 (2007)
11. Lundell, B., Lings, B., Lindqvist, E.: Perceptions and uptake of open source in Swedish organizations. In: International Conference on Open Source Systems (OSS), pp. 155–163 (2006)
12. Robles, G., Dueñas, S., Gonzalez-Barahona, J.M.: Corporate involvement of libre software: Study of presence in debian code over time. In: International Conference on Open Source Systems (OSS), pp. 121–132 (2007)
13. Henkel, J.: Champions of revealing–the role of open source developers in commercial firms. Industrial & Corporate Change 18(3), 435–472 (2009)
14. Arhippainen, L.: Use and integration of third-party components in software development. Technical Report 489:84, VTT (2003)
15. Li, J., Conradi, R., Slyngstad, O.P.N., Bunse, C., Torchiano, M., Morisio, M.: Development with off-the-shelf components: 10 facts. IEEE Software (2009)

Identifying and Tackling Diversity of Management and Administration of a Handover Process

Ahmad Salman Khan and Mira Kajko-Mattsson

KTH School of Information and Communication Technology,
Forum 120, 164 40, Kista, Sweden
{askhan, mekm2}@kth.se

Abstract. Software handover is a de facto process in all software organizations. It is one of the most business critical and complex processes. It is also one of the most diverse processes, and thereby, one of the most difficult processes to define. Despite this, software handover is not well recognized within the academia. Right now, there are no software handover process models whatsoever although software organizations desperately need guidelines for how to perform this important and critical task. To aid them in defining their handover process models, we are in the process of creating *Evolution and Maintenance Management Model (EM^3): Software Handover* focusing on handover (alias transition) of a software system from developer to maintainer. In this paper, we evaluate one of the *EM^3* components, *Management and Administration* (MA), addressing activities for planning and controlling the transition process, its schedule, budget and resources. We do it within 29 organizations. Our primary goal is to find out whether the component is realistic and whether it meets the needs and requirements of the software industry today. Using the feedback from the industry, we tackle process diversity using the *Context-Driven Process Orchestration Method* (CoDPOM).

Keywords: Transition, transition plan, transition schedule, diversity.

1 Introduction

Handing over a software system from developer to maintainer is a very complex and critical process. Hence, to assure its success, it must be treated with great care and caution, and thereby, it must be properly organized and managed. This implies that one must handle and direct it with the right degree of skill, experience, and caution. If not properly managed, it may lead to a communication gap between developer and maintainer, low quality maintenance service provision and customer dissatisfaction with the system. At its worst, it may lead to a delivery failure and loss of customer credibility to both developer and maintainer. For this reason, software organizations must have a solid and well-defined software handover process model in place.

Recently, software handover has become one of the most business critical processes within many software organizations. Due to its diversity, however, it is very challenging to define and implement it. Its design strongly depends on the business context, process lifecycle, product complexity, and customer needs. For instance, its context may range from handing over a software system from developer team to a

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 156–170, 2011.
© Springer-Verlag Berlin Heidelberg 2011

maintainer team within one and the same company, to handing over from a developer team in one partner company to another maintainer team in another partner company, and finally, to outsourcing the system to a totally unknown party in the clouds [8].

Despite the great importance and business criticality, there are no software handover process models whatsoever. To the knowledge of the authors of this paper, there exist a handful of publications dealing with software handover [1, 2, 5, 6, 7]. These are either old or they deal with handover on a very general level. Neither do they suggest how to approach the diversity of a handover process that the companies meet today. Hence, they do not fulfill current needs of the software industry which today is in a strong need of a well-defined handover process model.

To aid software organizations in developing their handover models, we are in the process of creating *Evolution and Maintenance Management Model: Software Handover* (*EM*^{3}*: Software Handover*) with the objective to provide guidelines on how to successfully handover (alias transition) a software system from developer to maintainer and guidelines for how to approach its wide diversity. On our way towards this journey, as an initial step, we have come up with seven components including classes of activities that are significant for a successful software transition and put them into *EM*^{3} *Taxonomy of Handover Activities* [4]. One of its components, *Management and Administration* (MA), addresses activities for planning and controlling the transition process, its schedule, budget and resources. In this paper, we evaluate it within 29 organizations. Our primary goal is to find out whether it is realistic and whether it meets the needs and requirements of the software industry today. Using the feedback from the industry, we identify diversity indicators whose feedback we then use for visualizing how the diversity of the process may be tackled using the *Context-Driven Process Orchestration Method* (CoDPOM) [9].

The remainder of this paper is as follow. Section 2 describes our research method. Section 3 presents the *EM*^{3} *Taxonomy of Handover Activities*. Section 4 evaluates the component using industrial feedback. Finally, Section 5 identifies diversity indicators and visualizes how to tackle the process diversity with the aid of CoDPOM.

2 Research Steps

Our research method consisted of three major steps. These were (1) questionnaire design, (2) data collection and (3) data analysis. In the *questionnaire design* step, we created a very simple and semi structured questionnaire consisting of only five questions to be asked for each activity in our taxonomy. These were: (1) Is this activity performed? (2) Who does exactly perform this activity? (3) When in the lifecycle of the system is it performed? (4) How is it performed? (5) Is any activity missing in this component? By stating these questions for each activity, we wished to gain understanding of the activity and its context.

In the *data collection* step, we used students to conduct interviews. The students were attending an international master program at KTH. Just because the students came from different countries, we were aware that it might be difficult for them to get in touch with the Swedish companies. For this reason, they were free to choose their organizations and they were encouraged to select them in the countries of their origin. The organizations could be large, medium, small, private or public. The only prerequisite was they must have a transition process in place.

Finally, in the *data analysis* step, we scrutinized the collected data and searched for missing or ambiguous answers. We asked students to clarify the problems, if required. We then analyzed the collected data and drew conclusions. It is these collected data and conclusions that we are presenting in this paper.

This study includes data collected from 29 organizations within eight countries such as Sweden, Russia, Pakistan, Iran, Bangladesh, Mexico, UAE and Nepal. Out of these 29 organizations, 15 are based in Sweden. The organizations studied are small, medium and large in size with the smallest organization having only eight employees and the largest one having 400000 employees. Their business domains are diverse ranging from ERP, B2B applications, Telecom, CRM systems, SAP applications, web based applications, financial products and E-commerce applications.

All case studies encounter validity threats. Regarding external validity, our data sample consisted of 29 small, medium and large organizations working in diverse domains and located in various parts of the world. Although we cannot claim that results of our study are generalizable, still due to the diverse nature of our sample we can say with confidence that our results are in the context of the organizations studied.

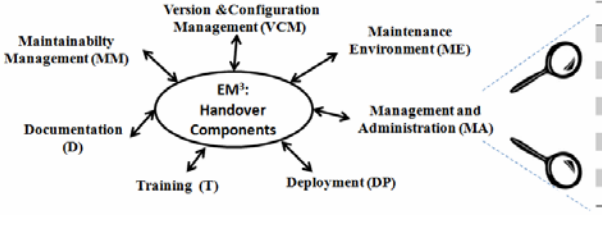
Regarding the construct validity, the risk was that the students might misinterpret the transition process and its results. To minimize this threat, we prepared students for conducting interviews. First, we gave one lecture on transition. We then presented the questionnaire and its purpose and we provided counseling hours. To ensure that all the questions were answered for each transition activity, we created templates listing each question for each activity and arranging space to be filled in with the answers. In this way we ensured the uniformity of the data and the completeness of all the answers. Finally, to enable additional validation of the answers, we requested that the students provided the contact details of their interviewees.

3 EM^3 Handover Taxonomy

In this section, we first describe all the components in *EM^3: Software Handover*. We then focus on the MA component. Due to space restrictions, we cannot fully describe *EM^3: Software Transition*. We only list its components and briefly describe them. Interested readers are however welcome to study them in [4].

3.1 EM3: Software Handover

Right now, *EM^3: Software Handover* contains types of activities required for transferring the software system from a development team to a maintenance team. As



ID	Activity title
MA1	Identify maintenance organizaiton.
MA2	Establish transition team.
MA3	Create a transition plan
MA3.1	Determine transition budget.
MA3.2	Ctreate a transition schedule.
MA3.3	Establish transition procedures.
MA3.4	Define transition resource requirements
MA3.4.1	Define maintenance manpower requirements.
MA3.4.2	Define maintenance facility requirements.
MA4	Develop management plans

Fig. 1. *EM^3* and its components

shown on the left side of Fig. 1, it has seven components where each component includes a set of highly cohesive activities having a common goal in the transition process. The components are the following:

- *Management and Administration* listing activities required for managing and administrating a transition process. It includes activities for identifying a maintenance team, establish a transition team and develop a transition plan.
- *Maintenance Environment* containing activities required for establishing a maintenance environment at the maintainer's site.
- *Version and Configuration Management* listing activities for tracking the changes made to the software system during transition.
- *Deployment* encompassing activities required for installing the system on the acquirer site.
- *Training* comprising activities for providing training on system, maintenance process, support process and training on new technology.
- *Documentation* listing activities needed for developing and transferring documents necessary for future maintenance.
- *Maintainability Management* including activities required for assessing the system and data maintainability.

3.2 Management and Administration (MA)

The *Management and Administration* (MA) component includes activities required for handling and controlling transition. As shown on the right side of Fig. 1, they concern identification of maintenance organization and transition team, creation of a transition plan and of management plans. Before describing the activities, we wish to point out that we present their role within a handover process. We do not present their sequence. These activities may or may not be performed in sequence or in parallel.

Maintenance team is responsible for evolving and maintaining a software system after transition. Its organizational membership and formation varies in different contexts. Overall, maintenance either stays with the development where it is delegated to one or several individuals or it is transferred to another team within the same organization, or it is transferred to a totally separate organization. To manage this wide spectrum of the maintenance actors, we refer to them all as maintenance teams.

It is important that the maintenance team be identified as early as possible (*Activity MA 1*). According to [6], it should be identified before contract writing. Only then the maintenance team members are prepared for taking over the new system.

All critical processes require authorities for managing and administrating them. If such an authority is not in place, then the risk is that one may fail to implement the process. This applies to all types of activities, and handover is no exception here. Hence, a handover process must be handled by a team specifically dedicated to managing and administrating it. In the context of EM^3, we call it a transition team.

To create a transition team is not trivial. Its role portfolio should include a representative set of stakeholders coming from all the organizations involved, such as developer, maintainer, acquirer and COTS suppliers. One should see to it that the stakeholders possess right capabilities, responsibilities, and experience. However, one should always keep in mind that the stakeholders may have different organizational cultures, processes, languages, working hours and workload. Therefore, a transition

team must be established in such a way so that all its members can work together in an as effective way as it is possible. In our model, we include the establishment of a transition team in *Activity MA 2* in Fig. 1.

All software engineering activities, whether complex or not, should have a plan for achieving their objectives. Transition is no exception here as well. It should have a plan worked out beforehand for the successful accomplishment of the handover from developer to maintainer. The EM^3 transition plan maps out transition budget, transition schedule, procedures for realizing the transition and for defining maintenance resource requirements (*Activity MA 3* and its sub-activities in Fig. 1).

The cost of transition varies depending on the system size, complexity and the number of the parties involved [3]. In complex cases, it may cost almost 40% of the overall development cost [3]. If the budget is not determined in advance, then the risk is that too few resources will be assigned to it, and thereby, the overall transition realization may fail. For this, reason, the determination of the budget is very important and critical for the success of a transition process (*Activity MA 3.1*).

A transition plan should have a schedule (*Activity MA 3.2*). To create it, however, is not always easy. First, scheduling in general is a very time-consuming activity. Second, it is always a challenge to balance the needs of a process and the availability of the roles involved in the process. In the context of transition, it is an extremely challenging task. This is because multiple parties participate in transition and these parties may be geographically distributed. Moreover, they are highly experienced professionals involved in many critical tasks in their respective organizations. Transition is only one of the many tasks they are engaged in. Hence, their availability, although crucial for the transition, may be significantly limited. Therefore, it is always difficult to develop a transition schedule so that all the important transition team members can participate in and contribute to the transition project in a timely manner.

A transition plan should establish transition procedures listing all the major transition activities, their sequence and communication channels (*Activity MA 3.3*). Establishing transition procedures is extremely important bearing in mind the fact that the transition team members may belong to organizations having diverse individualistic, collectivist and organizational cultures. All of them may have their own views and understanding of the transition procedures. In order not to fail, one must make sure that they have all agreed upon the common transition procedures.

A transition plan should define transition resource requirements (*Activity MA 3.4*). These requirements include (1) definition of maintenance manpower requirements (*Activity MA 3.4.1*) and (2) definition of maintenance facility requirements (*Activity MA 3.4.2*). Regarding the first requirement, a maintenance team must include a right number of personnel to handle and manage customer requests. Their number depends upon the size and complexity of the transitioned system. Regarding the second requirement, maintenance team cannot perform their duties without adequate resources. Hence, a transition team should determine the appropriate hardware and software facilities for maintaining the system such as hardware and software suites, system software baseline and support suites for maintaining the system.

Handover is not a standalone activity. It intersects with many other process areas, where each such area should be planned and managed as well. For this reason, their management plans should be in place before the implementation of a handover process starts. These plans include software configuration management plan, training

program plan, test plan, quality control plan, and many other plans that may be relevant for enabling and/or facilitating the transition process (*Activity MA 4*).

4 Component Evaluation

In this section, we report on the evaluation of the *MA* activities. When presenting each activity, we follow the order of our questionnaire.

4.1 Identify Maintenance Organization (MA 1)

All the twenty nine organizations studied identify maintenance teams or maintenance organizations. However, the identification process strongly varies depending on its timing within the software lifecycle and the roles involved in it.

In five organizations (17%), maintenance stays with the development team (Fig. 2.a). These companies have limited manpower resources and they cannot afford to have separate development and maintenance teams. Eight organizations (28%) transition their systems to separate maintenance teams within their respective companies. The complexity of their organizational structures and systems require full time maintenance teams dedicated to resolving the problems. Seven organizations studied (24%), transfer maintenance responsibilities to a separate organization.

The remaining nine organizations (31%) do not follow any specific pattern. They make their transition decisions by analyzing four parameters: (1) system size, (2) system criticality, (3) number of manpower resources available to maintain the system, and (4) outsourcing cost. Maintenance of large and critical systems is always dedicated to separate maintenance teams. Maintenance of small systems, on the other hand, is always delegated to developers. Here, the organizations do not make any distinction between development and maintenance teams. They assign maintenance responsibilities based upon development and maintenance workload.

The time point in the lifecycle and the roles involved for identifying maintenance teams vary. Regarding the roles, project manager is the main actor when identifying maintenance teams. However, as shown in Table 1, he is supported by other roles.

The time point when the organizations designate maintenance teams varies. As shown in Fig. 2.b, fifteen organizations (52%) perform this activity during the development phase, three other organizations (10%) do it during the system testing

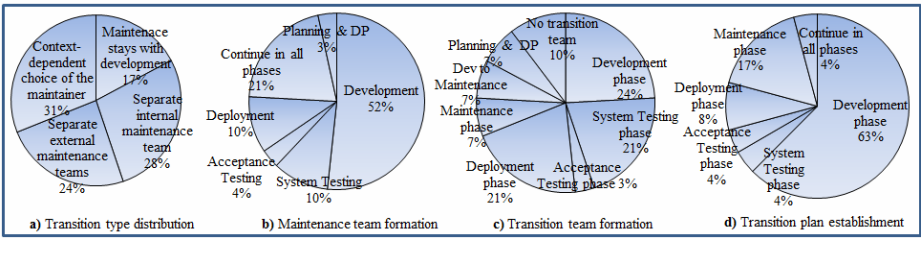


Fig. 2. Lifecycle phases for maintenance and transition teams formation. (DP= Deployment)

phase, one organization (3%) performs it in the acceptance testing phase, and three organizations (10%) designate maintenance teams during the deployment phase. Six organizations (21%) consider maintenance team formation to be a continuous activity starting during development and continuing till the maintenance phase. Finally, one organization (3%) identifies maintenance team during development, however, the final structure and members of the team are determined in the deployment phase.

Summing up, the organizations studied designate maintenance teams during development at the earliest and during deployment at the latest. All the organizations, however, preliminarily decide upon the choice of the maintenance team during the system planning phase. The choice is only preliminary because of many uncertainties involved such as system size, criticality and the number of available resources. The final choice is made later as soon most of these uncertainties are removed.

4.2 Establish Transition Team (MA 2)

All but three organizations (10%) studied establish a transition team. The three organizations do not do it because they wish to cut down project expenditure. They still, however, perform transition activities.

Ways of establishing transition teams strongly vary depending on who does maintenance. In case when maintenance stays with development (two organizations, 7%), the team consists of a project manager and a few key developers. In case when maintenance is delegated to a separate team within the organization (eight organizations, 28%), the team mainly consists of development manager, maintenance manager and project manager. Depending on the context at hand, other roles may be involved as well, such as, for instance, acquirer. However, the key role is played by the project manager who creates, manages and coordinates the team. Five organizations (17%) establish a separate maintenance team only for large projects. They establish a transition team headed by the project manager. For small projects, maintenance stays with development team and no transition team is formed.

Three organizations (10%) either delegate the maintenance responsibilities to internal maintenance teams or they outsource maintenance services to separate organizations. In the previous case, they establish an internal transition team comprising project manager, developers and maintainers. In the latter case, they establish a transition team comprising project manager, developers and acquirer. Finally, in one organization (3%) maintenance stays with development and, therefore, they do not establish any transition team. In some special cases, however, they may

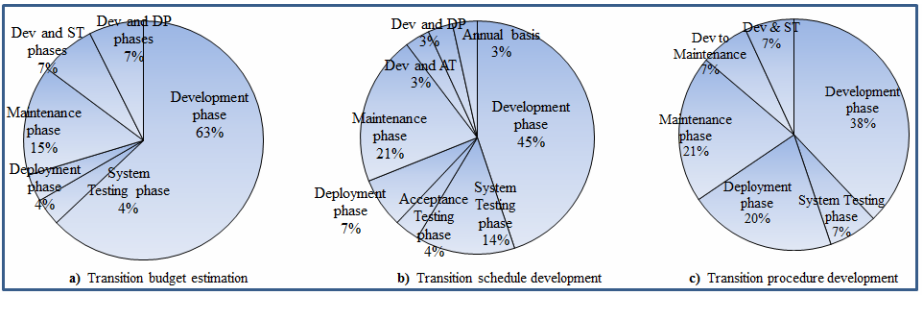


Fig 3. Lifecycle phases for transition plan activities (MA 3). Dev= Development, DP= Deployment, ST= System Testing, AT= Acceptance Testing.

feel forced by their acquirers to delegate maintenance responsibilities to other organizations and, thereby, to establish transition teams. The point in time when the organizations studied form the transition team varies. As shown in Fig. 2.c, seven organizations (26%) form a transition team during the development phase. Six organizations (22%) establish a transition team during the system testing phase, while one organization (3%) does it during the acceptance testing phase. Six organizations (22%) establish a transition team in the deployment phase and two organizations (7%) do it at the beginning of the maintenance phase. Two organizations studied (7%) establish a transition team in the project planning phase, however, they reassess its constellation during the deployment phase. They include or exclude team members based upon the system complexity and the workload of the team members. Finally, two organizations (7%) consider transition team formation as a continuous activity. They start this activity during the development phase and continue to evolve it till the start of maintenance.

4.3 Establish a Transition Plan (MA 3)

All the organizations studied develop a transition plan. Twenty three organizations (79%) develop an independent transition plan while six organizations (21%) include transition plan as part of their overall project plan.

The structure of a transition plan considerably varies from case to case. It is a simple document when the same development team continues with maintenance activities. Here, usually a project manager together with developers analyzes the completed percentage of a project and discusses the software modules to be delivered.

Transition plan is a comprehensive document in cases when transition takes place between separate development and maintenance teams and organizations. It includes: (1) deliverables including software packages and documentation, (2) schedule for handing over deliverables, (3) identification of the transition tasks, (4) identification of task owners, (5) start date, end date, priority and status of each task, (6) estimated time for completing each task, (7) sequence of tasks and their dependencies, and (8) estimated time span for the transition process.

Creation of a transition plan is not a one-off activity within the organizations studied. Only 17% of the organizations continuously review their transition plans and make appropriate modifications, if need arises. The point in time when the most changes are done is right before the deployment. This is where new conditions usually arise and force organizations to revise their transition plans.

The time point in the system lifecycle phase for establishing a transition plan varies for the organizations studied (see Fig. 2.d). Fifteen organizations (52%) establish transition plans during the development phase. One organization (3%) does it in system testing phase while another one (3%) establishes it during the acceptance testing phase. Two organizations (7%) do it in the deployment phase and four organizations (14%) develop it at the beginning of the maintenance phase. One organization (3%) starts performing this activity in the development phase and continues working on it till the maintenance phase. Four organizations (14%) develop a transition plan during the development phase but they finalize it during system testing. Finally, one organization (3%) develops or modifies it on an annual basis. Regardless of whether transition is internal or external, a project manager emerges as the main role responsible for establishing a transition plan (17 organizations, 69%).

However, as shown in Table 1, he is assisted by twelve different roles.

4.3.1 Determine Transition Budget (MA 3.1)

All but two organizations have stated that they determine transition budget. The interviewees from the two organizations were software engineers and they did not have insight into the project budget estimation process. Regarding the remaining 27 organizations (93%), they either treat transition budget as part of the overall project budget (3 organizations, 10%) or they treat it separately (24 organizations, 83%).

As shown in Table 1, the organizations studied have identified seventeen different roles participating in transition budget determination. Usually, however, project managers and finance department representatives are the major players in determining the transition budget. Development, maintenance and QA team representatives estimate the resource requirements for transition and share their estimation results with the project managers, who then calculate the overall transition budget. Their estimations are mainly based upon previous experience. Management board then finally approves the budget. In cases when the system is transitioned to another organization, the sales representatives from development organizations and product owners from customer organizations also participate in deciding on the budget.

The point in time when transition budget gets established varies within the studied organizations. As shown in Fig. 3.a, 17 organizations (63%) determine transition budget in the development phase. One organization (3%) does it during the system testing phase and another organization (4%) determines transition budget in the deployment phase. Four organizations (15%) determine transition budget at the beginning of maintenance phase. In case of a standard product development, two organizations (7%) estimate transition budget at the beginning of the project while negotiating the contract. But for the customized products, they discuss it in detail before the deployment phase. At this point, the transition budget may be renegotiated for providing extra services like training, changing features and scope of installation. Finally, two organizations (7%) determine transition budget in the development phase but they reassess and readjust it during the system testing phase.

Table 1. Roles participating in Management and Administration component

Activity	Participating roles
MA 1	Project manager, developer, maintainer, acquirer, IT manager, product manager, team lead, system architect, test lead
MA 2	Project manager, developer, maintainer, acquirer, product manager, CTO, CEO, team lead
MA 3	Project manager, developer, maintainer, acquirer, product manager, management board, support team, administrator, system architect, transition manager, operation team, finance team
MA 3.1	Project manager, developer, maintainer, acquirer, IT manager, product manager, program manager, team lead, management board, support team, CEO, administrator, operation team, finance team, quality assurance, delivery team
MA 3.2	Project manager, developer, maintainer, product manager, CTO, management board, management board, support team, CEO, administrator, operation team, VCM manager
MA 3.3	Project manager, developer, maintainer, product manager, team lead, management board, support team, system architect, transition manager, operation team
MA 3.4	Project manager, developer, maintainer, acquirer, product manager, team lead, management board, support team, administrator, system architect, transition manager, operation team
MA 3.4.1	Project manager, developer, maintainer, acquirer, product manager, team lead, management board, support team, administrator, transition manager, operation team, solution manager
MA 3.4.2	Product manager, developer, maintainer, acquirer, product manager, management board, support team, administrator, system architect, transition manager, operation team, VCM manager, test lead
MA 4	Project manager, developer, maintainer, acquirer, team lead, management board, support team, transition manager, sales team, VCM manager, customer service team

4.3.2 Create a Transition Schedule (MA 3.2)

All the organizations studied create a transition schedule. However, three of them do not develop a separate transition schedule. They treat it as part of a project schedule. The transition schedule specifies tasks with their start date, end date, description and priority. It also includes fixed dates for delivering software packages and documentation and dates for providing training.

The organizations studied have identified eleven roles participating in the creation of a transition schedule. They are shown in Table 1. However, in 69 % of the organizations, it is the project manager that develops a transition schedule. He is usually supported by experienced developers, maintainers and support personnel. He creates the schedule of the tasks to be performed and informs the concerned staff to work according to the scheduled tasks.

Transition schedule is created at the same time when the transition plan is being created by the project management. Initially, however, it only defines deadlines for the main transition activities to be completed. The schedule details are then handled by development and maintenance teams that setup a time to complete each activity. Regarding the point in time in the software lifecycle, as shown in Fig. 3.b, thirteen organizations (45%) develop transition schedule during the development phase, four organizations (14%) do it in the system testing phase, one organization (3%) does it in the acceptance testing phase, two organizations (7%) do it in the deployment phase, and finally, six organizations (21%) perform this activity at the beginning of the maintenance phase. One organization (3%) develops transition schedule in the development phase but readjusts it according to project complexity in the acceptance testing phase. Another organization (3%) develops it in the development phase as well. However, it readjusts it in the deployment phase. Finally, one organization (3%) develops transition schedule on an annual basis.

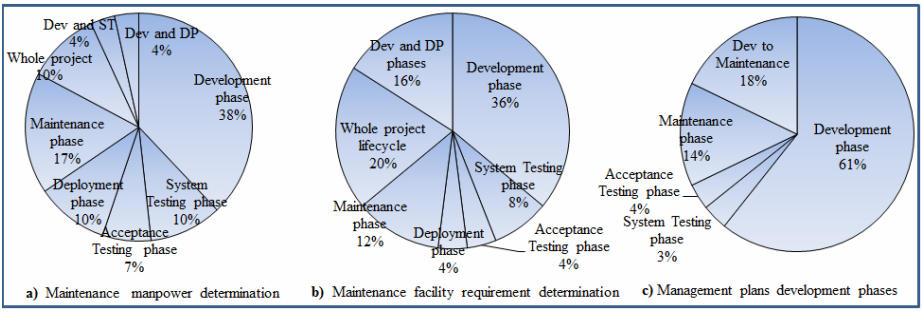


Fig. 4. Lifecycle phases for transition resource requirements and management plans. Dev= Development, DP= Deployment, ST= System Testing, AT= Acceptance Testing

4.3.3 Establish Transition Procedures (MA 3.3)

All the organizations studied have stated that they establish procedures for implementing a transition process. They either define general procedures to be then followed by all projects (65% organizations) or they define separate procedures for each project (35% organizations). Usually, the procedures concern installation, deployment, acceptance testing, contract signing, documentation, and reporting on problems

and accomplishments of the transition tasks. They are realized in form of breakdown structures for each of the major transition task.

As shown in Table 1, the organizations studied have identified ten roles participating in the establishment of transition procedures. Out of those, it is the project manager and management board who are the key players in defining the procedures. It is the project manager who creates the procedures in consultation with developers, maintainers and customer representatives and it is the management board who finally accepts them.

The point in time for establishing transition procedures varies for the organizations studied. As shown in Fig. 3.c, eleven organizations (38%) establish transition procedures in the development phase, two organizations (7%) in the system testing phase, six organizations (20%) in the deployment phase and six organizations (20%) at the beginning of the maintenance phase. Two organizations (7%) develop transition procedures in the development phase but reassess and modify them during system testing. Finally, two organizations (7%) start developing transition schedule in the development phase and continue improving it till the start of the maintenance phase.

4.3.4 Define Transition Resource Requirements (MA 3.4)

All the organizations studied define resource requirements for transition. Resource requirements are an important part of the transition plan. They vary with respect to each project. If maintenance is performed within the same organization then the only resource would be manpower. In other cases, it may include hardware and software resources. Irrespective of the case, decisions on the resource requirements are based on previous experience.

Project manager and acquirer representatives discuss and decide on the resources needed. Acquirer representative belongs to the support department on the acquirer side. The decision is then sent to management board for final approval. Support team manager chooses the right candidate for performing maintenance and support activities. The right candidate must possess technical expertise and experience of working with the transitioned system. Project manager, development and maintenance team representatives discuss and finalize the resource requirements. In case when there is need for extra manpower resource requirements on the acquirer side, an acquirer representative also participates in the discussion.

Define maintenance manpower requirements (MA 3.4.1)

All the organizations studied define manpower requirements for maintenance and estimate them. Their estimations are based on project complexity and previous experience with similar projects. For instance, maintenance personnel are selected based on their skills, expertise and knowledge about the transitioned system.

It is mainly project manager's responsibility to estimate and decide upon personnel resources. He may however do it in consultation with the representatives from development and maintenance teams. He evaluates the personnel skill and knowledge, and workload while making the final decision.

A number of maintenance team members are selected based upon the maintenance workload and priority of maintenance tasks. New maintenance team members are recruited only if the maintenance budget permits. In case when the development team continues with the maintenance, no special manpower requirements are needed. In case when the system is transitioned from development team to maintenance team,

both people from development and maintenance teams participate in the transition process. In addition, support team and QA team members are also involved. The team should include at least one person with complete system knowledge.

As shown in Fig. 4.a, eleven organizations (38%) determine maintenance manpower requirements in the development phase, three (10%) do it in the system testing phase, two (7%) in the acceptance testing phase, three (10%) in the deployment phase, and five (17%) at the beginning of the maintenance phase. Three organizations (10%) perform this activity during the whole lifecycle of the project. One organization (4%) estimates maintenance manpower requirements during the project planning phase. However, according to the complexity of the system, they reassess it during the system testing phase, if required. Finally one organization (3%) determines maintenance manpower requirements in the development and deployment phases. The organizations studied have identified twelve roles involved in defining maintenance manpower requirements. These are shown in Table 1.

Define maintenance facility requirements (MA 3.4.2)

All but four organizations studied (86%) define maintenance facility requirements. The requirements include software development tools, database servers and hardware. Regarding the four organizations (14%) that do not define maintenance facility requirements, three of them have already well-established maintenance facilities and one of them installs and maintains the system by remote online access.

The roles that are mainly involved in defining the facility requirements are project manager, system architect and configuration manager. After having created a list of requirements, the project manager gets feedback from development, maintenance and QA teams for defining additional software or hardware needs. Acquirer may also be involved in this process if maintenance and support activities are performed on the acquirer site. In such a case, project manager and acquirer representative discuss extra hardware and software resource requirements that are needed at the acquirer site.

Regarding the time point in the lifecycle, nine organizations (31%) define maintenance facility requirements in the development phase, two in the system testing phase (8%), one in the acceptance testing phase (4%), one in the deployment phase (4%) and three (12%) at the beginning of the maintenance phase. Five organizations (20%) start this activity in the development phase and continue with it till the end of the project lifecycle. Finally, four organizations (18%) define facility requirements during the development phase, but they finalize them in the deployment phase.

4.4 Develop Management Plans (MA 4)

All but two organizations studied develop management plans. These plans include project management plan, stress test plan, deployment plan, software configuration management plan, software quality program plan, software test plan, training program plan, transition plan, data migration quality plan, quality assurance plan, risk identification plan, communication plan, software test plan, and acceptance test plan. Each plan is developed by utilizing previous experience.

As shown in Table 1, the organizations studied have identified eleven different roles participating in developing and continuously revising management plans. These plans are then included in the project plan by the project manager. Project manager consults the configuration manager, development team, manager operations, data

manager and system administrator to revise the management plans. The management plans are discussed and agreed upon with the acquirer

Regarding time point in software lifecycle, 17 organizations (61%) develop management plans in the development phase (see Fig. 4.c), one organization (3%) does it in the system testing phase, one (4%) does it in the acceptance testing phase, and five organizations (17%) develop management plans at the beginning of the deployment phase. Finally, five organizations (18%) consider it as a continuous activity. They start developing management plans in the development phase and finalize them at the beginning of the maintenance phase.

5 Final Remarks

In this paper, we have evaluated the *Management and Administration* component in 29 organizations. Our results show that almost all the activities are implemented by the organizations. Those that are not implemented are usually not right for the context. Many times, their implementation depends on the complexity of the transitioned system, transition type and the like. Our results also show strong diversity in the implementation of the *MA* activities, their placement within a lifecycle and the roles involved. They have helped us to identify the following diversity indicators:

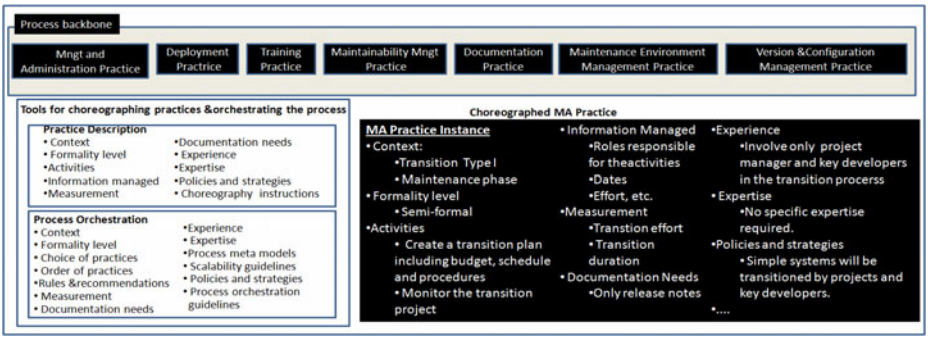


Fig. 5. CoDPOM method for EM[3]: Software Handover definition

- *Transition complexity depends on the transition type*: Transition process complexity strongly varies depending on who transfers to who. It may range from merely a handful of activities to very monumental processes.
- *Choice of maintainer depends on the software lifecycle the software system is in*: In cases when an existing software system has been evolved and maintained, the maintainer is already known in advance. Hence, it is not an issue to identify him. In case of new development, there is a need to identify the future maintainer.
- *Choice of maintainer depends on the context change*: Even if about 50% of the organizations studied identify their maintenance teams early in the development phase, they are still not certain whether the identified maintainer is the right one. They may realize later in the project that the system has become more or less complex or that they cannot afford to let maintenance stay in house, and therefore, they may have to decide to transition to a separate maintenance organization.

- *The constellation of the transition team changes with time*: Even if the transition teams are created early, they are still not fixed. The constellation of its members change as there is need for more roles to be involved in the transition with time.
- *Roles responsible for MA activities strongly vary*: Even if project manager is the key person, still the set of roles responsible for performing *MA* activities varies depending on the type of transition and system complexity and criticality.

The above-listed diversity indicators show evidence that it is not easy to create a generic handover process model. The diversity is visible in almost every aspect of the *MA* component. For this reason, when will use assistance of *Context-Driven Process Orchestration Method* (CoDPOM) when defining *EM³: Software Handover* [9]. Due to space restrictions, we cannot describe the CoDPOM method herein. Instead, we advise our reader to study [9]. Below, however, we give a flavor of how the diversity may be managed using the CoDPOM method.

As shown in Fig. 5, the overall handover process will be based on a process backbone and practices. The backbone (see the upper greyish part of Fig. 5) is a container of the core elements – practices that are choreographed for a specific context at hand. In our case, the *MA* component would constitute one practice and the other EM³ components would constitute the other practices.

The overall process design would be created using the tools for choreographing the practices and for orchestrating the process. As can be seen at the bottom left hand side of Fig. 5, the design of a process and practice is made using various process attributes whose contents is dependent on the first two attributes which are context and formality level. Depending on them, we then choose the right set of activities to be performed, the set of data and documents to be managed and created, and so on. Because this paper only focuses on the *MA* component, we cannot illustrate how the whole transition process instance will be orchestrated. However, we may illustrate how the simplest *MA* practice may be adapted to the right context at hand. Using feedback on the process diversity and practice description tool, we may now arrive at different practice instances. The black box in Fig. 5 illustrates one instance of the simplest possible practice. It is adapted to the context of an in-house transition and late maintenance phase where developer transfers the system from self to self, the formality levels of the process are semi-formal and the transition procedure covers only the minimal set of activities required for the transition. This, in turn, impacts the amount of information managed, measurements to be made and documentation to be created. Finally, in the context of an in-house transition, no external expertise is needed to manage the process and no major experience is required except for the fact that project manager and key developers must be involved in it.

References

1. April, A., Abran, A.: Software Maintenance Management, Evolution and Continuous Improvement, pp. 169–174. John Wiley & Sons, Chichester (2008)
2. International Organization for Standardization: ISO/IEC Standard 14764, Standard for Information Technology (2006)
3. Kajko-Mattsson, M., Khan, A.S., Tyrberg, T.: Evaluating a Taxonomy of Handover Activities in One Swedish Company. In: 36th EUROMICRO Conference on Software Engineering and Advanced Applications, Lille, France (SEAA 2010), pp. 342–349 (2010) ISBN: 978-0-7695-4170-9

4. Khan, A.S., Kajko-Mattsson, M.: Taxonomy of Handover Activities. In: Software Maintenance Maturity Model Workshop, S3M 2010, Limerick, Ireland, pp. 131–134 (2010)
5. MIL-HDBK-347: Military Handbook, Mission critical computer resource software support (1990)
6. Pigoski, T.: Practical Software Maintenance, pp. 117–162. John Wiley & Sons, Chichester (1997)
7. Vollman, T.: Transitioning from development to maintenance. In: Software Maintenance Conference, San Diego, CA, USA (1990) ISBN 0-8186-2091-9
8. Kajko-Mattsson, M., Gustafsson, L.: Cloud outsourcing requires a proper handover process. In: 6th International Conference on Advance Information Management and Service (IMS 2010), Seoul, Korea, pp. 142–146 (2010) ISBN: 978-1-4244-8599-4
9. Kajko-Mattsson, M.: Maturity is also about the capability to conform the process to the right context. In: Workshop on Future of Software Engineering Research (FSE/SDP), pp. 181–185. ACM, New York (2010) ISBN: 978-1-4503-0427-6

A Process Complexity-Product Quality (PCPQ) Model Based on Process Fragment with Workflow Management Tables

Masaki Obana[1], Noriko Hanakawa[2], and Hajimu Iida[1]

[1] Nara Institute science and Technology,630-0192
8916-5 Takayama-cho, Ikoma, Nara Japan
{masaki-o,iida}@is.naist.jp
[2] Hannan University,Information management,580-8502
5-4-33, Amami-Higashi, Matsubara, Oasaka Japan
hanakawa@hannan-u.ac.jp

Abstract. In software development projects, large gaps between planned development process and actual development exist. A planned process is often gradually transformed into complicated processes including a base process and many process fragments. Therefore, we propose a metric of process complexity based on process fragments. Process fragments mean additional and piecemeal processes that are added on the way of a project. The process complexity depends on three elements; the number of group of developers, the number of simultaneous process, and ratio of an executing period for a period of the whole project. The process complexity was applied to six industrial projects. As a result, changes of process complexities in the six projects were clarified. In addition, we propose a procedure of making a PCPQ (Process Complexity-Product quality) model that can predict post-release product quality on the way of a project. As a result of making a PCPQ model using the six projects, a post-release product quality was able to be predicted.

1 Introduction

In software development projects, large gaps between planned development processes and actual executed development processes exist[1]. Even if a development team has originally selected a waterfall model unplanned small processes are often triggered as shown the following examples.

i. One activity of waterfall-based process is changed into new iterative process because of urgent specification changes.
ii. Developers design GUI using a new prototype development process at design phase.
iii. In an incremental development process, developers correct defects in the previous release while developers are implementing new functions in current release.

That is, the waterfall-based process at planning time is often gradually transformed into the combination of the original waterfall-based process and several unplanned

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 171–185, 2011.
© Springer-Verlag Berlin Heidelberg 2011

small processes (hereinafter referred to as process fragments). In consequence, actual development processes are more complicated than planned one (see also Figure 1).

In this paper, we firstly assume that complicated process decreases product quality, and then propose a new metric for process complexity based on the number of unplanned process fragments, the number of simultaneous execution processes, and the number of developers' groups. It can be used to visualize how process complexity increases as actual development process proceeds.

An aim of measuring process complexity is to prevent software from becoming low quality product. A process complexity is derived from a base process and process fragments. A base process means an original process that was planned at the beginning of a project. Process fragments mean additional and piecemeal processes that are added to the base process on the way of a project. Process fragment occurs by urgent changes of customers' requirement, or sudden occurrence of debugging faults. Process fragments can be extracted from actual workflow management tables which are popular in Japanese industrial projects [2]. We especially focus on simultaneous execution of multiple processes to model the process complexity. Simultaneous execution of multiple processes is caused by adding process fragments on the way of a project. Finally, we propose a "process complexity – product quality" (PCPQ) model for predicting final product quality. We perform an industrial case study in order to show the usefulness of PCPQ model. In this case study, we found that PCPQ model was able to indicate the degree of post-release faults.

Section 2 shows related work about process metrics, and risk management. The process complexity is proposed in section 3. Section 3 also describes how the proposed complexity would be used in industrial projects. Case studies of six projects are shown in section 4. In section 5, we propose a PCPQ model to predict post-release product quality. Summary and future works are described in Section 6.

2 Related Work

Many software development process measurement techniques have been proposed. CMM [3] is a process maturity model by Humphrey. Five maturity levels of organizations have been proposed in CMM. When a maturity level is determined, various values of parameters (faults rate in total test, test density, review density) are collected. In addition, Sakamoto et al. proposed a metrics for measuring process improvement levels [4]. The metrics were applied to a project based on a waterfall process model. Theses process measurement metrics' parameters include the number of times of review execution and the number of faults in the reviewed documents. The aim of these process measurement techniques is improvement of process maturity of an organization, while our research aims to measure process complexity of a project, not organization. Especially, changes of process complexity in a project clearly are presented by our process complexity.

Many researches of process modeling techniques have been ever proposed. Cugola et al. proposed a process modeling language that describes easily additional tasks [5]. Extra tasks are easily added to a normal development process model using the modeling language. Fuggetta et al. proposed investigated problems about software development environments and tools oriented on various process models [6]. These process

modeling techniques are useful to simulate process models in order to manage projects. However, these process modeling techniques make no mention of process complexity in a project.

In a field of industrial practices, Rational Unified Process (RUP) has been proposed [7]. The RUP has evolved in integrating several practical development processes. The RUP includes Spiral process model, use-case oriented process model, and risk oriented process model. Moreover, the RUP can correspond to the latest development techniques such as agile software development, and .NET framework development. Although problems of management and scalability exist, the RUP is an efficient integrated process for practical fields[8]. The concept of the various processes integration is similar to our process fragments integration, while the RUP is a pre-planned integration processes. The concept of our process complexity is based on more flexible model considering changes of development processes during a project execution. Our process complexity focuses on changes of an original development process by process fragments, and regarded as a development processes change.

Garcia et al. evaluated maintainability and modifiability of process models using new metrics based on GQM [9]. They focus on additional task as modifiability. The focus of additional task is similar to our concept of process complexity. Although their research target is theoretical process models, and our research target is practical development processes. In this way, there is few studies for measuring complexity of process during a project. Therefore, originality of our proposed complexity may be high.

3 Process Complexity Based on Process Fragment

3.1 Process Fragment

In software development, a manager makes a plan of a single development process like a waterfall process model at the beginning of a project, because developers and managers yet not have sufficient information about the development system at planning phase of a project. However, the planned single process usually continues to change until the end of the project. For example, at the beginning of a project, a manager makes a plan based on a waterfall process model. However the original process is changed to an incremental process model because several functions' development is shifted to next version's development. Moreover, at the requirement analysis phase, prototyping process may be added to an original process in order to satisfy customers' demands. If multiple releases like an incremental process model exist, developers have to implement new functions while developers correct faults that were caused in the previous version's development. In this paper, we call the original process "a base process", we call the additional process "a process fragment". While an original process is a process that was planned at the beginning of a project, a process fragment is a process that is added to the original process on the way of a project. "Fragment" means piecemeal process. Process fragments are simultaneously executed with a base process. Process fragment does not mean simple refinement of a base process, but rather may separately executes from a base process execution.

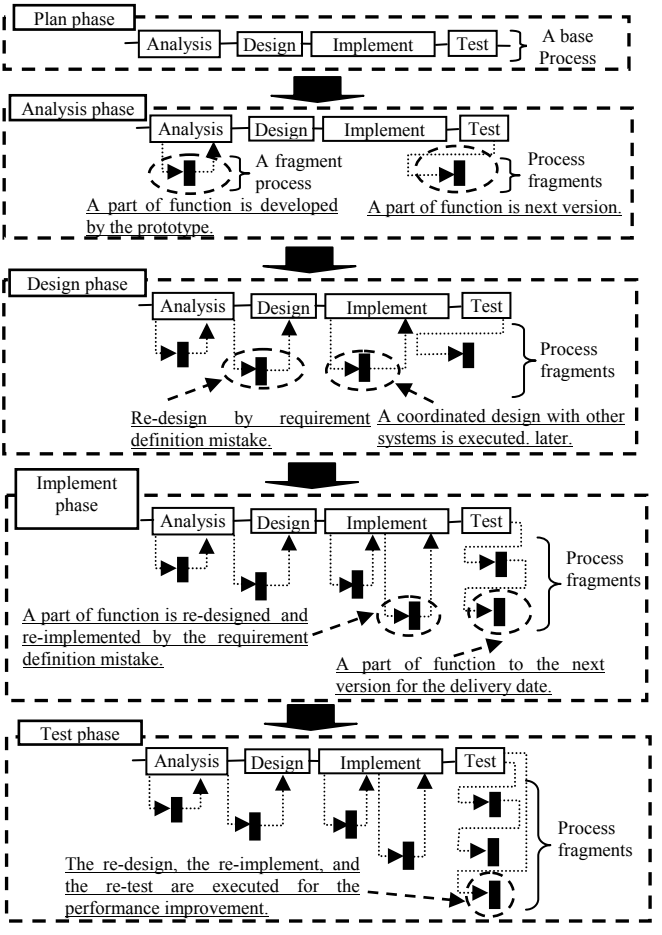


Fig. 1. A concept of process fragments

Figure1 shows an example of process fragment. At the planning phase, it is a simple development process that consists of analysis activity, design activity, implementation activity, and testing activity. In many cases, because of insufficient information, a manager often makes a rough simple process (macro process) rather than a detailed process (micro process) [10]. However, information about software increases as a project progresses, and the original simple process changes to more complicated processes. In the case of Figure 1, at the analysis phase, an unplanned prototype process was added to the original process because of customers' requests. As a result of analysis phase, implementations of some functions were shifted to next version development because of constraints of resources such as time, cost, and human. The process fragments were shown at the Figure1 as small black boxes. In the design phase, because customers detected miss-definitions of system specifications that were determined in the previous analysis phase, a process for reworking of requirement

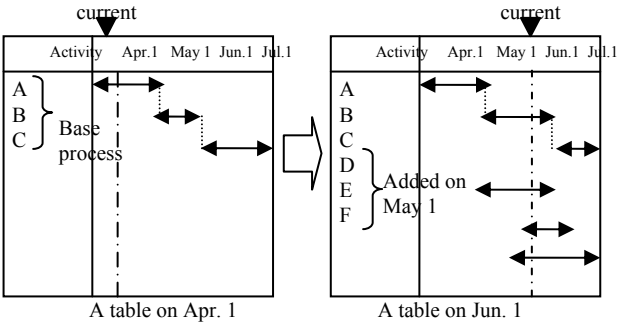


Fig. 2. Extracting process fragments from configuration of a workflow management table

analysis was added to the development process. Moreover, the manager shifted the development of a combination function with the outside system when the outside system was completed. During the implementation phase, several reworks of designs occurred. In the test phase, reworks of designs occurred because of low performance of several functions.

Process fragments are caused by urgent customers' requests, design errors, combination with outside systems on the way of development. Various sizes of process fragments exist. A small process fragment includes only an action such as document revision. A large process fragment may include more activities for example, a series of developing activities; design activity, implementation activity, and test activity.

3.2 Calculation of Process Complexity

3.2.1 Extracting Process Fragments

Process complexity is calculated based on process fragments. Process fragments are identified from a series of workflow management table. That is a continuator revised along the project. Figure 2 shows two versions of a workflow management table. Each column means a date, each row means an activity. A series of A, B, C activities is a base process. D, E, F activities are added to the base process on May 1. Therefore, D, E, F activities are process fragments. On Jun. 1, the base process and three process fragments are simultaneously executed. In this way, process fragments can be identified from configuration management of a workflow management table. Difference between current version and previous version of a workflow management table means process fragments. Of course, the proposed complexity is available in various development processes such as an agile process as long as managers manage process fragments in various management charts.

3.2.2 Definition of Process Complexity

Process complexity is defined by the two following equations.

$$PC_{(t)\_acummulate} = \sum_{i=1}^{N_{(t\_accumulate)}} (Num\_dev_{(t)i} \times L_{(t)i} \times term_{(t)i}) \quad (1)$$

$$PC_{(t)\_moment} = \sum_{i=1}^{N_{(t\_moment)}} (Num\_dev_{(t)i} \times L_{(t)i} \times term_{(t)i}) \quad (2)$$

$PC_{(t)\_accumulate}$: process complexity on time t including finished processes
$PC_{(t)\_moment}$: process complexity on just time t
$N_{(t\_accumulate)}$: the total number of process on time t including finished processes
$N_{(t\_moment)}$: the total number of process on just time t
$Num\_dev_{(t)i}$: the number of group of developers of the *i-th* process fragment on time t
$L_{(t)i}$: the number of simultaneous processes of the *i-th* process fragment on time t. But the *i-th* fragment is eliminated from these multiplications in $L_{(t)}$.
$term_{(t)i}$: ratio of an executing period of the *i-th* process fragment for the whole period of the project on time t, that is, if $term_{(t)i}$ is near 1, a executing period of the process fragment is near the whole period of the project.

Basically, we have two type process complexities. $PC_{(t)\_accumulate}$ is process complexity is accumulation of all process fragments including finished processes. $PC_{(t)\_moment}$ is a process complexity is on just time t not including finished processes. Finished process means that all tasks of the process already are completed. The reason of the two type complexities is based on different management viewpoints. If a manager wants to see the whole project characteristics, $PC_{(t)\_accumulate}$ is useful because the value of the complexity presents total accumulation of all process fragments. If a manager wants to see change of process on every day, $PC_{(t)\_moment}$ can be useful for grasping change of complexity on every day. For example, many process fragments occur at the first half of a project. Even if the process fragments have finished, the fragments' executions may harmfully influence products and process at the latter half of the project. In this management view, a manager uses a value of $PC_{(t)\_accumulate}$. On the other hand, tasks of every day change. The change of tasks of every day can be controlled by a value of $PC_{(t)\_moment}$

Theses process complexities basically depend on three elements; the number of group of the *i-th* process fragment on time t: $Num\_dev_{(t)i}$, the number of simultaneous processes of the *i-th* process fragment on time t: $L_{(t)i}$, and ratio of an executing period of the *i-th* process fragment for the whole period of project time t: $term_{(t)i}$. Granularity of group of developer($Num\_dev_{(t)i}$) depends on the scale of process fragments. If a process fragment is in detail of every hour, a group simply correspond to a person. If process fragment is large such as design phase, a group unit will be an organization such as SE group and programmer group. The granularity of group of developer will be carefully discussed in future research. The ratio of an executing period of the *i-th* process fragment for the whole period of project time t ($term_{(t)i}$) means impact scale of a process fragment. If a process fragment is very large, for example an executing period of the process fragment is almost same as the whole period of a project, the process fragment will influence largely the project. In contrast, if a process fragment is very small, for example an executing period is only one day, the process fragment will not influence a project so much.

In short, when more and larger scale process fragments are simultaneously executed, a value of process complexity becomes larger. When fewer and smaller scale process fragments simultaneously are executed, a value of process complexity becomes smaller. Values of the parameters of equation (1) and (2) are easily extracted from configuration management data of a workflow management table.

3.3 Setting a Base Process and Extracting Process Fragments

At the beginning of a project, a base process is determined. If a planned schedule is based on a typical waterfall process model such as the base process in Figure 1, the parameters' values of process complexity are $t=0$, $N_{(\square)}=1$, $Num\_dev_{(\square)}=3$ (SE group developer group, customer group), $L_{(\square)}=1$, *and* $term_{(\square)}=1$. Therefore the value of process complexity $PC_{(0)}=3$.

As a project progresses, unplanned process fragments are occasionally added to the base process at time *t1*. A manager registers the process fragments as new activities to the workflow management table. The manager also assigns developers to the new activities. Here, we assume that the planned period of a base process is 180 days. A manager adds two activities to the workflow management table. The period of each additional activity is planned as 10 days. Therefore the total number of process $N_{(t1)} = 3$, and the process complexity is calculated as follows;

(1) for $i = 1$ (a base process)

- $Num\_dev_{(t1)} = 3$
- $L_{(t1)1} = 3$
- $term_{(t1)1} = 180/180 = 1.0$

(2) for $i = 2$ (the first process fragment)

- $Num\_dev_{(t1)2} = 1$
- $L_{(t1)2} = 3$
- $term_{(t1)2} = 10/180 = 0.056$

(3) for $i = 3$ (the second process fragment)

- $Num\_dev_{(t1)3} = 1$
- $L_{(t1)3} = 3$
- $term_{(t1)3} = 10/180 = 0.056$

Finally, a value of process complexity at $t=t1$ can be calculated as $PC_{(t1)\_moment} = 9.000 + 0.168 + 0.168 = 9.336$. In this way, the value of process complexity at time t can be calculated based on workflow management table.

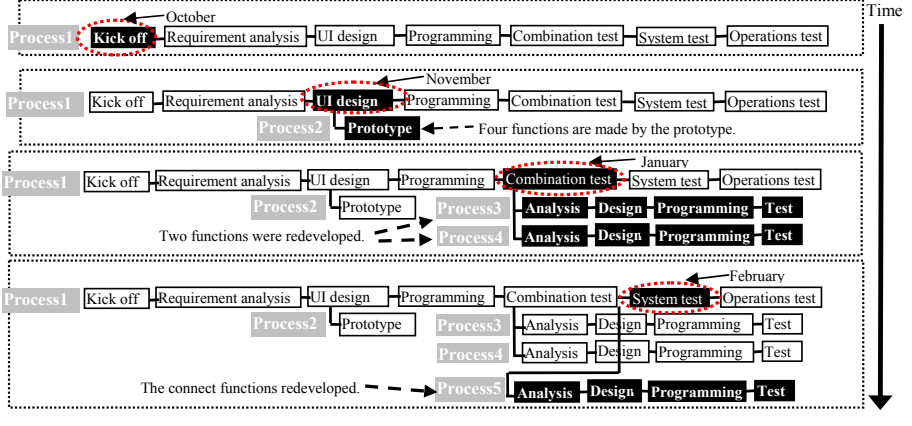


Fig. 3. A variation of development process of the HInT V2 project

4 Application to Six Industrial Projects

The process complexity has been applied to six practical projects; two versions of HInT project [13] and four versions of p-HInT project [11][12]. These projects executed by a same organization of a system development company. One of the 6 projects is presented at the following subsection.

4.1 The HInT V2 project

The version 2 of HInT project developed a web-based educational portal system. The development began from October 2007, the release of the HInT was April 2008. Because the workflow management table was updated every week, 20 versions of the workflow management table are obtained. At the beginning of the project, the number of activities in the workflow table was 20. At the end of the project, the number of activities reached to 123. Each activity had a planned schedule and a practice result of

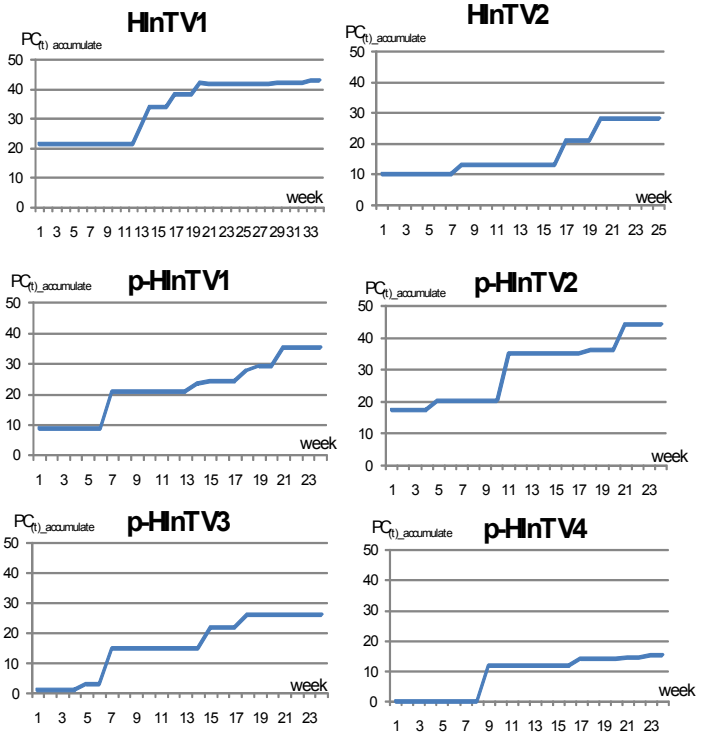


Fig. 4. Changes of values of process complexities (accumulate version) of 6 projects

execution. Figure 3 shows a rough variation of development process of the project. At the beginning of the project, the shape of the development process was completely a waterfall process. However, at the UI design phase, a prototype process was added to the waterfall process. In the prototype process, four trial versions were presented to customers. At the combination test phase, developers and customers found significant

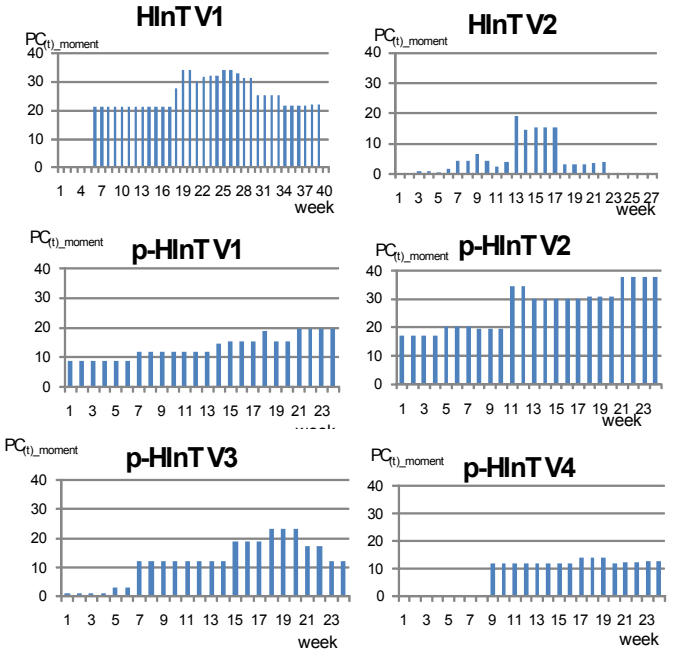


Fig. 5. Changes of values of process complexities (moment version) of 6 projects

errors of specifications and two process fragments were added to the development process in haste. Reworks such as re-design, re-implement, and re-test for the specification errors continued until the operation test phase. At the system test phase, an error of a function connecting with an outside system occurred and a new process fragment was introduced to the development process. The introduced process fragment consists of various activities such as investigating network environments, investigating specification of the outside system, and revising the programs. These activities continued until the delivery timing.

4.2 Changes of Process Complexities of the 6 Projects

Figure 4 shows the changes of process complexities (accumulate version) of the 6 projects. Figure 5 shows the changes of process complexities (moment version) of the 6 projects. The highest value of process complexity (accumulate version) is *44.4* of the p-HInT V2 project. A minimum value of process complexity (accumulate version) is *15.4* of the p-HInT V4 project. Therefore, the p-HInT V2 project included a complicated base process and many process fragments. The p-HInT V4 project consisted of a simple base process and few process fragments.

On the other hand, the changes of process complexities (moment version) in Figure 5 present different trends. Process complexities (moment version) of HInT V1 and p-HInT V2 were relatively high during the projects. That is, many fragment processes concurrently executed in HInT V1 and p-HInT V2. HInT V1 project was a new

development system. Because customers could not image the new system, customers' requests frequently changed. Therefore, developers should concurrently execute various development activities including activities for customers' new requests. In contrast, process complexities of HInT V2 keep low during the project. Customers could easily image new functions because customers mastered the original system. In p-HInT project, process complexities of p-HInT V2 were high. In the p-HInT V2, many system troubles occurred. The troubles were caused by development of p-HInT V1. Therefore, developers should concurrently execute not only development of new functions of version2 but also debugging activities of the errors. The p-HInT V2 project had complicated processes with many process fragments. After that, in p-HInT V3, the manager decided a product refactoring without developing new functions. Process complexities were relatively low in the p-HInT V3. Because the product refactoring executed smoothly in the p-HInT V3, process complexities of p-HInT V3 and p-HInT V4 did not become so high during the projects.

4.3 A Trial Tool for Visualizing Process Complexity

We have developed a trial tool for visualizing changes of process complexity (moment version) during a project. Figure 6 shows images of the changes in three projects; HInT V1, p-HInT V2, p-HInT V3. One block means one process fragment. A size of block means complexity of a process fragment. A big block is more complicate than a small block. In addition, X axis means *time*, Y axis means accumulated values of $L_{(t)}$ of equation (2) at time t, Z axis means accumulated values of $Num\_dev_{(t)}$ of equation (2) at time t. That is, if a block is long X-axially, the process fragment has long development time. If a block is long Y-axially, the process fragment has to execute concurrently with many other processes. If a block is long Z-axially, the process fragment has to execute with many development groups. In HInT V1, many fragment processes concurrently executed on the latter half of the project. Therefore, many blocks were piled, moreover, the blocks were long Y-axially. However because the blocks of HInT V1 is short X-axially, the process fragments finished quickly. In addition, concurrent executed process fragments of HInT V1 are more than one of p-HInT V2 and p-HInT V3. The piled process fragments of HInT V1 are higher than the piled process fragments of p-HInT V2 and p-HInT V3. In this way, characteristics of process complexity of each project are visualized in the tool.

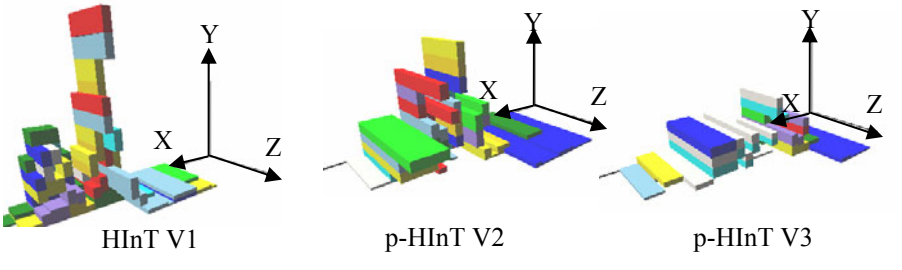


Fig. 6. Visualized images of changing process complexities

5 Process Complexity – Product Quality (PCPQ) Model

We propose a Process Complexity – Product Quality (PCPQ) model. A PCPQ model is built by pairs of a final process complexity and specific gravity of failure in several projects. A final process complexity means a value of process complexity (accumulate version) at the ending of project. A specific gravity of failure means a product quality based on importance and the number of post-release failures. A PCPQ model is built on each organization because a way of creating workflow management tables and management of post-release failure is different. For example, a manager of an organization creates a daily workflow management table. In this case, processes are divided into small process fragments. Scale of process depends on a management way of each organization. In addition, a way of management of failures is different among organizations. For example, a manager decides that a system-down error is a most important failure. Another manager decides that requirement analysis error is a most important failure. The way of deciding importance of failures is different on each organization. Therefore, a PCPQ model is built by each organization.

After building a PCPQ model, a manager can predict a post-release product quality even if the project is not finished. On the way of a project, a value of process complexity (accumulate version) can be calculated based on a scheduled workflow management table. When a manage remake a plan for process fragments, a post-release product quality can be predicted based on the PCPQ model.

5.1 A Procedure of Building a PCPQ Model

A PCPQ model is built in the following procedure;

Step1: Preparing several finished projects in a same organization.
Step2: Calculating a value of final process complexity (accumulate version) of each project from workflow management tables.
Step3: Collecting post-release failures, deciding importance of the failures.
Step4: Calculating a value of specific gravity of failure by each project.
Step5: Making an approximate value curve of the relation between the process complexities and the specific gravity of failures.

The approximate value curve of the relation between the process complexities and the specific gravity of failures means a PCPQ model. The next subsection shows an example of making a PCPQ model based on the 6 projects of section 4.

5.2 An Example of Making a PCPQ Model

Step1:We prepare six finished projects; HInT V1, HInT V2, p-HInT V1, p-HInT V2, p-HInT V3, p-HInT V4 in a same organization.

Step2:Six final process complexities (accumulate version) are calculated. Table 1 shows the values of the final process complexities.

Step3:We collected post-release failures. Each failure's importance was decided. The Table 1 also shows the number of the failures with importance ranks; SS, S, A, B, C.

In the organization, the failures are categorized into 5 ranks. SS rank means system-down level error. S rank means errors of missing important functions and performance. A rank means errors of missing middle important functions, B rank means low quality of user-interface, C rank means errors of presentation such as messages and button names. The failures were accumulated on failure management tables on each project after release. The importance rank of each failure is decided by the manager and customers.

Step4:We calculated values of specific gravity of failures. The specific gravity of the failures is a value that multiplied the number of failures and the importance rank. In the calculation, we set up that a constant of SS is 5, a constant of S is 4, a constant of A is 3, a constant of B is 2, and a constant of C is 1. For example, a value of specific gravity of HInT V1 is calculated by "5*4 + 4*4 + 3*6 +2*9 + 1*11". The value is 83. In the same way, the specific gravity of failure of HInT V2 is 66, one of p-HInT V1 is 82, and one of p-HInT V2 is 82, one of p-HInT V3 is 88, one of p-HInT V4 is37, and one of p-HInT V5 is 28.

Step5:We plotted relations between process complexities and specific gravity of failures. Figure 7 shows the relations and an approximate value curve. The approximate value curve is as a follow;

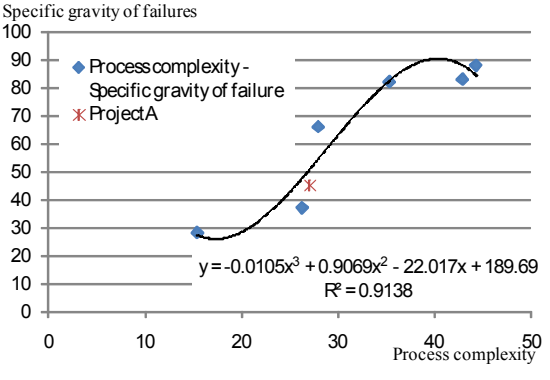


Fig. 7. An approximate value curve of relation between process complexities and specific gravities of failures

$$y = -0.0105X^3 + 0.9069X^2 - 22.017X + 189.69 \quad (3)$$

y: a value of specific gravity of failures
x: a value of final process complexity (accumulate version)

Equation (3) is a PCPQ model of the organization. Coeffient of determination (R-squared) is 0.9138. Of course, if a value of coeffient of determination is too small,

the PCPQ model is meaningless. Although the PCPQ model is based on a polynomial expression, the relation can present using linear approximation, logarithm approximation, and exponential approximation.

5.3 Predicting Product Quality Based on the PCPQ Model

Using the PCPQ model of Figure 7, product quality of another project is predicted. A target project is "Project A" developing an e-learning system in the same organization. Of course Project A is different from the six projects. Change of process complexity is shown Figure 8. At the thirteenth week, the value of process complexity is *25.21*. The value of the process complexity is applied to the PCPQ model (Equation(3)). Therefore, a value of specific gravity of failure is calculated as *42.78*. Therefore, the manager can predict post-release product quality as not better than p-HInT V3, and better than HInT V2.

The prediction of Project A is evaluated using real failure data. The number of post-release failure of Project A is *20*. The number of failure with SS and S rank is 0, the number of failure with A rank is 8, the number of failure with B rank is 9, and the number of failure with C rank is 3. The value of specific gravity of failure is *45*. The prediction value based on the PCPQ model is *42.78*, the real value is *45*. A plot "*" in Figure 7 is a relation between the process complexity and the specific gravity of failure of the Project A. We may judge that the product quality prediction of Project A at the thirteenth week is useful.

Table 1. Parameter values of a PCPQ model

Project	Final process complexity	Specific gravity of failures	Importance of failure				
			SS	S	A	B	C
HInT V1	43.0	83	4	4	6	9	11
HInT V2	28.0	66	2	1	6	10	13
p-HInT V1	35.4	82	2	3	19	1	1
p-HInT V2	44.4	88	10	6	4	1	0
p-HInT V3	26.3	37	3	3	2	1	2
p-HInT V4	15.4	28	0	0	7	3	1

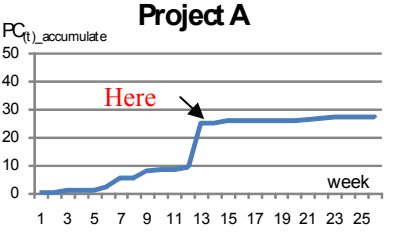


Fig. 8. Change of process complexity (accumulate version) of Project A

6 Summary

We propose a metric of process complexity based on process fragments. the process complexity has three elements; the number of group of developers, the number of simultaneous process, and ratio of an executing period for a period of the whole project. Process complexity can present changes of development processes with additional and piecemeal process fragment during a project. Process complexity is applied to six industrial projects. We could grasp how the development processes of the six projects became complicated as the projects progressed. In addition, we also proposed a way of making a PCPQ (Product Complexity-Product Quality) model. A PCPQ model is useful to predict post-release product quality on the way of a project. The PCPQ model is an approximate curve of a relation between process complexity and product quality. The model is built using process complexity and actual product quality of finished projects. A PCPQ model using the six projects' data was built. As a result, a post-release product quality was able to be predicted by the PCPQ model.

In future, we will evaluate several PCPQ models. A PCPQ models is built every organization because management ways of workflow management tables and post-release product quality are different among organizations. Therefore, we need building PCPQ models in different organizations. Then usefulness of PCPQ models will be evaluated.

References

1. Royce, W.: Software Project Management: A unified Framewaork. Addison-Wesley Professional, USA (1998)
2. Guide to the Software Engineering Body of Knowledge (SWEBOK), http://www.computer.org/portal/web/swebok
3. Humphrey Watts, S.: Managing the software process. Addison-Wesley Professional, USA (1989)
4. Sakamoto, K., Tanaka, T., Kusumoto, S., Matsumoto, K., Kikuno, T.: An Improvement of Software Process Based on Benefit Estimation (in japanese). IEICE Trans. Inf. & Syst. J83-D-I(7), 740–748 (2000)
5. Cugola, G.: Tolerating Deviations in Process Support Systems via Flexible Enactment of Process Models. IEEE Transaction of Software Engineering 24(11), 982–1001 (1998)
6. Fuggetta, A., Ghezzi, C.: State of the art and open issues in process-centered software engineering environments. Journal of Systems and Software 26(1), 53–60 (1994)
7. Kruchten, R.: The Rational Unified Process. Addison-Wesley Professional, USA (2000)
8. Manzoni, L.V., Price, R.T.: Identifying extensions required by RUP (rational unified process) to comply with CMM (capability maturity model) levels 2 and 3. IEEE Transactions on Software Engineering 29(2), 181–192 (2003)
9. Garcia, F., Ruiz, F., Piattini, M.: Definition and empirical validation of metrics for software process models. In: Proceedings of the 5th International Conference Product Focused Software Process Improvement, pp. 146–158 (2004)
10. Obana, M., Hanakawa, N., Yoshida, N., Iida, H.: Process Fragment Based Process Complexity with Workflow Management Tables. In: International Workshop on Empirical Software Engineering in Practice, pp. 7–12 (2010)

11. Hanakawa, N., Yamamoto, G., Tashiro, K., Tagami, H., Hamada, S.: p-HInT: Interactive Educational environment for improving large-scale lecture with mobile game terminals. In: Proceedings of the16th International Conference on Computers in Education, pp. 629–634 (2008)
12. Hanakawa, N., Obana, M.: Mobile game terminal based interactive education environment for large-scale lectures. In: Proceeding of the Eighth IASTED International Conference on Web-based Education, pp. 7–12 (2010)
13. Hanakawa, N., Akazawa, Y., Mori, A., Maeda, T., Inoue, T., Tsutsui, S.: A Web-based integrated education system for a seamless environment among teachers, students, and administrators. International Journal of System & Computer in Japan 37(5), 14–24 (2006)

Using Web Objects for Development Effort Estimation of Web Applications: A Replicated Study

Sergio Di Martino[1], Filomena Ferrucci[2], Carmine Gravino[2], and Federica Sarro[2]

[1] University of Napoli "Federico II", 80126, Napoli, Italy
sergio.dimartino@unina.it
[2] University of Salerno, Via Ponte Don Melillo, 84084, Fisciano (SA), Italy
{fferrucci,gravino,fsarro}@unisa.it

Abstract. The spreading of Web applications has motivated the definition of size measures suitable for such kind of software systems. Among the proposals existing in the literature, Web Objects were conceived by Reifer specifically for Web applications as an extension of Function Points. In this paper we report on an empirical analysis we performed exploiting 25 Web applications developed by an Italian software company. The results confirm the ones obtained in a previous study and extend them in several aspects, showing the robustness of the measure with respect to the size and technologies of the applications, and to the employed estimation techniques.

Keywords: Web applications, Size measure, Effort estimation technique, Empirical studies.

1 Introduction

Even if Web applications are becoming the de-facto standard in many domains, such as B2B [7], software engineering is still missing to fully support their development. Among others, there is to date the need of suitable measures to size this kind of applications and support critical management activities, such as cost/effort estimation, quality control, and productivity assessment. Indeed, size measurement methods, conceived and widely accepted for traditional software systems, such as the Function Point Analysis (FPA) [14], can fail to capture some specific features of Web applications [25]. Some measures have been defined so far (see e.g., [10][11][26]), and among them, *Web Objects* were introduced by Reifer [26] by adding four new Web-related components (namely Multimedia Files, Web Building Blocks, Scripts, and Links) to the five function types of the FPA method. In his original formulation [26][27], Reifer reported improved prediction performances of Web Objects over Function Points, but no details were provided about the empirical analysis he performed. In [29][30] Ruhe *et al.* described two studies assessing the effectiveness of Web Objects for estimating Web application development effort, by exploiting a dataset of 12 industrial Web applications. In the first analysis [30], they applied Ordinary Least Squares Regression (OLSR) [24], a widely used estimation technique, while in the second analysis [29] they employed Web-COBRA that is an extension for

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 186–201, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Web applications of the COBRA[1] method proposed by Briand *et al.* [4]. Web-COBRA can be considered a composite method, according to a widely accepted taxonomy [3], since it exploits expert's opinions, gathered in a controlled fashion, together with other cost drivers, within an algorithmic approach. To assess the obtained estimations, authors applied a leave-1-out cross validation, highlighting that Web Objects performed well and better than Function Points. It is obvious that, as the authors themselves pointed out, there is the need of replicated studies to further assess the measure with different (and possibly) larger datasets and to generalize the results in different contexts [2]. To this aim, in this paper we report on a replication of Ruhe *et al.*'s studies [29][30] performed by exploiting data on 25 Web applications developed by an Italian software company. This analysis also extends Ruhe *et al.*'s studies, since, in addition to OLSR and Web-COBRA, we applied an Artificial Intelligence prediction method, namely Case-Based Reasoning (CBR) [1], and exploited a different validation method. In particular, we performed a *hold-out* cross by using 15 applications as training set and 10 further applications as test set. We applied this validation since it is considered theoretically the best option in specific cases, e.g., when using projects started after a certain date as hold-out, as in our case [16]. Moreover the split reflects the real outcoming of the software company development process since the observations included in the test set were developed after the ones included in the training set. Moreover, the 25 Web applications used in our study are more recent, thus exploiting newer technologies, development environments, etc.., and are much bigger than those used in [29].

The remainder of the paper is organized as follows. In Section 2 we report on the experimental method we exploited to establish whether Web Objects can be used to predict the development effort of Web applications. The results of the empirical analysis are reported and discussed in Section 3, while a discussion about the empirical study validity is presented in Section 4. Section 5 reports on the related work while Section 6 concludes the paper giving some final remarks.

2 Experimental Method

This section presents the design of the empirical study carried out to assess the effectiveness of Web Objects for sizing Web applications[2]. The research question we addressed is:

[RQ1] Can the Web Objects measure provide good estimations of Web applications development effort when used in combination with OLSR / CBR / Web-COBRA?

It is worth noting that our experimental settings allowed us to gain insight on two consequent research questions:

[RQ2] Are the estimates obtained using Web Objects statistically superior to the estimates obtained using Function Points?

---

[1] COBRA is a trademark of the Fraunhofer Institute - http://www.fraunhofer.de/

[2] Details on the design of the case study can be find in the technical report available at: http://www.dmi.unisa.it/people/gravino/www/work/Report_WO_Gravino2011-01-18/TechReport_WO_Gravino2011-01-18.pdf

[RQ3] Which estimation method, among OLSR, CBR, and Web-COBRA, provides the best predictions, when used in combination with Web Objects?

2.1 The Dataset

Data for our empirical study were provided by an Italian medium-sized software company, whose core business is the development of enterprise information systems, mainly for local and central government. Among its clients, there are health organizations, research centers, industries, and other public institutions. The company is specialized in the design, development, and management of solutions for Web portals, enterprise intranet/extranet applications (such as Content Management Systems, e-commerce, work-flow managers, etc.), and Geographical Information Systems. It has about fifty employees, it is certified ISO 9001:2000, and it is also a certified partner of Microsoft, Oracle, and ESRI.

The company first provided us data on 15 projects, and then data on further 10 applications. These two sets include e-government, e-banking, Web portals, and Intranet applications, developed between 2003 and 2008, and are quite homogeneous in terms of adopted technologies and development teams. In particular, all the projects were developed by exploiting SUN J2EE or Microsoft .NET technologies. Oracle has been the most commonly adopted DBMS, but also SQL Server, Access, and MySQL were employed in some of these projects.

Table 1 reports some summary statistics on these 25 projects, aggregated on the two datasets. The variables employed in our empirical analysis are *EFH*, i.e. the actual effort, expressed in terms of person/hours, *WO*, expressed in terms of number of Web Objects, and *FP*, expressed in terms of number of Function Points. Further details on how these data were collected are discussed in Section 4.

Table 1. Descriptive statistics of EFH, WO, and FP for the study

Dataset	Var	Min	Max	Mean	Median	Std. Dev.
I (15 observations)	EFH	1176	3712	2677.867	2792	827.115
	WO	465	2258	1464.867	1389	543.986
	FP	110	601	360.200	355	167.009
II (10 observations)	EFH	782	4537	2511.778	2498	1265.208
	WO	323	3078	1503.000	1271	960.927
	FP	175	973	459.600	327.5	273.612

2.2 The Web Objects Method

The Web Objects method was proposed by Reifer to measure the size of Web applications [26]. In particular, Reifer added four new Web-related components, namely Multimedia Files, Web Building Blocks, Scripts, and Links, to the five predictors of the FPA method. A detailed description of these components can be found in the Reifer "*white paper*" explaining the counting conventions of the Web Objects method [27].

To size a Web application accordingly to the method, a Measurer has to compute the Function Points in the traditional way. Then he/she has to identify the Web-related components that have not yet counted. Similarly to FPA, the further step is to

determine the complexity of the identified instances of the nine components. To support this task, Reifer provided a calculation worksheet in [65] that was subsequently modified by Ruhe [28]. We used this latter version, since we were interested in replicating Ruhe's studies. Thus, the application size in terms of Web Objects is obtained by adding the identified component instances taking into account the weights that are related to each component.

2.3 The Employed Effort Estimation Methods

Several techniques have been proposed in the literature to be employed for effort estimation [3]. In our empirical analysis, we applied OLSR [24][23] and Web-COBRA [29], since they have been applied in previous studies to assess the effectiveness of Web Objects in estimating Web application development effort [29][30]. Furthermore, we also employed CBR [1] since, together with OLSR, it is one of the most diffuse techniques in the industrial context and in several researches to estimate Web application development effort (see e.g., [10][11][19][21]).

OLSR. It is a statistical technique that explores the relationship between a dependent variable and one or more independent variables [24][23], providing a prediction model described by an equation

$$y = b_1x_1 + b_2x_2 + ... + b_nx_n + c \quad (1)$$

where y is the dependent variable, x_1, x_2, ..., x_n are the independent variables, b_i is the coefficient that represents the amount the variable y changes when the variables x_i changes 1 unit, and c is the intercept. In our empirical study we exploited OLSR to obtain a linear regression model that use the variable representing the effort as dependent (namely EFH) and the variable denoting the employed size measure (namely WO) as independent. Once the prediction model is constructed, the effort estimation for a new Web application is obtained by sizing the application in terms of Web Obejcts, and using this value in the obtained model.

Web-COBRA. It is an adaptation of COBRA, proposed to estimate the development effort of Web applications, taking into account "the needs of a typical Web application company" [29]. In the following we describe only the key aspects of this method; the interested reader can consult [28][29] for further details.

To apply Web-COBRA, two key aspects have to be setup for a specific environment:

1. The set of external factors that can lead to a rise of the cost for an application within the specific domain. These factors are modeled by introducing the concept of cost overhead, defined as "the additional percentage on top of the cost of an application running under optimal conditions" [4].
2. The relationship between cost overhead and effort.

The first aspect is captured by a *causal model*, i.e. a list of all the cost factors (and their relationships) that may affect a development cost within a specific domain. This conceptual, qualitative model is obtained through the acquisition of experts' knowledge. Then, the experts are asked to "quantify" the effect of each of these identified factors on the development effort, by specifying the percentage of overhead above an "optimal" application that each factor may induce. Since different experts may

provide different estimations of these percentages, basing on their previous experience, these factors are modeled as uncertain variables requiring a minimal, most likely, and maximal values. For example, experts may agree that the factor "safety of the Web application" may affect the development effort ranging from 10% (minimal), through 50% (most likely), to 80% (maximal). Then, a triangular distribution of these cost overheads is calculated. It is worth noting that the range of the distribution provides an indication on how uncertain the experts are about the overhead inducted by the specific cost factor [29].

As for the second step, the relationship between the cost overhead and the development effort is modeled by using the OLSR and employing past data of the company. The causal model and the determined relationship between effort and cost overhead are used to obtain the effort estimations for new applications. In this step a Monte Carlo simulation can be run to provide a distribution from where an estimate of the effort can be obtained by taking the mean of the distribution [29].

CBR. It is an Artificial Intelligence technique that allows us to predict the effort of a new Web application (target case) by considering some similar applications previously developed (case base) [1]. In particular once the applications are described in terms of some features (such as the size), the similarity between the target case and the others in the case base is measured, and the most similar ones are selected, possibly with adaptations, to obtain the estimation. To apply the method, a Measurer has to choose an appropriate similarity function, the number of analogies to select the projects to consider for the estimation, and the analogy adaptation strategy for generating the estimation. Some supporting tools can help doing these tasks.

2.4 Validation Method and Evaluation Criteria

In order to validate the obtained effort estimation models we performed a hold-out cross validation approach [16], employing datasets I and II of Table 1. Dataset I (training set) was used to train the effort estimation techniques while dataset II (test set) was used to validate the obtained models.

To assess the derived estimations, we used some summary measures, namely MMRE, MdMRE, and Pred(25) [8]. In the following, we briefly recall their main underlying concepts.

The *Magnitude of Relative Error* (MRE) [8] is defined as

$$MRE = |EFHreal - EFHpredl/EFHreal \quad (2)$$

where EFHreal and EFHpred are the actual and the predicted efforts, respectively. MRE has to be calculated for each observation in the test set. Then, the MRE values have to be aggregated across all the observations. We used the mean and the median, giving rise to the *Mean of MRE* (MMRE), and *Median of MRE* (MdMRE), where the latter is less sensitive to extreme values [20]. According to [8], a good effort prediction model should have a MMRE$\leq$0.25, to denote that the mean estimation error should be less than 25%.

The *Prediction at level l*%, also known as Pred(*l*), is another useful indicator that measures the percentage of estimates whose error is less than *l*%, where *l* is usually set at 25% [8]. It can be defined as

$$Pred(25) = k/N \quad (3)$$

where k is the number of observations whose MRE is less than or equal to 0.25, and N is the total number of observations. Again, according to [8], a good prediction approach should present a Pred(25) $\geq$ 0.75, meaning that at least 75% of the predicted values should fall within 25% of their actual values.

Moreover, we tested the statistical significance of the obtained results, by using absolute residuals, in order to establish if one of employed estimation measures provides significantly better results than the other [18][20]. In particular, we performed statistical tests (i.e., T-Test or Wilcoxon signed rank test when the distributions were not normally distributed) to verify the following null hypothesis: "the two considered populations have identical distributions". This kind of test is used to verify the hypothesis that the mean of the differences in the pairs is zero.

To have also an indication of the practical/managerial significance of the results we verified the effect size [15]. Effect size is a simple way of quantifying the difference between two groups. Employing the Wilcoxon test and the T-test, the effect sizes is determined by using the formula: r = Z-score/sqrt(N), where N is the number of observations. In particular, we first calculated the effect size and then compared it to the Cohen's benchmarks [6]: so r=0.20 indicates a small effect, r=0.50 indicates medium effect, and r=0.80 indicates a large effect.

Finally, as suggested in [21], we also analyzed the values of the summary statistics MMRE, MdMRE, and Pred(25) obtained by employing the mean effort (MeanEFH) and the median effort (MedianEFH) of the training set as estimated effort. Indeed, if the prediction accuracy obtained with complex measures/techniques is comparable with those got with the mean or median effort, then a software company could simply use the mean or the median effort of its past applications rather than dealing with complex computations of software sizes, such as Web Objects, to predict development effort.

3 Empirical Results

The following subsections present the results of the empirical analysis we carried out to establish whether the Web Objects measure is a good indicator of Web application development effort, when used in combination with OLSR, Web-COBRA, or CBR. As benchmark, we compared the predictions with those obtained with traditional Function Points.

3.1 Obtaining Estimates with OLSR

We performed the OLSR analysis to build the effort estimation model by using the training set of 15 Web applications (i.e., dataset I of Table 1). We applied OLSR two times: as independent variable we used in the first one WO, while in the second run FP. In both the cases, we preliminarily carried out an outlier's examination to remove potential extreme values which may influence the models, and then we verified the assumptions underlying the OLSR analysis. Table 2 shows the results of the OLSR applied with WO, in terms of R^2 (an indication of the goodness of the model), F-value and the corresponding p-value (denoted by Sign. F), whose high and low values,

respectively, denote a high degree of confidence for the estimation. Moreover, we performed a t statistic and determined the p-value and the t-value of the coefficient and the intercept for the obtained prediction model, to evaluate its statistical significance. A p-value lower than 0.05 indicates we can reject the null hypothesis that the variable is not significant in the considered model, with a confidence of 95%. As for the t-value, a variable is significant if the corresponding t-value is greater than 1.5. As we can see from Table 2, both the criteria are matched.

Table 2. The results of the OLSR analysis with WO

	Value	**Std. Err**	**t-value**	**p-value**
Coefficient	1.246	0.241	5.162	0.000
Intercept	851.912	375.814	2.267	0.041
R^2	**Adjusted R^2**	**Std. Err**	**F**	**Sign. F**
0.672	0.647	491.513	26.645	0.000

The results of the application of the OLSR with FP are reported in Table 3. Even if the coefficient and the intercept can be considered accurate and significant as from the t statistic, the R^2 and F values are lower than those obtained with WO, pointing out a weaker correlation between FP and EFH.

To understand the effectiveness of these models in predicting the development effort, their accuracy has been evaluated on a test set of 10 Web applications (i.e., dataset II of Table 1). The results are reported in Table 4.

Table 3. The results of the OLSR analysis with FP

	Value	**Std. Err**	**t-value**	**p-value**
Coefficient	3.853	0.863	4.464	0.001
Intercept	1290.121	340.651	3.787	0.002
R^2	**Adjusted R^2**	**Std. Err**	**F**	**Sign. F**
0.605	0.575	539.3	19.93	0.001

Based on the commonly accepted thresholds provided in [8], even if the value of Pred(25) is slightly less than 0.75, we can conclude that WO is a good indicator of Web application development effort, when used in combination with OLSR. Furthermore, we can note that the estimates obtained using WO are much better than those obtained with FP, with about half the mean and median error. Also the T-test confirmed the superiority of WO, highlighting that their estimations are significantly better than those obtained with FP (p-value=0.008). Finally we computed the effect size, whose analysis revealed a medium effect size (r=0.54), according to the widely used Cohen's benchmarks [6].

Table 4. The results of OLSR

	MMRE	**MdMRE**	**Pred(25)**
OLSR with WO	0.21	0.15	0.70
OLSR with FP	0.46	0.28	0.40

3.2 Obtaining Estimates with Web-COBRA

To apply the Web-COBRA method, the following steps were conducted:

1) Identification and quantification of cost factors.
2) Data collection for the Web applications involved in the case study.

As for 1) it is worth noting that a large number of cost drivers may affect the development cost of software applications. However, for each domain, only a subset of these factors turns out to be relevant [4][29]. We drafted an initial list including the cost factors identified in [28][29] that was submitted to five experts of the software company involved in our empirical study. Then a Delphi method [18] was adopted until they agreed on the final set of cost drivers. They were asked to comment, basing on their experience, on the clarity of the factors (to avoid that different project managers could interpret them in different ways), on their completeness (to avoid that some key factors might not be considered), and on relevance for the Web application development domain, working also to reduce as much as possible redundancies and overlaps. A final list of 10 cost drivers was devised. They are reported in Table 5. It is worth noting that this list includes four cost factors employed by Ruhe *et al.* in [28][29]: Novelty of Requirements, Importance of Software Reliability, Novelty of Technology, and Developer's Technical Capabilities[3]. Then, the experts were asked to quantify the cost factors, specifying their minimal, most likely, and maximal inducted overhead (see Table 5). Again, a Delphi method was used to obtain a single representative triple for each cost factor. Subsequently, for each Web application p, the corresponding project manager specified the influence of the cost factors on p by a value in the range 0..3, where 0 means that no influence was due to that factor, and 3 represents the highest impact. Thus, the information on the cost overhead for each project p was obtained by the sum of all the triangular distributions of cost factors specified for p, taking into account their minimal, most likely, and maximal values of Table 6.

Table 5. Identified cost factors and their influence

Cost Factor	Minimal	Most Likely	Maximal
Novelty of Requirements (CF1)	10%	35%	70%
Importance of Software Portability (CF2)	7%	25%	60%
Importance of Software Reliability (CF3)	5%	20%	60%
Importance of Software Usability (CF4)	7%	30%	65%
Importance of Software Efficiency and Performance (CF5)	7%	20%	50%
Novelty of Technologies (CF6)	5%	25%	65%
Integration/Interaction with legacy systems (CF7)	20%	35%	70%
Temporal Overlap with other projects (CF8)	10%	35%	60%
Productivity of the adopted technological platform (CF9)	15%	45%	65%
Developer's Technical Capabilities (CF10)	10%	35%	65%

[3] Importance of Software Reliability was not included in the final list selected by the project managers in the experiment presented in [29].

The information on Effort (namely EFH), Size (expressed in terms of WO or FP), and *co_overhead* obtained from cost factors was exploited to build a model and validate it. Observe that Web-COBRA assumes that the relationship between effort and size is linear [29]. We have performed the required statistical tests to verify this linearity in our dataset. The obtained equation is:

$$Effort = 0.477 \cdot WO^{*}co\_overhead + 1095.89 \quad (4)$$

Moreover, the size of a Web application is modeled as an uncertain variable, which underlies a triangular distribution and an uncertainty of 5% was considered in [29]. Then, we applied a hold-out cross validation, by employing the training and the test sets in Table I. Moreover, we run a Monte Carlo simulation (considering 1000 iterations) that allowed us to use the relationship between cost overhead and effort together with the causal model to obtain a probability distribution of the effort for the new project [29]. Then, the mean value of the distribution was used as the estimated effort value. Table 6 shows the results of the validation we obtained in terms of MMRE, MdMRE, and Pred(25), by applying Web-COBRA in combination with WO and FP (this latter analysis was not performed by Ruhe *et al.* in [29]).

Again we got a superiority of WO, whose predictions fit the acceptable threshold defined in [8]. This does not hold for FP. Also statistical tests highlight that the estimates obtained with WO are significantly better than those obtained with FP (p-value=0.003) with a medium effect size (r=0.71).

Table 6. The results of Web-COBRA

	MMRE	**MdMRE**	**Pred(25)**
Web-COBRA with WO	0.18	0.12	0.80
Web-COBRA with FP	0.29	0.25	0.50

3.3 Obtaining Estimates with CBR

To apply CBR, in our empirical study we exploited the tool *ANGEL* [31] [30]. It implements the Euclidean distance as similarity function, using variables normalized between 0 and 1, and allows users to choose the relevant features, the number of analogies, and the analogy adaptation technique for generating the estimations. Since we dealt with a not so large dataset, we used 1, 2, and 3 analogies, as suggested in many similar works [20]. To obtain the estimation, once the most similar cases were determined (exploiting information on the size, i.e., Web Objects), we employed three widely adopted adaptation strategies: the mean of k analogies (simple average), the inverse distance weighted mean [20], and the inverse rank weighted mean [31]. So, we obtained 10 estimations and the corresponding residuals, for each selection of the number of analogies and of the analogy adaptation techniques. Since we carried out a hold-out cross validation, each estimation was obtained by selecting a target observation from the test dataset, and by considering as case base the observations in the training dataset. Table 7 shows the best results in terms of MMRE, MdMRE, and Pred(25), for both WO and FP. These results are the best we got, being obtained by employing 2 analogies and the mean of k analogies as adaptation strategy.

Table 7. The results of CBR using ANGEL

	MMRE	MdMRE	Pred(25)
CBR with WO	0.22	0.12	0.70
CBR with FP	0.49	0.17	0.60

As for OLSR and Web-COBRA, WO outperformed FP also with CBR. In particular, the MMRE and MdMRE values satisfy the usual acceptance thresholds of [8], while Pred(25) value is slightly less than 0.75. In contrast with the results achieved with OLSR and Web-COBRA, the statistical tests revealed that the estimates with WO are not significantly superior to those obtained with FP (p-value = 0.072), with a small effect size (r=0.42).

4 Discussion and Comparison

In this section we discuss the results we have gathered and compare them with those achieved by Ruhe *et al.* in [29][30].

The MMRE, MdMRE, and Pred(25) values reported in Table 4, Table 6, and Table 7 suggest that the Web Objects measure is a good indicator of Web application size, when used in combination with the prediction techniques we considered. These results also highlight that Web Objects outperforms Function Points in terms of prediction accuracy. This is a confirmation to an expected result, since the Web Objects method was conceived to overcome the limitations of FPA when dealing with Web applications. Moreover, we can observe that Web-COBRA provided slightly better results than OLSR and CBR, in terms of MMRE, MdMRE, and Pred(25).

The above results corroborate what suggested by the common sense: Web-COBRA, taking into account also many non-functional aspects of the software process and product, provides improved estimations than the two other techniques relying only on the Web Objects size measure. On the other hand, it is very interesting to point out that Web-COBRA applied with FP provided worse results than OLSR with WO. This means that the four new components sized by the Web Objects method are much more correlated to the effort than the non-functional factors handled by Web-COBRA. This is also confirmed by the fact that there is no statistically significant difference between the three techniques.

As designed, we compared the predictions with those obtained by the simple mean or median of the effort of the whole training set. These predictions are very poor, as reported in Table 8, since they do not satisfy the typical acceptance thresholds [8]. Moreover, predictions obtained with WO and FP based models are significantly better than those obtained using MeanEFH and MedianEFH.

Table 8. The results of MeanEFH and MedianEFH

	MMRE	MdMRE	Pred(25)
MeanEFH	0.63	0.37	0.40
MedianEFH	0.68	0.34	0.40

Summarizing, regarding the research questions RQ1, RQ2, and RQ3, the results of the performed empirical analysis suggest that:

[RQ1] The Web Objects measure resulted to be a good indicator of Web application development effort, when used in combination with OLSR, CBR, and Web-COBRA, since the values of summary measures are very close or match the thresholds usually adopted in this domain [8].

[RQ2] The estimates obtained with Web Objects turned out to be statistically superior to the ones achieved with Function Points in combination with OLSR and Web-COBRA.

[RQ3] Even if Web-COBRA provided slightly better results than OLSR and CBR in terms of summary measures there is no statistically significant difference in the estimations obtained by applying the three methods in combination with Web Objects.

It is worth mentioning that the present study confirmed and extended two our previous studies employing a different validation method and using a larger dataset. In particular, in [13] we assessed the effectiveness of Web Objects as indicators of development effort, when used in combination with OLSR, by employing dataset I of Table 1 (of 15 Web applications) as training set and further 4 Web applications as test set. The results revealed that the Web Objects measure is a good indicator of the development effort since we obtained MMRE=0.14, MdMRE=0.06, and Pred(25)=0.75. Moreover, in [12] we assessed the use of Web Objects in combination with Web-COBRA, using only dataset I of Table 1, with a leave-1-out cross validation, obtaining MMRE=0.11, MdMRE=0.10, and Pred(25)=0.93.

4.1 Comparison with Ruhe *et. al.* Analyses

Ruhe *et al.* [29][30] carried out empirical analyses based on a dataset of 12 Web applications developed between 1998 and 2002 by an Australian software company, with about twenty employees. The most of these projects were new developments, even if there were also enhancements, and re-development projects. The Web Objects measure was used as size metrics in combination with OLSR and Web-COBRA and a leave-1-out cross validation was exploited to validate the obtained estimation techniques. Ruhe *et al.* also employed summary measures MMRE, MdMRE, and Pred(25) and statistical test (T-test) to evaluate the accuracy of the obtained estimates.

Table 9 reports on the values of the summary statistics on the estimation accuracy obtained in [29][30]. We can observe that the summary values we obtained in our empirical analyses are slightly better than those obtained by Ruhe *et al.* Thus, the study reported in the present paper is confirming the results of the previous researches showing the effectiveness of Web Objects. Moreover, in all the three studies, the performed statistical tests (i.e., T-test) revealed that the estimates achieved with Web Objects significantly outperformed the estimates obtained with Function Points. As for comparison of the employed estimation techniques, the statistical analysis also suggested that the estimates obtained with OLSR and Web-COBRA are comparable, i.e., there is no significant difference between them.

Table 9. Ruhe *et al.'s* results reported in [29][30]

	MMRE	MdMRE	Pred(25)
OLSR with FP	0.33	0.33	0.42
OLSR with WO	0.24	0.23	0.67
Web-COBRA with WO	0.17	0.15	0.75

The results we obtained extend the ones of Ruhe *et al.* in several aspects. Indeed, besides the techniques employed in their case study, we also exploited CBR, still obtaining good results, thus showing a sort of robustness of Web Objects with respect to the employed techniques. As for the performed empirical analysis, we exploited further benchmarks (i.e., MeanEFH and MedianEFH) and more tests (i.e., effect size).

From a managerial point of view, our results extend the ones provided by Ruhe *et al.*, showing the scalability of the Web Objects measure, in terms of technologies and size of the considered projects. Indeed, the 25 Web applications used in our empirical study are more recent, being developed between 2003 and 2008, thus exploiting newer technologies, development environments, etc.. Moreover, they are much bigger than those used in [29]. Table 10 provides some descriptive statistics about the set of Web applications we employed in our case study and the dataset considered by Ruhe *et al.* in their study [29]. In particular, we reported on the size, the actual effort (in terms of person/hours), and the peak staff. We can observe that the mean effort of our dataset is about three times the one of the dataset used in [29] and applications are characterized also by a bigger size in terms of Web Objects (about five times bigger than those in [29]). It is interesting to note that the number of Function Points is not so different among the two datasets, since in our case the applications are about 1.5 times bigger than those of Ruhe *et al*., in terms of this size measure. A possible interpretation we gave to this phenomenon is that our applications highly exploit Web Building Blocks and Multimedia elements, which are considered by the Web Objects method but not by the FPA method.

Table 10. Descriptive statistics of EFH, WO, and Peak Staff

Our study					
	Min	**Max**	**Median**	**Mean**	**Std. Dev.**
WO	323	3,078	1366	1480	720.602
EFH (person/hours)	782	4537	2686	2577	988.136
Peak Staff	6	7	6	6.2	0.4
Ruhe *et al.*'s study					
	Min	**Max**	**Median**	**Mean**	**Std. Dev.**
WO	67	792	Unknown	284	227
EFH (person/hours)	267	2,504	Unknown	883	710
Peak Staff	2	6	Unknown	3	1.5

5 The Empirical Study Validity

It is widely recognized that several factors can bias the construct, internal, external, and conclusion validity of empirical studies [17] [20]. As for the construct validity, the choice of the size measure and how to collect information to determine size

measure and actual effort represent crucial aspects. Regarding the selection of the size measure, we employed a solution specifically proposed for Web applications by Reifer [26] and used in the previous case studies we have replicated. The software company uses timesheets to keep track of effort information, where each team member annotates his/her development effort and weekly each project manager stores the sum of the efforts for the team. To calculate the size measure, the authors defined a form to be filled in by the project managers. To apply the Web Objects method they employed the counting conventions of the FPA method [14] and followed the suggestions provided by Reifer in his "Web Objects White Paper" [27]. One of the authors analyzed the filled forms in order to cross-check the provided information. The same author calculated the values of the size measure. As for the collection of the information on the cost factors, we defined a questionnaire together with some company experts. Then, this questionnaire was submitted to the project managers of the considered Web applications. Thus, the data collection task was carried in a controlled and uniform fashion, making us confident on the accuracy of the results.

With regards to internal validity no initial selection of the subjects was carried out, so no bias has been apparently introduced. Moreover, the Web applications were developed with technologies and methods that subjects had experienced. Consequently, confounding effects from the employed methods and tools can be excluded. Moreover, the questionnaires used were the same for all the Web applications and the project managers were instructed on how to use them. Instrumentation effects in general did not occur in this kind of studies. As for the conclusion validity we carefully applied the statistical tests, verifying all the required assumptions. Biases about external validity were mitigated by considering as dataset a representative sample of modern Web applications. However, it is recognized that the results obtained in an industrial context might not hold in other contexts. Indeed, each context might be characterized by some specific project and human factors, such as development process, developer experience, application domain, tools, technologies used, time, and budget constraints [5].

6 Related Work

Besides the Web-COBRA method and the Web Objects measure, other estimation techniques and size measures have been proposed in the literature to be employed for estimating Web applications development effort.

The COSMIC [9] method has been applied to Web applications by some researchers in the last years [10][19]. In particular, Mendes *et al*. applied it to 37 Web systems developed by academic students, by constructing an effort estimation model with OLSR [19]. Unfortunately, this model did not provide good estimations and replications of the empirical study were highly recommended. Subsequently, an empirical study based on the use of 44 Web applications developed by academic students, was performed to assess the COSMIC approach [10]. The effort estimation model obtained by employing the OLSR provided encouraging results.

Some authors investigated the usefulness of size measures specific for Web applications such as number of Web pages, media elements, internal links, etc., [11], [20]. Among them, Mendes *et al*. also built the Tukutuku database [21], which aims to collect data from completed Web sites and applications to develop Web cost estimation models and to benchmark productivity across and within Web Companies. Several studies were conducted to investigate and compare the effectiveness of these measures in combination with estimation techniques like OLSR, CBR, Regression Tree (RT), and Bayesian Networks (BN). In particular, in [20] a dataset of 37 Web systems developed by academic students was exploited and the empirical results suggested that Stepwise Regression (SWR) provided statistically significant superior predictions than the other techniques when using length size measures, such as number of Web pages, number of new media. On the contrary, a study exploiting a dataset containing data on 15 Web software applications developed by a single Web company (the ones also employed in the empirical study presented in this paper) revealed that none of the employed techniques (i.e., SWR, RT, and CBR was statistically significantly superior than others [11]. Recently, Mendes and Mosley investigated the use of Bayesian Networks for Web effort estimation using the Web applications of the Tukutuku database [22]. In particular, they employed two training sets, each with 130 Web applications, to construct the models while their accuracy was measured using two test sets, each containing data on 65 Web applications. The analysis revealed that Manual SWR provided significantly better estimations than any of the models obtained by using Bayesian Networks and is the only approach that provided significantly better results than the median effort based model.

7 Conclusions

In this paper, we investigated the effectiveness of the Web Objects measure as indicator of Web application development effort. In particular, we replicated the two studies carried out by Ruhe *et al.* [29]. The contribution of our work to the body of knowledge can be summarized as in the following:

- we confirmed the effectiveness of the Web Objects measure as indicator of Web application development effort, when used in combination with OLSR and Web-COBRA, and verified that this holds also using CBR;
- we confirmed that the Web Objects method provides statistically superior results than the FPA method when used in combination with OLSR and Web-COBRA;
- we showed that there are no statistically significant differences in the results obtained with OLSR, CBR, and Web-COBRA, i.e., the approaches are comparable when using the Web Objects measure.

Of course, the experimental results here presented hold only with respect to the dataset took into account and they should be assessed on further data as soon as they are available. However, they are surely interesting enough to suggest the use of the Web Objects measure as indicator of Web application development effort, also because confirm the results of Ruhe *et al*.. In the future, we intend to further assess Web Objects by considering a different context.

References

[1] Aamodt, A., Plaza, E.: Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communication 7(1), 39–59 (1994)

[2] Basili, V., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Transactions on Software Engineering 25(4), 435–437 (1999)

[3] Briand, L., Wieczorek, I.: Software resource estimation. In: Encyclopedia of Software Engineering, pp. 1160–1196 (2002)

[4] Briand, L.C., Emam, K.E., Bomarius, F.: COBRA: a hybrid method for software cost estimation, benchmarking, and risk assessment. In: Proceedings of the International Conference on Software Engineering, pp. 390–399. IEEE Computer Society, Los Alamitos (1998)

[5] Briand, L.C., Wust, J.: Modeling Development Effort in Object-Oriented Systems Using Design Properties. IEEE Transactions on Software Engineering 27(11), 963–986 (2001)

[6] Cohen, J.: Statistical power analysis for the behavioral science. Lawrence Erlbaum, Hillsdale (1998)

[7] Conallen, J.: Building Web Applications with UML. Addison-Wesley, Reading (1999)

[8] Conte, D., Dunsmore, H., Shen, V.: Software Engineering Metrics and Models. The Benjamin/Cummings Publishing Company, Inc. (1986)

[9] COSMIC (2007), `http://www.cosmicon.com`

[10] Costagliola, G., Di Martino, S., Ferrucci, F., Gravino, C., Tortora, G., Vitiello, G.: A COSMIC-FFP approach to Predict Web Application Development Effort. Journal of Web Engineering 5(2) (2006)

[11] Costagliola, G., Di Martino, S., Ferrucci, F., Gravino, C., Tortora, G., Vitiello, G.: Effort Estimation Modeling Techniques: A Case Study for Web Applications. In: Proceedings of the International Conference on Web Engineering, pp. 161–165. ACM Press, New York (2006)

[12] Di Martino, S., Ferrucci, F., Gravino, C.: An Empirical Study on the Use of Web-COBRA and Web Objects to Estimate Web Application Development Effort. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 213–220. Springer, Heidelberg (2009)

[13] Ferrucci, F., Gravino, C., Di Martino, S.: A Case Study Using Web Objects and COSMIC for Effort Estimation of Web Applications. In: Proceedings of Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2008), pp. 441–448 (2008)

[14] I. F. P. U. G., Function point counting practices manual, release 4.2.1

[15] Kampenes, V., Dyba, T., Hannay, J., Sjoberg, D.: A systematic review of effect size in software engineering experiments. Information & Software Technology 49(11-12), 1073–1086 (2007)

[16] Kitchenham, B., Mendes, E., Travassos: Cross versus Within-Company Cost Estimation Studies: A systematic Review. IEEE Transactions on Software Engineering 33(5), 316–329 (2007)

[17] Kitchenham, B., Pickard, L., Pfleeger, S.L.: Case Studies for Method and Tool Evaluation. IEEE Software 12(4), 52–62 (1995)

[18] Kitchenham, B., Pickard, L.M., MacDonell, S.G., Shepperd, M.J.: What accuracy statistics really measure. IEE Proceedings Software 148(3), 81–85 (2001)

[19] Mendes, E., Counsell, S., Mosley, N.: Comparison of Web Size Measures for Predicting Web Design and Authoring Effort. IEE Proceedings-Software 149(3), 86–92 (2002)

[20] Mendes, E., Counsell, S., Mosley, N., Triggs, C., Watson, I.: A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. Empirical Software Engineering 8(23) (2003)

[21] Mendes, E., Kitchenham, B.: Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. In: Proceedings of International Software Metrics Symposium, pp. 348–357. IEEE press, Los Alamitos (2004)
[22] Mendes, E., Mosley, N.: Bayesian Network Models for Web Effort Prediction: A Comparative Study. IEEE Transactions on Software Engineering 34(6), 723–737 (2008)
[23] Mendes, E., Mosley, N., Counsell, S.: Investigating Web Size Metrics for Early Web Cost Estimation. Journal of Systems and Software 77(2), 157–172 (2005)
[24] Montgomery, D., Peck, E., Vining, G.: *Introduction to Linear Regression Analysis*. John Wiley and Sons, Inc., Chichester (1986)
[25] Morisio, M., Stamelos, I., Spahos, V., Romano, D.: Measuring Functionality and Productivity in Web-based applications: a Case Study. In: Proceedings of the International Software Metrics Symposium, pp. 111–118. IEEE press, Los Alamitos (1999)
[26] Reifer, D.: Web-Development: Estimating Quick-Time-to-Market Software. IEEE Software 17(8), 57–64 (2000)
[27] Reifer, D.: Web Objects Counting Conventions, Reifer Consultants (March 2001), `http://www.reifer.com/download.html`
[28] Ruhe, M.: The Accurate and Early Effort Estimation of Web Applications, PhD Thesis, Fraunhofer IESE 2002 (2002)
[29] Ruhe, M., Jeffery, R., Wieczorek, I.: Cost estimation for Web applications. In: Proceedings of International Conference on Software Engineering, pp. 285–294. IEEE press, Los Alamitos (2003)
[30] Ruhe, M., Jeffery, R., Wieczorek, I.: Using Web Objects for Estimating Software Development Effort for Web Applications. In: Proceedings of the International Software Metrics Symposium (2003)
[31] Shepperd, M., Schofield, C.: Estimating software Project Effort using Analogies. IEEE Transactions on Software Engineering 23(11), 736–743 (2000)

Applying EFFORT for Evaluating CRM Open Source Systems

Lerina Aversano and Maria Tortorella

Departement of Engineering, University of Sannio
Via Traiano,
82100 Benevento, Italy
{aversano,tortorella}@unisannio.it

Abstract. Free Open Source Software (F/OSS) for Customer relationship management (CRM) represents a great opportunity for small and medium enterprises as they can significantly impact their competitiveness. These software solutions can be adopted with success, whatever the size, and offer customized solutions for clients, even with few people working of it. This paper presents an analysis of the quality of CRM F/OSS systems using a predefined quality model. The analysis focuses on the main relevant aspect of both open source domain and application domain proposing a trade off assessment among these.

Keywords: FlOSS Free/Open Source Software, Software quality, Software Metrics, CRM Systems, Standard.

1 Introduction

Customer relationship management (CRM) systems are software systems aiming to support enterprises to automate many services from sales to marketing to customer services. They are efficient solutions that, however, were mainly used from big enterprises which could provide deep pockets and time to undertake these huge implementation projects. Nevertheless, even small business organizations can consider the introduction of CRM. The problem is that CRM is too expensive and difficult to be properly implemented, and it seemingly requires radical change. Actually, Small and Medium Enterprises – SMEs – have to deal with major difficulties as they have few resources to dedicate to selection, acquisition, configuration and customization of such complex systems. Moreover, enterprise software systems are generally designed to fit needs of big companies.

Adoption of Free/Open Source Systems – F/OSS – partially fill up this gap. F/OSS CRM systems are actually available to any business, whatever the size, and offer customized solutions for clients, even with few people that can be up and running in two to three weeks. This problem was already faced with reference to ERP systems and even in that case the adoption of a F/OSS ERP was proved to be very advantageous for SME [2, 3]. As an example, the possibility of really trying the system (not just using a demo), reduction of vendor lock-in, low license cost and possibility of in-depth personalization are some of the advantages.

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 202–216, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Likewise ERP systems, even adopting a Customer Relationship Management systems could represent an important competitive advantage for a company, but it could also be useless or even harmful if the system does not adequately fit the organization needs. Then, the selection and adoption of such a kind of system cannot be faced in a superficial way and evaluation supports are needed. Many quality models for evaluating F/OSS systems have been proposed in literature [10-16]. Nevertheless, they do not cover all the relevant aspects of quality and operative context of such systems and are not always applicable to operative contexts. An evaluation of these models is provided in [1].

This paper proposes a framework for a quantitative evaluation of the quality of F/OSS CRM systems. The framework is defined by specializing a more general one, called EFFORT – Evaluation Framework for Free/Open souRce projects – defined for evaluating open source software projects [1]. The EFFORT framework was already assessed for ERP Systems [5]. It is conceived to properly evaluate F/OSS projects and was defined on the basis of the Goal Question Metric (GQM) paradigm [6].

The rest of the paper is organized as follow: Section 2 presents EFFORT that is a necessary background for discussing the quantitative analysis of the CRM systems; Section 3 provides the specialization of EFFORT for evaluating CRM system; Section 4 presents a case study, regarding the evaluation of 4 open source CRM F/OSS projects: SugarCRM (www.sugarcrm.com), CreamCRM (http://sourceforge.net/projects/cream-crm/), Concursive ConnectCRM (www.concursive.com), VTigerCRM (www.vtiger.com). Concluding remarks are given in the last section.

2 Background

EFFORT is a framework defined for evaluating F/OSS systems [1]. In this paper. it is considered as a base framework to be specialized to the context of CRM systems.

As told in the introduction, EFFORT was defined on the basis of the GQM paradigm [6]. This paradigm guides the definition of a metric program on the basis of three abstraction levels: Conceptual level, referred to the definition of the *Goals* to be achieved by the measurement activity; Operational level, consisting of a set of *Questions* facing the way the assessment/achievement of a specific goal is addressed; and Quantitative level, identifying a set of *Metrics* to be associated to each question.

The GQM paradigm helped to define a quality model for F/OSS projects, providing a framework to be actually used during the evaluation. It considers the quality of a F/OSS project as synergy of three main elements: *quality of the product* developed within the project; *trustworthiness of the community* of developers and contributors; and *product attractiveness* to its specified catchment area.

Figure 1 shows the hierarchy of attributes that composes the quality model. In correspondence of each first-level characteristic, one *Goal* was defined. Then, the EFFORT measurement framework included three goals. *Questions*, consequentially, mapped second-level characteristics, even if, considering the amount of aspects to take into account, *Goal 1* was broken up into sub-goals, because of its high complexity. For question of space, the figure does not present the third level related to the metrics used for answering the questions.

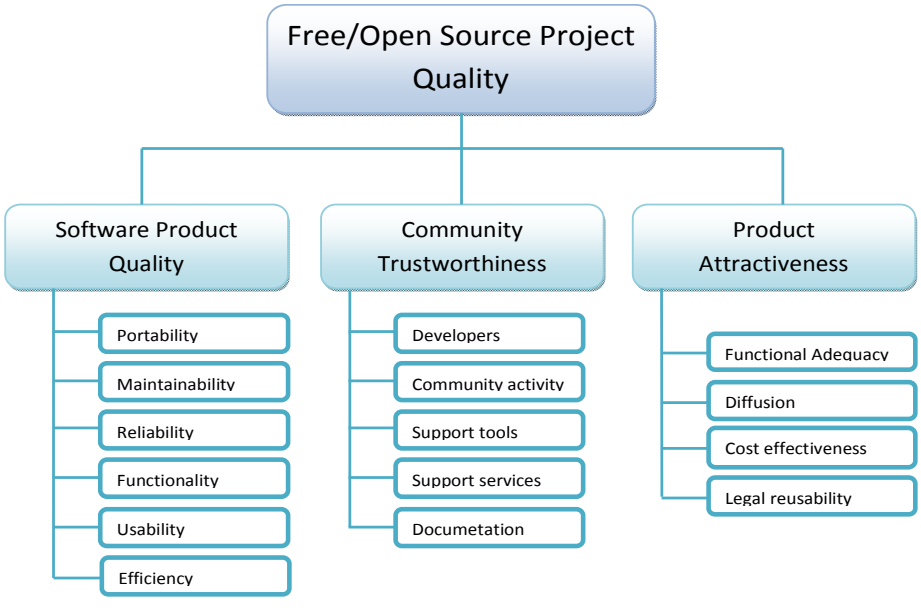


Fig. 1. Hierarchy of the quality model

The following subsections summarily describe the three goals providing their formalization, incidental definitions of specific terms and the list of questions. The listed questions can be answered through the evaluation of a set of associated metrics. For reason of space, the paper does not present all the metrics, and includes some references to them in the final subsection that discusses how the gathered metrics can be aggregated for quantitatively answering the questions.

2.1 Product Quality

One of the main aspects that denotes the quality of a project is the product quality. It is unlikely that a product of high and durable quality was developed in a poor quality project. So, all the aspects of the software product quality were considered, as defined by the ISO standard [8, 9].

Goal 1 was defined as follows:

*Analyze the **software product** with the aim of **evaluating** its **quality**, from the **software engineer**'s point of view.*

Table 1 shows all the sub-goals and questions regarding *Goal 1*. As it can be noticed almost all the attributes that the questions reference regards the ISO 9125 standard.

Table 1. Questions about Product Quality

GOAL 1- **PRODUCT QUALITY**	
Sub-goal 1a: *Analyze the software product with the aim of evaluating it with reference to portability, from the software engineer's point of view*	
Q 1a.1	What degree of adaptability does the product offer?
Q 1a.2	What degree of installability does the product offer?
Q 1a.3	What degree of replaceability does the product offer?
Q 1a.4	What degree of coexistence does the product offer?
Sub-goal 1b: *Analyze the software product with the aim of evaluating it with reference to maintainability, from the software engineer's point of view*	
Q 1b.1	What degree of analyzability does the product offer?
Q 1b.2	What degree of changeability does the product offer?
Q 1b.3	What degree of testability does the product offer?
Q 1b.4	What degree of technology concentration does the product offer?
Q 1b.5	What degree of stability does the product offer?
Sub-goal 1c: *Analyze the software product with the aim of evaluating it with reference to reliability, from the software engineer's point of view*	
Q 1c.1	What degree of robustness does the product offer?
Q 1c.2	What degree of recoverability does the product offer?
Sub-goal 1d: *Analyze the software product with the aim of evaluating it with reference to functionality, from the software engineer's point of view*	
Q 1d.1	What degree of functional adequacy does the product offer?
Q 1d.2	What degree of interoperability does the product offer?
Q 1d.3	What degree of functional accuracy does the product offer?
Sub-goal 1e: *Analyze the software product with the aim of evaluating it with reference to usability, from the user's point of view*	
Q 1e.1	What degree of pleasantness does the product offer?
Q 1e.2	What degree of operability does the product offer?
Q 1e.3	What degree of understandability does the product offer?
Q 1e.4	What degree of learnability does the product offer?
Sub-goal 1f: *Analyze the software product with the aim of evaluating it with reference to efficiency, from the software engineering's point of view*	
Q 1f.1	How the product is characterized in terms of time behaviour?
Q 1f.2	How the product is characterized in terms of resources utilization?

2.2 Community Trustworthiness

With *Community Trustworthiness*, it was intended the degree of trust that a user give to a community, regarding the offered support. Support could be provided by the communities by means of: good execution of the development activity; use of tools, such as wiki, forum, trackers; and availability of services, such as maintenance, certification, consulting and outsourcing, and documentation.

Goal 2 was defined as follows:

*Analyze the offered **support** with the aim of **evaluating** the **community** with reference to the **trustworthiness,** from the **(user/organization) adopter**'s point of view.*

Questions regarding *Goal 2* are shown in Table 2.

Table 2. Questions about Community Trustiworthiness

GOAL 2- **Community Trustiworthiness**	
Id question	**Question**
Q 2.1	How many developers does the community involve?
Q 2.2	What degree of activity has the community?
Q 2.3	Support tools are available and effective?
Q 2.4	Are support services provided?
Q 2.5	Is the documentation exhaustive and easily consultable?

2.3 Product Attractiveness

The third goal had the purpose of evaluating the attractiveness of the product within its application area. The term *attractiveness* indicates all the factors that influence the adoption of a product from a potential user, who perceives convenience and usefulness to achieve his scopes.

Goal 3 was related to product attractiveness and formalized as follows:

*Analyze **software product** with the aim of **evaluating** it with reference to the **attractiveness** from the **(user/organization) adopter**'s point of view.*

Two elements to be considered for evaluating a F*l*OSS product were *functional adequacy* and *diffusion*. The latter could be considered as a marker of how the product was appreciated and recognized as useful and effective. Other factors that could be considered were *cost effectiveness*, estimating the *TCO* (Total Cost of Ownership) [7], and the *type of license*.

Questions for Goal 3 are shown in Table 3; while, as an example, Table 4 lists the metrics related to question 3.2.

Table 3. Questions regarding Product Attractiveness

GOAL 2- **Community Trustiworthiness**	
Id question	**Question**
Q 3.1	What degree of functional adequacy does the product offer?
Q 3.2	What degree of diffusion does the product achieve?
Q 3.3	What level of cost effectiveness is estimated?
Q 3.4	What degree of reusability and redistribution is left by the license?

Table 4. Metrics related to question Q 3.2

Id Metric	**Metric**
M 3.2.1	Number of thread per year
M 3.2.2	Index of unreplied threads
M 3.2.3	Number of forums
M 3.2.4	Average of threads per forum
M 3.2.5	Average of posts per year
M 3.2.6	Degree of internationalization of the forum
M 3.2.7	Number of trackers
M 3.2.8	Wiki volume
M 3.2.9	Number of frequently asked questions

2.4 Data Aggregation and Interpretation

Once data were collected by means of the metrics, it was necessary to aggregate them, according to their interpretation, for answering the questions and obtaining useful information. Aggregation of answers gives an indication regarding the achievement of the goals.

In doing aggregation, some issues needed to be considered. These are listed below:

- Metrics have different types of scale, depending on their nature. Then, it was not possible to directly aggregate measures. After the measurement was done and to overcome that problem, each metric was mapped to a discrete score in the [1-5] interval, where: 1 = inadequate; 2 = poor; 3 = sufficient; 4 = good; and, 5 = excellent.
- An high value for a metric could be interpreted in either positive or negative way, according to the context of the related question; even the same metric could contribute in two opposite ways in the context of two different questions. So, the appropriate interpretation was provided for each metric.
- As questions did not have the same relevance in the evaluation of a goal, a relevance marker was associated to each metric in the form of a numeric value in interval [1-5]. Value 1 is associated to questions with the minimum relevance, while value 5 means maximum relevance. The definition of the relevance markers depended on the experience and knowledge of the software engineer and the organizational needs and requirements.

The aggregation function for Goal g was defined as follows:

$$q(g) = \left[\sum_{id \in Q_g} r_{id} * m(id) \right] / \sum_{id \in Q_g} r_{id}$$

where:

r_{id} is the relevance associated to question *id* (sub-goal for goal 1);
Q_g is the set of questions (sub-goals for goal 1) related to goal g.
m(q) is the aggregation function of the metrics of question *q,* defined as:

$$m(q) = \left\{ \sum_{id \in M_q} i(id) * v(id) + [1 - i(id)] * [v(id) \bmod 6] \right\} / |M_q|$$

where *v(id)* is the score obtained for metric *id* and *i(id)* is its interpretation. In particular:

$$i(id) = \begin{cases} 0 \; if \; the \; metric \; has \; negative \; interpretation \\ 1 \; if \; the \; metric \; has \; positive \; interpretation \end{cases}$$

and M_q is the set of metrics related to question q.

3 EFFORT Specialization

CRM solutions are enterprise software systems whose goal is to learn more about customers' needs and behaviors in order to develop stronger relationships with them, and facilitate acquiring, enhancing and retaining of customers. Although CRM has emerged as a major business strategy for e-commerce, little research was conducted in

Table 5. Integration of the EFFORT specialization

EFFORT Integration for Goal 1			
Id Question	**Question**	**Id Metric**	**Metric**
1a.1	What degree of adaptability does the product offer?	**1a.1.2**	Number of Application Servers supported
1a.3	What degree of replaceability does the product offer?	**1a.3.1**	Availability of functionality for creation of data backup
		1a.3.2	Availability of functionality for restoration of data backup
		1a.3.3	Availability of backup services
		1a.3.4	Number of file formats for the reporting
		1a.3.5	Number formats per the importing of data
1b.1	What degree of analyzability does the product offer?	**1b.1.3**	Number of Package
		1b.1.4	Number of Class
		1b.1.6	Methods for Class
		1b.1.7	Javadoc density (MB)/NOC
1b.2	What degree of changeability does the product offer?	**1b.2.1**	Lack of methods cohesion, as defined by Henderson-Sellers
		1b.2.2	Efferent coupling
		1b.2.3	Afferent coupling
		1b.2.5	Average value of the number of methods per class
1b.3	What degree of testability does the product offer?	**1b.3.2**	Average value of the height of the inheritance tree
		1b.3.3	Average of the Number of subclass for class
		1b.3.6	Average of the Number of attribute for class
		1b.3.7	Average of the Number of override method for class
		1b.3.8	Average number of test drivers respect to the number of classes

evaluating the available CRM solutions. In particular, in the best of the authors' knowledge, research did not propose either comparative analyses of open source solutions or analyses of the quality of such a kind of products. With the aim of overcoming this lack, the EFFORT framework was specialized to this specific application context.

As already stated in the previous section, EFFORT needs to be specialization to the context of the CRM systems before being applied for evaluating such a kind of systems. The specialization tasks was performed at the level of questions or goals. The first kind of task regarded the integration of additional metrics for answering some baseline EFFORT questions; while the intervention at the goal level concerned the extension of some goals with the adding of further questions.

In particular, with reference to the integration, Table 5 shows all the metrics that were added for answering some baseline questions. A large part of integration was performed with reference to Goal 1 and all the choices are explained by the strategic role played in the context of the CRM systems by both data and necessity of integrating an application with the information system of an organization. Therefore, additional metrics were considered for evaluating the Adaptability and Replaceability (and, consequentially, Portability), as it can be evicted from Table 5. In fact, the Number of Application Servers supported was considered for the adaptability

characteristic. Whereas, Availability of functionality for backup and restore data, Availability of backup services and Numbers of reporting formats were taken into account for the Replaceability characteristic. Evaluating the Analysability required the addition of a Javadoc density metric as all the analysed CRM systems were based on the Java technology. Moreover, as the analysed software systems were based on the object-oriented paradigm, metrics relate to this paradigm were considered. This required the instantiation of the measurement framework with the adoption of the specific object-oriented metrics.

With reference to Goal 3, Table 6 contains the integration performed for understanding the economical advantage when a CRM F/OSS system is adopted. This advantage does not depend just on the license costs, but also on those costs to be spent

Table 6. Integration for instantiating EFFORT for Goal 3

EFFORT Specialization of Goal 3			
Id Question	**Question**	**Id Metric**	**Metric**
Q 3.3	What degree of economical advantages is estimated?	**3.3.5**	Cost of migration among different versions
		3.3.6	Cost of system population
		3.3.7	Cost of system configuration
		3.3.8	Cost of system customization
Q 3.5	What degree of support for migration between different releases is it offered?	**3.5.1**	Availability of functionality for creation of data backup
		3.5.2	Availability of functionality for restoration of data backup
		3.5.3	Availability of backup services
		3.5.4	Availability of documentation of migration between versions
		3.5.5	Availability of automatic migration tools
		3.5.6	Availability of documentation for migrating the database
Q 3.6	What degree of support for population of the system is it offered?	**3.6.1**	Number of standard formats for importing data
Q 3.7	What degree of support for configuration of the system is it offered?	**3.7.1**	Availability of a wizard for configuring the system
		3.7.2	Number of supported languages
		3.7.3	Availability of documentation for supporting the starting setup
		3.7.4	Availability of documentation for supporting language configuration
Q 3.8	What degree of support for customization of the system is it offered?	**3.8.1**	Availability of functionality for installing an extension from the user interface
		3.8.2	Availability of functionality for creating a new module from the user interface
		3.8.3	Availability of functionality for customization of the user interface
		3.8.4	Availability of functionality for creating customized report
		3.8.5	N° of standard template for the creation of report
		3.8.6	Availability of documentation for the product customization

for both adapting the adopted software system to own needs and maintaining it by installing updated versions or adoption of new releases.

Goal 3 mainly regarded the way a software system should be used for being attractive. Then, it strongly depended on the application domain of the analysed software system and needs a customization to the specific context. Therefore, in the CRM context, the EFFORT framework was extended and customized taking into account additional specific attraction factors by considering additional questions referred to *Goal 3*. In particular the following aspects were considered:

- *Migration between different versions of the software*, in terms of support provided for switching from a release to another one. In the context of CRM systems, this cannot be addressed like a new installation, as it would be too costly, taking into account that such a kind of system is generally customized and hosts a lot of data.
- *System population*, in terms of support offered for importing big volumes of data into the system.
- *System configuration*, intended as support provided, in terms of functionality and documentation, regarding the adaption of the system to specific needs of the company, such as localization and internationalization: higher the system configurability, lower the start-up time.
- *System customization*, intended as support provided, without direct access to the source code, for doing alteration of the system, such as the definition of new modules, installation of extensions, personalization of reports and possibility of creating new workflows. This characteristic is very desirable in CRM systems.

Table 6 shows questions that extend Goal 3. As it can be noticed, the new questions are referred to the listed characteristics.

During the instantiation of EFFORT, a relevance regarding the CRM context was associated to each metric by using numeric values of the [1-5] interval. The EFFORT relevance factors that were additionally considered are the following:

- $rFlOSS_{id}$, representing the relevance indicator in the F*l*OSS context associated with question *id* or sub-goal *id* of Goal 1;
- $rCRM_{id}$, indicating the relevance indicator in the CRM context associated with question *id* or sub-goal *id* of Goal 1;
- Q_g, the set of questions (or sub-goals for Goal 1) related to Goal g;

Then, the aggregation function for evaluating Goal g is defined as follows:

$$q(g) = \frac{[\sum_{id \in Qg}(rFlOSS_{id} + rCRM_{id}) * m(q)]}{\sum_{id \in Qg}(rFlOSS_{id} + rCRM_{id})}$$

where $m(q)$ is the aggregation function of the metrics of question q.

4 Results

After specializing the EFFORT framework, an analysis of CRM F*l*OSS was performed. It permitted to choose the following four CRM systems considered for being evaluated: SugarCRM (www.sugarcrm.com), CreamCRM (http://sourceforge.net/

projects/cream-crm/), Concursive ConnectCRM (www.concursive.com), VTigerCRM (www.vtiger.com). These systems were selected from the top ten classification of open source CRM systems [4].

SugarCRM is an open source Customer Relationship Management (CRM) software that runs on Windows/Linux. SugarCRM is web-based and can be accessed with a web browser locally connected or set up for the internet access by any machine with a browser and an internet connection. SugarCRM for Microsoft Windows requires the open source software Apache Web Server, MySql Database and PHP. Functionality includes sales force automation, marketing campaigns, support cases, project management, calendaring, documents and more.

Cream CRM is a multilingual application designed for supporting the following services: tracks sale orders, payments, shipments, services, online and print subscriptions, and the effectiveness of promotional campaigns. Modules allow communication with customers via newsletters, email and a Web interface. Cream CRM runs on FreeDSB, Linux and Windows 2000/XP. It is written in Java and JavaScript.

vTiger CRM is built upon the LAMP/WAMP (Linux/Windows, Apache, MySQL and PHP) architecture, with the main development team based in Chennai, India. vTiger CRM includes SFA (Sales Force Automation), customer-support and -service, marketing automation, inventory-management, multiple database support, security-management, product-customization, calendaring and email-integration features. It also offers add-ons and support for other add-ons. Vtiger is written in JavaScript, PHP and Visual Basic. It is compatible with ADOdb, MySQL and PostgreSQL databases.

Concursive offers two robust products that enable businesses and other organizations to harness the power of collaboration, leverage social networks and manage their activities. ConcourseConnect is a platform that enables the creation of true commercial networks. Commercial networks are the intersection of social networking, web 2.0 collaboration and sales and marketing tools. ConcourseConnect enables organizations to create dynamic communities to connect various stakeholders and manages the entire ecosystem with an integrated management console and backend CRM tools.

Figure 2 and 3 show the results regarding the product quality obtained by applying the specialized framework. In particular, Figure 2 shows that vTiger CRM exhibits the best product quality. This results are also graphically confirmed by Figure 3 highlighting the detailed results regarding each characteristic of Goal 1. The higher value of the **Portability** was achieved by SugarCRM, that is 3.4. Indeed, all the portability metrics achieved good values for this system. On the contrary, the portability value reached by Cream CRM was 1.83 that is relatively low. This fact mainly depended on the lack of functionality for the creation and backup of data, deficiency of installation manuals, and customization of installation procedures that assumed a default value in just the 25% of cases for the data import/export. The value of Concursive was medium, 2.7. This was mainly due to the high values of metrics M1a3.1, M1a3.2, M1a3.3, M1a3.4, M1a3.5, related to the software replaceability and data import export, and to the low value of metrics, related to question Q1a2, regarding the knowledge required about the third part software system and guide effectiveness.

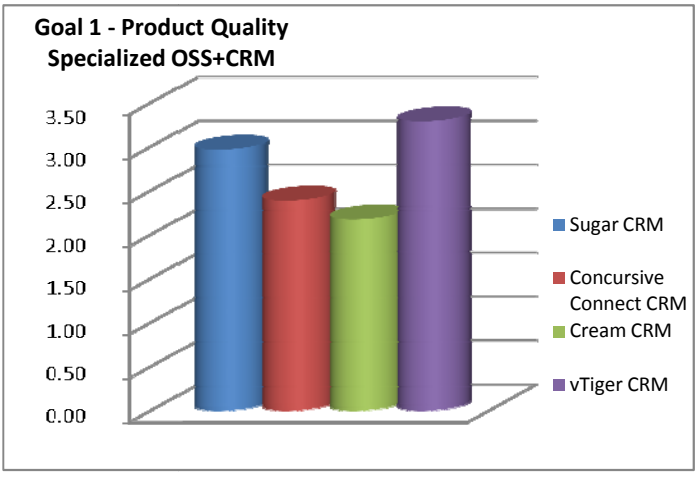


Fig. 2. Overall results obtained with reference to the Goal 1- Product Quality

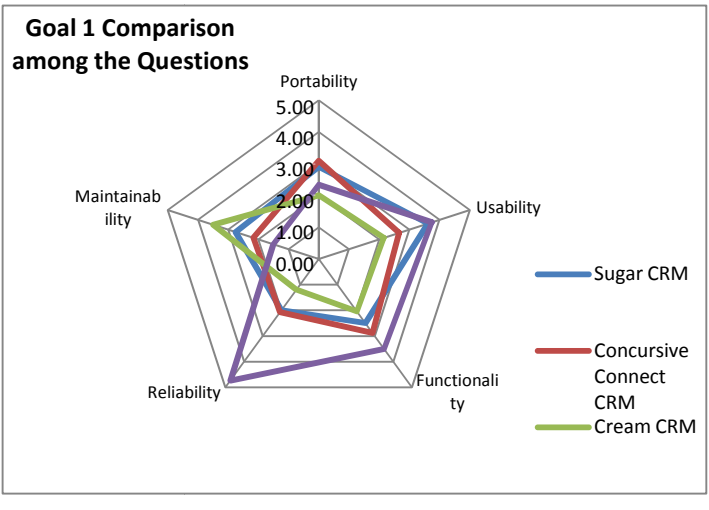


Fig. 3. Detailed results obtained with reference to the Goal 1- Product Quality

With reference to the **Maintainability,** the value achieved by SugarCRM is 2.5, that was higher than the value 3 obtained by CreamCRM. This was due to the value of Methods per Class (M1b1.6), that was very high in CreamCRM, and even to the good value 3 obtained by the index of LOC evaluated in Q1b3. As it can be noticed, Maintainability was very low for vTiger CRM, as it exhibited a very low value of modifiability (Q1.b.2), testability (Q1.b.3) and technology concentration (Q1.b.4).

Reliability generally achieved low values: value 2 for SugarCRM; value 1.7 for Cream CRM. These results were essentially penalized by the lack of backup services and bug tracker, that influence the values of the metrics related to Q1c. In any case,

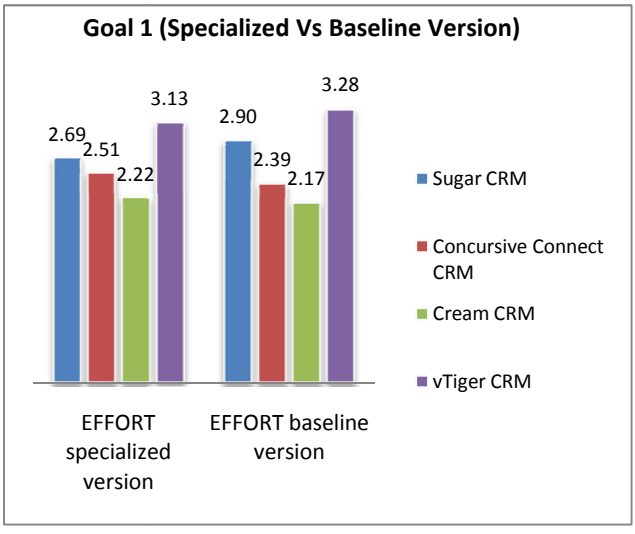


Fig. 4. Comparing the results of the Product Quality between the Specialized Framework and Baseline Framework

Reliability reached a high value for vTiger for the high capability of robustness and recoverability.

The best value regarding the **Functionality** characteristic was reached by vTiger that is 3.33; while the lowest value was reached by Cream CRM, that is 1.97, essentially due to the lack of web services support.

Similar values of **Usability** were reached by SugarCRM and vTiger CRM; while the worst value was still that one obtained by Cream CRM, mainly due to the limited documentation and videos available for the training of the product.

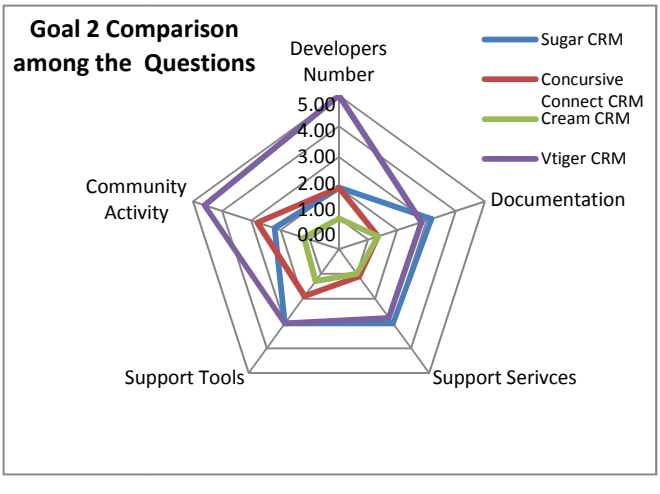


Fig. 5. Comparing the results of the Community Trustworthiness for each question

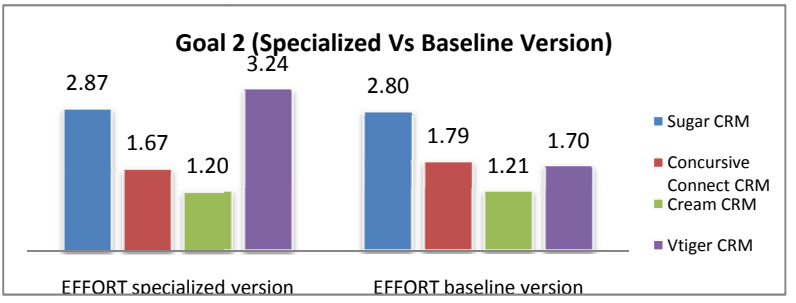


Fig. 6. Comparing the results of the Community Trustworthiness between the Specialized Framework and Baseline Framework

Overall, observing Figure 4, it can be noticed that the CRM system achieving the best results is vTiger CRM, even with the EFFORT specialized version. While the worst results are those of Cream CRM, as emerged by the data analysed for Goal 1.

Moving to the Goal 2, regarding the Community Trustworthiness, the results confirm the outcomes above. Indeed, vTiger CRM obtains the best results even with reference to Goal 2; while the worst value is achieved by Cream CRM that just obtains the value 1.2. The reason of this result is mainly due to the lack of supporting services and poor use of the forums, as shown in Figure 5.

Figure 6 shows that vTiger CRM achieved better results with the EFFORT specialized version instead of the baseline one. Actually, this was due to the better results obtained for those aspects having an higher relevance score in the specialized version.

Results of Goal 3, related to the Product Attractiveness, are shown in Figure 7 and Figure 8. In this case, the differences among the results obtained by the different CRM systems are less relevant. Nevertheless, the overall results confirm that vTiger CRM achieved the best values, and the lowest values are again those of Cream CRM.

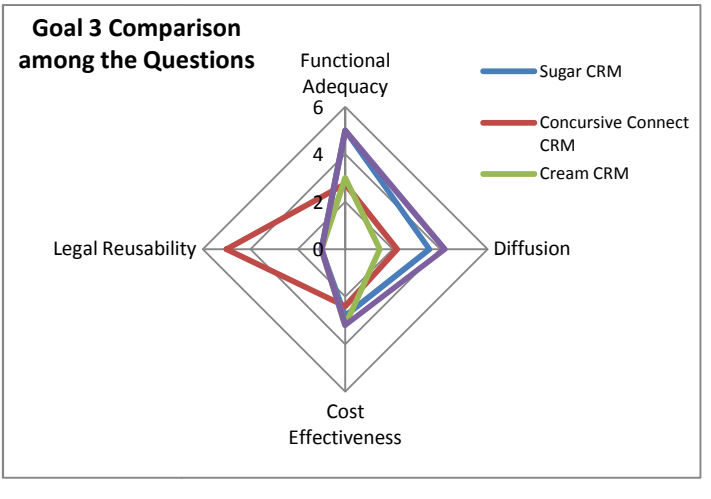


Fig. 7. Comparing the results of the Product Attractiveness for each question

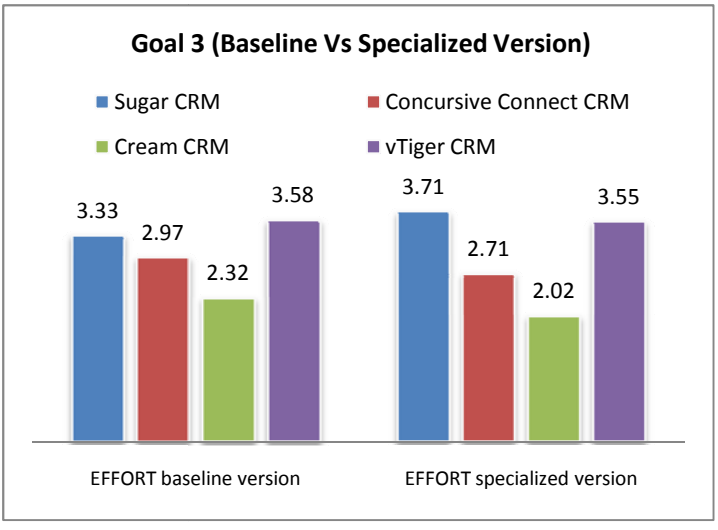


Fig. 8. Comparing the results of the Product Attractiveness between the Baseline Framework and Specialized Framework

Some reasons of the low results of CreamCRM are the following: question Q 3.1 regarding the Functional Adequacy achieved a very low value because not all the modules work correctly; question Q 3.2, regarding the diffusion, reached a low value due to the few visibility on the net; question Q 3.5, regarding the migration support, assumed a low value mainly for the lack of support for automating the migration and data backup.

5 Conclusion

The introduction of an open source CRM system into an organization can lead to the increase of its productivity, but it could also be an obstacle, if the implementation is not carefully faced. The availability of methodological and technical tools for supporting the process of evaluating and adopting a CRM system is desirable.

The work presented in this paper is related to the application of EFFORT, a framework for the evaluation of F/OSS projects, after its specialization to explicitly fit the CRM software system domain. The specialization mainly regarded the product attractiveness characterization.

The proposed framework is compliant to the ISO standards for product quality. In fact, it considers all of characteristics defined by the ISO/IEC 9126 standard model, but in-use quality. Moreover, it considers major aspects of F/OSS projects.

The usefulness of the framework is described through its concrete application. Indeed, EFFORT was used for evaluating four CRM open source systems, selected among the most diffused F/OSS CRM. The obtained results are quite good for product quality and product attractiveness. They are less positive with reference the community trustworthiness. The interpretation of the obtained results is strictly connected to the adopting enterprise's environment, needs and requirements.

Future investigation will regard the integration in the framework of a questionnaire for evaluating customer satisfaction. This obviously includes more complex analysis. In particular, methods and techniques specialized for exploiting this aspect will be explored and defined.

In addition, the authors will continue to search for additional evidence of the usefulness and applicability of the EFFORT and customizations, by conducting additional studies also involving subjects working in operative realities. In particular, EFFORT will be extended for better considering evolution aspects.

References

1. Aversano, L., Pennino, I., Tortorella, M.: Evaluating the Quality of FREE/OPEN Source Project. In: ENASE (Evaluation of Novel Approaches to Software Engineering) Conferences, INSTICC (2010)
2. Hyoseob, K., Boldyreff, C.: Open Source ERP for SMEs. In: ICMR 2005. Cranfield University (2005)
3. Wheeler, D. A.: How to evaluate open source software/free software (OSS/FS) programs (2009), http://www.dwheeler.com/oss_fs_eval.html#support
4. Hakala, D.: The Top 10 Open-Source CRM Solutions. Focus Expert Network (2009), http://www.focus.com/briefs/crm/top-10-open-source-crm-solutions/
5. Aversano, L., Tortorella, M.: Evaluating the quality of free/Open source systems: A case study. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2010. Lecture Notes in Business Information Processing, vol. 73, pp. 119–134. Springer, Heidelberg (2011)
6. Basili, V.R., Caldiera, G., Rombach, H.D.: The goal question metric approach. Encyclopedia of Software Engineering. Wiley Publishers, Chichester (1994)
7. Kan, S.H., Basili, V.R., Shapiro, L.N.: Software quality: an overview from the perspective of total quality management. IBM Systems Journal (1994)
8. International Organization for Standardization, 2001-2004. ISO standard 9126: Software Engineering – Product Quality, part 1-4. ISO/IEC
9. International Organization for Standardization, ISO standard ISO/IEC 25000:2005, Software Engineering – Software product Quality Requirements and Evaluation, SQuaRE (2005)
10. Golden, B.: Making Open Source Ready for the Enterprise: The Open Source Maturity Model. In: Succeeding with Open Source, Addison-Wesley Publishing Company, Reading (2005)
11. Kamseu, F., Habra, N.: Adoption of open source software: Is it the matter of quality? PReCISE PReCISE, Computer Science Faculty, University of Namur, rue Grandgagnage, Belgium (2009)
12. OpenBRR. Business Readiness for Open Source. Intel (2005)
13. QSOS. Method for Qualification and Selection of Open Source software. Atos Origin (2006)
14. Samoladas, I., Gousios, G., Spinellis, D., Stamelos, I.: The SQO-OSS quality model: measurement based open source software evaluation. In: IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, OSS 2008, Springer, Heidelberg (2008)
15. Spinellis, D., Gousios, G., Karakoidas, V., Louridas, P., Adams, P.J., Samoladas, I., Stamelos, I.: Evaluating the Quality of Open Source Software. Electr. Notes Theor. Comput. Sci., vol. 233, pp. 5–28. Springer, Heidelberg (2009)
16. Sung, W.J., Kim, J.H., Rhew, S.Y.: A quality model for open source selection. In: Sixth International Conference on Advanced Language Processing and Web Information Technology, China, pp. 515–519. IEEE Press, Los Alamitos (2007)

A Factorial Experimental Evaluation of Automated Test Input Generation
– Java Platform Testing in Embedded Devices

Per Runeson, Per Heed, and Alexander Westrup

Department of Computer Science, Lund University
Box 118, SE-211 00 Lund, Sweden
http://serg.cs.lth.se/

Abstract. Background. When delivering an embedded product, such as a mobile phone, third party products, like games, are often bundled with it in the form of Java MIDlets. To verify the compatibility between the runtime platform and the MIDlet is a labour-intensive task, if input data should be manually generated for thousands of MIDlets. **Aim.** In order to make the verification more efficient, we investigate four different automated input generation methods which do not require extensive modeling; random, feedback based, with and without a constant startup sequence. **Method.** We evaluate the methods in a factorial design experiment with manual input generation as a reference. One original experiment is run, and a partial replication. **Result.** The results show that the startup sequence gives good code coverage values for the selected MIDlets. The feedback method gives somewhat better code coverage than the random method, but requires real-time code coverage measurements, which decreases the run speed of the tests. **Conclusion** The random method with startup sequence is the best trade-off in the current setting.

1 Motivation

Many embedded devices, like mobile phones, come with a Java execution platform. The platform runs third party applications, like games as the most common type, in the form of Java MIDlets. Some applications are bundled with the embedded device or possible to download from the supplier's web site. Hence it is important for the user's impression of the product quality, that the platform supports running the applications, or at least does not crash the embedded device.

Thousands of applications may be downloaded, and hundreds of versions and variants of the embedded devices are developed. Hence, in order to get a cost efficient verification of the platform's interaction with the applications, an automated approach to input generation must be used, without requiring specific input modeling for each application. Further, the installation and execution of the MIDlets must also be efficient. Typically, there is a certain time frame of hours or days to spend on this particular type of testing, and thus there is a trade-off between how much time should be spent on running each application,

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 217–231, 2011.
© Springer-Verlag Berlin Heidelberg 2011

and how many applications may be run within the given time frame to test the platform. Before starting this study, only heuristics were used in the company under study, when deciding which MIDlets to download and which input sequences to generate. It was a feasible approach when the number of MIDlets was limited, but with the growth of applications available, a systematic and automated method was needed.

We designed four different input generation methods, 1) random selection of input keys, 2) random with a pre-defined startup sequence, 3) feedback based maximizing code coverage, and 4) feedback based with a predefined startup sequence. In order to evaluate the effectiveness of the generated input sequences, we monitored the code coverage of the MIDlets. The input generation methods were evaluated using ten different applications. We designed and executed a factorial design experiment [1] to evaluate the performance of each input generation method.

The paper is organized as follows. In Section 2 we present some related work on Java platform testing and input sequence generation. In Section 3 the technical context of the research is presented and in Section 4 we define the research questions and methods. The results are reported in Section 5 and discussed in Section 6. The paper is concluded in Section 7. More details can be found in a technical report [2].

2 Related Work

One approach to testing *Java platforms* in general, is to design specific MIDlets which stress tests a certain function of a Java platform [3]. This technique is aimed at testing one function at the time whereas the method presented in this paper is focused on testing the entire platform at once. Similarly, there exist benchmark MIDlets, for example JBenchmark [4] that focuses on graphics and gaming. These MIDlets use functions that are performance critical and do time measurements of how long a function or a sequence of functions take to execute.

Automatic *generation of input sequences* for testing was early researched in the software reliability community. Musa presented operational profile testing [5] and others proposed similar approaches of usage based testing [6] or random testing [7]. The principle is to model the intended usage of a system and generate input sequences from that model. Our approach is similar to this, except that we aim for code coverage rather than usage coverage. Recent developments of random testing include *adaptive random testing* [8] which selects among candidate next test cases in order to find a test case which is most different from the previous one. Our feedback based approaches are somewhat related to this, although their approach is based on failure patterns, rather than code coverage.

The area of *model based testing* (MBT) has grown significantly during the last decade, although rarely acknowledging the origins in the reliability community. In MBT, a model is defined using some formalism, and test cases are generated from the model. Models may be external or internal [9] i.e. modeling the system behavior or user behavior, respectively. The approach under investigation in this paper can be seen as a model based approach, where the user behavior model

defines the keys of the device, and the system behavior model is that the Java platform is expected not to crash.

3 Experiment Setup

3.1 Input Generators

We wanted to come up with an effective and efficient input generator that covers as much as possible of the MIDlet, in order to verify the stability of the Java platform. Since the MIDlets are third party software, we had no access the source code, nor to specifications. A typical MIDlet has a startup screen, where the user may select *new game, high scores, settings, help* and *exit*. The game is controlled by the joystick, arrow keys or numeric pad keys, see Figure 1. The MIDlets are fed with input from the keyboard, or from a test driver connected to the embedded software.

We defined four different approaches to automatic input generation and used manual input generation as a reference.

Manual (M). A tester running the applications manually. The tester tries to achieve as high code coverage by exploring as much of the functions of the application as possible.

Random (R). The next key to push during testing is selected randomly from the input key set.

Startup random (SR). The application is first executed with a predefined startup sequence that starts the application. After the startup sequence is done, the input generation is performed as for the random input generator. The startup sequence typically selects *new game* by two down arrow clicks and a confirm key click. The game is then fed by randomly generated key strokes.

Feedback based (F). The input is generated based on feedback from the debug information. First, the input generator selects the next key to push randomly from the current subset. During execution, the generator observes code coverage change and if it is below a threshold, it switches to a different key subset. Key subsets also have a predefined minimum number of key presses before the generator should change key subset to prevent the input generator from changing key subsets too often. A maximum value is also defined to prevent unwanted behavior.

Startup feedback based (SF). This input generator works in the same way as the feedback based input generator but has a predefined startup sequence. The startup sequence is the same as for the startup random input generator.

3.2 Testing Environment

The system under test is the Java execution platform on a mobile phone from Sony Ericsson. It runs in an embedded device, as a part of the embedded software. On top of the Java execution platform, Java MIDlets may be downloaded and executed. The phone has input keys as defined in Figure 1.

Fig. 1. The embedded device and its input keys

For the feedback based method, the keys are grouped into three subsets which have different use in most applications, 1) Joystick keys, 2) Soft keys, and 3) Numeric pad keys. Depending on the characteristics of the key sets, different number of keys are chosen from the sets. From the joystick set, a minimum of 10 keys and maximum of 50 keys are selected; from the soft keys, two keys are selected; and from the numeric pad set, five to 30 keys are selected.

The phone may be connected to a computer with an USB cable; hence the computer may act as the test driver. Then an Attention (AT) [10] connection is established between the program running on the computer and the phone. AT commands can the be sent to the phone, for example "Start MIDlet X" and "Press key 1". Since the processing speed is higher on the computer than on the embedded device, commands may be sent faster from the computer than the phone can process. In this case, commands are put in a queue until the queue overflows. We decided to send one key stroke every five seconds to minimize the risk for queue overflow, which is far from an optimal solution, but sufficient for the evaluation purposes.

To be able to access the phone's file system to, for example, transfer and delete files, an Object Exchange (OBEX) connection is needed [11]. A Java MIDlet is executed on the phone according to the following automated procedure. First the MIDlet is transferred using the OBEX connection. Once the MIDlet is transferred to the phone, it is installed and initiated by sending AT commands to the phone. When the MIDlet is running, input can be given with AT commands simulating key presses, also debug info is sent from the phone to the computer.

This environment allows us to automate the installation and de-installation of applications, entering input sequences to the phone and analyzing the debug information.

In order to evaluate the code coverage of the application during test, which is used both for evaluating the tests, and guiding the feedback based input generation, we needed code coverage measurements. However, since most of the applications came from third party suppliers, only Java bytecode was available.

Table 1. Bytecode injection example. Injected rows are marked with an expression mark (!) at the start.

Original bytecode	Bytecode after injection
`public final example()V`	`public final example()V`
`  FRAME FULL [] []`	`! L0 (0)`
`  ALOAD 0`	`! LINENUMBER 1 L0`
`  GETFIELD h.f : Z`	`  FRAME FULL [] []`
`  IFNE L0`	`  ALOAD 0`
`  RETURN`	`  GETFIELD h.f : Z`
`L0 (6)`	`  IFNE L2`
`  RETURN MAXSTACK = 3`	`! L1 (5)`
`  MAXLOCALS = 3`	`! LINENUMBER 2 L1`
	`  RETURN`
	`  L2 (6)`
	`! LINENUMBER 3 L2`
	`  RETURN MAXSTACK = 3`
	`  MAXLOCALS = 3`

Tools for monitoring code coverage require the Java code be compiled with a debug flag, which enables the insertion of monitoring instructions. Hence we could not use the publicly available tools.

To solve the problem, we extended the test tool to inject bytecode into class files inside a jar file. As we do not know which bytecode lines correspond to a source code line, we measure bytecode coverage rather than source code coverage. When the application is running on the phone, the Java VM sends an event to the test tool at every line counting instruction. In order to not impact on the real-time performance too much, we tried to have as few line counting instructions as possible while still being able to get accurate results. An example of the injection is shown in Table 1.

3.3 Selected Application

Ten applications were selected for our experiments out of the thousands available. Each application out of the thousands is not unique, rather there are different groups of similar kind. We identified ten different categories of games, and selected one representative for each. The selected applications are listed below with a short description.

3D Need For Drift is a car game where the car auto accelerates and the user only has to steer the car.

NHL represents a classic hockey or football game where the user may control all players and pass and shoot the puck or ball.

Pro Golf is a golf game where the user aims with the joystick and then presses a key twice to determine the power of the stroke.

Karpov 2 is a chess game that represents many games under the board game category.

Tetris Pop is a classic Tetris game where the blocks fall automatically and the user only has to rotate and move the blocks.

Indiana Jones is an adventure, platform game where the user moves the player horizontally and make him jump by pressing upwards.

Cooking Mama represents games with many smaller games in it and the controls are different depending on the mini game.

Virtua Fighter is a 3D fighting game where the user steers the player in three dimensions and tries to defeat an opponent by using different attacks.

Prehistoric Tribes represents strategy games where the user controls a lot of units and can move them around and give them orders.

IQ Booster is a sort of quiz game where the user is given a question and different answer alternatives.

4 Research Methods and Question

4.1 Research Question

The problems involved in this study concern how to use third party applications with no access to source code, to test the stability of a runtime platform. The testing should be as effective as possible and test as much of the platform as possible in as little time as possible. Our goal was stated as a research question.

RQ. *How can input sequences be effectively generated to the applications running on the platform?* Is it good enough to give random input or do we need to implement some sort of intelligence, possibly based on feedback from the application?

By effective we mean the trade-off between the six criteria, performance, portability, run speed, cost, scalability, and applicability, as defined in Table 2.

Table 2. Description of the evaluation criteria

Criteria	Description
Performance	Average code coverage value an input generator is able to reach.
Portability	The amount of work needed to make the input generator work with a different embedded device and possibly different types of applications.
Run speed	The input generation method might require additional information from the phone that will decrease the performance of the embedded device and thus reduce the run speed of the application.
Cost	Labor cost to run a test.
Scalability	How the cost scales when running multiple tests at the same time.
Applicability	The type of applications the input generation method can handle.

4.2 Research Methods

First, we piloted execution of the input generators to evaluate the execution time for each of the applications. The applications were executed during 20 minutes, with an input keystroke every 5 seconds. The code coverage grew steadily during the first 10 minutes and then flattened out. To have a safety buffer we decided to set the time for each run in the input generator performance test up to 15 minutes. Note, however, that the application may quit earlier due to a generated key stroke sequence that quits the application.

We designed a factorial design experiment to evaluate the performance of the proposed input generation methods [1]. The factors are four input generation methods and ten applications. The setup was primarily limited by the duration of the test; they were designed to be executed during two weeks on two parallel test environments. First, a full factorial design was executed, and then a fractional replication was executed to resolve identified interactions.

In the first experiment, we executed 20 input sequences on each of the ten applications with each of the four input generation methods, split on the two test environments, which took $20 * 10 * 4 * 15/60/2 = 100$ hours, i.e. four days. Then in the replication, we selected a fraction of input generators (i.e. two of the four) and executed additionally 30 input sequences on each of the ten applications with two input generators, to resolve the interaction between input generators and applications. These tests lasted $30 * 10 * 2 * 15/60/2 = 75$ hours, i.e. three days.

4.3 Threats to Validity

We analyze threats to the validity of the study and report countermeasures taken to reduce them. The definitions follow Wohlin *et al.* [12].

Conclusion validity is concerned with the statistical analyses underpinning the conclusions. The statistical technique of fractional factorial designs is robust. The test environment allows us to generate sufficiently large data sets to get significant results.

Internal validity is about the risk for other factors impacting on the relation between what is manipulated and the outcome. The four input generation methods are well defined and are not biased by any human intervention, and hence the internal validity is good. The manual approach is only used as a reference and is hence not critical.

Construct validity is concerned with the alignment between what is measured and what is the underlying construct. Byte code coverage was used to monitor how well the MIDlets were covered. This is a threat to the construct validity, which is due to the lack of access to 3rd party code. However, since all comparisons are relative, using the same measurement, it is not critical for the study validity.

External validity is related to generalizability of the results. The use of ten different applications adds to the external validity, while the single platform increases the threat to the external validity. Similarly, the applications are only

of game type, which reduces the external validity. If planning to apply the results to another context, a replicated study is recommended to validate the results before launching the approach widely.

5 Results

In this section, we first report the pilot executions, then the descriptive statistics of the main execution, and finally the statistical analysis of the factorial design experiments.

5.1 Pilot Execution

The pilot execution of the input generators lead us to the conclusion that the largest growth in code coverage is achieved during the first 10 minutes. Analysis of the outcome shows the average byte code coverage over time for all applications and input generators. Longer tests were executed as well, but the code coverage did not increase much for the automated input generators, indicating that the limit of 15 minutes for each application run is sufficient.

5.2 Main Execution – Descriptive Statistics

The main execution comprises 20 input sequences on each application for the random and feedback based input generators, 20 plus 30 sequences for the startup random and startup feedback based input generators, which were replicated, and 4 sequences for the manual input generation. Each sequence is 15 minutes long, processing 180 key strokes, one every five seconds.

In Figure 2 the average execution time for each combination of application and input generator is reported. Note that all executions are stopped after 15 minutes, if it not stopped before by a generated key stroke sequence selecting the 'quit' option of the MIDlet.

The code coverage of the same data is shown in Figure 3. The manual approach gives the highest average, still increasing over the whole 15 minute period. Among the automated input generators, the *startup feedback based* is best, followed by *startup random*, *feedback based* and *random*.

It is however worth noticing that these are average figures for all applications together. For individual applications, the results may be different. For example, for the Virtua Fighter game, presented in Figure 4, two of the automated generators have higher coverage than the manual during the initiation, and for this game, both *feedback based* generators give higher code coverage than the *random* approaches.

The average and standard deviation for code coverage of all the runs are tabulated in Tables 3 and 4 respectively. As noted from Figure 3, the manual approach gives highest code coverage on average, and also for all individual applications. However, the variation is large across the applications, from 32 to 75% code coverage. The variation across applications is clearly visible in the standard devisions, which vary across programs. Statistical analyses to find significant patterns among the variations are presented in the next section.

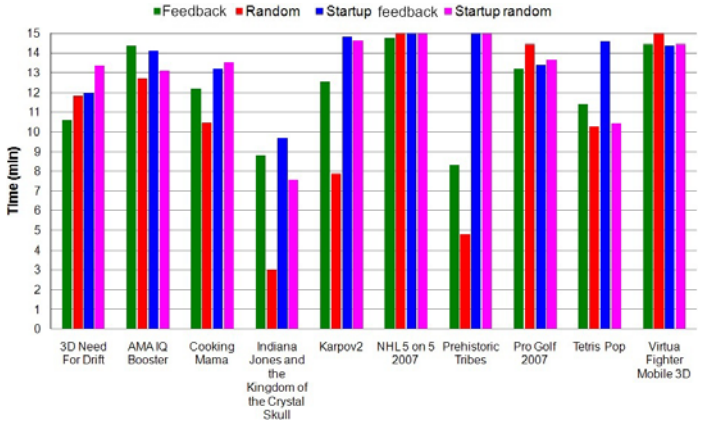


Fig. 2. Average run time for the applications and input generators. The execution is stopped by the test tools after 15 minutes.

5.3 Main Execution – Statistical Analysis

The first round of the experiment comprised 20 input sequences for each combination of application and input generator, summing up to 800 data points. The results of the variance analysis results of the first round are presented in Table 5. Tabulated F statistics for the actual values are: $F_{0.01,3,760} = 3.78$; $F_{0.01,9,760} = 2.41$; $F_{0.01,27,760} = 1.79$.

Since all F_0 values are larger than the values from the F statistics, we conclude that there are significant differences between both factors and their interaction at the level of 0.99. This means that the code coverage values depend on which applications are run, which input generator is used and also the combination of application and input generator. Further calculations have to be made to determine how they impact.

We conducted Tukey's test to identify which of the comparison that are significant. The test statistics for the comparison between all the input generators are presented in Table 6. The test values are the average code coverage of the first input generator for the specific application, subtracted by the average code coverage of the second input generator. If the absolute value of the result is above a threshold, defined by the student T distribution, there is a significant difference. The threshold is $T_{0.05} = 8.38$ for 95% significant results.

In Table 6 four different cases can be observed.

1. For *3D Need For Drift* and *Prehistoric Tribes* the significant difference is between the two input generators with the startup sequence and the two without.
2. For *Indiana Jones* and *Karpov 2* the random input generator is significantly worse than the other three input generators.
3. For *Virtua Fighter Mobile 3D* the two input generators with feedback based behavior are significantly better than the two random.
4. For the rest of the games there is no significant difference between the input generators at all.

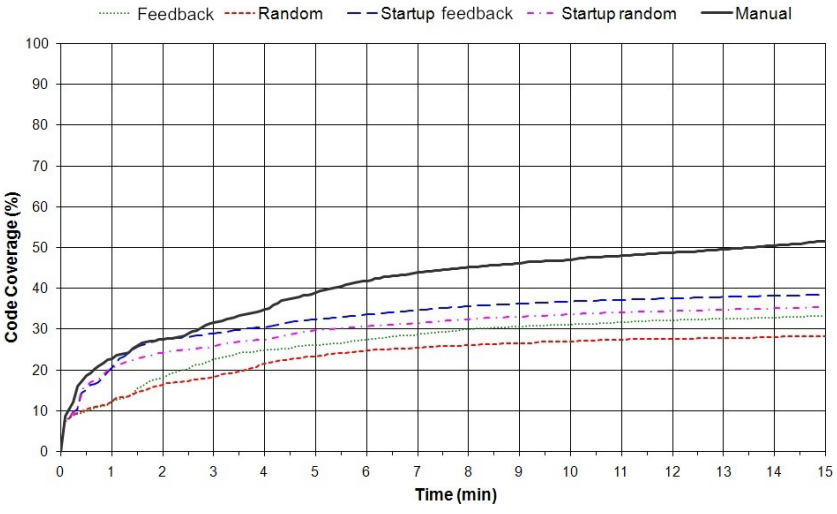


Fig. 3. Average code coverage for input generators for all applications

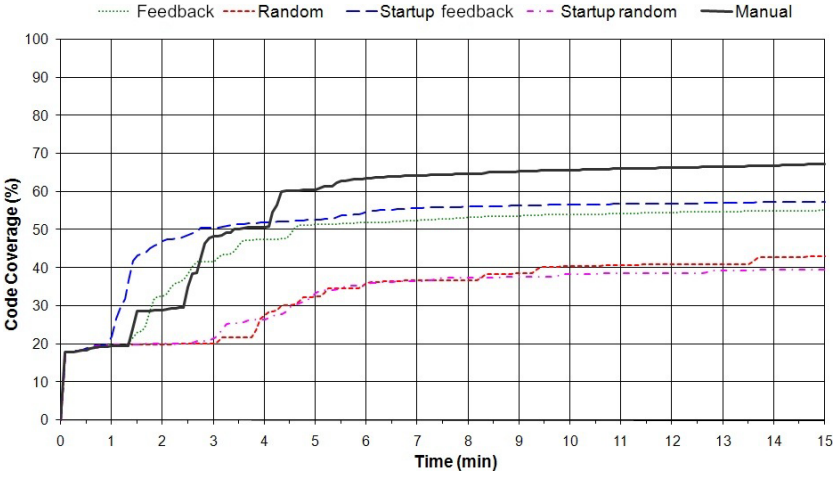


Fig. 4. Average code coverage for input generators for the Virtua Fighter application

Table 3. Average percentage value of code coverage

	Manual	Feedback	Random	Start_feed	Start_rand	All
3D Need for Drift	74.63	42.54	40.97	61.82	62.19	56.91
AMA IQ Booster	71.92	63.34	61.75	65.26	63.73	64.19
Cooking Mama	48.25	18.17	15.60	20.32	19.57	19.49
Indiana Jones	39.88	27.80	15.45	28.04	26.20	26.09
Karpov 2	60.55	37.19	21.51	40.36	39.20	36.97
NHL 5 on 5	33.92	23.56	24.74	29.61	28.47	27.77
Prehistoric Tribes	33.92	14.15	11.77	29.62	29.21	26.07
Pro Golf	46.90	28.43	27.27	27.33	25.57	27.38
Tetris Pop	32.53	21.05	21.92	23.84	20.57	22.25
Virtua Fighter Mobile 3D	67.30	55.34	42.03	57.33	39.55	48.98
All	51.45	33.13	27.15	38.16	34.70	34.99

Table 4. Standard deviation of percentage value of code coverage

	Manual	Feedback	Random	Start_feed	Start_rand	All
3D Need for Drift	2.64	24.83	26.89	3.28	2.36	16.34
AMA IQ Booster	4.29	2.34	2.18	2.21	2.31	2.88
Cooking Mama	3.96	4.20	4.22	1.57	1.66	5.55
Indiana Jones	0.73	7.89	9.79	8.82	7.08	9.33
Karpov 2	2.43	15.02	8.34	5.56	4.96	10.86
NHL 5 on 5	12.26	10.59	11.33	7.56	7.44	8.95
Prehistoric Tribes	2.04	8.89	8.75	1.60	1.58	8.19
Pro Golf	1.88	10.93	9.20	10.10	9.75	10.31
Tetris Pop	1.32	3.90	5.38	3.12	3.92	4.44
Virtua Fighter Mobile 3D	0.69	15.32	19.18	13.02	19.38	18.56
All	15.82	19.50	18.45	17.24	16.51	17.99

To be able to make a better distinction between the two highest performing input generation methods an additional 30 input sequences on each of the ten applications were executed in a fractional replication of the experiment. The results from calculations with these additional runs are presented in Table 7, where we thanks to the fractional factorial design, may use all the 20 plus 30 input sequences in the analysis. F-tables give: $F_{0.01,1,980} = 6.63; F_{0.01,9,980} = 2.41$.

As all F_0 values are larger than the F statistics value, we conclude that there are significant differences between all three factors at the level of 0.99. This means that the code coverage values depend on which applications are run, which input generator is used, and also the combination of application and input generator. Further calculations needs to be made to determine in what way they impact. Since the interaction is significant there is a need to compare the input generators for each application individually.

Table 5. ANOVA for the first experiment: four input generators with startup sequence, ten applications and 20 input sequences on each application

Source of Variation	Sum Squares	of Deg Free	of Mean Square	F_0	P-Value
Application	171224.75	9	19024.97	178.53	<< 0.01
Input generator	11343.01	3	3781.00	35.48	< 0.01
Interaction	14650.14	27	542.60	5.09	< 0.01
Error	80989.61	760	106.57		
Total	278207.51	799			

Table 6. Test statistics for comparison of all input generators over all applications, first round, ten applications and 20 input sequences for each application. Significant differences at 95% level in favor of the first input generator is marked in **bold** and in favor of the second generator is marked in *italics*.

	SF vs SR	SF vs F	SF vs R	SR vs F	SR vs R	F vs R
3D Need for Drift	-0.79	**18.49**	**20.06**	**19.28**	**20.86**	1.57
AMA IQ Booster	1.55	2.44	4.03	0.89	2.48	1.58
Cooking Mama	0.96	2.27	4.79	1.31	3.83	2.52
Indiana Jones	3.07	1.96	**14.55**	-1.12	**11.46**	**12.60**
Karpov 2	1.67	4.14	**18.64**	2.47	**16.98**	**14.51**
NHL 5 on 5	2.37	6.70	6.12	4.33	3.75	-0.58
Prehistoric Tribes	0.47	**15.36**	**18.62**	**14.89**	**18.15**	3.26
Pro Golf	4.57	0.07	0.89	-4.50	-3.68	0.82
Tetris Pop	1.21	2.98	2.81	1.77	1.61	-0.17
Virtua Fighter Mobile 3D	**16.01**	0.32	**12.60**	*-15.68*	-3.40	**12.28**

Table 7. ANOVA for the replication: two input generators with startup sequence, ten applications and 50 input sequences on each application

Source of Variation	Sum Squares	of Deg Free	of Mean Square	F_0	P-Value
Application	234694.25	9	26077.14	477.61	<< 0.01
Input generator	2347.12	1	2347.12	42.99	< 0.01
Interaction	6320.37	9	702.26	12.86	< 0.01
Error	53507.48	980	54.60		
Total	296869.23	999			

Table 8. Test statistics for comparison of two input generators over all applications. Significant differences at 95% level in favor of the first input generator is marked in **bold**.

	SF vs SR
3D Need for Drift	-0.56
AMA IQ Booster	1.70
Cooking Mama	0.76
Indiana Jones	**2.97**
Karpov 2	1.08
NHL 5 on 5	1.00
Prehistoric Tribes	0.36
Pro Golf	2.41
Tetris Pop	**3.14**
Virtua Fighter Mobile 3D	**17.78**

Table 9. Comparison of input generators for the chosen criteria. Ranking position in parentheses

	Manual	Random	Start_rand	Feedback	Start_feed
Performance	Very good	Very poor (4)	Medium (2)	Poor (3)	Good (1)
Portability	Good	Good (1)	Medium (2)	Bad (3)	Very bad (4)
Run speed	Fast	Fast (1)	Fast (1)	Slow (3)	Slow (3)
Cost	Expensive	Cheap (1)	Cheap (1)	Cheap (1)	Cheap (1)
Scalability	Bad	Good (1)	Good (1)	Good (1)	Good (1)
Applicability	Everything	Games (1)	Games (1)	Games (1)	Games (1)

The comparison between the two startup input generators is presented in Table 8. The test values are the average code coverage of the first input generator for the specific application subtracted by the average code coverage of the second input generator. If the absolute value of the result is above a threshold, defined by the student T distribution, there is a significant difference. The threshold is $T_{0.05} = 2.89$ for 95%.

In Table 8 it can be seen that the startup feedback (SF) performs better in three games and that there are no significant difference in the other seven. This means that the startup feedback input generator is significantly better than startup random (SR) on two more games compared to the leftmost data column of Table 6.

6 Discussion

To answer the research question in Section 4.1 we discuss the results related the criteria presented in Table 2. From the analysis, it is concluded that there are significant *performance* differences between input generators. The manual approach gives the best performance by far, however the problem with manual is, as stated before, that it is very costly and does not scale up.

An important issue is the reduction of *run speed* when running in debug mode to get code coverage calculations. The manual and random generation methods do not require debug mode execution, while the feedback based input generation requires code coverage calculations. In our case the debug mode reduced the performance significantly, but this might very well differ depending on the specific implementation and amount of information gathered in the debug mode.

For the automated input generators we notice that the startup sequence helps providing good *performance* (code coverage) values. We used the same startup sequence for all applications and it worked well in most cases. However, in one application the soft key usage was switched compared to all other games, which for our startup sequence meant that the game was terminated directly. This indicates that the startup sequence must be checked manually for each of the applications under test.

When comparing the two input generators with startup sequence to those without, it is clear that the feedback based approach gives slightly better *performance*, however at the cost of reduced *run speed* due to the use of debug mode execution.

The *portability* is best for the fully random input generator, since porting only means redefining the key set. For the generators with startup sequence, it has to be redefined when porting to a new device, and at least checked for each application. When you move on to the feedback based input generation you also need to define the key subsets, as well as implementing the feedback connection from the code coverage monitor.

Table 9 presents all the pros and cons with each input generation technique. For each criterion (see Table 2) the ranking position for each input generator is presented in parenthesis, manual has no position as it is only included as reference. Our recommendation in this case is to use the startup random input generator. It performs well for most applications tested and does not require the applications to be run in debug mode.

7 Conclusions and Further Work

The presented study aims at investigating automated input generation strategies to test a Java platform in an embedded device through executing MIDlets on it. We defined four input generation approaches and conclude that for most of the tested applications, it is important to make sure the startup sequence is critical to get high code coverage. However, for the continued input modeling the random generation is feasible for the MIDlet under study.

Depending on the performance of the execution system, the random or the feedback based method may be most efficient. The feedback based method requires the embedded device be executing in debug mode and if the performance of the device drops a lot in debug mode, it is probably not worth running the feedback based approach but the less computational random method.

Future work includes replicating the study with applications with access to the Java code. This would enable better code coverage analysis, which might

improve the adaptivity of the input generation as well as the test monitoring. Further, other types of applications than games, as well as different embedded platforms, should be investigated in improve the external validity of the results.

Acknowledgment

The authors would like to thank Mr Erik André at Sony Ericsson for guidance and cooperation. The work is partly funded by the Swedish Research Council under grant 622-2004-552 for a senior researcher position in software engineering.

References

1. Montgomery, D.C.: Design and Analysis of Experiments, 5th edn. Wiley, New York (2001)
2. Heed, P., Westrup, A.: Automated platform testing using input generation and code coverage. Technical report, Dept. of Computer Science, Lund University (2009)
3. Mazlan, M.A.: Stress test on J2ME compatible mobile device. Innovations in Information Technology, 1–5 (2006)
4. Jbenchmark, http://www.jbenchmark.com/index.jsp
5. Musa, J.D.: Operational profiles in software reliability engineering. IEEE Software, 14–32 (1993)
6. Wohlin, C., Runeson, P.: Certification of software components. IEEE Transactions on Software Engineering 20(6), 494–499 (1994)
7. Hamlet, R.: Random testing. In: Encyclopedia of Software Engineering, pp. 970–978. Wiley, Chichester (1994)
8. Chen, T.Y., Kuo, F.-.C., Merkel, R.G., Tse, T.H.: Adaptive random testing: The ART of test case diversity. Journal of Systems and Software 81(1), 60–66 (2010)
9. Malik, Q., Jaaskelainen, A., Virtanen, H., Katara, M., Abbors, F., Truscan, D., Lilius, J.: Model-based testing using system vs. test models – what is the difference? In: 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems, pp. 291–299 (2010)
10. Sony Ericsson (AT commands online reference), http://developer.sonyericsson.com
11. Infrared Data Association, IrDA: (Object exchange protocol (IrOBEX). ver. 1.4, http://www.irda.org
12. Wohlin, C., Höst, M., Ohlsson, M.C., Regnell, B., Runeson, P., Wesslén, A.: Experimentation in Software Engineering: An Introduction. International Series in Software Engineering. Kluwer Academic Publishers, Dordrecht (2000)

Automating and Evaluating Probabilistic Cause-Effect Diagrams to Improve Defect Causal Analysis

Marcos Kalinowski[1,2], Emilia Mendes[1,3], and Guilherme H. Travassos[1]

[1] COPPE/UFRJ – Federal University of Rio de Janeiro,
68511 Rio de Janeiro, Brazil
[2] UVA-RJ – Veiga de Almeida University,
68507 Rio de Janeiro, Brazil
[3] Computer Science Department – The University of Auckland,
92019 Auckland, New Zealand
mkali@cos.ufrj.br, emilia@cs.auckland.ac.nz,
card@computer.org, ght@cos.ufrj.br

Abstract. Defect causal analysis (DCA) has shown itself an efficient means to obtain product-focused software process improvement. A DCA approach, called DPPI, was assembled based on guidance acquired through systematic reviews and feedback from experts in the field. To our knowledge, DPPI represents an innovative approach integrating cause-effect learning mechanisms (Bayesian networks) into DCA meetings, by using probabilistic cause-effect diagrams. The experience of applying DPPI to a real Web-based software project showed its feasibility and provided insights into the requirements for tool support. Moreover, it was possible to observe that DPPI's Bayesian diagnostic inference predicted the main defect causes efficiently, motivating further investigation. This paper describes (i) the framework built to support the application of DPPI and automate the generation of the probabilistic cause-effect diagrams, and (ii) the results of an experimental study aiming at investigating the benefits of using DPPI's probabilistic cause-effect diagrams during DCA meetings.

Keywords: Bayesian Networks, Defect Causal Analysis, Defect Prevention, Defect Prevention-based Process Improvement, DPPI, Probabilistic Cause-Effect Diagrams, Product Focused Process Improvement.

1 Introduction

Causal analysis and resolution encompasses the identification of causes of defects and other problems, and ways to prevent them from recurring in the future. Many popular process improvement approaches (e.g., Six Sigma, CMMI, and Lean) incorporate causal analysis activities.

Defect causal analysis (DCA) [1] represents the application of causal analysis and resolution to a specific type of problem: defects introduced in software artifacts throughout a software lifecycle. Thus, DCA can be seen as a process to discover and analyze causes associated with the occurrence of specific defect types, allowing the identification of improvement opportunities for the organization's process assets, and

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 232–246, 2011.
© Springer-Verlag Berlin Heidelberg 2011

the implementation of actions to prevent the recurrence of those defect types in future projects. Effective DCA has helped to reduce defect rates by over 50% in organizations such as IBM [2], Computer Science Corporation [3], and InfoSys [4].

However, despite its benefits and industrial adoption, there are still numerous unanswered questions concerning DCA implementation in software organizations, in addition to a small number of related publications [5]. Thus, in practice, the causal analysis process and techniques are often taught to staff, but little guidance is provided on how to apply them. Consequently, the application of causal analysis becomes *ad-hoc* [5]. Therefore, in order to provide guidance on how to efficiently implement DCA in software organizations, a systematic review (SR) was conducted in 2006 and replicated in 2007, 2009, and 2010. The results allowed producing and updating guidance on how to implement DCA in software organizations and the identification of opportunities for further investigation [6]. For instance, "the DCA state of the art did not seem to include any approach integrating learning mechanisms regarding cause-effect relations into DCA meetings". This means that, in the approaches found, the knowledge on cause-effect relationships gathered during each DCA session was only used to initiate actions to improve the process, and afterwards discarded.

To our knowledge, the first effort to bridge this gap is reported in the initial concept of a DCA approach described in [7]. In this initial concept, the integration of knowledge gathered in successive causal analysis events as means to assemble a deeper understanding of the defects' cause-effect relations by using Bayesian networks is suggested. This integration aims at facilitating the creation and maintenance of common causal models to be used to support defect cause identification in each DCA event. Such causal models can help answering questions during DCA meetings, such as: "Given the past projects within my organizational context, with which probability does a certain cause lead to a specific defect type?". Additionally, in order to allow the usage of the Bayesian diagnostic inferences during DCA meetings, the traditional cause-effect diagram [8] was extended into a probabilistic cause effect diagram [7].

Later this initial concept was evolved and tailored into the DPPI (Defect Prevention-Based Process Improvement) approach [9]. Besides using and feeding Bayesian networks to support DCA meetings with probabilistic cause-effect diagrams, DPPI addresses all the specific practices of the CMMI CAR (Causal Analysis and Resolution) process area, described in [10]. The experience of applying DPPI to a real Web-based software project indicated its usage feasibility and provided insights into its required tool support [9]. Moreover, during this experience DPPI's Bayesian diagnostic inference predicted the main defect causes efficiently, motivating further investigation.

Given this context, in this paper we extend our research by describing the computational framework built to support the application of DPPI and the results of an experimental study conducted to evaluate DPPI's main innovation, using probabilistic cause-effect diagrams during DCA meetings. The remainder of this paper is organized as follows. In Section 2, an overview of DPPI is provided with examples of how its activities could be performed in a real software project. In Section 3, the computational framework built to support the application of DPPI is described. In Section 4, the design and results of the experimental study are presented. Final considerations are given in Section 5.

2 DPPI: Defect Prevention-Based Process Improvement

DPPI represents a practical approach for defect prevention. Its main innovation is the integration of knowledge gathered in successive causal analysis events in order to provide a deeper understanding of the organization's defect cause-effect relations. Such integration allows establishing and maintaining common causal models (Bayesian networks) to support the identification of causes in each causal analysis event. Those causal models support diagnostic reasoning, helping to understand, for similar projects of the organization, which causes usually lead to which defect types.

Additionally, DPPI follows the guidance for implementing DCA in software organizations [7] in order to tailor the defect prevention activities into specific tasks, providing further details on the techniques to be used to accomplish these tasks efficiently. Moreover, it integrates defect prevention into the measurement and control strategy of the development activity for which defect prevention is being applied, allowing one to observe whether the implemented improvements to the development activity brought real benefits. This is done by defining defect-related metrics to be used to measure and control the development activity.

Given that DPPI aims at continuous improvement, it was designed to take place right after the inspection of each main development activity artifacts. Note that the inspection process expects the correction of defects by the author [11]. Thus, when the DPPI is initiated the defects corresponding to the development activity have already been corrected.

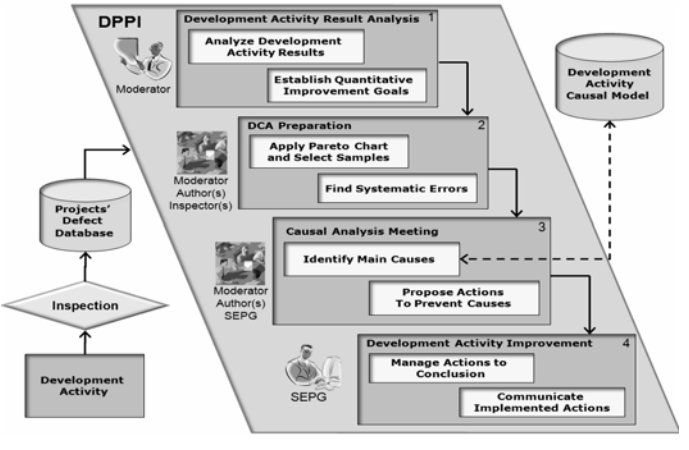


Fig. 1. DPPI Approach Overview

DPPI includes four activities: (i) Development Activity Result Analysis; (ii) DCA Preparation; (iii) DCA Meeting; and (iv) Development Activity Improvement. The main tasks for each of these activities, as well as the roles involved in their execution, are depicted in Fig. 1. Note that the software development activity itself and its inspection are out of DPPI's scope.

Fig. 1 also shows the development activity's causal model. DPPI considers those causal models to be dynamically established and maintained by feeding and using Bayesian networks. Pearl [12] suggests that probabilistic causal models can be built using Bayesian networks, if concrete examples of cause-effect relations can be gathered in order to feed the network. In the case of DPPI the examples to feed the Bayesian network can be taken from each DCA meeting results.

A brief description of the four DPPI activities and their tasks, with examples of how they could be performed in a real software project, is provided in the following subsections. Further details can be found in [9].

2.1 Development Activity Result Analysis

The Development Activity Result Analysis aims at quantitative measurement and control of the development activity. It comprises two tasks to be performed by the DPPI moderator. Further details on these tasks follow.

Analyze Development Activity Results. This task aims at analyzing the development activity's defect-related results by comparing them against historical defect-related results for the same development activity in similar projects. Therefore, the number of defects per unit of size and per inspection hour should be analyzed against historical data using a statistical process control chart. As suggested by [13], the type of the statistical process control chart for those metrics should be a U-chart, given that defects follow a Poisson distribution. Those charts can indicate if the defect metrics of the development activity are under control by applying basic statistical tests. An example of a real project U-chart for defects per inspection hour is shown in Fig. 2.

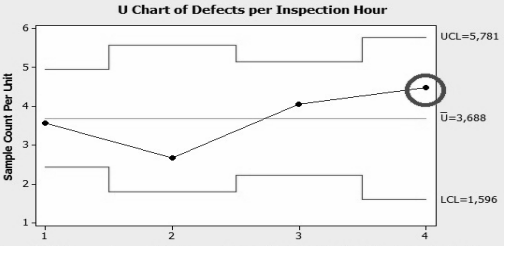


Fig. 2. Defects per Inspection Hour U-Chart

Establish Quantitative Improvement Goals. This task involves establishing improvement goals for the development activity. A typical example of quantitative improvement goal is: "reducing the defect rate per inspection hour (or per unit of size) by X percent".

If the development activity is out of control the focus of the causal analysis meeting becomes revealing assignable causes and the improvement goal should be related to stabilizing its behavior. If it is under control the focus is on finding the common causes and the improvement goal should be improving its performance and capability.

2.2 DCA Preparation

This activity comprises the preparation for defect causal analysis by selecting the samples of defects to be analyzed and identifying the systematic errors leading to several of those defects.

Apply Pareto Chart and Select Samples. This task refers to finding the clusters of defects where systematic errors are more likely present. Systematic errors lead to defects of the same type. Thus, Pareto chart can be used to find those clusters, by using the defect categories as the discriminating parameter. In DPPI the samples should be related to the defect categories that contain most of the defects. An example of a real project Pareto chart is shown in Fig. 3. It shows that most defects were of type incorrect fact, and that the sum of incorrect facts and omissions represents about 60% of all defects found.

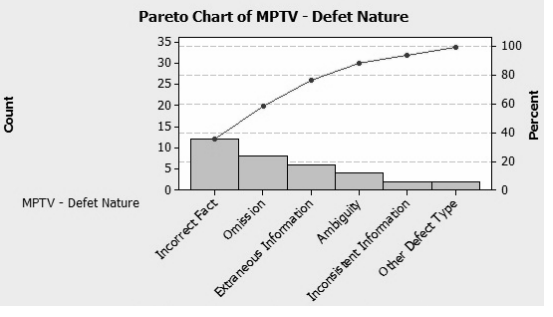


Fig. 3. Defect Pareto Chart

Find Systematic Errors. This task comprises analyzing the defect sample (reading the description of the sampled defects) in order to find its systematic errors. Only the defects related to those systematic errors should be considered in the DCA meeting. At this point the moderator could receive support from representatives of the document authors and the inspectors involved in finding the defects.

2.3 DCA Meeting

In this activity the moderator is supported by mandatory representatives of the authors and of the software engineering process group (SEPG). A description of the two tasks involved in this activity follows.

Identifying Main Causes. This task represents the core of the DPPI approach. Given the causal model elaborated based on prior DCA meetings for the same development activity considering similar projects (by feeding the Bayesian network with the identified causes leading to the defect type related to the analyzed systematic error), the probabilities for causes to lead to the defect types related to the systematic errors being analyzed can be calculated (using the Bayesian diagnostic inference).

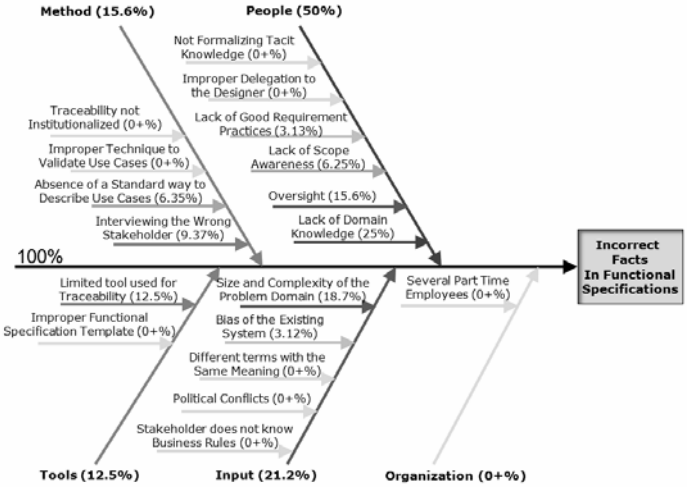


Fig. 4. DPPI's Probabilistic Cause-Effect Diagram

Afterwards, those probabilities can be used to support the DCA meeting team in identifying the main causes. Therefore, probabilistic cause-effect diagrams for the type defects related to the systematic errors being analyzed can be used. The probabilistic cause-effect diagram was proposed in [7] and extends the traditional cause-effect diagram by (i) showing the probabilities for each possible cause to lead to the analyzed defect type, and (ii) representing the causes using grey tones, where the darker tones are used for the causes with higher probability. This representation can be easily interpreted by causal analysis teams and highlights the causes with greater probabilities of causing the analyzed defect type.

Fig. 4 shows the probabilistic cause-effect diagram for defects of type "incorrect fact" for a real project. In this diagram it can be seen that, in this project, typically the causes for incorrect facts are "lack of domain knowledge" (25%), "size and complexity of the problem" (18.7%), and "oversight" (15.6%). Fig. 5 shows the underlying Bayesian network and its diagnostic inference for defects of type "incorrect fact".

We believe that showing such probabilistic cause-effect diagrams could help to effectively use the cause-effect knowledge stored in the Bayesian network to support the identification of causes during DCA meetings. In the particular experience described in [9], the identified causes for the analyzed systematic errors were "lack of domain knowledge", "size and complexity of the problem", and "oversight". Those causes correspond to the three main causes of the probabilistic cause-effect diagram and were identified by reading the defect descriptions and involving the author, inspectors, and SEPG members. Thus, the probabilistic cause-effect diagram showed itself helpful in this case, motivating further investigation.

According to DPPI, at the meeting end the Bayesian network should be fed with the resulting causes for the defect type, so that the probabilities of the causes can be dynamically updated, closing a feedback cycle for the next DCA event based on the consensus result of the team on the current DCA event. The framework that allows

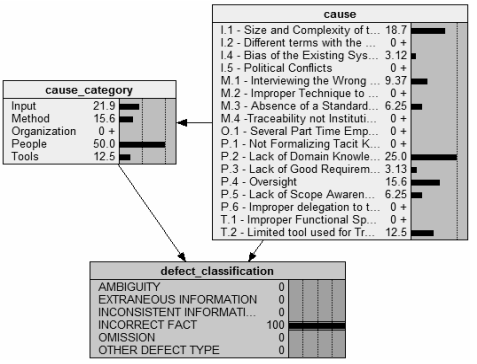


Fig. 5. Bayesian network inference for incorrect facts

automating this dynamic feedback cycle is a new contribution described in Section 3 of this paper. The further investigation on the use of the resulting probabilistic cause-effect diagrams during DCA meetings is described in Section 4.

Propose Actions to Prevent Causes. In this task, actions should be identified to improve the process assets in order to prevent the identified causes. Examples of actions proposed during a real project, addressing the causes "lack of domain knowledge" and "size and complexity of the problem", are: (i) studying the domain and the pre-existing system; and (ii) modifying the functional specification template by creating a separate session for the business rules, listing them all together.

2.4 Development Activity Improvement

Finally, the action proposals should be implemented by a dedicated team and managed until their conclusion. After implementation, the changes to the process should be communicated to the development team. The conclusion of the actions and the effort of implementing them should be recorded.

The next section presents the computational framework built to support the application of DPPI in industrial contexts.

3 DPPI Framework

Institutionalizing DPPI and keeping all the related documentation (e.g., improvement results and goals, main defect categories, systematic errors, identified causes, and action proposals) without any tool support, has shown itself a complex task during the experience of applying it to a real Web-based project [9].

Therefore, the purpose of building a computational framework supporting the application of DPPI was to facilitate its usage by the industry providing a step by step support that embeds DPPI's suggested DCA practices. Additionally, the resulting framework should automate the proposed DCA feedback cycle, allowing establishing and maintaining the Bayesian network's learning cases and generating the

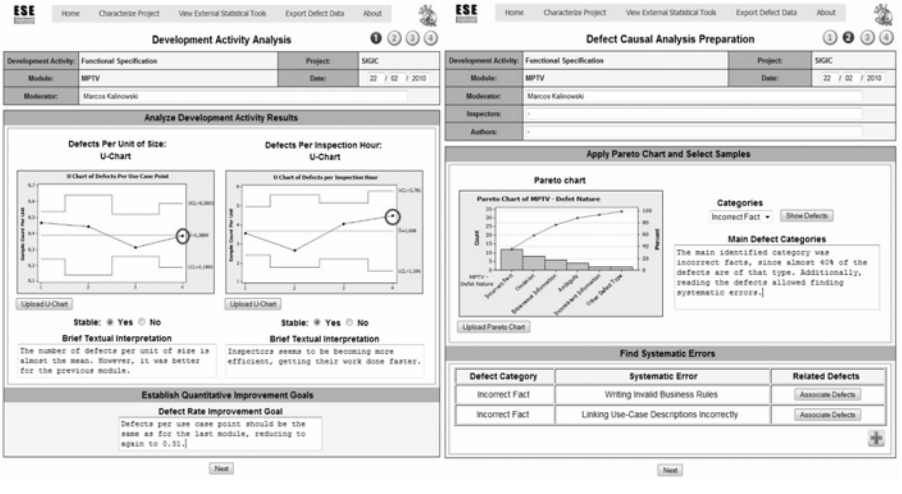
probabilistic cause-effect diagram. The support provided by the framework to each DPPI activity is described in detail hereafter. The data shown in the screens is the data of the experience described in [9], when no tool support was available. This data was used to enact the framework's usage as an informal proof of concept.

Considering the framework's input, defect data of the current project/module can be imported. Therefore it has to follow a specific XML schema format, which is the same for importing defect data as in the ISPIS inspection support framework [14]. To transform other XML formats into the required one the XMapper integration tool described in [15] may be used, avoiding manual XML transformation. Once the project is characterized and the defects where imported DPPI can be launched.

The screen for supporting DPPI's first activity, Development Activity Analysis, is shown in Fig. 6 (a). In it, the moderator analyzes the U-charts defined in DPPI (defects per unit of size and defects per inspection hour). The framework allows exporting defect data so that the U-charts can be generated by using one of the registered external statistical tools and the resulting chart uploaded as an image to be analyzed. After analyzing the charts the moderator records the improvement goal.

The next DPPI activity is the DCA meeting. In it each systematic error's main causes should be identified. Therefore, the framework allows reading the associated defects and generates a probabilistic cause-effect diagram for the defect type to which the systematic error is related. After identifying the causes, action proposals to address them can be recorded. The DCA Meeting screen is shown in Fig. 7.

To enable the dynamic generation of the probabilistic cause-effect diagrams, the WEKA java API [16] is used internally to handle the Bayesian network learning and inference. The learning cases are the causes related to defect types, according to results of prior DCA meetings conducted for the same development activity on similar



(a) (b)

Fig. 6. Support for the Development Activity Analysis (a) and DCA Preparation (b) activities

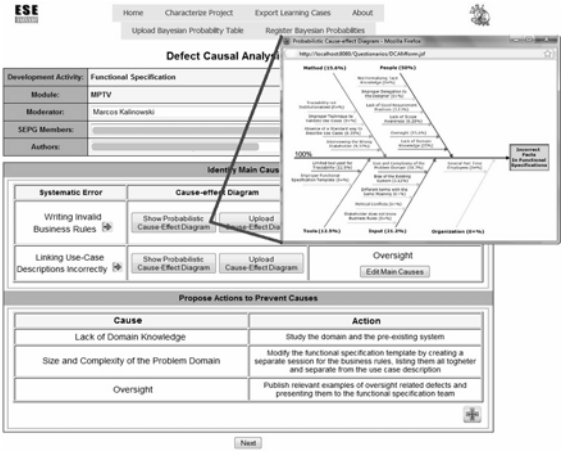


Fig. 7. DCA Meeting Support

projects. This activity allows the framework to automate DPPI's proposed dynamic feedback cycle, since, at the end of the meeting the newly recorded causes automatically become learning cases (one learning case for each defect associated to the analyzed systematic error) for the next DCA meetings concerning projects with similar characterization.

Finally, the development activity improvement screen (Fig. 8) allows managing the proposed actions until their conclusion and registering how the communication of changes to the project team happened. Optionally, the actions and their estimated due dates may be exported as a project data interchange XML to be managed using external project management tools. At the end a report on the complete DPPI session may be generated.

Fig. 8. Development Activity Improvement Support

The framework closes DPPI's proposed automated learning case feedback cycle, allowing generating probabilistic cause-effect diagrams (reflecting Bayesian diagnostic inferences). However, the benefits of using such diagrams during DCA meetings had not been evaluated objectively; although it showed promising results during the experience of applying DPPI manually to a specific software project [9]. The experimental study described hereafter addresses this evaluation.

4 Experimental Study

4.1 Experimental Study Design

Three hypotheses where stated in the experimental study plan to evaluate benefits (regarding effectiveness, effort and usability) of using the probabilistic cause-effect diagrams (DPPI's approach) to identify causes of defects:

- **H1:** The use of DPPI's approach to identify causes of defects results in more effective cause identification, when compared to *ad-hoc* cause identification.
- **H2:** The use of DPPI's approach to identify causes of defects reduces the effort of cause identification, when compared to *ad-hoc* cause identification.
- **H3:** The use of DPPI's approach to identify causes of defects improves user satisfaction in identifying causes, when compared to *ad-hoc* cause identification.

The experimental study design had to handle several constraints, since it was planned to happen in the context of a real software project and its real development team. The decision to conduct the experimental study in a real context was to eliminate confounding factors and bias regarding defect causes that could easily happen in the context of toy problems with students.

The scope of this project was to develop a new information system to manage the activities of a Foundation that manages research and development initiatives. The project involved several departments, such as: human resources, financial, accountability, protocol, project monitoring, among others. The system to be implemented was modularized and developed in an iterative and incremental lifecycle, allowing the gradual substitution of the existing information system. Software inspections were performed on each of the modules' functional specifications, using the ISPIS framework [14]. By the end of the project, all inspection data were available, including details on the defects found and removed from the functional specifications before the actual implementation.

Given this context, the experimental design used one factor (identifying causes of defects), two treatments (using DPPI's approach and using an *ad-hoc* approach), and two objects (the defects of two functional specification activities performed on two consecutive modules of the project).

To instrument the experimental study DPPI's DCA preparation activity was performed by the researchers, plotting the Pareto chart in order to identify the main defect categories and identifying the systematic errors and their related defects. For each module 5 systematic errors were identified. At all, the instrumentation encompasses a consent form, a subject characterization form, the task, and a follow-up questionnaire.

The task was to identify up to three main causes for the systematic errors, given the set of related defects. All subjects participated in the inspection meetings in which the set of defects was discussed throughout the project. Thus, they were familiar with the modules (for effective DCA they should be [2]). On the other hand, the constraint of using subjects familiar with the modules limited the amount of possible subjects to 5.

Another constraint of performing the study in the context of a real development project was that the researchers could not know the correct causes in advance. This issue was addressed by using one of the 5 possible subjects to, for each systematic error, based on the set of causes identified by the others, select the up to three most relevant ones. For this role the most experienced subject, which was also the inspection moderator of those modules, was chosen. Being the inspection moderator, he was very familiar to those defects, since during the inspection meetings he was responsible for leading the discussion of those defects that led to their categorization according to the defect natures described in [17]. When choosing the most relevant causes for each systematic error he did not know that the provided set of causes was assembled by joining all the different causes cited by the other 4 subjects. Moreover, to avoid any bias, his choices were reviewed by the CTO responsible for the project, which participated in weekly status report meetings, where the project's main issues, risks and schedule were discussed.

Hence, only 4 subjects were available to conduct the experimental study. But how could it be possible to observe something relevant with only 4 subjects? Thus, we studied possible arrangements of those subjects to lower the impact of this threat to validity. The resulting arrangement for the study design is shown in Fig. 9. Each subject accomplished the task of identifying the main causes for the sets of defects in three separate experimental study trials. The first two trials were consecutive, the third happened three months later. The observation scenarios are depicted by the arrows.

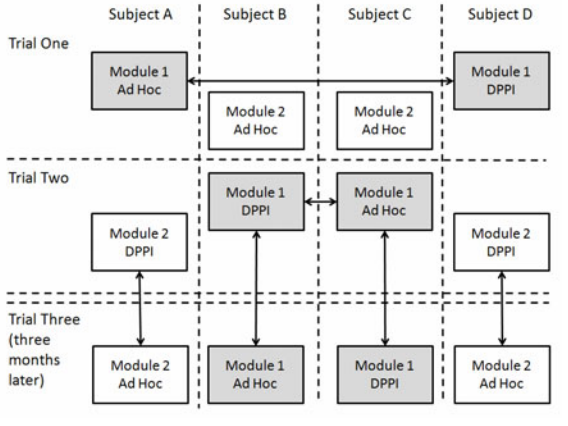


Fig. 9. Experimental Study Design

In the first trial, the performance of subject A was compared with the performance of subject D, since both applied different treatments on the same object, not being, at this moment familiar to the task to be performed. In the second trial, the performance

of subject B was compared to the performance of subject C, again both used different treatments, but this time they were already aware of the task to be accomplished, since they performed it *ad-hoc* in the first trial on another module.

Thus, joining the two trials, the confounding factor "training" (trial 1 without training and trial 2 with training) and "individual ability to identify causes" (trials 1 and 2 used different subjects) would be someway isolated. However, given the limited amount of subjects, the results would still be strongly dependent on the subjects and those trial's results would only allow an initial understanding of possible effects of using one treatment instead of the other one.

Therefore, the plan involved a third trial, three months after the two initial ones, changing the treatment for the same subject acting on the same object. The aim was to afterwards compare the results of the subject's in trials 2 and 3, using a *crossed design* [18].

The *crossed design* took place between trials 2 and 3 on module 1, where one subject B applied first the *ad-hoc* treatment and subject C applied first the DPPI treatment. In this way, the confounding factor "using one treatment before the other" was treated. Still regarding this confounding factor, the pause of three months between trial two and three was to assure that short term memory could not be used by the subjects to relate the main causes to each of the five systematic errors to be analyzed. Therefore, the subjects were interviewed to assure that they did not remember the main causes for the systematic errors. All of them mentioned not to remember, but even if they did, for the analysis of the results for module 1 this factor was isolated by the crossed design. For the analysis of the results for module 2, however, the short term memory could eventually benefit the ad hoc treatment.

During the operation phase of each experimental study's trial, subjects were asked to fulfill the consent and characterization forms and afterwards to identify up to three main causes for each systematic error, by reading the related defects. One of the researchers monitored them while performing the task. At the end the follow-up questionnaire had to be fulfilled to gather the effort (time in minutes) and qualitative data on their satisfaction in performing the task.

Participants of the *ad-hoc* treatment received the 5 systematic errors and the related defects, and the cause categories suggested by the guidance (input, method, people, tools, and organization) [6]. Participants of the DPPI treatment instead of the cause categories received the probabilistic cause-effect diagrams for the types of defects being analyzed. Those cause-effect diagrams were built by the researchers associating 197 defects of 4 prior modules to causes (possible causes were brainstormed with the project team before this association to facilitate this task), building the Bayesian network and performing the diagnostic inference for the analyzed defect types. Participants of both treatments were encouraged to list any causes they considered relevant. The obtained results are described in the following subsection.

4.2 Experimental Study Results

The quantitative results of the three trials for each subject are shown in Table 1. This table shows the effectiveness (percentage of main causes found) for each systematic error, the average effectiveness and the effort (time spent in minutes to identify the causes).

Table 1. Quantitative results of the three trials for each subject

Subj.	Trial	Module	Treat.	Effectiveness						Effort (min.)
				1.1	1.2	1.3	1.4	1.5	Avg.	
A	1	1	*ad-hoc*	33.3%	0%	33.3%	66.7%	33.3%	33.3%	45
A	2	2	DPPI	66.7%	66.7%	66.7%	33.3%	66.7%	60.0%	20
A	3	2	*ad-hoc*	66.7%	33.3%	33.3%	0.0%	33.3%	33.3%	27
B	1	2	*ad-hoc*	66.7%	33.3%	0%	0%	0%	20%	95
B	2	1	DPPI	33.3%	33.3%	66.7%	33.3%	0%	33.3%	30
B	3	1	*ad-hoc*	33.3%	33.3%	33.3%	0%	0%	20.0%	75
C	1	2	*ad-hoc*	33.3%	0%	33.3%	66.7%	66.7%	40.0%	50
C	2	1	*ad-hoc*	33.3%	33.3%	33.3%	0%	66.7%	33.3%	45
C	3	1	DPPI	33.3%	66.7%	66.7%	33.3%	33.3%	46.7%	30
D	1	1	DPPI	66.7%	66.7%	66.7%	33.3%	0%	46.7%	30
D	2	2	DPPI	66.7%	33.3%	66.7%	0%	33.3%	40.0%	20
D	3	2	*ad-hoc*	0%	33.3%	33.3%	0%	33.3%	20.0%	25

Some observations concerning the hypotheses and the experimental study results could be made:

- **H1:** The use of DPPI's approach to identify causes of defects results in more effective cause identification, when compared to *ad-hoc* cause identification.
  - Trial 1. Comparing the two treatments for Module 1 the effectiveness of the subject that used the DPPI treatment was 40% higher.
  - Trial 2. Comparing the two treatments for Module 1, the effectiveness of the subjects was the same.
  - Trial 3. Comparing trials 2 and 3 it is possible to observe that all subjects were more effective when applying the DPPI treatment. This result supports the alternative hypothesis H1. For module 1, subjects B and C had higher effectiveness with the DPPI treatment (66.7% and 40% higher, respectively). For module 2, subjects A and D also had higher effectiveness with the DPPI treatment (80% and 100% higher, respectively).
- **H2:** The use of DPPI's approach to identify causes of defects reduces the effort of cause identification, when compared to *ad-hoc* cause identification.
  - Trial 1. Comparing the two treatments for Module 1 the effort of the subject that used the DPPI treatment was 33.3% lower.
  - Trial 2. Comparing the two treatments for Module 1, again the effort of the subject that used the DPPI treatment was 33.3% lower.
  - Trial 3. Comparing trials 2 and 3 it is possible to observe that all subjects were required less effort when applying the DPPI treatment. This result supports the alternative hypothesis H2. For module 1, subjects B and C had required less effort to perform their task with the DPPI treatment (60% and 33.3% less, respectively). For module 2, subjects A and D also required less effort with the DPPI treatment (25.9% and 20% less, respectively).
- **H3:** The use of DPPI's approach to identify causes of defects improves user satisfaction in identifying causes, when compared to *ad-hoc* cause identification.

- Based on the qualitative results obtained from the follow up questionnaires, nothing could be observed regarding this hypothesis. In fact, all subjects kept or increased their user satisfaction degree in performing the task throughout the three trials, independent of the adopted treatment.

An interesting analysis is combining the effort and effectiveness results to allow observing that on average the DPPI treatment required less effort to produce more accurate results. Even though the results suggest a significant improvement for the DPPI treatment, we would like to state that the study's design doesn't allow any external validity inferences, since it was performed in the context of one specific Web-based software project with its real context. Actually, because of the limited amount of subjects there is also no conclusion validity, since no significant statistical tests could be applied. However, the arrangement of the study allowed us to provide several arguments that could serve as a preliminary indication of benefits of using DPPI's probabilistic cause-effect diagrams to support DCA meetings.

5 Conclusions

DCA is an effective software process improvement practice, as shown in success cases reported in several industrial environments [2][3][4]. DPPI [9] represents an approach for conducting, measuring and controlling DCA in order to use it efficiently, assembled based on unbiased DCA guidance obtained from SRs [6] and on feedback gathered from experts in the field. According to the results of our SRs, and to our knowledge, it represents the only approach that integrates cause-effect learning mechanisms (Bayesian networks) into DCA meetings. In order to facilitate the use of such Bayesian network to support DCA meetings a graphical representation, called probabilistic cause-effect diagram, was designed [7]. The experience of applying DPPI manually to a real Web-based software project indicated its feasibility and provided insights into the required tool support [9]. During this experience DPPI's probabilistic cause-effect diagram supported the identification of the main defect causes efficiently, motivating further investigation.

In this paper we extended this research by presenting the computational framework built to support DPPI's systematic application. This framework automates the proposed DCA feedback cycle, by establishing and maintaining the Bayesian learning cases (relations of causes to defect types, obtained as a result of each DCA event), without requiring additional effort, and generating the probabilistic cause-effect diagram based on the Bayesian diagnostic inference results for a given defect type.

Additionally, the benefits of using such probabilistic cause-effect diagrams during DCA meetings to support the identification of the causes were evaluated through an experimental study conducted on a real problem with software professionals. The preliminary results indicated benefits regarding both, effectiveness and effort. Moreover, the study's design reduced the effect of confounding factors such as the human influence and the learning bias. However, we are aware that the specific context for which the study was conducted and the limited amount of subjects are threats to the study's external and conclusion validities. Therefore, despite the benefits observed in our experimental study results, we are aware that further investigation is needed.

Acknowledgments. We would like to thank David N. Card for his contributions to our research. We would also like to Lucas Paes for his software implementation efforts to get the framework ready and functional. And finally, thanks to CAPES, CNPq, the subjects involved in the experimental study, and the COPPETEC Foundation, without their support this paper would not have been possible.

References

1. Card, D.N.: Defect Causal Analysis Drives Down Error Rates. IEEE Software 10(4), 98–99 (1993)
2. Mays, R.G., Jones, C.L., Holloway, G.J., Studinski, D.P.: Experiences with Defect Prevention. IBM Systems Journal 29(1), 4–32 (1990)
3. Dangerfield, O., Ambardekar, P., Paluzzi, P., Card, D., Giblin, D.: Defect Causal Analysis: A Report from the Field. In: Proceedings of International Conference of Software Quality, American Society for Quality Control (1992)
4. Jalote, P., Agrawal, N.: Using Defect Analysis Feedback for Improving Quality and Productivity in Iterative Software Development. In: 3rd ICICT, Cairo, pp. 701–713 (2005)
5. Card, D.N.: Defect Analysis: Basic Techniques for Management and Learning. In: Advances in Computers, vol. 65, ch. 7, pp. 259–295 (2005)
6. Kalinowski, M., Travassos, G.H., Card, D.N.: Guidance for Efficiently Implementing Defect Causal Analysis. In: VII Brazilian Symposium on Software Quality (SBQS), Florianópolis, Brazil (2008)
7. Kalinowski, M., Travassos, G.H., Card, D.N.: Towards a Defect Prevention Based Process Improvement Approach. In: 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2008), Parma, Italy, pp. 199–206 (2008)
8. Ishikawa, K.: Guide to Quality Control. Asian Productivity Organization, Tokyo (1976)
9. Kalinowski, M., Mendes, E., Card, D.N., Travassos, G.H.: Applying DPPI: A defect causal analysis approach using bayesian networks. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 92–106. Springer, Heidelberg (2010)
10. SEI: CMMI for Development (CMMI-DEV), Version 1.3. CMU/SEI-2010. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (2010)
11. Fagan, M.E.: Design and Code Inspection to Reduce Errors in Program Development. IBM Systems Journal 15(3), 182–211 (1976)
12. Pearl, J.: Causality Reasoning, Models and Inference. Cambridge University Press, Cambridge (2000)
13. Hong, G., Xie, M., Shanmugan, P.: A Statistical Method for Controlling Software Defect Detection Process. Computers and Industrial Engineering 37(1-2), 137–140 (1999)
14. Kalinowski, M., Travassos, G.H.: A Computational Framework for Supporting Software Inspections. In: Int. Conf. on Automated Soft. Eng. (ASE 2004), Linz, Austria, pp. 46–55 (2004)
15. Spínola, R.O., Kalinowski, M., Travassos, G.H.: Uma Infra-Estrutura para Integração de Ferramentas CASE (in portuguese). In: XVIII Brazilian Symposium on Software Engineering, Brasilia, Brazil, pp. 147–162 (2004)
16. WEKA: WEKA - Open Source Machine Learning Software in Java (2011), `http://www.cs.waikato.ac.nz/ml/weka/` (last accessed at February 15, 2011)
17. Shull, F.: Developing Techniques for Using Software Documents: A Series of Empirical Studies. Ph.D. thesis, University of Maryland, College Park (1998)
18. Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering – An Introduction. Kluwer Academic Publishers, Dordrecht (2000)

A Genetic Algorithm to Configure Support Vector Machines for Predicting Fault-Prone Components

Sergio Di Martino[1], Filomena Ferrucci[2], Carmine Gravino[2], and Federica Sarro[2]

[1] University of Napoli "Federico II" Via Cinthia, 80126 Napoli, Italy
sergio.dimartino@unina.it
[2] University of Salerno, Via Ponte Don Melillo, 84084 Fisciano (SA), Italy
{fferrucci,gravino,fsarro}@unisa.it

Abstract. In some studies, Support Vector Machines (SVMs) have been turned out to be promising for predicting fault-prone software components. Nevertheless, the performance of the method depends on the setting of some parameters. To address this issue, we propose the use of a Genetic Algorithm (GA) to search for a suitable configuration of SVMs parameters that allows us to obtain optimal prediction performance. The approach has been assessed carrying out an empirical analysis based on jEdit data from the PROMISE repository. We analyzed both the inter- and the intra-release performance of the proposed method. As benchmarks we exploited SVMs with Grid-search and several other machine learning techniques. The results show that the proposed approach let us to obtain an improvement of the performance with an increasing of the Recall measure without worsening the Precision one. This behavior was especially remarkable for the inter-release use with respect to the other prediction techniques.

Keywords: Fault prediction, Support Vector Machines, Genetic Algorithm.

1 Introduction

Software testing is one of the most expensive phases of the software development life cycle and at the same time it is very critical for the quality of a product. Thus, it is valuable for the competitiveness of a software company to have tools able to better support this phase. A huge amount of research in Software Engineering has been devoted to improve the efficiency of testing. Among these, considerable efforts have been aimed towards the definition of techniques able to predict the components of a software system that more likely will contain faults (e.g., [8, 13, 18, 19, 23, 26]). The rationale is that, once identified these potentially defective components, project managers can better decide how allocate resources to test the system concentrating their testing efforts to fault-prone components. As a result, the reliability, the quality, and the cost/effectiveness of the software product can be improved.

Usually the approaches for predicting fault-prone components exploit statistical models able to relate some software structural metrics with the probability of the presence of faults. Typical employed metrics are the Lines of Code (LOCs), the Halstead measures [15], or the Chidamber and Kemerer (CK) metrics for Object-Oriented

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 247–261, 2011.
© Springer-Verlag Berlin Heidelberg 2011

systems [7]. Among the machine learning techniques, Neural Networks, Bayesian Networks, Logistic Regression, and Decision Tree algorithm C4.5 are widely used [1, 8, 11, 17, 25, 29].

Many efforts have been devoted to understand what are the most explicative metrics for the phenomenon at the hand, and to tailor machine-learning techniques in order to provide more accurate identification of the defective components (e.g., [1]). In this context, some studies have reported a good performance of Support Vector Machines (SVMs) to predict fault-prone components (see e.g., [8, 12]).

SVMs are supervised learning methods that can be applied to classification tasks; i.e., given a set of data, each marked as belonging to one of two groups, a model is constructed to predict whether a new item falls into one group or the other. SVMs have turned to be a powerful tool in several contexts. Nevertheless, the application of the technique is not straightforward since some parameters must be carefully set to get more accurate classifications. Moreover, the suitable values of such parameters depend on the characteristics of the dataset, thus no rule of thumb can be defined, but a search technique has to be employed. The most of the studies presented in the literature adopts a "Grid-search" [6] approach, whose grain is very coarse.

To overcome these limitations, in this paper we propose a solution for the identification of error-prone components, based on the combination of a Genetic Algorithm (GA) and SVMs. In particular, a GA is exploited to search for a suitable SVM parameter setting exploiting a fitness function. Each criterion that is used to determine the performance of fault prediction methods can be used as fitness function.

The proposed approach has been assessed by an empirical analysis meant to verify the effectiveness of the approach to configure SVMs. As datasets, we have employed data on jEdit, a well-known text editor written in Java, whose information in terms of metrics and faults are available in the PROMISE database [24] for different releases. This has allowed us to analyze both the inter- and the intra-release performance of the method. In particular, we used a 10 fold cross-validation for the intra-release analyses, and a hold-out validation for an inter-release assessment, thus replicating with the latter validation a situation that typically arises in real software testing context, where data from the former releases are exploited to train the model that is used to predict faults for a new release. As benchmarks we employed SVMs in combination with the use of Grid-search for parameter selection [6] and six other widely used machine learning techniques, namely Logistic Regression, Decision Tree Algorithm C4.5, Naïve Bayes, Multi-Layer Perceptrons, K-Nearest Neighbor, and Random Forest. As measures to compare performance, we have employed Accuracy, Precision, Recall, and F-Measure. All these measures were also experimented as fitness function of GA.

The remainder of the paper is structured as follows: in Section 2 we describe the proposed approach reporting the main concepts of SVMs and GA and illustrating the key aspects of the defined GA. Section 3 is devoted to present the planning of the empirical analysis, in terms of the investigated research goals, the employed datasets, and the adopted validation method and evaluation criteria. In Section 4 we report and discuss the results, while in Section 5 we analyze the threats to the validity of the empirical study. A review of the related work is presented in Section 6, while some final remarks conclude the paper.

2 Support Vector Machines and Genetic Algorithms

In the following, we provide a brief description of Support Vector Machines and Genetic Algorithms, and how we tailored a Genetic Algorithm to configure Support Vector Machines.

2.1 Support Vector Machines

Support Vector Machines (SVMs) are a classification technique developed by Vapnik at the end of '60s [28, 29]. Since then the technique has been deeply improved, being applied in many different contexts. SVMs are known as maximum-margin classifiers, since they find the optimal hyperplane between two classes, defined by a number of support vectors [13]. The well-known generalization feature of the technique is mainly due to the introduction of a penalty factor, named *C*, that allows us to prevent the effects of outliers by permitting a certain amount of misclassification errors.

Although the original technique was able to provide only linear classification, thanks to the use of the kernel trick, SVMs can handle also non-linear problems. Indeed, in this case a kernel function is used to implicitly map the data points into a higher-dimensional feature space. Consequently, every dot product is replaced by the nonlinear kernel function, allowing the technique to find the maximum-margin hyperplane in a transformed, higher dimensional space. The rationale is that data is more likely to be linearly separable in the higher feature space [16]. A lot of kernel functions have been proposed in the literature. Among them, the ones based on Radial Basis Functions (RBF) are widely employed. In this study we decided to employ the RBF kernel since it was previously used in defect prediction context [13, 26, 27] and usually yields better performance than other kernels [5, 16].

When SVM is used with the RBF kernel, two parameters, *C* and γ, have to be set by the user. The selection of appropriate values for these parameters is crucial to obtain good classification performance. As described before, *C* is the penalty factor for misclassified points. If it is too large, a higher penalty for non-separable points is added, leading to store too many support vectors and thus over fit. On the other hand, if *C* is too small, an underfitting can occur. The γ parameter specifies the radius of the RBF, also having a strong impact on the accuracy.

A suitable combination of *C* and γ is often selected by a Grid-search with exponentially growing sequences of values, as done in LibSVM [6], one of the most employed, freely available library for SVMs. However, this approach has a twofold problem: 1) it has a very coarse grain, and thus it is likely to miss optimal values; 2) always the same couples of values for *C* and γ are explored, without taking into account the problem at the hand. In other contexts, the use of some search-based approaches [16] has been investigated to address the problem. For instance, in [5] Tabu Search (TS) has been employed to configure Support Vector Regression (SVR), i.e., the regression version of SVM, for software development effort estimation. TS is a meta-heuristic relying on adaptive memory and responsive exploration of the search space that has been used to address several optimization problems [10]. The results of the case study presented in [5] revealed that TS was able to suitably set the parameters

of SVR. In this paper, to configure SVMs for fault prediction we investigate the use of another search-based approach, namely a Genetic Algorithm. In the next subsection we detail the approach.

2.2 A Genetic Algorithm to Configure SVMs

Genetic Algorithms [11] belong to the family of evolutionary algorithms that, inspired by the theory of natural evolution, simulate the evolution of species emphasising the law of survival of the strongest to solve, or approximately solve, optimisation problems. The idea of exploiting this method to configure SVMs for fault prediction is based on the observation that the setting of SVMs can be formulated as an optimisation problem. As a matter of fact, among the possible configurations (solutions), we have to identify the one which leads to the optimal SVMs performance.

A typical Genetic Algorithm (GA) creates consecutive populations of individuals (i.e., chromosomes), considered as feasible solutions for a given problem, to search for a solution which gives the best approximation of the optimum for the problem under investigation. To this end, a fitness function is used to evaluate the goodness (i.e., fitness) of the chromosomes and genetic operators based on selection and reproduction are employed to create new populations (i.e., generations). Despite of a number of variations, the elementary process of a GA is the follows: (i) first a random initial population, i.e., a set of chromosomes, is generated; (ii) then, new individuals (i.e., offspring) are created by applying genetic operators (i.e., crossover and mutation) and a selection based on individual's fitness value is applied to determine who will survive among the offspring and their parents; (iii) the second step is repeated until either the fitness of the best solution has converged to the optimal value or if the optimal is not knew until a certain number of generations have been made. The individual that gives the best solution in the final population is taken in order to define the best approximation to the optimum for the problem under investigation. The analysis of this process suggests that the following design choices have to be made for tailoring GA to a given optimisation problem [11]:

1. defining the chromosome for representing a solution (i.e., *solution encoding*) and the number of initial solutions (i.e., *population size*);
2. choosing the criterion (i.e., *fitness function*) to measure the goodness of a chromosome;
3. defining the combination of *genetic operators* to explore the search space;
4. defining the *stopping criteria*.

In the following we provide the details regarding the choices we made for points 1-4 to design a GA for configuring SVMs. Concerning the *solution encoding*, since a solution has to represent an SVM configuration, the corresponding chromosome is composed by two genes, i.e., one for each SVM parameter, and the values for the genes are selected in the ranges [0.01, 32000] and [1.0E-6, 8] for the genes representing C and γ, respectively. These ranges are the same adopted in the Grid-search included in LibSVM [6]. The initial population is composed by 100 chromosomes that are created assigning random values to each gene. To assign the fitness value, the goodness of chromosomes is evaluated by running SVMs with the configuration

represented by each chromosome and employing as *fitness function* some widely used performance measures (i.e., Accuracy, F-measure, Precision, and Recall) which are described in section 3.3. As for the *genetic operators*, we employed a single point crossover which combines two individuals (i.e., parents) to form a new individual by randomly selecting a point of cut and swapping all genes beyond that point in either parent. Concerning the mutation operator, it selects a random gene of the chromosome and randomly changes the associated value. Crossover and mutation rates are fixed to 0.5 and 0.1, respectively. To determine the individuals that are included in the next generation (i.e., survivals) we employed the tournament selector, where only the best *n* solutions are copied straight into the next generation.

The evolutionary process is terminated according two *stopping criteria*, i.e. after 300 generations or if the fitness value of the best solution does not change after 30 generations.

3 Case Study Planning

In this section, we present the design of the empirical study we performed to get an insight on the use of the proposed GA to configure SVMs for fault prediction. In particular, to verify the effectiveness of the combination of GA and SVMs described in the previous section, we compared the obtained predictions with those achieved using SVMs configured with the Grid-search provided by LIBSVM [6]. Since the choice of the fitness function can play a crucial role in the application of a GA we have also experimented the impact of using different fitness functions. Furthermore, to understand the actual effectiveness of the proposed approach with respect to other methods we have also investigated some techniques available in the Weka tool [14], i.e., Logistic Regression (LR), Decision Tree Algorithm C4.5 (C4.5), Naïve Bayes (NB), Multi-Layer Perceptrons (MLP), K-Nearest Neighbor (KNN), and Random Forest (RF).

To perform this analysis we employed two datasets, namely *jEdit: 4-0-final_4-2-final* (in the following *jE1*) and *jEdit: 4-2-final_4-3-pre12* (in the following *jE2*)[1], obtained from different releases of the same software (*jEdit*) and included in the PROMISE repository [24]. In particular, we carried out two kinds of analysis: intra-release and inter-release similarly to [30].

Thus, the research questions of our study can be outlined as follows:

- RQ1: Can the choice of the fitness function affect the prediction performance of the combination of GA and SVMs?
- RQ2: Is the proposed GA able to effectively configure SVMs parameters for fault prediction?
- RQ3: Is the fault prediction performance of the combination of GA and SVMs superior to the ones obtained by other techniques?
- RQ4: Are there differences in the performance of the considered techniques for intra- and inter-release fault prediction?

[1] jE1 and jE2 are available at http://promisedata.org/?p=74 and http://promisedata.org/?p=73, respectively.

3.1 Dataset

To carry out the empirical evaluation of the proposed technique, we selected two datasets from the PROMISE repository, which contains data made publicly available in order to encourage repeatable, verifiable, refutable, and/or improvable predictive models of software engineering [24].

The datasets we employed are related to versions 4.0, 4.2 and 4.3 of the jEdit system, a well-known text editor written in Java. They contain data for the Chidamber and Kemerer (CK) metrics [7], a suite of six complexity metrics widely employed to measure product attributes of Object Oriented software systems and described in Table 1 together with Number of Public Methods and Number of Lines of Code.

Metrics data were computed by exploiting Understand IDE, a tool able to measure many characteristics of a software application. As for the fault data, they were obtained from SVN log files (where "true" means fault existence and "false" means fault nonexistence) [24]. In particular, for dataset *jE1* the metric data were computed based on jEdit release 4-0-final, while fault data were extracted between releases 4-0-final and 4-2-final. Similarly, for dataset *jE2* the metrics data were computed based on jEdit release 4-2-final, while fault data were extracted between releases 4-2-final and 4-3-pre12. The descriptive statistics of the metrics and fault data for the employed datasets are reported in Tables 2 and 3.

3.2 Validation Method

To assess the effectiveness of the fault predictions obtained using the techniques described herein, we have employed a k-fold cross validation for the intra-release analyses and a hold-out validation for the inter-release analysis.

Table 1. Metrics of the employed datasets

Name	Definition
Weighted Methods per Class (WMC)	The WMC metric represents the number of methods in the class (assuming unity weights for all methods).
Depth Inheritance Tree (DIT)	The DIT metric provides for each class a measure of the inheritance levels from the object hierarchy top.
Number Of Children (NOC)	The NOC metric measures the number of immediate descendants of the class.
Coupling Between Object Classes (CBO)	The CBO metric represents the number of classes coupled to a given class.
Response For Class (RFC)	The RCF metric measures the number of different methods that can be executed when an object of that class receives a message.
Lack of Cohesion Metric (LCOM)	The LCOM metric counts the sets of methods in a class that are not related through the sharing of some of the class fields.
Number of Public Methods (NPM)	The NPM metric counts all the methods in a class that are declared as public. The metric is known also as Class Interface Size (CIS).
Lines Of Code (LOC)	The LOC metric is the number of instructions in each method of the class.

Table 2. Descriptive statistics for the employed datasets

Dataset	Variable	Min	Max	St.dev	Mean
jE1	WMC	0	407	31.2	11.73
	DIT	0	7	1.98	2.50
	NOC	0	35	3.1	0.72
	CBO	0	105	14.13	12.64
	RFC	0	843	269.59	174.98
	LCOM	0	100	33.52	46.24
	NPM	0	193	17.12	7.78
	LOC	3	6191	529.66	206.21
jE2	WMC	0	351	26.46	11.66
	DIT	0	8	2.01	2.4
	NOC	0	38	3.04	0.71
	CBO	0	125	14.34	13.46
	RFC	0	862	268.58	169.98
	LCOM	0	100	33.66	48.86
	NPM	0	214	16.60	7.68
	LOC	3	5317	478.13	225.28

Table 3. Existence of faults

Dataset	# Elements with No Faults	# Elements with Faults
jE1	140	134
jE2	165	204

Cross validation is widely used in the literature to validate prediction models when dealing with medium/small datasets (see e.g., [4]). In particular, the k-fold cross validation is applied by partitioning each original dataset into k training sets, for model building, and test sets, for model evaluation. This is done in order to avoid optimistic predictions [20]. The errors from applying a given prediction technique are summarized using several performance measures (described in the following). In our study, we applied a 10-fold cross validation on the selected datasets, thus obtaining k=10 randomly test sets, and then for each of them, the remaining observations formed the training set to build the prediction model. As for the hold-out validation we have employed the first dataset, i.e., *jE1*, as training set and the second dataset, i.e., *jE2*, as test set.

3.3 Evaluation Criteria

Concerning the evaluation of the predictions obtained with the analyzed methods, we have used the following widely employed performance measures: Accuracy, Recall, Precision, and F-measure [1, 3]. In order to calculate them, we have exploited the concepts of True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN). This classification is represented in the Confusion Matrix reported in Table 4.

Accuracy is defined as the ratio between the number of components correctly predicted (i.e., classified as TP and TN) and the total number of components (i.e., the sum of TP, TN, FP, and FN).

Table 4. The confusion Matrix

		Predicted	
		Defective	Non Defective
Actual	Defective	True Positive	False Negative
	Non Defective	False Positive	True Negative

Precision is defined as the ratio between the number of components classified as TP and the number of components classified as TP or FP.

Recall is defined as the ratio between the number of components classified as TP and the number of components classified as TP or FN.

This means that Precision concerns the correctness of the responses provided by the method, while the completeness of the responses is measured by employing Recall. It is easily understandable that in the context of fault prediction, the consequence of low Recall is far more important than low Precision [22]. Thus, it is fundamental to employ a technique able to maximize Recall [30]. Indeed, in this way, the technique is able to detect all the potential defective components, even at the cost of a lower Precision, in order to avoid that potential defective components are missed. In this scenario, a subsequent manual check is responsible of deleting false positives inserted in this step. On the other hand, having a too low Precision requires an excessive post-processing, making the technique useless. A measure that provides an indication of a balance between correctness and completeness is the harmonic mean of Precision and Recall (or F-measure), defined as:

$$F-measure = 2 * \frac{precision * recall}{precision + recall} \quad (1)$$

A key advantage of GA is that it allows project managers to select the preferred criterion to drive the search for the prediction model and try to optimize a given performance measure (see e.g., [9, 17]).

4 Results and Discussion

The following subsections present and discuss on the results achieved in the empirical study.

4.1 Fitness Function Impact and Effectiveness of GA to Configure SVMs

In this section, we consider the first and the second research questions. The results of the application of the combination of GA and SVMs (GA+SVM in the following) for the inter- and the intra-release analysis and related to each of the investigated fitness functions (i.e., Accuracy, Precision, Recall, and F-measure) are reported in Table 5. These results suggest that the performance of GA+SVM are affected by the specific fitness function employed. In particular, the best value for each criterion has been obtained in almost all the cases using such a criterion as fitness function. Moreover,

there are some fitness functions that pay this with a worsening of the other criteria. In particular, the fitness function Recall was not able to provide high scores in terms of Accuracy and Precision. This is an expected result since it is widely recognized that any attempt to improve Recall is often paid back by a decreasing of Precision because it gets increasingly harder to be precise as the sample space increases [3]. Nevertheless, this phenomenon is not observed for all the fitness functions. Indeed, the best overall performances on both datasets *jE1* and *jE2* have been obtained by using F-measure as fitness function. Indeed, it has allowed us to obtain Recall and Precision values very close to the best ones. Furthermore, the fitness function F-measure has also provided the best Accuracy value in the case of the first dataset and a value close to the best one for the second dataset.

The results for the hold-out validation related to the inter-release assessment are also reported in Table 5. As expected these results are worse than those achieved by the intra-release analysis. Nevertheless, similarly to intra-release prediction models, we can observe that the best overall results were obtained using F-measure as fitness function. Thus, we can positively answer the research question RQ1 outlined in Section 3, i.e., the choice of the fitness function affected the accuracy of the prediction models built using GA+SVM.

In order to answer the research question RQ2 we compared the results achieved with GA+SVM with those obtained by applying SVMs configured with Grid-search of LIBSVM [6] (SVM-Grid in the following). The obtained results are shown in Table 6 together with the ones obtained with the other considered techniques and discussed in the next subsection. To facilitate the comparison in Table 6 we have also reported the results obtained with GA+SVM using F-measure as fitness function (since it provided the best performance).

The analysis of these results reveals that, for the first dataset, GA+SVM has allowed us to obtain better predictions than SVM-Grid in terms of Accuracy (c.a., 0.7%), Recall (c.a., 11.2%), and F-measure (c.a., 5%), while it has produced almost the same results in terms of Precision. With regard to the second dataset, GA+SVM

Table 5. Results using the combination of GA and SVMs with different fitness functions (the best results are in bold)

Dataset	Fitness function	Results			
		Accuracy	Precision	Recall	F-measure
jE1	Accuracy	0.719	0.677	0.813	0.739
	Precision	0.723	**0.704**	0.746	0.725
	Recall	0.635	0.582	**0.903**	0.708
	F-measure	**0.741**	0.701	0.821	**0.756**
jE2	Accuracy	0.637	0.610	0.951	0.743
	Precision	**0.648**	**0.635**	0.853	0.728
	Recall	0.572	0.564	**1**	0.721
	F-measure	0.629	0.601	0.980	**0.745**
jE1&jE2 (inter-release)	Accuracy	0.580	0.616	0.637	0.627
	Precision	0.588	**0.626**	0.632	0.629
	Recall	0.591	0.604	0.755	0.671
	F-measure	**0.615**	0.609	**0.848**	**0.709**

Table 6. Results for the employed prediction techniques

Dataset	Technique	Results			
		Accuracy	Precision	Recall	F-measure
jE1	GA+SVM F-measure	0.741	0.701	**0.821**	**0.756**
	SVM-Grid	0.712	0.704	0.709	0.706
	LR	0.715	0.719	0.687	0.715
	C4.5	0.708	0.696	0.716	0.706
	NB	0.693	**0.784**	0.515	0.622
	MLP	0.701	0.676	0.746	0.709
	KNN	0.723	0.730	0.687	0.708
	RF	**0.748**	0.721	0.791	0.754
jE2	GA+SVM F-measure	0.629	0.601	**0.980**	**0.745**
	SVM-Grid	0.648	0.657	0.754	0.702
	LR	0.634	0.658	0.706	0.681
	C4.5	0.612	0.660	0.618	0.638
	NB	0.531	**0.763**	0.221	0.342
	MLP	0.607	0.632	0.691	0.660
	KNN	**0.661**	0.691	0.701	0.696
	RF	0.653	0.673	0.725	0.698
jE1&jE2 (inter-release)	GA+SVM F-measure	**0.615**	0.609	**0.848**	**0.709**
	SVM-Grid	0.553	0.602	0.564	0.582
	LR	0.561	**0.615**	0.549	0.580
	C4.5	0.534	0.575	0.598	0.587
	NB	0.501	0.585	0.338	0.429
	MLP	0.542	0.579	0.627	0.602
	KNN	0.537	0.588	0.539	0.563
	RF	0.558	0.596	0.623	0.609

scored better than SVM-Grid for Recall (c.a., 22.6%) and F-measure (c.a., 4.3%), while it has produced worse results than SVM-Grid for Accuracy (c.a., 1.9) and Precision (c.a., 0.1%). As for the inter-release prediction, GA+SVM was able to outperform SVM-Grid for all the considered measures, with an impressing improvement in terms of Recall of more than 28%, having at the same time also a slight improvement in Precision (c.a., 0.7%). Consequently, the F-measure value growth of almost 13%.

The above results suggest that the use of GA to configure SVMs allowed us to obtain better results than SVM-Grid. This improvement is particularly evident and significant for Recall and is paid back by a little decrease in the values of Precision (only for *jE2*). Moreover, if the project manager is particularly interested in maximizing Recall also accepting a decrease of Precision, he/she can even exploit Recall as fitness function.

Thus, we can also positively answer the research question RQ2 outlined in Section 3, i.e., GA is able to effectively set SVMs configuration parameters for fault prediction.

4.2 Comparison of GA+SVM with other Estimation Techniques and between Intra- and Inter-release Prediction Performance

In this section, we report the results related to the research questions RQ3 and RQ4 and obtained by comparing the predictions provided by the combination of GA and

SVMs with the ones achieved by using other classification techniques available in the Weka tool [14], i.e., LR, C4.5, NB, MLP, KNN, and RF. Preliminarily, we compare the results obtained with these other techniques and reported in Table 6. We can observe that the best performance for the intra-release analysis was obtained with RF while NB provided the best Precision value but also the worst Recall and F-measure values. Regarding the results for the inter-release analysis, we can observe that the best Recall and F-measure values were obtained with RF and MLP, while the best Accuracy and Precision values were achieved with LR and RF. The results also suggest that NB provided the worst Precision, Recall, F-measure, and Accuracy values.

Concerning the comparison between the results we achieved with GA+ SVM and those obtained using the above techniques we can observe that GA+SVM, exploiting F-measure as fitness function, allowed us to get the best results in terms of Recall and F-measure for all the considered experimental settings.

Regarding the results obtained for the inter-release analysis (see Table 6), we can observe that among the prediction techniques we employed only the predictions obtained with GA+SVM with F-measure are very similar to the ones achieved in the intra-release assessment. Indeed, the predictions provided by the other techniques are worse than the ones obtained in the intra-release context. Moreover, also in this case GA+SVM with F-measure yielded to the best results in terms of Accuracy, Recall, and F-measure and comparable results in terms of Precision with respect to the other techniques we employed.

Thus, we can positively answer to research question RQ3, i.e. the predictions obtained by using the combination of GA and SVMs are superior to the ones obtained by the other techniques. As for RQ4 there are differences in the performance achieved with intra- and inter-release prediction for all the considered techniques except for GA+SVM using F-measure as fitness function.

5 Validity Evaluation

Several factors can bias the validity of empirical studies. Here we consider three types of validity threats: *Construct validity*, related to the agreement between a theoretical concept and a specific measuring device or procedure; *Conclusion validity*, related to the ability to draw statistically correct conclusions; *External validity*, related to the ability to generalise the achieved results. As highlighted by Kitchenham *et al.*[20], in order to satisfy construct validity a study has "to establish correct operational measures for the concepts being studied". Thus, the choice of the measures and how to collect them represents the crucial aspects. We tried to mitigate such a threat by evaluating the employed prediction methods on publicly available datasets from the PROMISE repository [24], that have been previously used in other empirical studies (see e.g., [19, 25, 30]). In relation to the conclusion validity, we carefully calculated the employed performance measures and the obtained data was cross-checked by two authors. Finally, observe that also the fact of having used open source software systems could affect the results of the presented study. This means that we cannot conclude that the results of this study promptly apply to other software development settings. This could represent an important external validity threat that can be mitigated only replicating the study on other datasets. Accordingly, we plan to conduct a further investigation on commercial software systems.

6 Related Work

Fault prediction is a very active research field within Software Engineering and many studies have addressed this key issue using a variety of different methods. Some interesting literature reviews are available in [1, 2]. For sake of space, we limit our description to the research that employed Support Vector Machines (SVMs) and Genetic Algorithms (GA) for fault prediction.

SVMs were used in many works together with other classification techniques, obtaining different results. In [8] SVMs were compared against eight modeling techniques in terms of several performance measures (i.e., Accuracy, Recall, Precision, and F-measure) using four datasets from the NASA Metrics Data Program Repository (MDPR) [21]. The results revealed that none of the employed techniques was significantly better than the others. On the other hand, Gondra [12] reported that, on the JM1 dataset in the NASA MDPR [21], SVMs significantly outperformed an Artificial Neural Network, achieving a percentage of correct classifications on the validation set of 87.4%, versus the 72.61% got by the Artificial Neural Network, thus suggesting that SVMs could be a promising technique for predicting fault-proneness software components.

Gray *et al.* [13] also carried out an empirical study employing SVMs on eleven NASA datasets but with a different purpose. Indeed, they aimed to analyze the performance of this technique when only static code metrics were used. Moreover, in this paper a data-preprocessing step was performed including selection of instances and variables, data normalization, and balancing of faulty and non faulty classes. The obtained results, evaluated only in terms of Accuracy, showed that SVMs yielded at an average Accuracy of 70% on the employed datasets. Similarly, Singh *et al.* [26] exploited SVMs on the KC1 NASA dataset [21] with the goal to analyze the relationship between the Objected-Oriented metrics given by Chidamber and Kemerer [7] and fault proneness by means of many indicators, such as Precision, Recall, and Receiver Operating Characteristic (ROC) analysis. The results revealed that some metrics (i.e., CBO, RFC, and SLOC) were significantly related to fault proneness. The authors replicated the study applying SVMs to jEdit [27], unfortunately, authors did not provide enough information to replicate the settings of the experiment, neither in terms of dataset (version, validation, etc.), nor in terms of the SVMs setting employed.

A variation of the regression version of SVMs (namely SVR) for fault prediction was presented in [18]. Indeed, the authors proposed the use of a Fuzzy SVR (FSVR) for predicting software fault number and analyzed whether the fuzzification of the input allowed SVRs to handle unbalanced software metrics datasets. The results obtained employing the MIS and the RSDIMU datasets revealed that FSVR yielded to a lower Mean Squared Error and higher Accuracy respect to the use of SVR.

As we can observe the majority of the above works employed datasets contained in the NASA Metrics Data Program Repository [21]. This is a useful database for fault prediction empirical analyses, but almost all the contained projects are highly unbalanced, since non-faulty components are many times more than the faulty ones. As a consequence, the most of the studies about the use of SVMs employed a random undersampling of the datasets, making impossible for us to replicate the experimentations or compare the results we achieved in this study. We do not know other studies that have addressed the problem of configuring SVMs parameters for

fault prediction: usually the parameter Grid-search feature included in LibSVM has been employed (e.g., [13, 26]). We recall that this feature represented the baseline for our experimental study.

The use of a GA for the fault prediction problem is presented in [25], where the authors reported on the results of, the application of GA to dataset jEdit 4.0. Again, no details at all are provided, either on the employed GA, either on the assessment protocol. Moreover, only a cumulative Accuracy of 80.14% is reported, but it is not clear how it was computed.

Finally, we want to highlight that we applied an intra-release prediction analysis and an inter-release prediction analysis and employed the same two datasets of the study presented in [30]. In particular, Watanabe *et al.* [30] exploited the algorithm C4.5 of Weka and a 10-fold cross validation. The results we have achieved using C4.5 on dataset *jE2* are different from those presented in [30] and maybe this is due to the different folds employed for the 10-fold cross validation. As expected, the same results were instead achieved when performing the inter-release prediction analysis. We observe that also RF was applied in the past on dataset *jE1* [19], however the authors did not declare what implementation of RF they employed and what k they used for the k-fold cross validation. Nevertheless, the RF results we achieved are very close to the ones they reported.

7 Conclusions

To improve software quality, it is fundamental to be able to predict defective software components. To address this issue, in this paper we have presented an approach that exploits a Genetic Algorithm (GA) to configure Support Vector Machines (SVMs) for predicting fault-prone software components, on the basis of some structural metrics. In this way, it is possible to fully exploit the potentiality of SVMs, whose performances depend on the configuration of parameters.

The proposed approach has been assessed in an empirical analysis, based on the jEdit data from the PROMISE repository. In particular, we have performed an inter- and an intra-release assessment of the proposal. The latter setting is the most realistic one, since it replicates the typical software development scenario, where data from previous releases are used to predict fault components in a newer version of the software under development. As benchmarks we exploited SVMs with the Grid search provided by LibSVM, and six other techniques frequently used in the context of fault prediction, namely Logistic Regression, Decision Tree Algorithm C4.5, Naïve Bayes, Multi-Layer Perceptrons, K-Nearest Neighbor, and Random Forest.

With respect to the four research questions addressed in the empirical analysis, the results reported in the paper show that:

- the prediction performance of GA+SVM is affected by the choice of the fitness function. The approach represents a flexible tool to support the strategies of project managers that might prefer to maximize a specific performance criterion (for instance Recall rather than Precision);
- GA is able to effectively set SVMs parameters in order to improve fault predictions;

- the fault predictions performance of GA + SVM was superior to the ones obtained by the other techniques;
- the improvement of GA+SVM with respect to the other investigated prediction techniques was especially remarkable for the inter-release use.

As we have mentioned in Section 5, we cannot promptly apply these results to other software systems different from the ones we employed. To address this issue, in the future we intend to replicate the performed analysis using other datasets. This is the only way to get better confidence on the generalizability of the results.

References

1. Arisholm, E., Briand, L., Johannessen, B.: Data mining techniques, candidate measures and evaluation methods for building practically useful fault-proneness prediction models. Simula Research Laboratory Technical Report, 2008-06
2. Arisholm, E., Briand, L., Johannessen, B.: A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Journal of Systems and Software 83, 2–17 (2010)
3. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
4. Briand, L., Langley, T., Wiekzorek, I.: A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. In: Procs of the International Conference on Software Engineering, pp. 377–386. IEEE press, Los Alamitos (2000)
5. Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E.: How Effective is Tabu Search to Configure Support Vector Regression for Effort Estimation? In: Procs of the International Conference on Predictive Models in Software Engineering, p. 4 (2010)
6. Chang, C.C., Lin, C.-J.: LIBSVM: a library for support vector machines, (2001), Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
7. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
8. Elish, K.O., Elish, M.O.: Predicting defect-prone software modules using support vector machines. Journal of Systems and Software 81(5), 649–660 (2008)
9. Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F.: Genetic Programming for Effort Estimation: an Analysis of the Impact of Different Fitness Functions. In: Procs of the 2nd International Symposium on Search Based Software Engineering, pp. 89–98. IEEE Computer Society, Los Alamitos (2010)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
11. Goldberg, E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
12. Gondra, I.: Applying machine learning to software fault-proneness prediction. Journal of Systems and Software 81, 186–195 (2008)
13. Gray, D., Bowes, D., Davey, N., Sun, Y., Christianson, B.: Using the Support Vector Machine as a Classification Method for Software Defect Prediction with Static Code Metrics. In: Palmer-Brown, D., Draganova, C., Pimenidis, E., Mouratidis, H. (eds.) EANN 2009. Communications in Computer and Information Science, vol. 43, pp. 223–234. Springer, Heidelberg (2009)
14. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1) (2009)
15. Halstead, M.H.: Elements of Software Science. Elsevier North-Holland, New York (1977)

16. Harman, M., Jones, B.F.: Search based software engineering. Information and Software Technology 43(14), 833–839 (2001)
17. Harman, M., Clark, J.A.: Metrics Are Fitness Functions Too. IEEE Metrics, 58–69 (2004)
18. Yan, Z., Chen, X., Guo, P.: Software Defect Prediction Using Fuzzy Support Vector Regression. In: Procs of the International Symposium on Neural Networks, pp. 17–24 (2010)
19. Kaur, A., Malhotra, R.: Application of Random Forest in Predicting Fault-Prone Classes. In: Procs of the International Conference on Advanced Computer Theory and Engineering
20. Kitchenham, B., Pickard, L., Peeger, S.: Case studies for method and tool evaluation. IEEE Software 12(4), 52–62 (1995)
21. NASA – Metrics data program, http://mdp.ivv.nasa.gov/
22. Ostrand, T.J., Weyuker, E.J.: How to measure success of fault prediction models. In: Procs of the Fourth Workshop on Software Quality Assurance, pp. 25–30 (2007)
23. Ostrand, T.J., Weyuker, E.J., Bell, R.M.: Predicting the Location and Number of Faults in Large Software Systems. IEEE Trans. Software Eng. 31(4), 340–355 (2005)
24. PROMISE Repository of empirical software engineering data, http://promisedata.org
25. Sandhu, P.S., Dhiman, S.K., Goyal, A.: A Genetic Algorithm Based Classification Approach for Finding Fault Prone Classes. World Academy of Science, Engineering and Technology 60 (2009)
26. Singh, Y., Kaur, A., Malhorta, R.: Software Fault Proneness prediction Using Support Vector Machines. In: Procs of the World Congress on Engineering, vol. I, pp. 240–245 (2009)
27. Singh, Y., Kaur, A., Malhorta, R.: Application of Support Vector Machine to Predict Fault Prone Classes. ACM SIGSOFT Software Engineering Notes 34(1) (2009)
28. Vapnik, V., Chervonenkis, A.Y.: Theory of Pattern Recognition (1974) (in Russian)
29. Vapnik, V.: The nature of Statistical Learning Theory. Springer, Heidelberg (1995)
30. Watanabe, S., Kaiya, H., Kaijiri, K.: Adapting a Fault Prediction Model to Allow Inter Language Reuse. In: Procs of the International Conference on Predictive Models in Software Engineering, pp. 19–24 (2008)

A Systematic Approach to Requirements Engineering Process Improvement in Small and Medium Enterprises: An Exploratory Study

Edward Kabaale[1] and Josephine Nabukenya[2]

[1] Makerere University Business School, P.O.Box 1337, Kampala, Uganda
[2] School of Computing and Informatics Technology, Makerere University, P.O.Box 7062, Kampala, Uganda
ekabaale@mubs.ac.ug, josephine@cit.mak.ac.ug

Abstract. Requirements Engineering (RE) studies have demonstrated that requirements errors affect the quality of software developed, making software requirements critical determinants of software quality. Requirements Engineering Process Improvement (REPI) models have been provided by different authors to improve the RE process. However, little success has been achieved in Small and Medium Enterprises (SMEs) software companies especially in transitional countries such as Uganda. This study reports on an exploratory study which provides insights into current RE practices in four Ugandan SME software companies, critical success factors and challenges that impede REPI. As a result a Systematic Approach to REPI has been designed following the design science approach. It provides guidelines and steps for SMEs in improving their RE processes.

Keywords: Requirements Engineering, Process Improvement, Software Process Improvement.

1 Introduction

Fast changing technology coupled with increased competition is placing a lot of pressure on software development process [9]. One of the most crucial parts of the software development process is requirements engineering (RE); and the process of developing new software products always starts with some kind of needs or wishes. The wishes and needs can be of help in finding the requirements that describe the properties and functions of the new software product. Discovering, documenting and maintaining requirements are often described as *requirements engineering* [31]. Effective RE lies at the heart of an organization's ability to produce software products that can meet the needs of the customers yet keeping pace with the rising wave of complexity [9]. The software industry in most countries is composed of Small and Medium Enterprises (SMEs) [18]. This study focused on SMEs mainly because they form the biggest number of software companies in developing countries and yet they produce important products for their clients [24]. Because SMEs are small in nature, then process improvement can easily be achieved through SMEs' flexibility, fast

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 262–275, 2011.
© Springer-Verlag Berlin Heidelberg 2011

reaction time and enhanced communication between members. In the Ugandan context a *Small Enterprise* is defined "as an enterprise employing maximum 50 people, annual sales/revenue turnover of maximum Ugandan Shillings 360 million and total assets of maximum Ugandan Shillings 360 million"; while a *Medium Enterprise* is defined as an "enterprise employing more than 50 people, annual sales/revenue turnover of more than Ugandan shillings 360 million and total assets of more than Ugandan Shillings 360 million" [29]. In the software industry, a *small organization* is defined as one with fewer than 50 software developers and a *small project* is one with fewer than 20 developers [18]. RE is a very important phase of the software process as errors at this phase inevitably lead to later problems in the system design and implementation [15]. RE can lead to better quality in software and systems development processes [16, 26]. It's only with efficient RE that the development process can be controlled and directed in terms of appropriateness and cost-effectiveness of the solution produced [9]. The main aim of a RE process is to come up with a set of necessary, verifiable and attainable requirements, which are acceptable to all the relevant stakeholders [26, 19].

The need to improve the RE process has been recognized for some time now and RE community has witnessed the emergency of models and standards for Requirements Engineering Process Improvement (REPI) and assessment. For example, the *Good practice guide* [27] gives basic guidelines on how to improve the RE process. However, even when the framework has been useful, it was intended for safety-critical domain project, hence lacking adaptation to different domains [30]. It is also too general and complex for SME software companies. The *Flexible and Pragmatic RE* framework for SMEs [22] aimed at providing a framework for RE improvement in SMEs that is more adaptable to support more domains and improve support for small, incremental improvements of the RE process. The *Requirements-Capability Maturity Model* [1] suggests key requirements practices within a maturity framework. Its main objective is to guide software practitioners to relate processes to goals in order to prioritize their requirements process improvement activities. However little progress has been registered in SMEs using these models in improving their RE process as witnessed by the continued failures in these companies

To this end there is a need for SMEs to access a systematic and reliable approach to REPI. In other words, what are the challenges for REPI in SME software companies, and how can these be used to derive recommendations and requirements that can be used to design a systematic approach to REPI in SME software companies? [33] define *Systematic process improvement* as a goal-oriented measurement and controlled way of introducing process change, with predictable outcome in terms of quality, time and productivity

Thus, to put our research in context, we looked at the state of art of SMEs in section 2. In section 3, we describe the research approach followed in order to undertake the exploratory study and later the design of the systematic approach. . In section 4, we describe the exploratory study in which the current practices with respect to REPI in Ugandan SME software companies were investigated and analyzed. We also present the derived requirements that lead to the design of the actual systematic approach to REPI for SMEs in section 5, and finally provide a way forward on future prospects of this research.

2 State of the Art

There is scanty literature about RE processes in SMEs; this is because they posse unique characteristics that originate from their make and ownership. For example [14] argues that requirements practices in SMEs are dependent on individuals implementing them. SMEs have got specific problems due to their size and the budget constraints under which they operate, their maturity level in software engineering is very low, little resources to consider quality and process improvements, very few SMEs document their requirements and there are no clear ways of RE process verification and validation [10]. [28] state that SMEs do not emphasis training, have pressing deadlines and so little time is spared for process improvement, SMEs have a simplified software process lifecycle where emphasis is put on development and testing only, SMEs also lack control procedures, project management and planning skills as well as risk management. [18] argues that SMEs cannot measure the process progress and benefits that accrue from such processes.

Many SME software companies are interested in improving their RE processes because of their confidence that RE can be the key to developing successful software systems [18]. However, SMEs find it difficult to implement these process improvements because they cannot bear the cost of implementing these REPIs as well as the limited resources and the strict time constraints in which they operate [18]. A survey of twelve SMEs in Finland reveal that SMEs management are not aware of the available REPI methods but there is desire to start them [20]. [22] argues that where SMEs have in place RE process, it's always very difficult to improve such practices because it has an economic implication to the organization. [10] found out that the most relevant topics to SMEs were requirements modeling, improvement of requirements document, inspections, and tools. In [16] field study, seven key factors were identified as critical to a successful RE process improvement effort and these are package consideration, managing the level of detail of functional process models, examining the current system, user participation, managing uncertainty, benefits of case tools and project management capability . The driving factors in SMEs for RE process improvements are the problems with testing and ISO 9001 certification as well as competing for big contracts with bigger software companies and yet the problems hindering effective RE process improvement in SMEs are the small budgets and tight project schedules. Demonstrated benefits of RE process improvement like the ones in [2] are a fundamental step towards encouraging SMEs to start RE process improvements [10].

Software development processes in Ugandan SMEs are still low with a few cases elaborating the RE processes. Therefore the situation and challenges facing Ugandan SMEs is not different from that discussed above. But perhaps it may even be worse in some cases.

3 Research Approach

In order to understand the REPI current practices in Ugandan SMEs from which the systematic approach to REPI was designed; we followed the design science (DS) approach. Design science is fundamentally a problem solving paradigm and therefore

seeks to produce constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations [8]. The created artifacts extend the boundaries of human problem solving within organizational capabilities [8] and are tangible recommendations that enable IT researchers to tackle the problems inborn in developing and deploying information systems within organizations. We chose design science research method because of its ability in solving practical problems that have besieged the Information system arena for some time by creating successfully IT artifacts [23]. To achieve our research goal, we followed the design science research process model suggested by [23]. This model was used because it provides a nominal design science research process with a template for conducting applicable and precise design science research with six clear steps and it was tested and verified using RE case studies, the subject matter of this research [23]. The suggested six steps for conducting and evaluating good design-science research that were used in this research are;

Problem identification and motivation; The problem in this research was the lack of a systematic approach to RE process improvements in SMEs. This is due to the inadequate RE processes being used. *Objectives of a solution;* The main objective of this research was to understand the challenges impeding REPI in SMEs and come up with requirements from which we developed a systematic approach for improving RE processes and practices in SMEs. *Design and development*; in this step *a* literature review was conducted and corroborated with data from the field studies in the selected cases. These led to appropriate requirements for the design of the systematic approach to RE process improvement for SMEs. *Demonstration and Testing;* We planned to demonstrate the systematic approach to the selected cases and get their feedback about the new approach. *Evaluation and validation;* We planned to evaluate and validate the systematic approach in the selected cases against recommended design science value criteria. Questionnaires and interviews will be used in evaluation and validation of the new approach. *Communication;* Presentation of findings and the systematic approach other relevant audiences like SMEs' management, conferences and different reports.

Our exploratory study findings and analyses are based on the data collected through interviews, questionnaires and existing requirements documents. The cases for this study were purposively selected so as to represent different application domains, experiences in software development and company sizes in Uganda. These cases are operating in a wide range of application domains and have been in business between 5 to 10 years which gives them the necessary experience in the RE area. The respondents were software developers, system analysts, project managers as well as system administrators. These SMEs employ five to thirty people.

Case 1: *Makerere University Business School (Socket works project).* This project aimed at developing an Education Information Management System to be used by the school. Its key areas included; requirements elicitation, analysis, design and implementation. It provided respondents in these specific areas of RE. Case 2: *Department of Innovations and Software Development (DISD), Faculty of Computing and Information Technology, Makerere University.* It focuses on the growth of software conception, design and development capacity at the Faculty. DISD believes in the development of local capacity to build and exploit ICT innovations in the Country. It

is also involved in different external and internal projects ranging from Business Information Systems to web based systems.

Case 3: *Crystal Clear Software Ltd*. Its mission is to deliver high quality, user friendly and tailor made business software for her clients. It is the sole developer of Loan Performer software. This software is a database for recording and evaluating microloans. Case 4: *Software Factory Ltd*. It deals in a range of solutions from Business Information Systems to Web Based Systems. RE is one of the areas emphasised in this SME in order to develop quality software for their clients.

4 Field Study and Analysis of RE Process and Practices in Ugandan SMEs

The findings of this study are based on the data collected through observations, interviews and questionnaires that were issued to the different experts in the area of requirements engineering. The purpose of the interviews was to gain more information on how practitioners defined and managed requirements in practice as well as the strengths and weaknesses of the existing RE practices and the challenges they faced in REPI. Fifty (50) questionnaires were given out to purposively selected respondents and only 30 informants responded giving a 60% response in the four SME software companies selected. Collected data was categorized, quantified, coded and arranged in themes according to RE practices, REPI challenges and recommendations.

4.1 Current RE Processes and Practices in Ugandan SMEs

To get an overview of current RE practices and improvement needs, interviewees with a company-wide role and knowledge of high level RE practices and involvement were sought. The outcomes were assessed using standard RE process practices suggested in the Requirements Engineering Adaptation and IMprovement for Safety and dependability (REAIMS) maturity model [27] and Flexible and Pragmatic RE framework for SMEs [22] due to the fact that they spell out the main activities of a RE process.

Table 1. Current RE processes and Practices in Ugandan SMEs

RE Processes and Practices	Frequency	Percent
Elicitation	8	26.7
Elicitation, analysis, specification, validation	3	10.0
Elicitation, analysis, specification, validation, verification	2	6.7
Elicitation, analysis, specification	8	26.7
Elicitation & analysis	6	20.0
Requirements documentation	1	3.3
Requirements negotiation	2	6.7
Total	**30**	**100.0**

From table 1, the results revealed that SMEs were involved more in requirements elicitation, analysis and specification (26.7%) as compared to documenting their requirements (3.3%). This shows that these SMEs were acquainted with the first four stages of RE process suggested in [32, 22]. We also observed that there were no SMEs involved in requirements traceability and requirements change management. This implies that the SMEs could not validate the requirements against their sources nor in a clear position to manage requirements that keep changing during the course of the RE process. As it's always the case with SMEs, little documentation (3.3%) of requirements was found in the case studies. This is in line with [10] where requirements documentation is among the priority topics for the workshop. This implies that SMEs did not prepare requirement documents; this may be due to their tight schedules and lack of resources and skills to document the requirements. No SMEs were found to be using automated tools to support the RE process. This was attributed to lack of awareness (knowledge) about the availability and use of such tools; some SMEs thought that these tools were for large organizations and expensive for small ones. This is divergent from [20] who had SMEs with similar roles but were using automated tools to support their RE processes. These results revealed that the Ugandan SMEs were still at the lowest level of the RE process maturity, where the RE process is not explicitly defined, no standards for requirements documentation and requirements description [27]. The preceding observations were further analyzed to establish whether SMEs had a defined REPI approach as seen in table 2. . From table 2, we observed that majority of the SMEs (66.7%) didn't have a defined REPI approach; and when asked why majority (53.3%) responded that they were not aware (knowledge) of how to define one (see table 3). Meanwhile, 33.3% had a defined RE process approach; though it was customized by respective organizations.

Table 2. Defined REPI Approach

Approach	Frequency	Percent
Yes (Defined Approach)	10	33.3
No (Not defined Approach)	20	66.7
Total	**30**	**100.0**

Majority of the SMEs (66.7%) did not have any specific approach to REPI. This was largely caused by lack of knowledge of the improvement models available for use. This further indicates low levels of RE process maturity in these SMEs. This situation is consistent with [10] study in which REPI was among the top topics ranked by the respondents from different SMEs in Germany.

Table 3. Awareness of REPI Models

Awareness of RE process Improvement Models	Frequency	Percent
Yes (Aware)	14	46.7
No (Not aware)	16	53.3
Total	**30**	**100.0**

Respondents reported that management was not aware of any available REPI models (53.3%) despite their interest in improving the RE processes. Surprisingly the respondents were aware of the benefits of REPI and willing to start any process improvement in order to tap the benefits. However they did not know any REPI models available in practice and a few were using customized models to help in the improvement of the RE process. This was in disagreement with SMEs in [20, 10, and 30] who have used one or two REPI models in their process improvements despite their low levels of software maturity.

4.2 Challenges and Recommendations to Successful REPI in SMEs

Based on the above results, we observed that the Ugandan SMEs did have less to none understanding of REPI (see tables 1-3), and thus becoming important to investigate the hindrances and would-be recommendations to their success as represented in tables 4 and 5 respectively. While in table 5, training (36.7%) was reported as the most favored solution to these challenges.

Table 4. Challenges to REPI in Ugandan SMEs

Challenges	Frequency	Percent
Lack of User's Involvement	5	16.7
Lack of management support	5	16.7
Lack of Skills	2	6.7
Changing Requirements	1	3.3
Resistance to change	1	3.3
Ambiguous Requirements	9	30.0
Lack of proper Documentation	2	6.7
Measurement of RE benefits	2	6.7
Expensive	3	10.0
Total	**30**	**100.0**

From table 4, we observed that ambiguous requirements from the clients were the biggest challenge (30%) faced by the respondents while on the other hand a few respondents (3%) were faced with resistance to change and changing requirements as obstacles to REPI. Summarized, the challenges that face SMEs include:

Lack of user involvement; affects the acceptability of the new process in organizations. This study revealed that users are not fully involved in REPIs instead they are imposed on the users. This concurs with [15] who looked at user involvement in process improvement as a very important factor for user acceptance of new processes.

Lack of Management support; This challenge affects the availability of the facilities that support any process improvement efforts in any organization [15]. The study revealed that support from management was lacking towards REPI. This was hindering Process Improvements in these SME software companies. This concurs with [11]

were sustainable changes across an organization require management commitment and support at all levels.

Lack of Skills; There are generally low levels of REPI awareness in SMEs. SMEs lack the people with the right skills to carry out proper process improvements that can yield benefits to such organizations and yet they cannot use consultants because of their budget constraints [15]. In this study, it was established that SMEs lacked the necessary skills to start process improvements within their organizations despite their interest to start process improvements. This view is also shared by [11] who argues that a process improvement initiative is at risk if the developers, managers, and process leaders do not have adequate skills to carry out process improvement. *Ambiguous Requirements;* Requirements from users are never clear or complete, so the developers are always occupied with correcting and understanding requirements than starting process improvements in the organization. It's reported in [2] that prior to a REPI initiative; requirements were not clearly defined nor fully understood by developers. This was because clients were ambiguous in stating their requirements. In this study, it was established that clients did not state exactly what they wanted from the new system, thus they keep on changing their requirements from time to time. *Expensive;* Given the tight budgets of SME software companies, it's always difficult to spare some funds for process improvements. The study revealed that majority could not carry out REPI because it was very expensive and it concurs with [10] who highlighted small budgets as the main obstacles to REPI in SMEs. *Measurement of RE process benefits;* measurement of the REPI benefits to the organizations implementing the improvement is a problem to SMEs. Its argued in [30] that it's very difficult to measure the benefits of REPI because there is a long time lag between the requirements phase and system delivery to pinpoint how specific RE techniques contribute to a system's success or failure. *Resistance to change*; this can threaten the success of any new REPI. It's stated in [17] that there is a direct relationship between resistance to change and the total amount of change required of individuals. It's pointed out in [12] that people resist change because change initiatives are introduced too quickly and frequently.

The challenges above have impeded successful REPI in Ugandan SMEs; yet can be critical success factors if well handled. It's argued in [15] that managing the above challenges can lead to better management of REPI in software companies.

Table 5. Recommendations to REPI Challenges in Ugandan SMEs

Recommendations	Frequency	Percent
Workshops & RE process Documentation	2	6.7
Training	11	36.7
Management Support & Documentation	2	6.7
User involvement	5	16.7
Project Management Skills & planning	4	13.3
RE improvement strategy	2	6.7
Management of changing requirement	4	13.3
Total	**30**	**100.0**

From Table 5 above, it was established that most of the respondents thought training (36%) would provide the necessary skills that are lacking in most of the cases visited for REPI. On the other hand, a few respondents opted for management support and improvement strategies to bring about successful REPI in their organizations. Summarized, the recommendations that SMEs suggested include:

Training; [28] emphasis the need of training in process improvement in their OWPL model for software process improvement in SMEs. According to [5] education and training helps to promote the good understanding of the RE process to all the people involved in the improvement process and it's considered to be one of the critical success factor for any RE improvement process. Therefore, there is need to adequately train all people involved in the RE process improvement in order to ensure sustained change. [11] suggests that training helps the organization's members to have a common vocabulary and understanding of how to assess the need for change and how to interpret specialized concepts of the improvement model being used as well as to achieve a common understanding of the improvement process. *Management Support and Commitment;* Management support to process improvement can be in form of funding, encouragement, allocation of staff and providing a conducive environment for working [15]. Raising the management awareness and support of RE practices would make it easier to start RE process improvement efforts in organizations and thus eventually raise the RE process maturity in companies [19]. [15] concur that management commitment process is a fundamental requirement for a successful improvement in RE process.

Change Management; can help to manage resistance to change by employees during the RE process improvement. This is in line with [2] who suggested change management as a very important factor for RE process improvement where acceptance of new RE practices can be one of the key challenges in RE process improvement.

User involvement in the RE process is very important to the success and institutionalization of any system process improvement. Among the stakeholders from whom requirements are elicited are the potential users of the new system and this helps to come up with a useful system that will meet the needs of the users as well as acceptance of such systems [15]. Several authors have also pointed out that RE process improvement should be a team effort (See for example [2, 26]).

Use an evolutionary improvement strategy; [26] points out that it is not realistic to expect organizations to invest a lot of time and money in process improvements whose value is difficult to assess. Therefore, they recommend organizations to introduce small-scale improvements with a high benefit/cost ratio before expensive new techniques. [11] aligns with these statements and points out that, instead of aiming at perfection, it is important to develop a few improved procedures and to get started with implementation.

5 Design of the Systematic Approach to REPI in Ugandan SMEs

Many scholars have emphasized the importance of critical success factors in enabling any process improvement [15, 25]. Therefore these can help to overcome the challenges from the exploratory study. In performing the REPI, the existing generic steps suggested by [4] are followed to manage/effect the SAREPI requirements, more so

they are considered important in the REPI plan as shown in figure 1. While improving RE process the company (in our case SMEs) should follow the following steps to accelerate change and RE process acceptance across the company [4].

Challenges	Derived Requirements	Adopted step	Output
Lack of user involvement	Support User involvement	Define a simple RE process	Simple and easy RE processes
Expensive	Use of evolutionary REPI strategy	Pilot the new RE process	Unnecessary project risks avoided
Resistance to change	Support change management	Adapt the new RE process	Tailored RE process adapted
Lack of skills	Provide training and education	Create awareness and promotion of the new RE process	New RE process is promoted in the organization
Lack of Management support	Encourage management support and commitment	Integrate new RE process within the software product development lifecycle	RE process integrated in the organization

5.1 Requirements for the Systematic Approach

The requirements for the design of the proposed approach to systematic REPI in Ugandan SMEs were derived from the challenges and recommendations observed in the exploratory study. These requirements could be used as measures to overcome the challenges presented in the preceding section. Apart from the derived requirements from the results presented in the preceding sections, we also adopted the steps from the existing generic REPI suggested by [4] to manage the requirements. This is because they are considered important in the REPI plan.

Support user involvement – There is a need to support user involvement in REPI if the new process is to succeed and be institutionalized [11]. User should be involve in the assessment of the current state of the RE process in terms of its strengths and weaknesses. This can serve as starting point for REPI. The assessment made we users build a shared understanding of the improvement goals, planning and practical actions for these SMEs [13]. Users should also be involved while *defining RE processes* – This step follows the assessment of the current RE processes and practices. Simplicity and ease of use in REPI can be a determining factor for any RE process improvement efforts. Involving users in defining simple processes and practices makes it very easy for users to learn and work with the new improved processes, as well as integrating new processes incrementally and gradually.

Use evolutionary improvement strategy – [27] recommend organizations to introduce small-scale improvements with a high benefit/cost ratio before expensive new techniques. Where SMEs are budget constrained then small incremental processes can help in alleviating the problem. This is enabled through *piloting the new small RE process,* i.e. use the evolutionary improvement strategy while piloting the new RE processes in the organization. This will help in avoiding unnecessary project risks that may be caused by rapid changes in the organization. Improved processes and

practices should be introduced gradually and blended with existing practices. This will help to create RE process acceptance throughout the organization.

Support change management –It's important to manage change so as to minimize employee resistance to new and improved RE processes. This can be done by *adapting the new RE process,* i.e. tailor the improved process to the organization. There is need to set clear, quantifiable and measureable REPI benefits if the process is to succeed [2]. The benefits of the improved process should be known to all the team members involved in the software development process through a proper change management plan. The new RE process should be adapted to the needs of the organization and be integrated in the daily routines of the organization.

Support training and education **–** education and training helps to promote the good understanding of the RE process to all the people involved in the improvement process [5]. It's considered to be one of the critical success factors for any RE improvement process. This can be enabled by *creating awareness and promoting the new RE process.* This involves usage of the new RE process benefits to promote its use in the organization and persuading software product development teams to adopt the new RE process and secure the support of senior management. Communicate these practices through face-to-face discussions, staff meetings and newsletters so that everyone in the organization is informed. Organize training and education about the new RE processes to the employees so as to eliminate any chances of resistance to change to the new RE processes.

Encourage management commitment and support **–** Management support to process improvement can be in form of funding, allocation of staff and providing a conducive environment for working [15]. This can be done by promoting systematic use of the new processes throughout the organization.

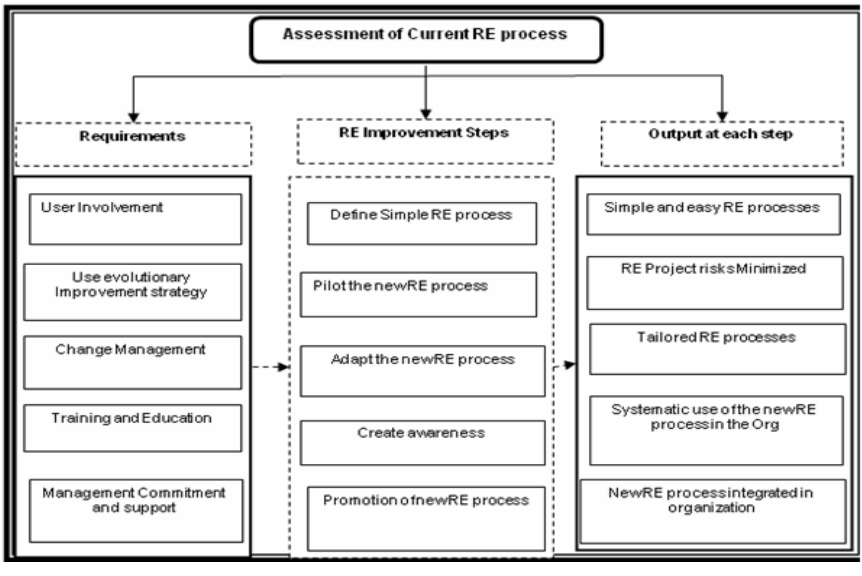


Fig. 1. Systematic Approach to RE Process Improvements (SAREPI)

6 Conclusion and Future Work

In this study we looked at the RE process practices and how these can be improved in order to minimize the number of information system development failures as a result of the RE process. This lead us to look at the current state of SMEs from which we established the challenges they face in RE process improvement, majority of which come from the size and characteristics of SMEs as small budgets, tight deadlines, lack of skills and trained personnel, management reluctance to support the endeavor as well as unclear requirements from users that keep on changing from time to time. We also established recommendations from the SMEs which we used to derive requirements (support user involvement, use evolutionary improvement strategy, support change management, encourage training and education and encourage management support) for the design of the systematic approach to REPI in SMEs that has been lacking for some time now and especially in Uganda. To operationalise the REPI requirements, we also co-opt the REPI steps as seen in the discussion above which should, if well addressed lead to effective output(s). This approach is a methodical approach that is learnable and understandable through a step by step procedure. Therefore it is very important to SMEs that are unable to improve their RE processes due to different reasons, thus should enable these companies to improve their RE processes systematically. However, during the exploratory study we noted that most of the cases visited had little to none defined RE process, though some SMEs were using customized ways of requirements development. It was hard to assess such cases using the more common known RE improvement and assessment models. More so this research focused on exploring possible approach to systematic improvement of requirements engineering processes in SME, thus far as future research, we propose to empirically validate the designed systematic approach in order to verify if it indeed improves the RE process in SMEs.

References

1. Beecham, S., Hall, T., Rainer, A.: Building a requirements process improvement model, Technical Report No. 378 of the department of Computer Science, Faculty of Engineering and Information Sciences. University of Hertfordshire, Centre for Empirical Software Process Research (2003)
2. Damian, D., Zowghi, D., Vaidyanathasamy, L., YogendraPal: An industrial case study of immediate benefits of requirements engineering process improvement at the Australian Center for Unisys Software. Empirical Software Engineering Journal (2003)
3. Davey, B., Cope, C.: Requirements Elicitation – What's missing? Issues in Informing Science and Information Technology (2008)
4. Dominic, T.: Seven steps to achieving better requirements engineering in your organization IBM. Requirements engineering to support your business objects. Rational Software (2009)
5. Francisco, A., Pinheiro, C., Julio, C., do Prado Leite, S., Castro, J.F.B.: Requirements Engineering Technology Transfer: An Experience Report. Journal of Technology Transfer 28, 159–165 (2003)
6. Gonzalez, R.A.: Validation of Crisis Response Simulation within the Design Science Framework. In: ICIS 2009 Proceedings (2009)

7. Hakim, C.: Research Design: Strategies and Choices in the Design of Social Research. In: Bulmer, M. (ed.) Contemporary Social Research: 13, Routledge, London (1987)
8. Hevener, R.A., March, T.S., Park, J., Ram, S.: Design Science in information systems Research. Management Information Systems Quarterly (2004)
9. Hull, E., Jackson, K., Dick, J.: Requirements Engineering, 2nd edn. Springer, Heidelberg (2004) ISBN 1-85233-879-2, Business Media, http://springeronline.com
10. Kamsties, E., Hormann, K., Schlich, M.: Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions, and Technology Transfer. In: Published at a Conference on European Industrial Requirements Engineering (CEIRE 1998), London, UK (1998)
11. Wiegers, K.E.: SoftwaREPI: Ten Traps to Avoid. Process Impact. Software development (1996), http://www.processimpact.com
12. Wiegers, K.E.: Why Is Process Improvement So Hard? Process Impact. Software development (1999), http://www.processimpact.com
13. Kauppinen, M., Kujala, S.: Starting Improvement of Requirements Engineering Processes: An Experience Report (2001)
14. Kauppinen, M., Tapani, A., Kujala, S., Laura, L.: Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice; Helsinki University of Technology; Software Business and Engineering Institute (2001)
15. Kauppinen, M., Vartiainen, M., Kontio, J., Kujala, S., Sulonen, R.: Implementing requirements engineering processes throughout organizations: success factors and challenges. Information and Software Technology 46, 937–953 (2004)
16. Khaled, E., Nazim, H.M.: A Field Study of Requirements Engineering Practices in Information Systems Development. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering, York, England (1995)
17. MCfeeley, B.: IDEAL: a user's guide for softwaREPI, Handbook CMU/SEI-96-HB-001, Software Engineering Institute, Carnegies Mellon University, Pittsburgh, PE, USA (1996)
18. Mishra, D., Mishra, A.: SoftwaREPI in SMEs: A Comparative View. ComSIS 6(1) (2009)
19. Niazi, M.K.: Improving the Requirements Engineering Process through the Application of a Key Process Areas Approach. In: Australia Workshop on Requirements Engineering (2002)
20. Nikula, U., Sajaniemi, J., Kälviäinen, H.: A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises. Telecom Business Research Center Lappeenranta. Research Report (2000)
21. Nikula, U., Sajaniemi, J., Kalviainen, H.: Management View on Current Requirements Engineering Practices in Small and Medium Enterprises. In: Proceedings of The Australian Workshop on Requirements Engineering (2000)
22. Olsson, T., Doerr, J., Koenig, T., Ehresmann, M.: A Flexible and Pragmatic Requirements Engineering Framework for SME. In: Proceedings of SREP 2005, Paris, France (2005)
23. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Published in Journal of Management Information Systems 24(3) (2007)
24. Pino, F.J., Garcia, F., Piattini, M.: SoftwaREPI in small and medium software enterprises: a systematic review. IEEE, Los Alamitos (2007)
25. Qadir, M., Asghar, I., Ghayyur, A.K.: Scaling of Critical success factors for Requirements engineering in the development of Large Scale Systems. International Journal of Reviews in Computing (2009)
26. Sawyer, P., Sommerville, I., Viller, S.: Improving the Requirements Process. Cooperative Systems Engineering Group Technical Report Ref. In: The Fourth International Workshop on Requirements Engineering: Foundation for Software Quality, Pisa, Italy (1998)
27. Sawyer, P., Sommerville, I., Viller, S.: Requirements Process Improvement through the Phased Introduction of Good Practice. SoftwaREPI and Practice (1997)

28. Simon, A., Alain, R., Naji, H.: OWPL: A Gradual Approach for Software Process Improvement In SMEs. In: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA 2006) (2006)
29. Small and Medium Enterprises (SMEs) Business Guide. Uganda Investment Authority (March 2008)
30. Sommerville, I., Ransom, J.: An Empirical Study of Industrial Requirements Engineering Process Assessment and Improvement. ACM Transactions on Software Engineering and Methodology 13(1), 85–117 (2005)
31. Sommerville, I.: Software Engineering, 6th edn. Addison-Wesley, Reading (2001)
32. Sommerville, I.: Integrated Requirements Engineering: A Tutorial. IEEE Software (2005)
33. Wohlin, C., Gustavsson, A., Höst, M.: A Framework for Technology Introduction in Software Organizations. In: Proceedings SoftwaREPI Conference, Brighton, UK, pp. 167–176 (1996)

Understanding the Dynamics of Requirements Process Improvement: A New Approach

A.S. (Aminah) Zawedde[1], M.D. (Martijn) Klabbers[2], D. (Ddembe) Williams[3], and M.G.J. (Mark) van den Brand[4]

[1] Faculty of Computing and Informatics Technology, Makerere University, Uganda
[2] LaQuSo, Laboratory for Quality Software, Eindhoven University of Technology
[3] Decision Innovation Systems, Limited, Uganda
[4] Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands
sazawedde@gmail.com, M.D.Klabbers@laquso.com, ddembe.williams@gmail.com, m.g.j.v.d.brand@tue.nl

Abstract. Many software development organizations invest heavily in the requirements engineering process programmes, and with good reason. They fail, however, to maximize a healthy return on investment.

This paper explores factors that influence requirements process improvement (RPI) with the aim to explain how the attributes of the underpinning process affect both the quality and associated costs of the requirements specification delivered to the customer. Although several tools and techniques have been proposed and used for RPIs, many lack a systematic approach to RPI or fail to provide RPI teams with the required understanding to assess their effectivity.

The authors contend that the developed quality-cost RPI descriptive model is a generic framework, discipline and language for an effective approach to RPI. This descriptive model allows a systematic enquiry that yields explanations and provides RPI stakeholders with a common decision making framework. The descriptive model was validated by practicing process improvement consultants and managers and makes a contribution towards understanding of the quality-cost dynamics of RPI. To address the acknowledged deficiencies of RPI, the authors further suggest a generic RPI model and approach that integrates statistical process control (SPC) into system dynamics (SD). The approach enables RPI teams to steer for a cost-effective and successful RPI.

Keywords: cost, quality, Requirements process improvement, system dynamics, statistical process control.

1 Introduction

Software systems project failures are partly a result of an inefficient requirements process improvement (RPI). RPI in such projects is characterized by ever changing requirements throughout the development process [8,20]. Requirements engineering (RE) practitioners are faced with the challenge of making informed

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 276–290, 2011.
© Springer-Verlag Berlin Heidelberg 2011

decisions on the effectiveness of carrying out any process improvement. Recent practical experience indicates that RPI process attributes are interlinked and dynamic, and should be analyzed in the context of the whole process instead of analyzing them in pairs. The objective of such analysis is to identify the interdependence and feedback structure amongst the process attributes [3,24,20].

It is important to note that methods currently used by practitioners for RPI do not capture the feedback relationships amongst RPI attributes. Many RPI practitioners instead focus on dominant requirements relationships while the underlying structure that gives rise to requirements volatility behavior is not explored [8,3,24]. This may result in deterioration of the end product's performance, and hence leads to software project failure or escalated post implementation system costs.

This paper builds a case for integrating System Dynamics (SD) and Statistical Process Control (SPC) to enhance the understandability and practicability of RPI of measuring the effectiveness of RPI over time. SD has been acknowledged for its ability to facilitate the understanding of the behavior of complex dynamic problems [9,24] while SPC is used by many process improvement practitioners to monitor and control the process variability [2]. The complementary advantages of combining the two methods lie in their ability to visualize feedback through the use of analytical tools in SD and control charts supported by SPC. The visualization enables the process stakeholders to talk about and discuss the emerging system behavior as a basis for process improvement. Visualizing these feedback interactions is a prerequisite in helping the RE stakeholders gain shared understanding of the decisions made in achieving effective requirements for process improvement.

The rest of the paper is structured as follows: Section 2 provides an overview of the methods currently used for process improvement and highlights the success factors of RPI; Section 3 is a discussion of the interactions amongst the factors that influence RPI; Section 4 discusses the field findings from RPI in regard to the RPI variables used in practice; Section 5 explains the dynamics of RPI through a practical case study; and Section 6 discusses further research directions and the suggested benefit of combining SD and SPC for enhancing RPI.

2 State of Practice in Process Improvement

A number of process improvement models exists, namely: Requirements Capability Maturity Model (R-CMM) [3], Software Capability Maturity Model (CMM) [17], Capability Maturity Model Integrated (CMMI) [11], Requirements Engineering Process Improvement Model (REPSIM) [24], Bootstrap [16], Trillium [1], and Software Process Improvement and Capability Determination (SPICE) [7]. However apart from R-CMM, CMMI and REPSIM, all the models that have been mentioned above are software process improvement models. The review of the literature reveals that the SPI models have both workable and economic potential for software process improvement but they do not adequately address requirements process improvement [16]. This paper will only discuss R-CMM, CMMI and REPSIM that are relevant for RPI.

R-CMM and CMMI use a capability based approach for process improvement. Although these models have been used in large organizations for process improvement, they are generalized and can not be used to define the actual benefits gained from the improvements for a specific organization [15]. It is therefore considered to be superficial for smaller organizations and the improvements are usually difficult to measure or implement [4,15]. In practice real systems are complex and their RE processes are also complex and dynamic; process variables interact amongst each other depicting feedback relationships that cut across all levels of organizational processes [24]. In addition, R-CMM is heavily dependent on organizational learning and therefore organizations that are at an initial stage of learning may not be able to measure the effectiveness of RPI. Given this scenario, there is need for an approach that is able to improve the analytical strengths of R-CMM and CMMI. Coupled with the above drawback, Ojala [16] asserts that capability based improvement only is not enough to start process improvement work. To be effective, process improvement methods also need to take into account costs created in processes and the need to measure capability and effectiveness of processes.

REPSIM [24] was used to predict cost, understand the cause and effect between processes and explain the impact of the resulting interrelationship, however, like [3] its supporting dynamic prescriptive model was not developed. Even though REPSIM is a more promising model for enhancing into a generic tool for RPI systems than R-CMM and CMMI due to its ability to model the RE process in a holistic manner and cutting across all levels of the organizational maturity, it has not been applied in an industrial setting. The term holistic analysis is drawn from the concept of Holism which stipulates that the properties of a given system can not be determined by its component parts alone; the system as a whole determines how the parts behave [18].

Given the relative strengths and deficiencies in the current RPI models, to successfully achieve a better understanding, an appropriate RPI model should primarily have the ability to capture feedback resulting from interaction amongst the requirements engineering processes. Secondly, the model should be able to support evaluation and synthesis of the requirements processes in a manner that promotes explanation and insight into the problem under investigation in order to improve communication amongst the requirements engineering stakeholders. Thirdly, given the above two conditions, the model should carry out a holistic analysis of the RPI process variability. This holistic analysis is key to understanding the dynamics of RPI and achieving a cost-effective state of the system under development.

2.1 System Dynamics Modeling

System Dynamics (SD) is an approach for modeling and simulation of dynamic behavior of complex systems over time [9,13]. The complexity of a system is defined by feedback loops, non-linearity and time delays that often affect the system behavior. System dynamics models being a representation of real world situations are well suited to offer explanation and generate insights into the root

causes of the behavior of complex systems. The insights generated can facilitate informed decision making before any improvements can be implemented [21,6]. SD adopts the big picture strategic viewpoint in capturing the overall system structure rather than that of the individual parts of the system [13,18]. Various techniques and methods have been used to model complexity but their main drawback is that analysis becomes complex with large models [6].

System dynamics analytical tools have the ability to handle large models, identify independent loop sets within models that determine particular behavior resulting from the interaction among model components [6,10,9]. SD has an advantage over statistical models because of its ability to incorporate in the model important soft variables, and also yield explanation and foster understanding of systematic problems [21].

2.2 Statistical Process Control

Statistical Process Control (SPC) is a statistical technique used in monitoring processes that are in control using means and ranges in a chronological order and shows how the values change over time. SPC determines the stability or instability of a process by discriminating between common cause variation and assignable cause variation. Common cause variations result from normal interactions among people, machines, environment, or techniques used while assignable cause variations arise from events that are not part of the process and make it unstable [5,2]. By using few data points, SPC is able to dynamically determine an upper and lower control limit of acceptable process performance variability. These control limits act as signals or decision rules, and give process improvement teams information about the process and its state of control. SPC has been shown to be effective in manufacturing contexts and has been extended successfully to software processes as well [2,17,5]. SPC uses several control charts together with their indicators through visualization to: establish operational limits for acceptable process variation; monitor and evaluate process performance evolution in time.

The control charts aid in suggesting whether corrective action can be taken to improve quality or in assurance that a process has the satisfactory quality [10]. The output from the control charts helps stakeholders in predicting results which enable preparation of achievable plans, meeting cost estimates, scheduling commitments and delivering the required product quality within the acceptable limits [5].

3 Factors that Influence Requirements Process Improvement

To be able to demonstrate the plausibility of combining SD and SPC, the factors that influence the RPI were identified and relationships among them defined. The reference model of factors influencing RPI is based on the requirements engineering and process improvement literature [23,24,19]. Eight (8) commonly

used variables were identified that influence RPI: (a) Productivity of Engineers, (b) Process Capability Index, (c) Process Improvement Capability, (d) Process Effectiveness, (e) Process Improvement Costs, (f) Errors Observed, (g) Perceived Effectiveness, and (h) Process Rigor. The importance of the preceding variables and why they were considered key for the reference model for RPI is justified below. Rico [19] asserts that to attain effective process improvement, it is inevitable to ignore productivity gains. The same author asserts that there is a strong relationship between productivity and process rigor. He goes further to emphasize that without improving process rigor, there will be minimal productivity gains. He however, acknowledges that improving productivity is a complex process that goes beyond process rigor.

Rico [19] and Starz [23] note that higher numbers of errors observed result into very high process improvement costs but can be improved by enhancing productivity of requirements engineers. Williams [24] and Starz [23] argue that practitioners, as a way of decreasing process improvement costs and increasing process effectiveness have to ensure that the RE processes that they use are capable of carrying out an activity. This therefore makes the process capability index and process improvement capability key to ensuring minimization of process improvement costs and enhancing process and perceived effectiveness.

All the three authors [23,24,19] concur that it is those eight variables discussed above that are most important for requirements process improvement considering that process improvement is a factor of cost, time and resources and it is these variables that determine these factors.

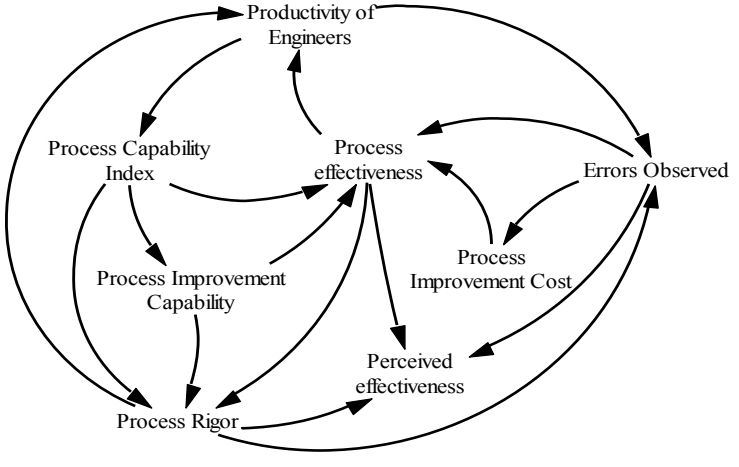


Fig. 1. A Reference Model of the Factors that influence RPI

Figure 1 is a visualization of the interrelationship amongst the RPI variables that are mentioned. A brief description of the RPI variables is as follows:

- (a) Productivity of Requirements Engineers: This is the rate at which requirements engineers deliver requirements specifications as agreed upon by the requirements engineering stakeholders [24].
- (b) Process Capability Index: This is the potential of a process to meet its specifications. The higher the index, the more capable the process is.
- (c) Process Improvement Capability: The extent to which a process can meet customer requirements, specifications or product tolerances.
- (d) Process Effectiveness: The is the measure of the extent to which the process goals and objectives are met.
- (e) Process Improvement Cost: This refers to the total cost of resources in terms of wages, documentation, training technology and initial set-up costs.
- (f) Errors Observed: This is the total number of defects identified by the requirements engineering stakeholders during the review process.
- (g) Perceived Effectiveness: This is the level of effectiveness achieved by the RPI team.
- (h) Process Rigor: This is the level of thoroughness adhered to established standards when carrying out or implementing process improvements.

The validity of the reference model projected in Figure 1 was tested through interviews of ten (10) requirements process improvement experts from different companies. The objective of the interviews was to confirm the relevance of the quality-cost variables considered for RPI in practice as summarized in Table 1. The identity of the companies has not been disclosed due to non disclosure agreements.

In the next section the results of validation of the RPI factors in Figure 1 are presented.

4 Modeling Requirements Process Improvement Dynamics

Validation of the model allows a large number of data collection techniques. For this research we used a semi-structered interview technique because it is the most efficient method when the number of experts to be interviewed is small (0 to 20) and the topic that is to be investigated is not multi-disciplinary [12]. It was considered that a consultation through interviews was the most efficient way of gathering information for validation of the model of RPI factors due to the limited number of available experts in requirements process improvement and given that the nature of the research is still in its infancy.

Semi-structured interviews were used as a strategy for data collection. Discussions with the RPI experts were conducted in a setting of maximal 2 hour interviews. Respondents included quality managers, process improvement managers, and system analysts among others, and they all had experience with requirements engineering. The interviews included questions about the methods used for RPI in practice, problems faced in using existing methods and techniques, the parameters considered for RPI in practice, as well as the measurements for the RPI parameters.

The above validation process took into consideration the threats to model validity. Some practitioners may not appreciate the value of combining the theoretical grounding of RPI with that of systems thinking and complexity theory. The practice has been to use work breakdown structures and the capability maturity models. To mitigate this factor, the assumptions made for the proposed approach are firmly grounded in RPI. Furthermore, changes in the interview instrument may produce changes in the outcome. This was mitigated by having a uniform agenda for the meetings with the RPI experts as well as standardizing the interview guides.

Table 1 shows the level of importance perceived by each expert in relation to RPI variables as experienced and used in practice. The positive signs (+) indicate that the expert acknowledged the variable as relevant for RPI in practice whereas the negative signs (-) indicate that the experts did not acknowledge the variable as important for RPI in practice.

Table 1. Summary of results from RPI Experts

RPI Experts' Job Descriptions	Errors Observed	Process Effectiveness	Perceived Effectiveness	Process Improvement Cost	Productivity of Engineers	Process Improvement Capability	Process Capability Index	Process Rigor	Customer Satisfaction	Management Commitment
1. Senior Scientist at A	+	-	-	+	+	-	-	-	+	+
2. Quality Manager at B	-	+	-	-	-	-	-	-	+	+
3. Senior consultant at C	-	-	-	-	-	-	+	+	-	+
4. Group Leader Software Testing & SDE at D	-	-	-	-	-	-	+	+	-	+
5. Consultant quality and process improvement at E	+	-	-	+	+	-	+	+	+	-
6. Software Development Manager at F	+	+	-	-	-	+	+	-	+	+
7. Managing Consultant at G	+	+	-	+	+/-	-	-	-	+	+
8. System Analyst at H	+	-	-	-	-	-	-	-	+	-
9. Process Improvement Manager at I	+	-	-	-	-	-	+	+	+	-
10. Change Manager at J	-	-	-	-	-	-	-	-	-	+
Total number of positives	6	3	0	3	2.5	1	3	4	7	7

Out of the 8 high level factors that were tested in the field, 6 variables were found to be essential in the RPI practice. Management Commitment and Customer Satisfaction were raised by all the interviewed RPI experts, as variables for RPI but they were not among the variables that were captured from RPI literature. Customer Satisfaction (70%), Management Commitment (70%) and Errors observed (60%) are the most influential parameters considered for RPI. These were followed by Process Rigor (40%), Process effectiveness (30%), Process Capability Index (30%) and Process Improvement Cost (30%). However, most experts considered Process Effectiveness as a component within Process Capability Index. Therefore for this research, Process Effectiveness will not be considered as one of the key variables for RPI. Productivity of Engineers (25%)

and Process Improvement Capability (10%) are of less importance in RPI according to the specialists.

Results from field studies indicate that Customer satisfaction, Management Commitment and Errors Observed are the most relevant variables for RPI in practice.

4.1 Descriptive Model for RPI

A descriptive model is a qualitative representation of the RPI process map that will in future work be translated into quantitative functional relationships using mathematical equations. The validated model of factors influencing RPI as presented in Table 1 facilitated the development of the feedback descriptive model by redefining the relationships among factors as well as determining their measures and values. Table 2 presents the measures for all variables considered and indicates the variables that were obtained from field studies and those that were obtained from RPI literature.

It should be noted that Customer satisfaction: the degree to which customers expectations of a product are met or surpassed and Management Commitment: the continuous support and involvement of executive management based on acknowledged benefits in the implementation and maintenance of a system under development, were not captured in RPI literature as important variables for RPI. This research adopts these two variables as some of the most relevant variables for RPI based on the field studies findings.

Figure 2 provides a systemic and descriptive view of the variables that influence decision making of teams during the RPI. It illustrates the revised descriptive model of the key variables that experts confirmed they considered important when carrying out RPI. In system dynamics a descriptive model is a causal loop diagram indicating the direction and polarity (+/-) of each loop [24]. The revised descriptive model has three (3) Balancing loops and six (6) Reinforcing loops. A Reinforcing feedback loop (R) represents growth or declining actions while a balancing feedback loop (B) is a goal seeking loop that seeks stability or return to control [24]. The loops show the interrelationship between the variables and how they link to each other. The linkages depict how the loop structures drive system behavior and the variables involved for each loop structure impacting on

Table 2. Measure of RPI variables for the revised descriptive model

Variables	Measure	Units	Variable obtained from Field (Yes/No)
Productivity of Engineers	No. of specifications/Engineer/Day	No. of Specifications	No
Process Capability Index	Unitless	0-1	No
Management Commitment	Unitless	0-1	Yes
Process Improvement Cost	Currency	US Dollars	No
Process Rigor	Unitless	0-1	No
Customer Satisfaction	Unitless	0-1	Yes
Errors Observed	Errors	Errors	No

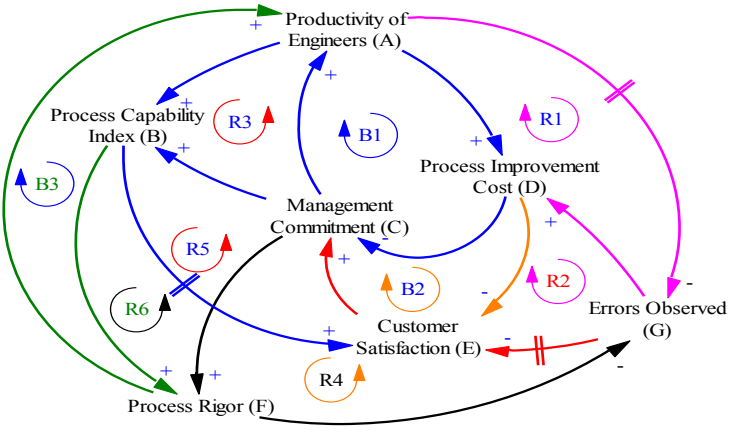


Fig. 2. The Descriptive Feedback Model for RPI

the system. Analyzing the behavior of each of these loops is key to understanding the impact of changes in one or more variables on system behavior and the limits within which cost effectiveness can be achieved for a set of variables.

An example of one of the loops that illustrates RPI is R6 that goes through A-B-F-G-D-C-A. It is a reinforcing loop where an increase in the the Productivity of Engineers (A) will cause an increment in the Process Capability Index (B) which in turn will increase the Process Rigor (F). An increase in the Process Rigor will cause a reduction in the Errors Observed (G) which will in turn reduce the Process Improvement Cost (D). The reduction in costs will in turn trigger the interest of Management Commitment (C) feeding back into further increasing the process productivity of requirements engineers (A).

This demonstrates the need to analyze these feedback loop structures in a holistic manner. The double line marks on some of the relationship arrows in Figure 2 indicate that there is a time delay between the variables. These indicate that the effect of that relationship is obtained over time. For continuous process improvement the reinforcing feedback loops have to be balanced by the balancing loops to avoid causing a system burn out or end in chaos.

The descriptive model in Figure 2 demonstrates that System dynamics uses feedback loops and time delays to represent complex systems. Causal loop diagrams representing feedback loops are used for conceptualizing the system structure of a complex system and for communicating model-based insights. These feedback descriptive models have explanatory power and insight to aid understanding to RE stakeholders.

Despite the difficulty in most approaches to convert qualitative data into quantitative information, this is a key strength of the system dynamics methodology. Inherent to the system dynamics methodology is the ability to convert qualitative data into quantitative values to aid better understanding [24] and this has been seen over the years as the key strength and advantage of using the system dynamics methodology [9]. System dynamics techniques [9] and

procedures make it possible to effectively model quantitative data based on qualitative models. This is done methodically starting with mental models through causal link relationships and then use of reference modes.

Analysis of the descriptive model offers a useful basis for identifying a number of propositions that can be drawn from it and suggests how they may be tested.

4.2 Propositions Derived from the Model

A system dynamics descriptive model offers a basis for capturing the mental models of process improvement teams and facilitates the understanding of their decision making strategies during the process improvement. There is no theory or research that may relate directly to the mental model of process improvement teams nor to the impact of process improvement on the mental models of process improvement teams [24].

As a tool the model can be used by practicing RPI managers and researchers in a training and learning situation. The model presented in Figure 2 offers a useful basis for research on RPI. Review of the process improvement literature suggests that differences in organizational culture may influence the nature of process improvement. Coupled with management commitment, process improvement teams who feel empowered to make decisions are less likely to experience a decline in engineer productivity than teams that feel isolated and unempowered. The following propositions can be derived from Figure 2.

Proposition1: The errors observed in the requirements specification are likely to increase during the process improvement.

During the RPI period, both the customers and process improvement teams have to take time to scrutinize the correctness of the requirements specification. According to the model, the period of time taken between reviews produces a temporary increase in errors observed.

Proposition2: The more process rigor in the organization's process improvement, the greater the decline of errors observed.

The more standards are strictly applied and novel technological tools and methods are applied during the process improvement the less the errors are observed in the process improvement document.

Proposition3: The extent of decline in the process capability index is dependent on the productivity of engineers and the level of management commitment.

The more requirements engineers are knowledgeable of tools and applied standards, the level of the resulting process capability index will be dependent on the productivity of engineers and the level of management commitment.

Proposition4: The extent of decline in the productivity of engineers will be moderated by the technological rigor of tools and commitment from senior managers with the organization.

The productivity of engineers will decline during their training for new methods or techniques. However, this decline will be mitigated by the technological support and motivation as a result of management commitment.

Proposition5: The extent of decline in management commitment to RPI will be dependent on the process improvement costs, and the productivity of engineers and feedback from customer satisfaction.

The decline in management commitment to RPI is dependent on process improvement costs, productivity of engineers and feedback from customer satisfaction.

Proposition6: The extent of decline in process improvement costs is dependent on the number of errors observed and productivity of engineers.

During the period of process improvement, the decline in process improvement costs is dependent on number of errors observed and productivity of requirements engineers.

Proposition7: The extent of decline in customer satisfaction is dependent on errors observed during a review and the level of process capability index.

The process improvement team in organizations that learn from their past project experiences are likely to have a higher process capability index and therefore higher process capability level. This leads to capturing errors included in the requirements specification during the review process. On the reverse customer satisfaction is likely to be low if the requirements engineers (process improvement team) encounters an increment in the number of errors observed.

In order to illustrate the above propositions and to show how the descriptive model can be applied, we will demonstrate it in a real life case study. In the following section we discuss a practical case study in relation to RPI. In order to actually validate the model and test feedback relationship, the authors intend to carry out a programme of research to confirm or reject the above propositions in the near future.

5 A Practical Case Study

In order to illustrate the applicability of the proposed feedback approach and to confirm some of the propositions in section 4.2, a practical case study was undertaken through a post mortem analysis of a real case study. The case study, a small project for a management information system receives a seemingly successful RPI and shows the consequences of lack of insight in RPI.

A consultancy firm's RE process is embedded in an iterative project based software development process. In a number of iterations the business requirements are detailed into user requirements and use-cases and further refined in a design, implementation, and tests phases. Where possible, design, implementation and tests are offshored. Dependent on the customer's input for the

specific project, RE steps can be skipped. The input, however, is always checked and improved where necessary. Generally, the first iteration produces an initial architecture which guides the primary assignment of articulating business requirements to the next specific iterations.

The firm decided to apply RPI that should result in (1) better quality assurance (QA), (2) generate insight into the traceability of business requirements to design, (3) be compliant to the firm's interpretation of the ISO 9126 non-functionals, and (4) understandable use cases for the customer. The QA was supported by a strict set of guidelines and related checklists.

This case study is one of the first projects executed with the improved RE process; an offshored project aimed at replacing an existing Dutch Government Budget Distribution System. The firm detailed the user requirements as delivered by the customer using 3 iterations into a set of complex use-cases. The in-company offshore team, implemented and tested the system, based on translated use cases and the existing distribution system, including source code and documentation. Finally, the system was tested and accepted by the customer.

After the system's deployment the project was evaluated using unstructured interviews with the RE stakeholders including the customers, and assessment of the requirements' quality. The interviews included a number of fixed RPI variables described in Figure 2. Furthermore, we assessed the number of errors using LaQuSo's Software Product Certification Model (LSPCM [14]). LSPCM is a check based rule method for assessing defects in software artifacts. Similar to the Independent Verification and Validation (IV&V) techniques it provides a stable basis to solve conflicts of interest and does not immediately suffer from a weakened analysis outcome due to time pressure. LSPCM checks technical aspects of software artifacts, but leaves the assessor (in this case computer science students) freedom to determine the appropriate, elegant, and fastest way of executing the check rules [22]. Note that LSPCM is also a fruitful addition to our proposed approach.

Based on the firm's intended improvements and the firm's approved assessment techniques in this project, the test cases focused on 5 of the 7 RPI variables of the descriptive model in Figure. 2. These variables are measured as follows: *(A)Productivity of REs* the delivered use cases and non-functionals as requested by the customer/stakeholders measured through the documents' traceability and customer's and REs' statements about the use-cases in the interviews; *(C)Management Commitment* Support and involvement of executive management as experienced by REs through the interviews. *(D)Process Improvement Costs* as additional hours for the required improvements; *(F)Process Rigor* in what standards and templates the IT experts at the firm were using as stated in the interviews. *(G)Errors Observed* as the defects in the requirements (assessed by using LSPCM) and reported bugs.

Figure 2 shows the 5 RPI variables and related loops that they form. Their paths are resp.: *(B1)*:A-D-C-A,*(B2)*:A-D-E-C-A, *(B3)*:A-B-F-A, *(R1)*:A-G-D-C-A, *(R2)*:A-G-E-C-A, *(R3)*:A-B-E-C-A, *(R4)*: F-G-D-C-F, *(R5)*: B-E-C-B, *(R6)*: B-F-G-D-E-C-B. Note that there are 3 balancing loops and 4 reinforcement loops. In

a real case these loops are important to monitor because they give an early insight where important bottlenecks are that endanger the success of the RPI. Balancing loops are the most demanding since they typically include variables operating between lower and upper boundaries, whereas reinforcement loops only deal with minimum or maximum boundaries hence the usefulness of the proposed integration with SPC. Below we demonstrate what we found on the evaluated variables:

- ad.*(A) in B1,B3,R1,R2,R3,R5* LSPCM assessments showed that traceability matrices were incomplete, use cases were as stated by the customer "too complex", and non-functional requirements like maintainability were not specified. Productivity did not score highly;
- ad.*(C) in B1,B2,R1,R2,R3,R4,R5,R6* Requirements engineers (REs) did not report any executive management involvement in the interviews other than the explicit focus on accuracy of the system. Reason was that wrong budget calculations would have resulted in a large claim against the firm;
- ad.*(D) in B1,B2,R1,R6* The additional process improvement costs were estimated and realized to be no more than a small percentage (1%);
- ad.*(F) in B3,R4,R6* The interviews with the project's expert REs revealed that the new firm's standards were perceived as guidelines for novice REs; skilled REs could deviate from them, provided that the documents were uniform and understandable to RE and developers. Time pressure made the customer countersign the use cases without understanding them. Budget constraints prohibited QA to find these anomalies.
- ad.*(G) in R1,R2,R6* In addition to observation in *(A)*, LSPCM revealed errors in the use cases and non-functionals. The customer observed at least one major bug in the little extra functionality that was added to the existing system.

The customer perceived the project as a success in delivered functionality and time/costs in the short-term. An explanation for this success can be found in the fact that the offshored development had full access to the existing system; both in source code and in produced results. Furthermore, the set of new functionalities was very small hence the lack of return on investment in process improvement. In the long-term, maintainability of the system could become a problem leading to system failure or escalating maintenance costs. One of the indicators to this was that later, the tender for the system's maintenance was assigned to the firm which had the lowest bid. In addition, the Dutch Government declared to look into maintainability more critically in response to the high maintenance costs of governmental software systems. Nevertheless, the belief of a successful RPI was not contradicted by any feedback and it was generally accepted within the firm.

With the proposed approach, an early warning system based on empirical boundaries and thresholds would have shown a number of problems: RE productivity *(A)* did not increase, against expectations, the higher number of errors observed *(G)* in the requirements by using LSPCM. A low productivity *(A)* could have been acceptable in *(B1)* because it would provide lower costs *(D)*. However, the reinforcement loop *R2* without *(A)* should show progress, but because of the errors *(G)*, that would not be the case. The combination of both

measuring RPI variables and providing a visual feedback to REs and managers, would have given them the real insight to the successfulness of the RPI and the opportunity to adjust it accordingly.

6 Summary and Further Research Directions

The contribution of this paper has been fourfold; (i) a literature survey that helped identify the gap that exists with current RPI tools in capturing the feedback interaction amongst systems requirements engineering processes; (ii) the proposal of a new approach to understanding the dynamics of RPI; (iii) validating the key variables for RPI in practice and the emerging feedback loops; and (iv) using a Practical case study to demonstrate the importance of holistic analysis to RPI and to verify some of the propositions stated in this paper.

At this initial stage we have been able to identify key variables and their feedback loops that are responsible for the software systems project behavior. In the immediate future, work on a full fledged system dynamics model will be developed to be able to determine the strengths of the relationships and thus the relative importance of each key variable discussed in this paper. We have further proposed how the gap with existing RPI tools can be closed by building a case for combining SD and SPC to attain an effective process improvement framework.

SPC can augment SD's ability to identify variability in requirements engineering process variables during process improvement [10]. Combining SD and SPC has been done for manufacturing [10] and software processes [5] but not in the requirements process improvement context.

Integrating SPC within SD is expected to help RE stakeholders gain insights about the RPI feedback structure. Furthermore, it is also expected to help carry out assessment of the potential effects of the high leverage points on the stability of the interacting requirements processes directly from the control charts leading to attainment of cost effective systems under development. More still, RE stakeholders will have ownership of the RPI process together with its results, and implement the changes recommended by the tool [10]. This will enhance the understanding of the causes of the process variations [5,10].

Further research direction will endeavour to integrate SPC within SD through initially creating a system dynamics model, then integrating the process of quality-cost variances and determining control limits using control charts in SPC.

References

1. April, A.A., Coallier, F.: Trillium V3.0: A Model for the Assessment of Telecom Software System Development Capability. In: 2nd International SPICE Symposium ASQRI, Griffith University, Australia, pp. 79–88 (1995)
2. Baldassarre, M.T., Boffoli, N., Caivano, D., Visaggio, G.: Managing software process improvement (SPI) through statistical process control (SPC). In: Bomarius, F., Iida, H. (eds.) PROFES 2004. LNCS, vol. 3009, pp. 30–34. Springer, Heidelberg (2004)

3. Beecham, S., Hall, T., Rainer, A.: Defining a Requirements Process Improvement Model. Software Quality 13, 247–279 (2005)
4. Brinkkemper, S., van de Weerd, I., Saeki, M., Versendaal, J.: Process improvement in requirements management: A method engineering approach. In: Rolland, C. (ed.) REFSQ 2008. LNCS, vol. 5025, pp. 6–22. Springer, Heidelberg (2008)
5. Caivano, D.: Continuous Software Process Improvement through Statistical Process Control. In: Proceedings of 9th European CSMR 2005 (2005)
6. Clempner, J.: A Hierarchical Decomposition of Decision Process Petri nets for Modeling Complex Systems. Int. J. Appl. Math. and Comp. Science 20(2), 349–366 (2010)
7. Dorling, A.: SPICE: Software Process Improvement and Capability Determination. Software Quality Journal 2(4), 209–224 (1993)
8. Ferreira, S., Collofello, J., Shunk, D., Mackulak, G.: Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. Systems and Software 82(10), 1568–1577 (2009)
9. Forrester, J.W.: System Dynamics and a lesson for 35 years. Sloan School of Management. MIT, Cambridge (1991)
10. Georgantzas, N., Fraser, J., Tugsuz, E.: Bipartisan process improvement in polymer coating: Combining SD with SPC. System Dynamics 2, 502–511 (1995)
11. Gibson, D.L., Goldenson, D.R., Kost, K.: Performance Results of CMMI-Based Process Improvement. Pittsburgh, SEI (August 2006)
12. Gustafsson, T., Ollila, M.: Expert consultation in the preparation of a national technology programme. Systems Analysis Laboratory, Helsinki University of Tech. (September 2003)
13. Harris, B., Williams, B.: System Dynamics Methodology. W.K. Kellogg Foundation (2005)
14. Heck, P., Klabbers, M.D., Van Eekelen, M.: A software product certification model. Software Quality Journal 18(1), 37–55 (2010)
15. Linders, B.: Building Process Improvement Business Cases Using Bayesian Belief Networks and Monte Carlo Simulation,Tech. Report, CMU/SEI-2009-TN-017 (2009)
16. Ojala, P.: Combining Capability Assessment and Value Engineering: A BOOTSTRAP Example. In: 5th Int. Conference of Profes, Kansai City, Japan (2004)
17. Paulk, M.C.: Applying SPC to the Personal Software Process. Proc. 10th Intl. Conf. Software Quality (October 2001)
18. Rafferty, M.: Reductionism, Holism and System Dynamics. In: 8th SD PhD Colloquium (2007)
19. Rico, D.F.: ROI of Software Process Improvement: Metrics for Project Manaers and Software Engineers (2004) ISBN 1-932159-24-X
20. Ruhe, G., Eberlein, A., Pfhal, D.: Quantitative WinWin- A New Method for Decision Support in Requirements Negotiation. In: SEKE, Italy (July 2002)
21. Saurabh, K.: Software Development and Testing: A System Dynamics Simulation and Modeling Approach. In: 9th WSEAS-SEPADS, UK, February 20-22 (2010)
22. Serebrenik, A., Mishra, A., Delissen, T., Klabbers, M.D.: Requirements certification for offshoring using LSPCM. In: 7th IEEE Computer Society's QUATIC 2010, September 29-October 2 (2010)
23. Statz, J.: Measure of Process Improvement. Technical Report on Practical Software and Systems Measurement (2005)
24. Williams, D.: Challenges of System Dynamics to Deliver RE Projects: faster, better and cheaper. In: 21st Int. Conf. of the System Dynamics Society (2003)

Precise vs. Ultra-Light Activity Diagrams - An Experimental Assessment in the Context of Business Process Modelling

Francesco Di Cerbo[1], Gabriella Dodero[1], Gianna Reggio[2], Filippo Ricca[2], and Giuseppe Scanniello[3]

[1] CASE, Libera Università di Bolzano-Bozen, Italy
{francesco.dicerbo,gabriella.dodero}@unibz.it
[2] DISI, Università di Genova, Italy
{filippo.ricca,gianna.reggio}@disi.unige.it
[3] Dipartimento di Matematica e Informatica, Università della Basilicata, Italy
giuseppe.scanniello@unibas.it

Abstract. UML activity diagrams are a commonly used notation for modelling business processes in the field of both workflow automation and requirements engineering. In this paper, we present a novel precise style for this notation. Further, the effectiveness of this style has been investigated in the context of the modelling of business processes through a controlled experiment conducted with master students in Computer Science at the Free University of Bolzano-Bozen. The results indicate that the subjects achieved a significantly better comprehension level when business processes are modelled using the precise style with respect to a "lighter" variant, with no significant impact on the effort to accomplish the tasks.

Keywords: Business Process modelling, UML activity diagrams, Controlled experiment, Precise and Ultra-light styles.

1 Introduction

In the last years, many organizations have been changing their business processes to be competitive in the global market [8]. In this context, modelling, management, and enactment of business processes are considered relevant to support organizations in their daily activities. Concerning the modelling of business processes, a number of process definition languages have been proposed in the literature, based on several formalisms such as BPMN (Business Process Modeling Notation) [17], event-condition-action mechanisms [1], graph rewriting mechanism [11], Petri Nets [2], etc. More recently, some authors have suggested exploiting UML (Unified Modeling Language) [19] to model business processes [14,16].

UML represents a natural choice for modelling business processes since it has been conceived for the communication among people and then can be easily understood and used by customers, managers, and developers [16]. Process

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 291–305, 2011.
© Springer-Verlag Berlin Heidelberg 2011

modelling also plays an important role in the requirement engineering field [10]. It is an essential mechanism for specifying the processes to be supported by a software system as well as for communication with customers and end-users, who have to understand and possibly review these processes [9]. In this scenario UML activity diagrams are a commonly used notation for business process modelling.

In favour of the UML notation is its flexibility allowing the modeller to choose the preferred degree of precision/abstractiveness to build models. Concerning the business process modelling, different options are available ranging from "light" styles, where nodes and arcs of the activity diagrams are simply decorated by natural language text, to more rigorous ones, where for example nodes and arcs are expressed in a formal language. "Light" activity diagrams are simple to write/use but their inherent ambiguity complicates the communication among participants. On the other hand, more precise/rigorous notations are more complex to use but limit ambiguity and have the good quality to be more easily transformed into executable models (e.g., expressed in BPEL).

In this paper, we sketch our *precise* UML activity diagrams to model business processes. This style has been proposed and used in the context of the TECDOC project[1] along with other variants: *ultra-light*, *light*, *precise* and *conceptual precise*[2]. The effectiveness of the *precise* style has been investigated through a controlled experiment conducted with master students in Computer Science at the Free University of Bolzano-Bozen. Indeed, we compared the comprehension of the subjects on business processes specified by using *precise* and *ultra-light* (the "lightest" variant) UML activity diagrams.

The remainder of the paper is organized as follows: Sect. 2 introduces both the precise and the ultra-light styles. Sect. 3 presents the design of the controlled experiment, while Sect. 4 shows and discusses the achieved results. Sect. 5 presents relevant related literature concerning experiments in the use of UML models in comprehension tasks, while final remarks conclude the paper.

2 Process Modeling with UML: Ultra-Light and Precise Styles

In this section we present the two styles for business process modelling that we intend to contrast with respect to comprehension level and comprehension effort.

We will use the following terminology: – the basic activities in a business process are the *basic task* of the process; – the *process objects* are the entities over which the activity of the process are performed, obviously these entities are passive, i.e., they are not able to do such activities by themselves; – the active entities that perform the various tasks are *process participants* (entities

[1] Funded in the framework of research activities of Ligurian Technology District SIIT (Integrated Intelligent Systems and Technologies), the TECDOC project aimed to define methodologies to efficiently schedule, coordinate, monitor and manage the different operational activities related to the management of Complex Organizations.

[2] See http://softeng.disi.unige.it/tech-rep/TECDOC.pdf for the complete TECDOC document.

playing a certain role in a domain), and whenever it will be relevant, we will distinguish the human participants from those corresponding to software and hardware systems.

2.1 Ultra-Light Style

The ultra-light style is the one currently used in the industry for UML business process modelling, see e.g., [15]. Following the *ultra-light style* a process is modelled by a UML activity diagram, where the nodes (activity and object) and the guards on the arcs leaving the decision nodes are decorated by natural language text, which follows neither rules nor patterns. Notice that it may happen that the sentences defining the activities may be either in active or passive form (e.g., "Clerk fills the form" and "'Form is filled") and that the entity executing the activity may be precisely determined or left undefined (e.g., "Form becomes filled"). It is also possible that nominal sentences are used instead of verbal phrases ("Filling the form"). Also, the objects over which the process activities are performed may be described in different ways, for example by a noun (e.g., "Form", "The form") or by a qualifying sentence (e.g., "Client form", "Filled form", "Sent form").

Participants of the process may be modelled only by introducing swim-lanes and titles of the various lanes. The objects produced and consumed by the activities of a business process may optionally be made explicit by using object nodes.

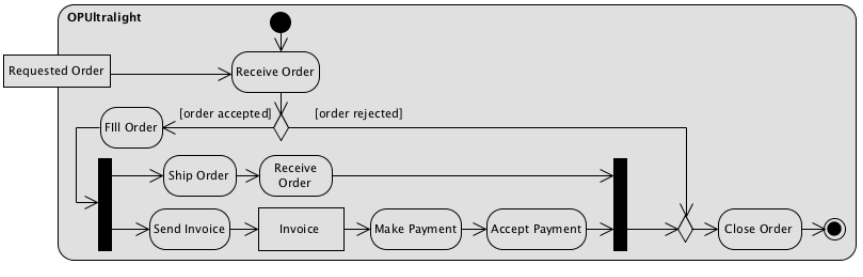


Fig. 1. Ultra-light model of Process Order

In Fig. 1, we present the *ultra-light* UML model of the business process Process Order, namely one of the objects used in the experiment. It is a parametric activity diagram that receives as input the Requested Order (see the node at the boundary of the activity diagram) for an on-line shop. Tasks are represented in the model as rounded rectangles, while the produced objects are depicted as rectangles. The activity diagram describes how the order is managed by the on-line shop. It is quite easy to understand and there is no need to further comment it.

2.2 Precise Style

The participants and the objects of a business process modelled by the precise style are explicitly listed and precisely modelled with UML by means of classes. The behavioural view of the process is given by an activity diagram where actions and conditions will be written by using respectively the language for the action of UML and OCL [18], the textual language for boolean expressions part of the UML. Whenever the object nodes will be used, they should be typed by UML classes and data types; and if swim-lanes are used, they should be given titles by participants.

Thus the *UML precise model of a business process* consists of a class diagram, introducing the classes needed to type its participants and objects, a list of its participants and objects, and an activity diagram representing its behaviour:

- Classes in the class diagram may be stereotyped by ≪object≫ (process objects), ≪businessWorker≫ and ≪system≫ (process participants distinguishing between human beings and hardware/software systems). For readability the stereotype ≪businessWorker≫ is usually omitted. Elements of those classes may be described using the many tools offered by the UML, for example constraints and behavioural diagrams, and their mutual relationships will be expressed by associations and specializations. Dependency (visually depicted by a dashed arrow) will be used to represent the fact that participants of a given class will act over objects of another given class.
- Participants will have a name and will be typed by a class with stereotype either ≪businessWorker≫ or ≪system≫, and objects also will have a name and will be typed by a class stereotyped by ≪object≫. Notice that participants/objects are roles for the entities taking part to the process, and not specific individuals. Constraints might be imposed on participants and objects of a process.
- Basic tasks in the activity diagrams are modelled by UML actions (i.e., calls to class operations, belonging to those classes describing types of participants and objects, and the standard statements, e.g., assignment, creation and destruction of objects). Nodes in the activity diagram will correspond to basic tasks, and thus they will be action nodes, and the conditions on arcs leaving the decision nodes will be OCL expressions (e.g., ORDER.acceptable in Fig. 2). Participants and objects will freely appear both in the actions and in the conditions.

Fig. 2 shows the precise model of the Process Order case. In this process we have two participants (i.e., human being) the Client and the Company, and three business objects: Order, Payment and Invoice. The three objects are related among them as shown by the constraints in the participants/objects box (see the box on the bottom of Fig. 2). The flow of the business object Order is shown by using its name in the various actions nodes, whereas the flow of Invoice has been emphasized by using an object node. The class diagram in Fig. 2 introduces the classes typing the participants and the objects with their relevant operations and attributes, together with their mutual relationships. For example we can

see that a Payment and an Invoice are relative to exactly one Order. The dashed arrows, i.e., the dependency relationships denote that the Company may work on the payments, the invoices and the orders, whereas the Client only on the Payment and the Order. Constraints may be used to finely describe the various classes; for example the constraint on the operation receives of class Company (see the note in Fig. 2) expresses that an Order is considered acceptable by the company if it is well-formed and available.

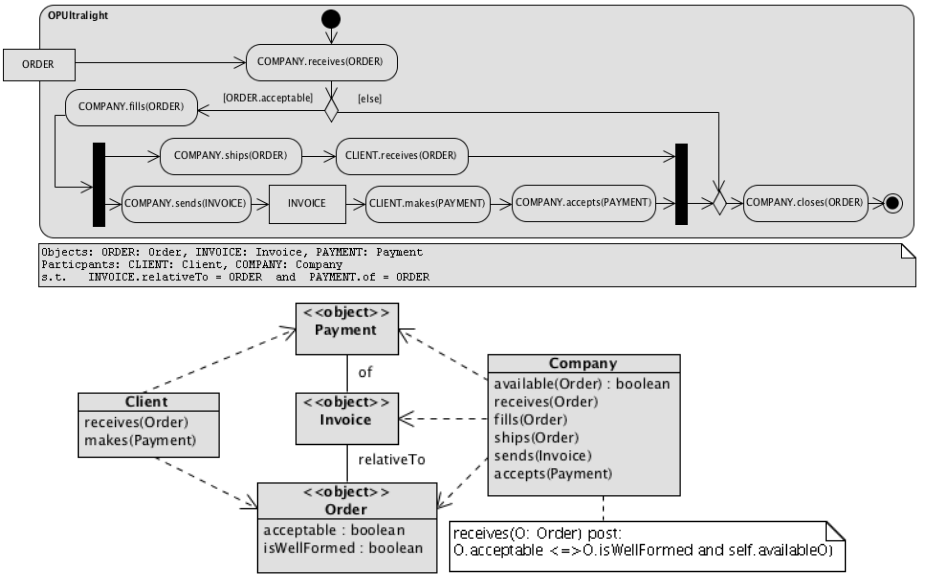


Fig. 2. Precise model of Process Order (activity and class diagram)

3 Experimentation Setup

In this section we present the design of the experiment. We followed the guidelines proposed in [13,24]. For replication purposes, the experimental package (in English) and the raw data are made available on the Web[3].

3.1 Context

The main experiment was conducted with 26 master students in Computer Science at the Free University of Bolzano-Bozen. In addition, 14 bachelor students from the same University took part in a pilot experiment, useful to assess the material, which was executed before the main experiment.

[3] www.scienzemfn.unisa.it/scanniello/BPM/ (please cut and past this URL into your web browser).

The main experiment represented an optional educational activity of two Software Engineering courses: Infrastructures for Open Service Oriented Architectures and Requirements and Design of Software Systems. The pilot experiment was an optional activity for the Business Information Systems course, attended by third-year bachelor students. As mandatory laboratory activity of the course Infrastructures for Open Service Oriented Architectures, students had previously developed Web services using specification documents that included UML models in terms of class, sequence, and activity diagrams. Students of the course Requirements and Design of Software Systems had already made use of full UML specifications in the design of non-trivial software systems. This year the assignments were defined in cooperation with the Italian National Transplant Organization (CNT): students redesigned the CNT's IT infrastructure according to the SOA paradigm. Note that all the students (master and bachelor) had also previously attended courses on basic and advanced object oriented programming, before carrying out the main experiment and the pilot.

The specifications of two different business processes were considered for the experiment. The business processes refer to application domains in which the subjects are familiar with. The first (i.e., Process Order) is in charge of processing orders for an on-line shop. In particular, this process takes as input an order. The order is accepted and info is filled, payment processing and shipment are done. Finally, the order is closed (see Fig. 1). Instead, the second business process (i.e., Document Management Process) is a business process for managing the on-line review process of any kind of documents. First a document is created by the author and then it is reviewed by a reviewer. Finally, a document is approved (if its quality satisfies the constraints imposed). Note that documents can also be updated and archived. The two business processes are comparable both in complexity and in size as well as in the number of activities and classes. Furthermore, they are small enough to fit the time constraints of the experiment but at the same time they are realistic for small/medium sized comprehension tasks.

3.2 Hypotheses Formulation

The perspective of this study is twofold. From the point of view of researchers, it is an investigation of the effectiveness of using precise activity diagrams in the specification of business processes; and from the point of view of project managers, it is an evaluation of the possibility of adopting this style. Accordingly, we have defined and tested the following null hypotheses:

- H_{lo}: The use of precise activity diagrams **does not significantly improve** the comprehension level of the subjects to perform a task.
- H_{to}: There **is no significant difference** in terms of effort when using precise or ultra-light activity diagrams to perform a comprehension task.

The objective of the statistical analysis is to reject the defined null hypotheses, thus accepting the corresponding alternative ones that admit a positive effect and so can be easily derived. It is worth mentioning that the null hypothesis H_{l0}

is one tailed since we expect a positive effect of the precise activity diagrams on the subject performance. On the other hand, H_{t0} is two-tailed since we cannot postulate an expectation of a difference in terms of effort.

3.3 Design

We adopted a counterbalanced design [24] as shown in Table 1. We considered four groups: A, B, C, and D. Each group was formed by subjects randomly selected (precisely: 7 subjects for groups A and D; 6 for groups B and C). Each subject worked on two comprehension *Tasks* (i.e., Task 1 and Task 2) on the following two experimental *Objects*: Process Order (PO) and Document Management Process (DMP). Each time subjects used the precise or ultra-light activity diagrams. For example, the subjects within the group A started to work in Task 1 on PO using the precise activity diagram and then they used the ultra-light activity diagram to perform Task 2 on DMP.

Table 1. Experiment design

	A	B	C	D
Task 1	PO Precise	PO Ultra-light	DMP Precise	DMP Ultra-light
Task 2	DMP Ultra-light	DMP Precise	PO Ultra-light	PO Precise

3.4 Selected Variables

The *control group* indicates the students working with the ultra-light activity diagram, while the *treatment group* indicates the students working with the precise activity diagram. Thus, the only independent variable is *Treatment*, which is a nominal variable that admits two possible values: *Precise* and *Ultra-light*. On the other hand, we selected the following dependent variables to investigate the defined null hypotheses: *comprehension level* and *comprehension effort*.

The *comprehension level* dependent variable is used to measure the comprehension of the subjects on each business process. Similarly to [21], the subjects were asked to answer a comprehension questionnaire (it is equal for both the treatments but different by objects) composed of multiple choice questions. In particular, on each considered business process the questions were 12, each admitting the same number of possible answers (i.e., 5), with one or more correct answers. Fig. 3 shows a sample question (question 1) regarding the comprehension questionnaire of the PO object. The goal of this question is to investigate whether the subjects understood who are the business process participants. Correct answers are the first and the third ones.

Correctness and completeness of the provided answers have been measured, similarly to [21], using an information retrieval based approach. To this end, we defined as: $A_{s,i}$, the set of answers provided by the subject s on the question i, and C_i, the correct set of answers of the question i. The correctness and the

1. Indicate the **participant/s**[1] of the workflow

- ☐ Company
- ☐ Invoice
- ☐ Client
- ☐ There are no participants in the workflow
- ☐ Order

Fig. 3. Question 1 of the comprehension questionnaire for PO

completeness of the answers have been measured using, respectively, *precision* and *recall*:

$$precision_{s,i} = \frac{|A_{s,i} \cap C_i|}{|A_{s,i}|} \quad recall_{s,i} = \frac{|A_{s,i} \cap C_i|}{|C_i|}$$

In order to get a single value representing a balance between correctness and completeness, we used the harmonic mean between precision and recall:

$$F{-}Measure_{s,i} = \frac{2 \cdot precision_{s,i} \cdot recall_{s,i}}{precision_{s,i} + recall_{s,i}}$$

For example, if a student had answered Question 1 of the PO task (Fig. 3) picking the first, second and fifth answer, her precision will be 0.33 (three answers given and only one correct) while her recall will be 0.5 (one correct answer out of two). Instead, her F-measure will be 0.39.

The overall comprehension level achieved by each subject has been computed using the overall average of the F-Measure values of all the questions. This average assumes a value ranging from 0 to 1. A value close to 1 indicates a very good understanding of the business process, while a value close to 0 indicates a very bad comprehension level.

The *comprehension effort* dependent variable measures the time, expressed in minutes, that each subject spent to accomplish a task. We got this value using the start and stop times the subjects were asked to record.

3.5 Experimental Material, Pilot and Execution

In order to assess the experimental material (mainly the comprehension questionnaire) and get an estimation of the time needed to accomplish the task a pilot experiment with 14 bachelor students was accomplished before the main experiment.

Regarding the main experiment, the subjects were asked to use the following procedure to execute both the tasks: *(i)* specify name and start-time in the comprehension questionnaire; *(ii)* answer the questions by consulting the provided material; *(iii)* mark the end-time.

To perform the experiment the subjects were provided with the following hard copy material: *(i)* a summary of the modelled business process, *(ii)* the comprehension questionnaires and the models of the business processes, *(iii)* a unique post-experiment questionnaire to be filled in after the two tasks.

The post-experiment questionnaire aimed at gaining insights about the subjects' behaviour during the experiment. The post-experiment questionnaire was composed of 5 questions concerning the availability of sufficient time to complete the tasks, the clarity of the experimental material and objects, and the ability of subjects to understand the business processes used in the experimentation. The questions expected answers according to the following five point Likert scale: (1) strongly agree, (2) agree, (3) neutral, (4) disagree, (5) strongly disagree. For space reasons, results of the post-experiment questionnaire are only marginally discussed in the following.

4 Analysis and Results

In this section, after a brief summary of the pilot experiment, results of the data analysis of the main study are presented with respect to the defined null hypotheses (Sect. 3.2). In all the performed statistical tests, we decided (as it is customary) to accept a probability of 5% of committing Type-I-error [24], i.e., rejecting the null hypothesis when it is actually true. We conclude the section discussing the effect of the co-factors (i.e., Object, Task and Group) and sketching the potential threats to validity.

4.1 Pilot Experiment

All students completed both comprehension tasks of the pilot experiment in 50 minutes. This let us conclude that the pilot was well suited for bachelor/master students. Minor changes were made to improve the comprehension questionnaires. A simplified data analysis showed that students with precise activity diagrams outperformed (the mean comprehension level was 0.69) in comprehension students with ultra-light ones (the mean comprehension level was 0.59). Concerning the effort, students with precise diagrams employed more or less the same time that students with ultra-light diagrams (median for both groups was 19 minutes).

4.2 Comprehension Level - Main Experiment

Table 2 reports some descriptive statistics (i.e., mean, median, and standard deviation) of comprehension level and the results of statistical analysis conducted on the data of the main experiment. Because of the sample size (26 subjects) and mostly non-normality of the data we adopted non-parametric tests to test the first null hypothesis. In particular, we selected Mann-Whitney test for unpaired analysis and Wilcoxon test for paired analysis. We used these tests since they are very robust and sensitive [13,24].

The overall comparison (i.e., without partitioning by object) is visually presented in Fig. 4 by means of boxplots. From them, it appears that students with precise activity diagrams outperformed in comprehension students with ultra-light ones. We evaluate the first hypothesis overall. The one-way unpaired Mann-Whitney test ($p - value < 0.001$) and the one-way paired Wilcoxon

test ($p - value < 0.001$) provide evidence that there exists a significant difference in terms of comprehension level between the two treatments. Therefore, we can reject the null hypothesis H_{l0}. The mean comprehension level improvement achieved with precise diagrams is of 17 points (see means of the "All" row in Table 2), i.e., 27,41%[4]. Similar results can be observed for the primary measures. For space reasons, we report only results of Mann-Whitney tests: precision ($p - value < 0.001$) and recall ($p - value < 0.001$) and for both objects, DMP ($p - value = 0.005$) and PO ($p - value = 0.003$).

Table 2. Descriptive statistics of comprehension level and the statistical test results

Object	Precise mean	med	σ	Ultra-Light mean	med	σ	MW test	Wilcoxon test
All	0.79	0.84	0.11	0.62	0.66	0.14	< 0.001	< 0.001
DMP	0.76	0.74	0.10	0.64	0.64	0.10	0.005	-
PO	0.80	0.84	0.11	0.58	0.69	0.19	0.003	-

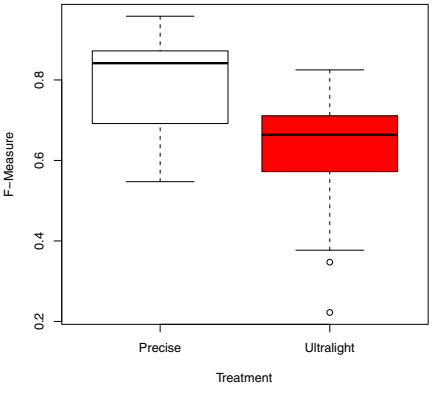


Fig. 4. Comprehension level

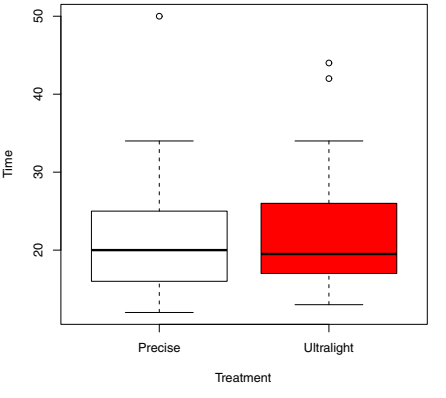


Fig. 5. Comprehension effort

4.3 Comprehension Effort - Main Experiment

Fig. 5 shows the boxplots of comprehension effort versus the treatments. Apparently, students with precise diagrams employed more time that students with ultra-light diagrams (see the two medians). Means and medians are respectively: 22'16" and 20 minutes for precise diagrams; 22'12" and 19'50" minutes for ultra-light diagrams. A two-tailed unpaired Mann-Whitney test returned 0.9 as $p - value$. A similar value is returned by paired Wilcoxon test ($p - value = 0.6$).

[4] The percentage comes from the following equation: 0.62+0.62*x%=0.79.

Therefore, we cannot reject the overall null hypothesis H_{t0}. Even analysing the two objects separately no significant difference was found. The results of the unpaired two-tailed Mann-Whitney test were 0.56 and 0.57 for DMP and PO, respectively.

4.4 Co-factors and Post Questionnaire Results - Main Experiment

We have analysed the effects of the co-factors on both comprehension level and effort, to find possible interactions with the treatments. This kind of analysis can be also useful to discover possible learning, fatigue, and order effects that could "contaminate" the obtained results. For this task, we used a two-way Analysis of Variance (ANOVA) [13,24]. Even if all the assumptions/conditions for using ANOVA were not checked, this test is quite robust. On the overall data set, we found no significant effect of object on comprehension level ($p-value = 0.77$) and no interaction with treatment ($p-value = 0.22$). Similarly, no significant results were obtained analysing the effects (and interactions) of the co-factors task and group on the comprehension level. Alike, we found no significant effect of objects on effort ($p-value = 0.78$) and no interaction with treatment ($p-value = 0.44$). Instead, an interaction ($p-value = 0.003$) was observed between treatment and task (without effect of task alone). Finally, we found a significant effect of group on effort ($p-value = 0.01$) but no interaction with treatment ($p-value = 0.79$). Looking more in details at the data, we found that students in group A used more time and put more effort than students in other groups. The cause of this difference will be investigated in future.

We computed medians of subjects perception, collected through the perceived agreeing level of the post-experiment questionnaire. Students judged sufficient the time to complete the task , they also found clear: the objectives of the experiment (median=2), comprehension questions (median=2) and answers given as possible options (median=2). Finally they found the exercise useful (median=2).

4.5 Threats to Validity - Main Experiment

Threats to validity that could affect our results belong to four categories [24]: *internal*, *external*, *construct*, and *conclusion*.

The counterbalanced design adopted in this experiment enabled us to mitigate *internal validity* threats. It is well-known that this design balances possible learning, fatigue, and order effects. This was also confirmed by the analysis of the co-factors (see Sect. 4.4). Another issue concerns the information exchanged among the subjects. This was prevented as much as possible by monitoring the students while performing the tasks. In addition, students were not evaluated on their performance to avoid apprehension.

External validity may be threatened when experiments are performed with students, throwing into doubt the representativeness of the subjects with respect to software professionals. However, performed tasks do not require high level of industrial experience, so we believe that this experiment can be considered appropriate, as suggested in the literature [3]. Another possible threat concerns the

size and complexity of the tasks. Hence, we plan to replicate the experiment with more complex tasks. Replications with different and more experienced subjects (e.g., professionals and PhD. Students) are planned as well.

In this study, the *construct validity* threats are related to the metrics used to get a quantitative evaluation of the subjects' comprehension and effort. We used questionnaires to assess the comprehension level of the subjects, and answers were evaluated using an information retrieval based approach (as in [21]), in order to avoid as much as possible any subjective evaluation. Furthermore, the comprehension questionnaires were defined to be complex enough without being too obvious. The comprehension effort was measured by means of proper time sheets, and it was validated qualitatively by researchers, who were present during the experiment.

Conclusion validity concerns data collection, reliability of measurements, and validity of statistical tests. Statistical non-parametric tests (Mann-Whitney and Wilcoxon) were used to reject the null hypotheses. Two-way ANOVA was used to detect possible co-factors effects and interactions between each co-factor and the main factor. Even if all the assumptions/conditions for using ANOVA were not checked, this test is quite robust and has been extensively used in the past to conduct analyses similar to ours (see, e.g., [23]).

5 Related Work

Out of the huge body of literature that is based on UML, we highlight in this section just a few papers, that provide useful terms of comparison for our work.

The UML activity diagrams provide an intuitive and easy to learn visual formalism to model business process [7,16,12]. For example, Di Nitto *et al.* [16] propose an approach to process modelling by using a subset of UML diagrams, including UML activity diagrams with object flow to model the control and data flow, class diagrams to model structural properties of the process, and state diagrams to model the behaviour of activities. The XMI standard representation of these models produced using a UML CASE tool can then be translated into an executable process description for the OPSS Workflow Management System [5]. Several are the differences between our approach and theirs. The most remarkable ones are that OCL is not used in the process modelling and the validity of the proposal has not been assessed through controlled experiments.

In [7] a case study is presented, mapping UML activity diagrams with object flow on the process definition language of the GENESIS environment [1]. The authors showed that UML activity diagrams do not support all the control flow and data flow rules of the GENESIS process definition language. As a consequence, the syntax and semantics of this type of UML diagrams often need to be extended to make them suitable for modelling business processes in workflow management systems. Similarly, De Lucia *et al.* in [6] present a system offering a visual environment, based on an extension of UML activity diagrams, that allows to graphically design a process model, and to visually monitor its enactment. The main difference with respect to notations used in this experiment

is that participants and objects are not explicitly considered. Furthermore, the behavioural conditions are not formally specified.

Differently from us, all the approaches discussed above do not assess the validity of the proposed formalism by means of controlled experiments. To our knowledge, only a few other studies perform empirical evaluations in business process formalisms comparisons. For instance, Peixoto *et al.*, [20] compare UML and BPMN (Business Process Modelling Notation) [17], with respect to their readability in expressing Business Processes. Their analysis is motivated by the consideration that many different stakeholders are interested in the results of a business process modelling. Given their different background, and their need to understand the results of modelling, it is very important that all stakeholders are able to understand business process diagrams. Peixoto *et al.* expected BPMN models to be easier to understand than UML 2.0 activity diagrams, as BPMN is a specialized language, designed for modelling business process and with the primary goal of being understandable by all business stakeholders [17]. However, an experiment with 35 undergraduate students of Computer Science, unskilled in business process modelling, could not confirm their initial hypothesis, therefore UML activity diagrams and BPMN seem to be equivalent in terms of understandability.

Gross and Doerr [10] conducted two experiments, comparing the UML activity diagrams and Event-driven Process Chains [22] with different perspectives. First, they considered the business processes specification from a requirements engineer perspective with a focus on model creation. Second, their attention was on model understandability, seen from a customer's or end user's point of view. The used methodology was in both cases a blocked subject-object study: participants were partitioned in two groups, each of them receiving an assignment in one of the considered formalisms. The authors found evidence that activity diagrams performed better than EPCs from a requirements engineer's perspective. When considering end users, no significant difference was identified between the two methods.

With different objectives, Coman and Sillitti [4] conducted an empirical study on the possibility of mapping low-level to high-level software development activities in an automatic way. The method is based on low level data automatically collected, that are used in order to identify high-level activities. The context of the experiment is similar to ours, as, among other commonalities, the analysed data was related to software systems similar to the one proposed in this work.

6 Conclusion

In this paper, we propose a variant of UML activity diagrams. This variant has been defined in the context of business processes modeling and its effectiveness has been investigated with respect to a less rigorous visual formalism. To this end, a controlled experiment with 26 master students has been conducted and the results have been presented and discussed in this paper. The data analysis indicated a significant effect of the more rigorous style on the comprehension of

business processes (+27.61%). Conversely, the effect of the effort to accomplish comprehension tasks is not statistically significant.

Acknowledgements. We would like to thank the students that took part in both the pilot and the main experiment.

References

1. Aversano, L., De Lucia, A., Gaeta, M., Ritrovato, P., Stefanucci, S., Villani, M.L.: Managing coordination and cooperation in distributed software processes: the genesis environment. Software Process: Improvement and Practice 9(4), 239–263 (2004)
2. Bandinelli, S., Di Nitto, E., Fuggetta, A.: Supporting cooperation in the spade-1 environment. IEEE Trans. Softw. Eng. 22, 841–865 (1996)
3. Basili, V., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Trans. Softw. Eng. 25(4), 456–473 (1999)
4. Coman, I.D., Sillitti, A.: An empirical exporatory study on inferring developpers' activities from low-level data. In: Proceedings of International Conference on Software Engineering and Knowledge, pp. 15–18. Knowledge Systems Institute Graduate School (2007)
5. Cugola, G., Di Nitto, E., Fuggetta, A.: The jedi event-based infrastructure and its application to the development of the opss wfms. IEEE Trans. Softw. Eng. 27, 827–850 (2001)
6. De Lucia, A., Francese, R., Scanniello, G., Tortora, G.: Distributed workflow management based on UML and web services. In: Encyclopedia of E-Commerce, E-Government, and Mobile Commerce, pp. 217–222. IGI Global (2006)
7. De Lucia, A., Francese, R., Tortora, G.: Deriving workflow enactment rules from uml activity diagrams: a case study. In: Symposium on Human-Centric Computing Languages and Environments, pp. 211–218 (2003)
8. Eriksson, H.E., Penker, M.: Business Modelling with UML. Wiley Computing Publishing, Chichester (2000)
9. Gonzalez, J.D., Diaz, J.S.: Business process-driven requirements engineering: a goal-based approach. In: Proc. of the 8th Workshop on Business Process Modeling Development and Support, pp. 1–9. Tapir Academic Press, London (2007)
10. Gross, A., Doerr, J.: EPC vs. UML activity diagram - two experiments examining their usefulness for requirements engineering. In: Proceedings of Requirements Engineering Conference, pp. 47–56. IEEE Computer Society, Washington, DC, USA (2009)
11. Heimann, P., Joeris, G., Krapp, C., Westfechtel, B.: Dynamite: Dynamic task nets for software process management. In: Proceedings of the International Conference on Software Engineering, pp. 331–341 (1996)
12. Jurack, S., Lambers, L., Mehner, K., Taentzer, G., Wierse, G.: Object flow definition for refined activity diagrams. In: Proceedings of the 12th International Conference on Fundamental Approaches to Software Engineering, pp. 49–63. Springer, Heidelberg (2009)
13. Juristo, N., Moreno, A.: Basics of Software Engineering Experimentation. Kluwer Academic Publishers, Dordrecht (2001)
14. Marshall, C.: Enterprise modelling with UML: Designing successful software through business analysis. Addison-Wesley, Reading (2000)

15. Monfared, R., West, A., Harrison, R., Weston, R.: An implementation of the business process modelling approach in the automotive industry. Journal of Engineering Manufacture 216(11), 1413–1428 (2002)
16. Nitto, E.D., Lavazza, L., Schiavoni, M., Tracanella, E., Trombetta, M.: Deriving executable process descriptions from UML. In: Proceedings of the 22rd International Conference on Software Engineering, pp. 155–165 (2002)
17. OMG. Business process model and notation (BPMN) Version 2.0. OMG Final Adopted Specification, Object Management Group (2006)
18. OMG. Object constraint language (OCL) specification, version 2.2. Technical report, Object Management Group (February 2010)
19. OMG. Unified modeling language (UML) specification, version 2.3. Technical report, Object Management Group (May 2010)
20. Peixoto, D., Batista, V., Atayde, A., Borges, E., Resende, R., Pádua, C.: A Comparison of BPMN and UML 2.0 Activity Diagrams. In: VII Simposio Brasileiro de Qualidade de Software (2008)
21. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: The role of experience and ability in comprehension tasks supported by uml stereotypes. In: 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, 2007, May 20-26, pp. 375–384 (2007)
22. Scheer, A.: ARIS-business process modeling. Springer, Heidelberg (2000)
23. Torchiano, M., Ricca, F., Tonella, P.: Empirical comparison of graphical and annotation-based re-documentation approaches. IET Software 4(1), 15–31 (2010)
24. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering - An Introduction. Kluwer, Dordrecht (2000)

If the SOK Fits, Wear It: Pragmatic Process Improvement through Software Operation Knowledge

Henk van der Schuur, Slinger Jansen, and Sjaak Brinkkemper

Department of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
{h.schuur,s.jansen,s.brinkkemper}@cs.uu.nl

Abstract. Knowledge of in-the-field software operation is nowadays acquired by many software-producing organizations. Vendors are effective in acquiring large amounts of valuable software operation data to improve the quality of their software products. For many vendors, however, it remains unclear how their actual product software processes can be advanced through structural integration of such information. In this paper, we present a template method for integration of software operation information with product software processes, and present four lessons learned that are identified based on a canonical action research study of ten months, during which the method was instantiated at a European software vendor. Results show that the template method contributes to significant software quality increase, by pragmatic but measurable improvement of software processes, without adhering to strict requirements from cumbersome maturity models or process improvement frameworks.

1 Introduction

Nowadays, software vendors are continuously striving for refinement and improvement of their software processes to achieve and extend competitive advantage. Simultaneously, software vendors are experienced in acquiring in-the-field operation data from their software products and services. It has become common practice, for example, to monitor software operation and identify operation failures by means of acquired operation information [5,12].

A wealth of software operation knowledge (SOK), gained from operation information analysis, can improve basically any software process in a product software company, such as software maintenance, software product management and customer support [11]. For many vendors, however, it remains unclear how software processes can be improved through integration of such information. As a consequence, acquired operation information is left untouched during execution of product software processes.

The research question we attempt to answer in this paper is '*How can product software processes effectively be improved with acquired information of in-the-field software operation?*'. To answer this question, we introduce a template

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 306–321, 2011.
© Springer-Verlag Berlin Heidelberg 2011

method for integration of software operation information with product software processes, and present four lessons learned that are identified based on a canonical action research study of ten months, during which the method was instantiated at a European software vendor. Several prescriptive 'one-size-fits-all' software process improvement (SPI) approaches, such as the Capability Maturity Model Integration (CMMI) and ISO/IEC 15504 (SPICE), are considered as too large, too extensive and too expensive to comprehend and implement effectively within small and medium-sized companies [1,8,10,13]. As opposed to those approaches, the SOK integration template method presented is pragmatic and inductive, i.e., potential resulting improvements are based on the particular situation of an organization. The template method particularly describes *what* are typical operation information integration activities and concepts; a method instantiation describes *how* these activities should take place and how related concepts are involved, specific to a vendor's situation.

This paper continues with placing our work in context. Next, the research approach is detailed, after which the SOK integration template method is presented (section 4). Section 5 details in-the-field instantiation of the method, as well as an valuation of integration experiences, resulting lessons learned and research limitations. Finally, conclusions and future work are presented (section 6).

2 Related Work

Many research efforts cover the subject of software process improvement, but only few consider knowledge of in-the-field software operation as an instrument for improving product software processes.

For instance, Pettersson et al. [10] have proposed iFLAP, a process improvement framework that is to some extent similar to the SOK integration template method we presented: the framework is inductive in nature and draws on knowledge that is already residing in the organization to improve processes. However, as opposed to our method, the framework is specifically directed towards, and evaluated with, requirements engineering processes. Miler and Górski [9] report on a case study in which they apply a risk-driven software process improvement framework in a real-life software project. Case study results demonstrate that the proposed framework is able to reveal new, previously undetected risks that provide important input for process improvement. Although various undetected risks were identified, the framework requires a high level of process description detail to be applied effectively. Iversen et al. [7] proposed a framework for understanding risk areas and resolution strategies within software process improvement, as well as a corresponding risk management process. Although both the framework and the process are comprehensive and detailed, the framework was not evaluated through empirical studies and practical use. Moreover, it was left unclear how and to which extent vendors over time actually benefit from it. Also, many efforts are directed to demonstrating the effectiveness of software process improvement by means of CMM(I) [2,4]. For example, Dangle et al. [4] analyze the role of process improvement in the context of small organizations

through an extensive case study in which CMM is applied. Based on this study, lessons learned are identified. Although one lesson is somewhat analogous to one of the lessons we have identified, it is left unclear to which extent the lessons learned of Dangle et al. are generalizable to vendors that have implemented a different maturity model, or no maturity model at all.

In this paper, we demonstrate pragmatic but measurable improvement of product software processes, and identify lessons learned that are generalizable to similar product software vendors.

3 Research Approach

To investigate how product software processes can be effectively improved with acquired information of in-the-field software operation, we designed a template method for integration of such information with product software processes, following a combination of design research and canonical action research principles [6,3]. We conducted an extensive action research study at a European software vendor, during which the method was used to integrate software operation information acquired by the vendor with the vendor's product software processes, and therewith improve those processes. Based on study results, lessons learned are identified that may serve as a guiding substrate for similar vendors in integrating operation information with their product software processes.

3.1 Research Site

The action research study presented in this paper was carried out at CADComp[1], a European software vendor founded in 1990. The vendor develops an industrial design application, CADProd, which is targeted on the Microsoft Windows platform and is used daily by more than 4,000 customers in five countries. Since the start of its development in 1995, four major versions of the application have been released. Currently, CADComp employs about 100 people and is established in the Netherlands, Belgium and Romania. From December, 2009 to September, 2010, we were present at the vendor's main development site to integrate acquired SOK in the vendor's product software processes.

Before our study commenced, CADComp already acquired data of the in-the-field operation of its software product CADProd in the form of customized error reports. However, these data were not structurally analyzed, no software operation information was extracted from these data and no operation information was integrated with CADComp's existing product software processes.

3.2 Canonical Action Research

The canonical action research method described by Davison et al. [3] was used to structure the research activities. We attempted to satisfy each of their 'canonical action research principles':

[1] Note that for reasons of confidentiality, the names of the vendor and its software products have been anonymized.

1. Principle of the Researcher-Client Agreement. The study was conducted at the European software vendor CADComp. Both the researchers and the vendor agreed on the action research approach; the vendor indicated that it is in need of an approach for improving its product software processes with the use of acquired operation data. We have agreed on a research plan that contains an overview of the shared research objectives as well as the data that was to be collected during the study (e.g. software architecture specifications, process descriptions, memos, semi-structured interviews, etc.)
2. Principle of the Cyclical Process Model. The cyclical process model [3], originally proposed by Susman and Evered [14], served as a basis for our work at the vendor. Structuring research activities by means of this model ensures adequate action planning, action taking and evaluation, as well as specifying what other vendors and researchers could learn from our study.
3. Principle of Theory. According to Davison et al. [3], action research theory takes the following form: in situation S with salient features $F_1, \ldots, F_n$, outcomes $O_1, \ldots, O_n$ are expected from actions $A_1, \ldots, A_n$. Our (grounded) theory, as reflected by the formulated research question, is consistent with this form: we expect vendors that acquire but not analyze or integrate operation information (S), to improve their product software processes (O) by means of implementing an instantiation of the SOK integration template method (A).
4. Principle of Change through Action. As stated by principle 1 and 3, both the researcher and the vendor were motivated to change the situation at the vendor in terms of operation data and information use. Planned actions were designed and taken to achieve the defined objectives (see section 5).
5. Principle of Learning through Reflection. Both the observations made as well as the actions taken were reported to, and evaluated with the vendor's employees and management. The researcher and vendor reflected outcomes of the study by means of semi-structured interview sessions (see section 5).

Adhering to these principles assisted us in establishing pure validity of our research approach, which contributes to realistic repetition of the study at similar software vendor sites.

4 SOK Integration

We have developed a template method to facilitate integration of acquired SOK into existing product software processes (*target processes*). The method is designed to guide vendors in (1) identification of relevant and valuable operation information, (2) analysis of target processes and their integration environment, (3) integration of selected information in, and transformation of, target processes and (4) presentation of integrated operation information.

When applied successfully, the method enables software vendors to increase the extent to which their practices, processes and tools are supported by SOK, which may result in directed software engineering, informed management and more intimate customer relationships. Figure 3 depicts the method as a Process-Deliverable Diagram (PDD) [15].

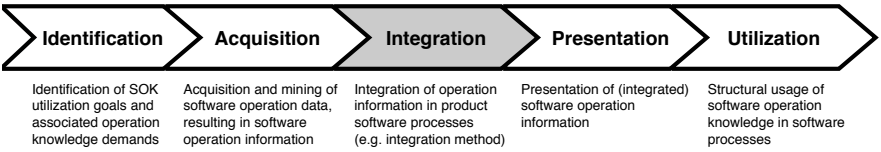


Fig. 1. Integration of operation information is part of the SOK life cycle [11]

In the context of the SOK framework [11], the SOK integration template method is an implementation of the SOK integration process, following identification [11] and acquisition [12] processes, and preceding presentation and utilization processes (see figure 1). In the SOK acquisition process, software operation data are acquired, mined and analyzed, resulting in software operation information that forms input of the method. After acquisition, operation information is presented on carriers determined during application of the method; software operation knowledge resulting from interpretation of such information is used with the frequency determined during method application.

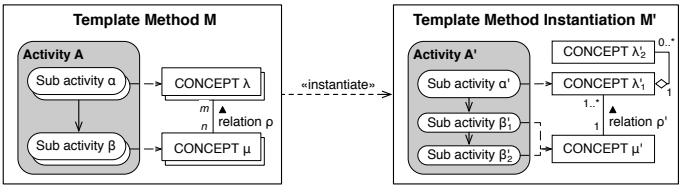


Fig. 2. Template method instantiation

4.1 Instantiation

The SOK integration template method can be used by software vendors that attempt to integrate operation information with product software processes. For each target process, a new method instance is created, each with corresponding object and activity instantiations. The template method prescribes *what* (rather than *how*) activities and concepts are to be implemented by software vendors. Therefore, the method is only composed of *open* activities and concepts [15], which should be instantiated with standard, open or closed equivalents. To anchor process improvement in the organization, vendors should instantiate both template activities and concepts consciously and diligently, taking into account daily practices. The method can be typically applied iteratively, continuously, and potentially in parallel with other method instantiations. The method's application duration and frequency depend on INTEGRATION RESOURCEs and environment. One or multiple people are responsible for successful execution of (sub) activities. We assume that both the SOFTWARE OPERATION INFORMATION as well as one or more TARGET PROCESSes are available and accessible, before instantiating the method. An instantiation example is visualized in figure 2.

4.2 Concepts

The eight template concepts the SOK integration template method refers to, depicted at the right side of the PDD (see figure 3), are detailed in table 1.

Table 1. Concepts of the SOK integration template method

Concept Name	Concept Description
ACTOR	A human who demands and utilizes SOFTWARE OPERATION INFORMATION with a certain frequency, potentially visualized by a CARRIER, and participates in one or more TARGET PROCESSes
CARRIER	Medium that can convey and visualize SOFTWARE OPERATION INFORMATION
INTEGRATION EVALUATION	A systematic determination of merit, worth, and significance of the performed integration of SOFTWARE OPERATION INFORMATION using criteria against the set of defined INTEGRATION OBJECTIVES involved
INTEGRATION OBJECTIVE	Goal of integration of SOFTWARE OPERATION INFORMATION with a TARGET PROCESS involving one or more INTEGRATION REQUIREMENTS
INTEGRATION REQUIREMENT	A SOFTWARE OPERATION INFORMATION integration necessity, demanding an amount of INTEGRATION RESOURCES
INTEGRATION RESOURCE	The (human) resources available for performing integration of SOFTWARE OPERATION INFORMATION
SOFTWARE OPERATION INFORMATION	Information resulting from data mining and abstraction of software operation data acquired from software operating in the field, possibly presented on a CARRIER
TARGET PROCESS	A collection of related, structured activities, tasks, tools and ideas that produce a specific service or product for (a) customer(s), possibly dependent on other processes or activities, with which SOFTWARE OPERATION INFORMATION is integrated

Each of the concepts referred to by the method, corresponds to at least one of the (sub) activities of which the template method is composed, which are detailed in the next section.

4.3 Activities

The method's activities and sub activities, depicted at the left side of the PDD (see figure 3) are detailed below.

1. Operation information selection. In the first activity of the method, a selection of relevant operation information resulting from data mining and abstraction of software operation data is made. First, the TARGET PROCESS is analyzed (*Analyze target process*). Questions like 'How does the process actually work?', 'How is the process used?', and 'What are process dependencies?' are answered during this activity. *Analyze target process* may be performed from a specific perspective (e.g. human interaction, data dependencies, etc.), which results in a comprehensive view of the process. Based on this process analysis, INTEGRATION OBJECTIVES are determined (*Determine integration objectives*). In this activity, integration incentives and goals are identified, for example by defining the role and functioning of the TARGET PROCESS after integration. Secondly, software operation information demands of the vendor are identified (*Identify operation information demands*). Next, operation information that is considered relevant and valuable to integrate with the TARGET PROCESS, is selected (*Select relevant operation information*)[2], based on the process analysis results and identified operation information demands.

[2] If operation information is missing, application of the method can be interrupted to first iterate through the identification and acquisition phases again (see figure 1), and therewith make sure that desired information is available and accessible.

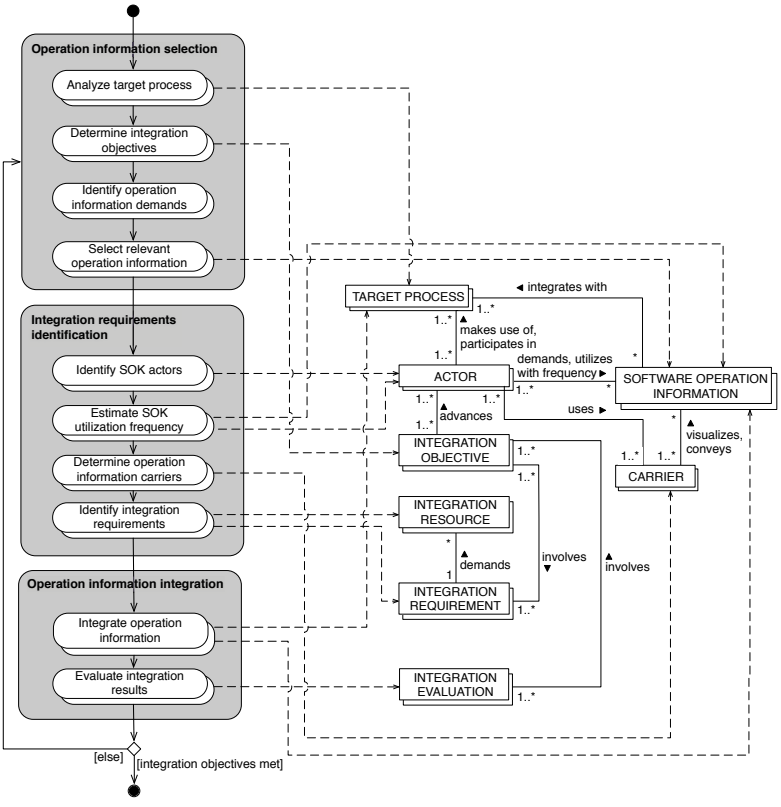


Fig. 3. SOK integration template method

2. Integration requirements identification. First, current and future actors involved with the TARGET PROCESS and acquisition, integration or presentation of operation information are identified (*Identify SOK actors*). Secondly, it is estimated how often SOK resulting from integration or presentation of operation information will be used by ACTORs within the TARGET PROCESS after integration (*Estimate SOK utilization frequency*). Thirdly, CARRIERs for presentation of operation information integrated with the TARGET PROCESS are determined (*Determine operation information carriers*). Finally, based on the sub activities prior to *Identify integration requirements*, INTEGRATION RESOURCEs and INTEGRATION REQUIREMENTs for effective integration of acquired SOFTWARE OPERATION INFORMATION are determined. Resulting requirements serve as input for the subsequent 'Operation information integration' activity.

3. Operation information integration. The TARGET PROCESS is altered to allow integration of relevant SOFTWARE OPERATION INFORMATION (*Integrate software operation information*) selected in 'Operation information selection'. Integration of selected operation information is dependent on, and constrained

by, the available INTEGRATION RESOURCEs identified earlier (e.g. external data sources, time, people, knowledge, etc.) Operation information integration results are evaluated in the second activity (*Evaluate integration results*). If, based on the subsequent INTEGRATION EVALUATION, can be concluded that INTEGRATION OBJECTIVEs are met, the result of this activity (and therewith of the SOK integration template method) is a process that is effectively supported by acquired operation information. Otherwise, the method is reinitiated by starting the 'Operation information selection' activity.

In the following section, we demonstrate instantiation of the template method.

5 Three Pragmatic In-the-Field Method Instantiations

CADComp uses its own software operation data mining and analysis tool called Denerr, to extract operation information from the acquired CADProd operation data. Denerr was deployed on CADComp's intranet, and is accessible to all CADComp employees. The tool presents acquired error reports, and provides comprehensive data filtering functionality allowing developers and maintainers to define constraints on each of the error report properties, for example to analyze a particular set of error reports. The SOK integration template method was used to integrate operation information extracted by the Denerr tool in three of CADComp's product software processes: (1) software maintenance, (2) software product management and (3) customer support. Each process was selected based on CADComps needs expressed by its CEO, and corresponds to a particular SOK framework perspective [11].

Tables 2 and 3 respectively list how the template activities and concepts of the integration method were instantiated for each process. As part of the integration process, adjustments were made to the Denerr tool and to CADComp product software processes. CADComp employees involved in the target processes were introduced to new Denerr functionality by means of e-mails describing the new functionality, meetings of both software development and customer support departments during which new functionality was presented, and short hands-on evaluation sessions during which new functionality was demonstrated to and evaluated by particular employees. The SOK integration template method activities were performed in parallel. All integration activities were performed by the researchers and were overseen by the CADComp CEO.

5.1 Observations

As stated in table 2, weekly one-hour 'Denerr harvest meetings' were introduced to frequently analyze and delegate received error reports six months after initiation of our study. The meetings were organized with two maintenance team leaders and one researcher, and were led by the product manager made responsible for all Denerr-related issues. We attended the first three meetings in preparation of the interviews. With around 8200 error reports received since the start of error report acquisition seven months earlier, aims of the first meeting were to

Table 2. Template activity instantiations

Template activity	Target process (corresponding perspective)					
	Software maintenance (development)		**Software product management** (company)		**Customer support** (customer)	
	Time period	Activity instantiation	Time period	Activity instantiation	Time period	Activity instantiation
Analyze target process	12/09–01/10	Software maintenance is dependent on testing, customer support, customer training and sales feedback: error reports are just acquired and stored, progress regarding repairing software failures during software maintenance is not registered. CADProd developers are not used to involving end-user error reports in their maintenance work	01/10	The software product management process relies on operation information stored in customer support (CRM) software, as well as end-user feature requests and bug fix wishes reported by salesmen in planning features and bug fixes for upcoming releases. Error reports are not taken into account by product management in the context of these activities	01/10	Customer support is used to answer end-user questions, aid customers with software usage, identify software failures and sustain customer contact. Customer supporters have limited knowledge of the in-the-field operation of their software at customers, and are dependent on the customer's willingness to explain its situation and reason for calling
Determine integration objectives		Integration objectives (e.g. accelerated software maintenance) based on analysis of the imperfections of the current software maintenance process (e.g. process dependencies slowing down the maintenance process)		Integration objectives (e.g. directed software product (release) management) were based on analysis of the imperfections of the current software product management process (e.g. limited use of operation information)		Integration objectives (e.g. increased customer intimacy) were based on analysis of the imperfections of the current customer support process (e.g. little a priori knowledge of customer's reason for calling)
Identify operation information demands		No significant demand identification was performed. Software engineers requested for maintenance status information of error reports by their own efforts		During evaluation of the Denerr tool, product managers and CEO requested for aggregated operation information and software operation trends		Through short talks with customer supporters, operation information supporting identification of particular customers was identified as information requested for
Select relevant operation information		Error report meta information was identified as relevant information, based on information demands of software engineers as well as analysis of the current software maintenance process		Aggregated error report properties and software operation trends were identified as relevant operation information based on information demands of product managers as well as analysis of the current software product management process		Operation information which enables identification of customers was identified as relevant operation information based on information demands of customer supporters as well as analysis of the current customer support process
Identify SOK actors	01/10	Actors were selected from the employees that requested for operation information, or were appointed by CADComp's CEO	02/10	Actors were selected from the employees that requested for operation information, or were appointed by CADComp's CEO	01/10–02/10	Actors were selected from the employees that requested for operation information, or were appointed by CADComp's CEO
Estimate SOK utilization frequency		By analyzing the frequency of current software maintenance process activities, it was estimated that SOK would be used at least once a week		By analyzing the frequency of current software product management process activities, it was estimated that SOK would be used at least every Scrum sprint		By analyzing the frequency of current customer support process activities, it was estimated that SOK would be used potentially with every customer support call
Determine operation information carriers		Based on (meta) operation information demands and feedback from CADComp employees, a Denner error report list view with status information was selected as carrier		Based on (meta) operation information demands and feedback from CADComp employees, a Denner aggregated error report view was was selected as carrier		Based on (meta) operation information demands and feedback from CADComp employees, Denner customer-specific error report views were selected as carrier
Identify integration requirements		Based on results of previous activities as well as discussions with CADComp's CEO, creating and introducing error report labeling functionality were identified as integration requirements.		Based on results of previous activities as well as discussions with CADComp's CEO. Also, it became clear that creating and introducing error report aggregation functionality were identified as integration requirements. Also, it became clear that frequent meetings had to be organized for analyzing, discussing and delegating aggregated error reports		Extending the error report format and Denerr functionality with customer-specific elements were identified as integration requirements, based on results of previous activities as well as discussions with CADComp's CEO
Integrate operation information	02-10–03/10	Denerr was extended to show error report status, people responsible were assigned and team leader responsibilities were expanded	03/10–09/10	Various graphs have been designed and implemented (e.g., error cause modules, error report submission time, etc.), as well as an aggregation view containing the most frequent occurring errors. Also, 'Denerr harvest meetings' were introduced to frequently analyze and delegate received error reports	02/10–06/10	IP-based location determination was implemented by means of IP geolocation library, customer-specific error report analysis view was created, integration with external tools was realized
Evaluate integration results		After deployment of new Denerr functionality, short evaluation talks were held with the involved software engineers		After deployment of new Denerr functionality, short evaluation talks were held with the involved product managers		After deployment of new Denerr functionality, short evaluation talks were held with the involved customer supporters

Table 3. Template concept instantiations

Template concept	Target process (corresponding perspective)		
	Software maintenance (development)	**Software product management** (company)	**Customer support** (customer)
ACTOR	2 software engineers, 4 team leaders, 3 product managers	3 product managers, CEO	3 customer supporters, 2 software engineers
CARRIER	Error report list view with status information	Statistics, graphs, aggregation error reports view	Customer-specific error report view, acquisition, mining and analysis of subjective operation information (e.g. end-user comments)
INTEGRATION EVALUATION	Employees suggested that error reports could (and should) be automatically be assigned a status, particularly when at least 95% of the equivalent error reports is assigned one and the same status, to streamline maintenance work and reduce repetitive administration tasks. Also, it was suggested to tighten integration of acquired operation knowledge with the software maintenance process by keeping track of error reports status change(r)s over time, to analyze past and delegate future maintenance tasks. Both ideas were implemented. Finally, it was suggested to integrate error report data with bug tracking software, to align error reports with identified bugs.	Initially, the graph- and aggregation views were always generated for all error reports. Later, it appeared that more specific queries were requested for generating these views, resulting in views generated for error reports with a particular build number, status or IP address. Therefore, both views were implemented as a search result view. Also, it appeared convenient to delegate errors to teams using 'error report bundle' URLs instead of URLs to individual reports: teams are provided with insight in the scope and 'in-the-field severity' of software failures. 'Bundle URLs' were created to support this form of communication.	A basic customer view was first accessible via the detail view of each individual error report. Supporters made very little use of this customer view. A global customer search was implemented to alleviate the task of associating customer support details with operation information and tighten integration of operation information in the customer support process. Also, it was suggested to extend CADComp's Salesforce 'customer prepsheet' with operation information of the particular customer, to provide salesmen and supporters with insight in the customer's recent in-the-field software operation
INTEGRATION OBJECTIVE	Increase error report status awareness; enable error report status updating and monitoring; faster software maintenance	Gain insight in trending error report characteristics; directed software product (release) management	Gain insight in software operation at particular customers; increased customer intimacy
INTEGRATION REQUIREMENT	Labeling error reports upon receive with 'New' status, visualize labels within Denerr tool, implement label update functionality	Adding aggregated error report view and trending graphs to Denerr tool	Extending error report format and Denerr mining, analysis and reporting functionality (include customer profile and comment data)
INTEGRATION RESOURCE	1 software engineer, 3 team leaders, 1 product manager	1 software engineer, 3 team leaders, 1 product manager	1 software engineer, 3 team leaders, 1 product manager
SOFTWARE OPERATION INFORMATION	Error report status labels	Aggregated operation information, software operation trends	Operation information which enables identification of customers, such as IP address, customer and user name, license number, etc.
TARGET PROCESS	Software maintenance	Software product management	Customer support

(1) investigate which error reports could be considered old or irrelevant and put their status to 'Ignore', and therewith get a recent, realistic view of the status of all error reports, (2) delegate each aggregated report in the resulting top 10 of aggregated error reports to a software development team, and (3) investigate reports received in the last week to identify potential bugs introduced recently.

During the second and third meeting, respectively, the status of tasks identified during first harvest were discussed again with the team leaders, and reports with automatically assigned statuses (see table 3) were verified to check if the correct status was assigned. Based on our attendance of the meetings, three main observations were made. First, during analysis of the error reports, employees were surprised about the amount of reports being submitted, particularly from versions that were considered old versions by the employees. As a consequence, questions like '*When can we ignore error reports originating from a particular release build?*' and '*To which extent is it desirable to see the number of error reports received from a new software product significantly increase month over month?*' were discussed. Secondly, although employees understood the significance of the reports ('*Those error reports represent the unhandled exceptions that are experienced by our end-users*)', occasionally, insufficient data was available to gain a clear understanding of an end-user's software operation. As a result, end-user comments accompanying the error reports were frequently analyzed to identify the usage history and goals of end-users. Also, team leaders formulated more accurate operation information demands. Thirdly, we observed a demand for fine-grained report analysis. After aggregation and statistic views were implemented for all error reports, employees desired to show these views

only for reports originating from release builds, internal builds and per (build) version. The aggregation view was used to quickly identify which bugs were not under investigation for the current sprint. Bug fix work items were created, and engineers were managed based on the aggregation view.

5.2 Experience Evaluation

Twelve semi-structured interviews consisting of 34 questions divided over seven sections[3] (*Integration Objectives*, *Process Improvement*, *Integration Challenges*, *Return On Investment*, *Future*, *Lessons Learned* and *Final Remarks*), were performed after application of the SOK integration template method. Interviews were conducted with CADComp employees that are involved in a particular product software process (see table 4), to reflect on the SOK integration process and identify lessons learned. The method, tables 2 and 3 as well as observations of the attended Denerr harvest meetings served as input for the interviews. The interviews took 1.5 hour on average and were conducted over a period of 68 days. Interview results of the first five sections are summarized in table 5; lessons learned are presented separately in section 5.3.

Table 4. Interviewees per target process

Interviewee type	Software maintenance	Software product management	Customer support	Years experience in IT (average)
Senior supporters			3	16.3
Senior software engineers	2			12
Team leaders	3			8.5
Product managers		3		16.5
CEO		1		25

Although increase of software quality was considered a significant return on investment, a particular rival hypothesis regarding software quality increase was postulated often during the action research study and the reflective interviews. Various employees pondered over the actual cause of the decrease of received error reports: had the software quality actually been improved, or was there a random downward trend in software usage (or, more particularly, error report submission)? Error report submission history (see figure 4) indicates that software quality actually has been improved during period our study was conducted. While the number of submitted error reports increased about linearly from February, 2009 until June, 2010, this trend was broken in September, 2010 (the drop during July and August could well caused by summer vacation). In this month, a new major release of CADProd was released and delivered to customers, causing a slight increase in number of CADProd users.

5.3 Lessons Learned

The lessons learned listed below have been identified based on interview session results as well as observations during our presence at the vendor.

[3] Interview questions can be found at http://people.cs.uu.nl/schuurhw/sokintegration

Table 5. Interview results

Area	Software maintenance	Software product management	Customer support
Integration objectives	Quickly identify software failures most customers are experiencing frequently, faster improve software quality and performance, increase customer satisfaction	Gain more precise insight in software failures and weak spots in the software code base, efficiently increase software quality	Increase customer intimacy, increase efficiency of product software processes, get insight in software usage of customers that do not call for support
Process improvements	Software maintenance (e.g. bug prioritization): time was saved because software failures are faster reproducible, better insight in and awareness of in-the-field software operation and quality was gained	Software maintenance, software product management: processes have been accelerated because software failures are bundled (clustered) and prioritized automatically and discussed on a weekly basis (instead of manual, subjective bundling)	Software maintenance, customer support: more detailed information of in-the-field software operation is available which helps to faster determine failure causes in collaboration with the software development department
Integration challenges	Procrastination of developers during identification and reparation of software failures based on software operation information, coping with large amounts of acquired operation information (identifying what information is most relevant in which situations and cope with diversity of information during analysis)	Assigning responsibilities to employees in involving operation information in product software processes, while ensuring a balance between (1) the liberty of an employee and its team, and (2) improving the performance and efficiency of a department as a whole	Based on large amounts of acquired operation information, correctly determine what are actual causes of software failures and what additional (operation environment) information is needed to do so if those causes can not be correctly determined
Integration side effects	Operation information can be used to convince development management in taking release planning decisions, exception handling mechanism of the software was extensively refactored to increase software quality	Information on software usage is also gained, e.g. insight in a customer's software update policy can be gained	Customers feel taken seriously, especially when they are contacted after providing information regarding their software operation
Return on Investment[a]	Software quality (robustness) increase of 25%, decrease of software maintenance time of 50%[b]	Unhandled exception occurrence decrease of 40%, customer satisfaction increase of 25%	Customer support time decrease of 50%, software quality increase of 25%
Main future challenge	Knowing what are the functional requirements of the main customer (end-user) types, and to ensure that relevant, reliable operation information is extracted while data acquisition increases	Realizing a customer-specific approach in terms of software licensing and customer support, and finding an optimal balance between steering processes through operation information, and sustaining a leading role in industry by implementing a software product vision	Making sure that operation information is used and prioritized as effective as possible, while data acquisition sources and resulting operation data amounts increase

[a] All percentages are averages of rough estimations made by interviewees.

[b] Before operation information integration, software failures were frequently unreproducible and were never repaired.

1. Integration Processes Should be Lean. The effects of integrating operation information in product software processes should not be underestimated. Additional processes with corresponding responsibilities may be required to ensure effective and continuous integration of acquired operation information (for example, registering software maintenance tasks based this information and delegating those tasks to the right employee(s)). Vendors should ensure that additional (administrative) tasks caused by integration of operation information are handled in a pragmatic and lean way: additional administration may negate the time gain caused by integration of operation information.

2. Integration Responsibilities and Results Should Be Evangelized. An internal manager that has affiliation and experience with development-, business- and customer-related processes should be made responsible for integration of acquired operation information with those processes, since acquired operation information will not integrate automatically: during our action research study at CADComp, we observed that making integration of operation information everyone's shared responsibility, is effectively equivalent to making no one responsible. Inter alia, such a manager should ensure that the potential and results of the 'SOK-supported' process both are communicated clearly and frequently: this increases awareness and acceptance of the new way of working, both among employees as well as at management level. Evangelism of SOK integration potential and results is key in integrating operation information.

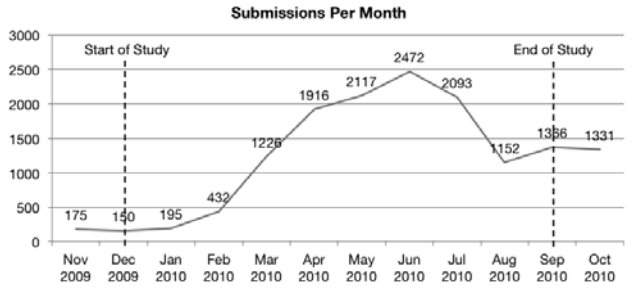


Fig. 4. CADProd error report submission decrease of about 45% during our study

3. SOK Integration Opens Up Black Boxes. In line with expectations of CADComp employees, integration of acquired operation information improved software maintenance, software product management and customer support processes: the time needed to reproduce software failures was decreased, deeper insight into (and awareness of) in-the-field software operation and end-user behavior was gained, and customer satisfaction was increased. Operation information is used for prioritization of fixes for software failures that are actually experienced by end-users, which may decrease the time required for software maintenance and customer support. However, as became clear after integrating CADProd operation information with CADComp processes, unanticipated improvements may result from effective SOK integration. For example, since developers are made aware of in-the-field software operation quality, SOK integration may result in a more customer-central, pro-active development mentality ('*build what the customer will use, before the customer asked for it*'). Also, integration of operation information in product software processes may clarify and speed-up interdepartmental communication as well as communication between employees and management. Improvement areas that were unnoticed before, are highlighted as such after (and potentially as a side effect of) SOK integration.

4. Continuous Refinement of SOK Integration Objectives and Requirements Leads to Optimization of Integration Results. Integration results are dependent on integration objectives and requirements. Since product software processes and activities change, as well as software operation environments and customer demands, integration objectives and requirements should be evaluated and refined continuously to correspond to both a vendor's product strategy as well as a vendor's customers needs. On the long term, software vendors should attain a balance between using operation information to steer their product software processes, and adhering to their product vision and strategy.

These lessons learned may serve as a guiding substrate for similar vendors in integrating information of in-the-field software operation.

5.4 Threats to Validity

The validity of the study results is threatened by several factors. First, construct validity of our study is threatened by the fact that the researchers conducting

the study were involved in objects of study (e.g. product software processes implemented at CADComp): observations or conclusions could be biased. This threat is addressed by adhering to the principles for canonical action research elicited by Davison et al. [3] (see section 3.2). For example, the researchers and software vendor being studied agreed on a research plan describing the shared objectives and data that was to be collected during the study. Also, both the researcher and the vendor reflected upon the outcomes of the study by means of semi-structured interview sessions.

Secondly, primary threat to the internal validity of the study is the relation between the instantiation of the SOK integration template method at CADComp and the subsequent software quality increase perceived by CADComp interviewees. Although it is a challenge to isolate the particular influence of template method instantiation on improvement of CADComp's product software processes, we regard CADComp's error report submission history (as analyzed in section 5) as representative of the extent to which CADComp's software maintenance, software product management and customer support processes are improved through instantiation of the SOK integration template method.

Thirdly, external validity is threatened by the fact that the SOK integration template method was instantiated at only one software vendor, during a limited period of time. While we acknowledge this threat, we regard the study as repeatable with the same results, presuming similar circumstances (e.g. similar operation information, processes, software vendors, etc.)

6 Conclusions and Future Work

All too often in industry, software vendors acquire large amounts of valuable software operation data, without effectively using these data in advancement of their processes. Operation information extracted from operation data is not structurally integrated with product software processes, leaving vendors in the dark regarding in-the-field software performance, quality and usage, as well as end-user feedback. Vendors are in need of an approach that supports them in accomplishing such integration.

We presented a template method that aids product software vendors in (1) identification of relevant and valuable operation information, (2) analysis of target processes and their integration environment, (3) integration of selected information in, and transformation of, target product software processes, and (4) presentation of integrated operation information. During an action research study of ten months performed at a European software vendor, the template method was instantiated to improve the vendor's product software processes through integration of acquired operation information.

Evaluation of the study shows that typical product software processes like software maintenance, software product management and customer support benefit from structural integration of operation information in terms of software quality, operation knowledge and customer intimacy. Based on this evaluation, four lessons learned are identified that may serve as a guiding substrate for similar

vendors, in integrating information of in-the-field software operation with their product software processes. We regard the SOK integration template method and lessons learned as an adequate early answer to the main research question of this paper, '*How can product software processes effectively be improved with acquired information of in-the-field software operation?*'. We demonstrated how product software processes can be improved pragmatically but measurably, without adhering to strict requirements from cumbersome maturity models or process improvement frameworks.

Future work will include additional action research or case studies to instantiate and evaluate the SOK integration template method in industry, and therewith further demonstrate its soundness and utility. Further research is also needed to mature the identified lessons learned towards generic guidelines or principles for effective integration of software operation information.

Acknowledgements

We would like to thank all CADComp employees for cooperating and contributing to our research by sharing their ideas and experiences.

References

1. Conradi, R., Fuggetta, A.: Improving Software Process Improvement. IEEE Softw. 19(4), 92–99 (2002)
2. Dangle, K.C., Larsen, P., Shaw, M., Zelkowitz, M.V.: Software Process Improvement in Small Organizations: A Case Study. IEEE Software 22, 68–75 (2005)
3. Davison, R.M., Martinsons, M.G., Kock, N.: Principles of canonical action research. Information Systems Journal 14, 65–86 (2004)
4. Fitzgerald, B., O'Kane, T.: A Longitudinal Study of Software Process Improvement. IEEE Software 16(3), 37–45 (1999)
5. Glerum, K., Kinshumann, K., Greenberg, S., Aul, G., Orgovan, V., Nichols, G., Grant, D., Loihle, G., Hunt, G.C.: Debugging in the (Very) Large: Ten Years of Implementation and Experience. In: SOSP 2009: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 103–116. ACM, New York (2009)
6. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28(1), 75–105 (2004)
7. Iversen, J.H., Mathiassen, L., Nielsen, P.A.: Managing Risk in Software Process Improvement: An Action Research Approach. MIS Quarterly 28(3) (2004)
8. Kuilboer, J.P., Ashrafi, N.: Software process and product improvement: an empirical assessment. Information and Software Technology 42(1), 27–34 (2000)
9. Miler, J., Górski, J.: Risk-driven Software Process Improvement - a Case Study. In: EuroSPI 2004: 11th European Conference on Software Process Improvement. Springer, Heidelberg (2004)
10. Pettersson, F., Ivarsson, M., Gorschek, T., Öhman, P.: A practitioner's guide to light weight software process assessment and improvement planning. Journal of Systems and Software 81(6), 972–995 (2008)

11. van der Schuur, H., Jansen, S., Brinkkemper, S.: A Reference Framework for Utilization of Software Operation Knowledge. In: SEAA 2010: 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 245–254. IEEE Computer Society, Los Alamitos (2010)
12. van der Schuur, H., Jansen, S., Brinkkemper, S.: Reducing Maintenance Effort through Software Operation Knowledge: An Eclectic Empirical Evaluation. Accepted for Publication in the Proceedings of the 15th European Conference on Software Maintenance and Reengineering, CSMR 2011 (2011)
13. Smite, D., Gencel, C.: Why a CMMI Level 5 Company Fails to Meet the Deadlines? In: Bomarius, F., Oivo, M., Jaring, P., Abrahamsson, P. (eds.) PROFES 2009. Lecture Notes in Business Information Processing, vol. 32, pp. 87–95. Springer, Heidelberg (2009)
14. Susman, G.I., Evered, R.D.: An Assessment of the Scientific Merits of Action Research. Administrative Science Quarterly 23(4), 582–603 (1978)
15. van de Weerd, I., Brinkkemper, S.: Meta-Modeling for Situational Analysis and Design Methods. In: Handbook of Research on Modern Systems Analysis and Design Technologies and Applications, pp. 38–58. Information Science Reference (2008)

Critical Issues on Test-Driven Development

Sami Kollanus

Faculty of Information Technology
University of Jyväskylä
Jyväskylä, Finland
sami.kollanus@jyu.fi

Abstract. During the last decade, Test-Driven Development (TDD) has been actively discussed in the software engineering community. It has been regarded as a useful and beneficial software development practice as well in industry as in academia. After a decade of active research, there is still very little critical discussion on TDD in the literature. This paper is based on a literature review and it is focused on identifying and introducing critical viewpoints on TDD. First, the current evidence on TDD's benefits is still weak and it includes several issues. Second, the paper presents a number of other possible issues and challenges with TDD that are referred in the literature. Finally, based on the findings, a list of concrete research questions for the future research is presented.

1 Introduction

Test-first programming is not a new idea, but it has been in known at least from the 1960s [9]. However, the concept has become well known in the software engineering community just during the last decade, when Test-Driven development (TDD) has been popularized as one of the key practices in eXtreme Programming (XP) [5]. Later TDD has been discussed as a stand alone practice [3] [6] and there is already a decade of active research on the field.

The basic idea of TDD is simply to write tests before code in small iterations. First, developer writes a test case that is just enough to define the next functionality. The next step is to write code that is just enough to pass the test. Finally, the code is refactored, if needed. These steps are iterated in short cycles through the whole development process. Originally TDD was introduced as a development, not a testing, method. [3] [6]

TDD has been suggested to provide a variety of different benefits, such as better productivity [21] [32], better quality [20] [47], high test coverage [7] [31], better program design [35] [57] and improved developers' confidence on their code [20] [35]. Regardless of the several suggested benefits, empirical evidence on them is still weak [37]. And there is very little critical discussion on TDD in the research literature.

This paper is focused on different critical viewpoints on TDD. The aim of the study is to identify the critical questions that have to be further studied in order to better understand TDD and its limitations in practice. The viewpoints

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 322–336, 2011.
© Springer-Verlag Berlin Heidelberg 2011

presented in this paper are based on a systematic literature review that was focused on the current empirical evidence on TDD's benefits [37].

The following section 2 will include description of the conducted literature review. Section 3 is focused on issues with the empirical evidence on TDD's benefits. Other possible issues, limitations and challenges with TDD are discussed in section 4. Finally, sections 5 and 6 include discussion and the main conclusions.

2 Literature Survey

This paper is based on a systematic literature review, in which the initial purpose was to evaluate the current empirical evidence on TDD. The main results of the literature review have been published in another paper [37]. Therefore the focus in this study is not on reporting the systematic literature review results, but on introducing critical viewpoints on TDD that have been found during the literature review process.

The review process was designed using the well known guidelines for systematic literature reviews on software engineering field [36]. In the beginning, a small preliminary review was conducted in order to identify the typical terminology and the potential sources of TDD related research. The following subsections will describe how the actual review process (search, data extraction, analysis) was conducted.

2.1 Sources

The search was limited to scientific journals, magazines and conference proceedings. The first step was to list the potential journals and magazines that most probably publish research on TDD. The list was created based on the preliminary review and the author's earlier experience on software engineering research. The final list included 15 publication series focused on software engineering. The electronic archives of each journal were searched by using their own search engines.

The next step was to find the relevant conference proceedings. Three important electronic databases (IEEE Explorer, ACM Digital library, Springer database) were included in the search. They were selected, because 1) they cover most of the relevant conference proceedings, 2) most of the publications are peer reviewed, and 3) they were easy to access.

Finally, the data extraction phase revealed several referred studies that were not found in the original search. All possibly relevant new references were included in the initial review data.

2.2 Search Procedure

Two searches were done on each of the sources using keywords: "TDD" and "test-driven development". The search results were manually checked based on tittles and abstracts, because most of the search results had nothing to do with test-driven development. At this stage, all the potentially relevant articles that are related to TDD were listed. The papers including empirical evidence were identified later.

2.3 Data Extraction and Analysis

The initial list of articles included 229 papers. Each relevant article of the initial list was read. Some of the papers were excluded, typically for three reasons. First, the papers discussing acceptance test-driven development or some related concept were excluded. Second, there were some XP papers that actually don't discuss anything about TDD. Third, there were also some publications that were not research papers. All the conference proceedings were regarded as research papers, but for example, short columns in IEEE Software were excluded. Finally, the initial review data included 165 articles and the following data was recorded of each paper.

- research question
- characteristics of the research method (e.g. empirical, controlled experiment, number of participants, target group)
- brief description of the research
- main results
- evidence on TDD (if any)
- **critical comments on TDD (if any)**
- how TDD was introduced in the paper

At this stage, all the relevant TDD related papers were still included in the material in order to create a broad view on the research field. During the data extraction process, the reported empirical evidence on TDD was recorded. There were finally 40 papers in the review data that included some kind of empirical evidence on TDD's benefits and this data was used in the systematic literature review.

2.4 Evaluation

The following actions were done during the review process to control the quality of the process and the results.

- A small preliminary review was conducted before the actual search phase to ensure good design of the review.
- The data extraction protocol was first piloted with 10 good quality journal articles. The plan was to revise the protocol after the pilot phase, but in this research there was no need for changes during the review process.
- References of the articles were searched during the data extraction phase to ensure that the review includes as many relevant articles as possible.

2.5 Role of Literature Review in This Paper

In order to identify different critical viewpoints on TDD, the literature review data is used in two different ways. First, the results of the systematic literature review were analyzed and found issues with the empirical evidence on TDD's benefits are introduced in section 3. Second, any critical comments on TDD were recorded of each 165 papers in the initial review data. Many of these comments are not based on empirical research, but more like anecdotal discussion in the publications. The main viewpoints of the discussion are summarized in section 4.

3 Issues with Empirical Evidence on TDD

There is an earlier paper published of the same project [37]. The paper is focused on reported empirical evidence of TDD. There was very limited number of critical viewpoints reported in the empirical studies. This section includes a summary of the analyzed empirical evidence as well as the few critical issues with TDD.

There were 40 papers in the review data that include some kind of empirical evidence on TDD. These papers represent different research questions, methods and varying quality. The main quality issue for the meta-analysis in the review data is reporting. Many of the research reports provide very limited information about the research setting. In a typical experiment, there is a group using TDD and a control group that follows "traditional development method". Both TDD and especially this traditional method are often very briefly (or not at all) defined. This makes interpretation of the results difficult. The same issue appears also in the case studies. They typically report some experiences of a TDD implementation, but often give very little details about the development process with TDD and the process used before TDD.

For the low number of high quality studies, all available evidence was included in the review data that is summarized in this section. The review data discuss a number of suggested benefits of TDD. However, most of the reported evidence on TDD is related to three general topics: 1) external quality, 2) internal code quality, and 3) productivity.

The following subsections 3.1-3.3 are focused on each of the main topics. They include rhe main conclusions of empirical evidence and also the raised questions about the suggested benefits of TDD.

3.1 External Quality

External code quality is measured using a couple of different metrics. Experiment settings typically include a well defined programming task and acceptance tests written by the researchers. In this kind of setting, external quality means number of passed acceptance tests. In the case studies, external quality usually means number of defects found before release or defects reported by customers.

There are 22 articles in the review data that include evidence on how TDD affects external code quality. The results of these studies are summarized in table 1. Most of the evidence (16 out of 22 studies) suggests that TDD improves external quality. However, the results are much more contradictory, if research

Table 1. How TDD affects external quality?

Method	Increase quality	No difference	Decrease quality
Contr. experiment	[24] [27]	[21] [30] [46] [48]	[41]
Case studies	[7] [13] [40] [45] [47] [52] [56] [61] [63]	[44]	
Other	[14] [19] [20] [49] [62]		
Total number	16	5	1

method is taken into account. Almost all the case studies support better external quality with TDD, whereas only two (out of 7) controlled experiments end up with the same conclusion. Most of the controlled experiments didn't find any difference in external and in one experiment the TDD group passed even less acceptance tests than the control group [41].

It may be concluded that the evidence provides weak support for better external quality with TDD. The case studies are quite consistent in reporting about better external quality after TDD implementation. However, the contradictory results from the controlled experiments leave some questions about the actual factors behind the improvement. For example, Huang and Holcombe [30] found in their controlled experiment that the TDD group passed more acceptance tests, but the trend was actually related to the time used for unit testing, not the development method.

The raised issues with external quality can be summarized to the following points:

- Case studies and controlled experiments give different results.
- In one of the experiments, TDD produced even less quality code.
- Are the reported quality improvements actually related to TDD or simply time invested in unit testing?

3.2 Internal Quality

It is more difficult to interpret the evidence on internal code quality, because there are so many different metrics used in the found 16 studies. The most common metrics were test coverage and number of test cases, but also method size, cyclomatic complexity, coupling and cohesion were measured in more than one study. So, the results are not completely comparable, but the evidence is summarized in table 2 based on the metrics used in each study. There may be also different results in a single study. For example, Siniaalto and Abrahamsson [54] found in their case studies that TDD improved test coverage, but suffered for lack of cohesion. In addition, they found no difference in few other internal code quality metrics. The study was classified in no difference category in the summary table.

Most of the controlled experiments found no difference in internal quality between TDD and a control group. For example, Madeyski [43] conducted a controlled student experiment, in which the control group was also advised to use iterative developments style and they were required to write unit tests after

Table 2. How TDD affects internal quality?

Method	Increase quality	No difference	Decrease quality
Contr. experiment	[11] [21]	[10] [22] [42] [43] [48]	
Case studies	[44] [51]	[54] [55]	
Other	[25] [31]	[32] [34]	[60]
Total number	6	9	1

the code. Madeyski measured branch coverage and mutation score of the students' code, but found no difference between the two groups. Also a couple other experiments have reported no difference in test coverage [48] [22]. The two controlled experiments that found TDD improving internal code quality, measured simply number of written test cases [21] [11]. They concluded that TDD may motivate to write more tests.

There are only few case studies that include some evidence on TDD and internal code quality. Madeyski and Szala [44] report on an academic case project that the team wrote less code per user story when using TDD. Rendel [51] concludes in his industrial case study that TDD improved code cohesion, whereas Siniaalto and Abrahamsson [54] [55] found lack of cohesion in their case studies and their results on the other internal quality metrics were contradictory. They concluded that TDD may produce less complex code, which is, however, more difficult to maintain. They also question whether the suggested benefits of TDD could be achieved just by emphasizing unit testing without the test first approach.

The raised issues with internal quality can be summarized to the following points.

- There are very few comparable studies, because the studies use different metrics.
- Most of the studies didn't find improvement with TDD.
- TDD may produce code that is more difficult to maintain.
- (again) Are the reported quality improvements actually related to TDD or simply time invested in unit testing?

3.3 Productivity

Productivity here is discussed very broadly and it includes any metrics that are related to development effort. In different studies, the used metric may be total development effort [7], lines of code per hour [44] or number of implemented user stories [21]. The evidence of the productivity is summarized in table 3.

As can be seen in table 3, there are very mixed empirical results of TDD's effect on productivity. The controlled experiments slightly support decrease in productivity and the case studies report more coherent results of decreased productivity. However, if all the studies are taken into account, majority (12 out of 23) of the studies report on no difference or even improved productivity with TDD. So, the evidence on productivity is not so clear and interpretation is even

Table 3. How TDD affects productivity?

Method	Incr. productivity	No difference	Decr. productivity
Contr. experiment	[21] [27]	[22] [25] [30] [46]	[10] [11] [23] [24]
Case studies		[44] [61]	[7] [13] [45] [47] [52] [63]
Other	[32] [62] [64]	[31]	[60]
Total number	5	7	11

more difficult. Several researchers (e.g. [47]) regard the reported increase in development report as a beneficial investment in improved external quality.

The raised issues with internal quality can be summarized to the following points:

- There are very few comparable studies, because the studies use different metrics.
- TDD may increase needed development effort.
- (again) Are the reported quality improvements actually related to TDD or simply time invested in unit testing? Especially several case studies report on both increased development effort and better quality with TDD, but they are not detailed enough to support further conclusions.

3.4 Summary of the Issues with the Empirical Evidence

The literature review revealed several issues with the empirical evidence on TDD's suggested benefits. Here are summarized the key issues.

Review data: The literature review data includes several issues for this kind of meta analysis. The number of the found relevant articles was relatively low. The found 40 articles represent different research questions, methods, metrics and quality. There are very few really comparable results.

Weak evidence: There is some evidence on better quality, but increased development effort with TDD. However, the reported results are contradictory and therefore it is hard to interpret them.

Is TDD the actual factor?: Several researchers raise the same question about the actual factors behind the claimed benefits of TDD (e.g. [30] [39] [55]). Better external quality may be more related to unit testing than the test first approach.

4 Other Issues and Challenges with TDD

The previous focused on issues raised up from the empirical evidence on TDD's benefits. In addition, the literature discusses several possible issues, limitations and challenges of TDD. The main points are introduced in the following subsections.

4.1 Lack of Design

One of the key ideas of TDD is the bottom-up approach with minimal upfront design. Several papers refer to lack of design as one of the potential issues in TDD [15] [39] [56]. For example, Kollanus and Isomöttönen found in their student experiment that even participants with several years of working experience suffered for lack of initial design in their relatively small assignments.

Lack of design may lead to other issues like heavy refactoring. Vodde and Koskela [59] give a small concrete example of how TDD may lead to unexpected refactoring. They don't regard this as a problem in their case, but in the bigger scale there may be more serious risk. The issue with TDD is not only refactoring

of production code, but also maintenance of the tests. Boehm and Turner [8] note that rapid changes may break the existing tests.

TDD has been claimed to produce better design in terms of less coupled and more cohesive code [4]. It was already discussed in the previous section that there is very little empirical evidence on better design with TDD. Siniaalto and Abrahamsson [55] found that TDD produced even less cohesive code. According to Madeyski [42], TDD may improve code quality in a single class, but in his experiment, TDD didn't affect quality on package level. Sangwan and LaPlante [53] suggest based on their experiences that TDD may work in the bigger scale, but there must be more focus on upfront design and integration tests.

4.2 Applicability

Applicability of TDD depends on the context. In some cases tests are simply too difficult write or writing them require too much effort. Beck [4] states that writing user interfaces test-first is hard. Grenning [26] introduces the challenges and possible solutions for TDD in embedded systems development. For example, it may be impossible to run unit tests in the target hardware.

In some cases, applicability of TDD is a question of available tools. Without proper tools, writing TDD may require too much effort. Test-driven database development is a good example of this. Ambler [2] presents concepts for it, but refers to challenges with the existing tools. Hamil et al. [28] have made the similar conclusion about TDD in developing web services with the common tools (Java + JUnit).

4.3 Test Code Size

The typical ratio of test to production code is typically at least one-to-one [16] [53]. A couple of possible issues are related to size of test code. First, full build may take too much time for TDD. Rasmussen [50] report in his case study how developers finally ran only selected (not optimal) set of tests, because full build took too much time. In their case the full build took 24 minutes at the end of the project. In large projects it may take several hours. So, it is often not possible to continuously run all the tests.

Another issue is maintenance of the test code. For example Rendel [51] presents in his case study a typical bad scenario with TDD. The most crucial point in the scenario is that refactoring work takes longer than expected, because also the tests need refactoring. In time pressure, it may lead developers to abandon the existing tests instead of maintaining them. So, if we rely on bottom-up development approach with refactoring, it possibly means substantial extra work on refactoring tests. Or possibly the skilled enough developers are able to minimize the need for refactoring and don't ever face this issue?

4.4 Required Skill Level

It has been claimed that TDD may need special skills and experience [15]. Kollanus and Isomöttönen [39] refer to one possible explanation for this in their

student experiment. They found that the participants had no problems with using TDD in an easy routine task, but with the more challenging task, the participants were in trouble. They would have needed some upfront design before the test-firsts programming. In TDD, the programmer must have the big picture of the program (at some level) in order to write tests for it. The more experienced programmers are able to form the design in their mind and therefore the bottom-up design approach suggested with TDD may work for them while the less advanced programmers end up with trouble.

Another viewpoint is type of the skills needed with TDD. Several studies agree that testing skills are different from programming skills. Writing proper test cases has been found as one of the most difficult issues with TDD [29] [38] [58].

4.5 Challenges in Adopting TDD

The most referred challenges in adopting TDD are learning curve and attitudes. First, several authors agree that learning TDD is not a simple task [21] [29] [38]. For example, Hedin et al. [29] found TDD as the most difficult to learn of all eXtreme Programming practices. Crispin [12] report in his case study that results improved significantly even during the second year after adopting TDD in the organization. So, learning TDD may take more effort and time than expected.

Second, the practitioners appear to intuitively have some negative attitudes on TDD. For example, Maximilien and Williams [45] report in their case study that many people both in development and management questioned productivity TDD. They also describe that typically some of the developers resisted change in their programming style.

Understanding the expected benefits of TDD may not be enough. Developers may have positive attitude towards TDD, but still have weak motivation to use it in practice, because they regard wring tests as extra effort on individual level [33] [1].

5 Discussion

There are several limitations in results presented in the previous sections 3 and 4. First, it must be understood that the purpose of this paper is not to give an objective picture of TDD, but broadly introduce reported critical viewpoints on it. Generally, most of the empirical evidence on TDD (section 3) is positive.

The main issue with the empirical evidence on TDD is quality of the primary data. There are very few comparable studies in the review data. The data includes 40 papers that represent different research questions, methods, metrics and quality. Often used methods and/or research setting are poorly introduced and therefore it would be impossible to replicate the study or interpret the results. For example, a typical case study presents positive results from a project that adopted TDD, but it does not describe in detail the changes in the development process or discuss about the other factors that may have impacted the results.

There are some issues to consider with generalizability of the empirical evidence on TDD. Most of the participants in the reported experiments were students or professionals with little or no experience on TDD. Can we make conclusions based on these results about TDD in an experienced team? And possibly the more important issue is typical research setting in the experiments, in which the participants do a small programming assignment that can be done within hours. What can we say about TDD's actual benefits based on such small programming task?

The mentioned issues with the empirical evidence on TDD appear to be more general issues on software engineering field. Dybå and Dingsøyr [18] systematically analyzed strength of evidence in their earlier systematic literature review on agile practices [17]. They concluded that strength of empirical evidence on agile practices is very low. The conclusion is based on similar issues that are related to TDD research in this study.

It can be questioned why it is so hard to find critical studies on TDD (or any other software engineering method). Of course, it is not an easy question to answer, but here are some guesses that may explain at least part of the phenomenon. First, companies don't want to publish any negative results. Second, also the researchers prefer publishing positive results. In many cases, it feels like the authors were "believers" of TDD (or some other method) and trying to proof its superiority instead of critically studying it.

Third, natural development of research seems to be part of the explanation. In order to publish a new software engineering method, the researchers have to evaluate it and reason how it is better than the other competitive methods. So, the first studies naturally present positive results. Then, for some reason, the next stage (at least in TDD/agile case) includes both case studies and experiments that report extremely positive results. It takes time, before more critical viewpoints develop on the research field.

Fourth, the software engineering community publishes new methods all the time and the research focus also moves on before a strong body of knowledge develops on any method. Such progress is a characteristic of the software engineering community. On individual level, most of the researchers are more motivated on studying new methods than creating substantial body of knowledge on a narrow research field.

6 Conclusions

This paper has discussed several critical viewpoints on test-driven development. First, there are several issues with the current empirical evidence on TDD, which was studied based on a systematic literature review. It was concluded that there is some evidence on better quality but decreased productivity with TDD. However, the primary review data included very few comparable studies and overall the empirical evidence on TDD is weak. In many cases, it can be questioned if TDD is the actual factor behind the reported results.

Table 4. Suggested questions for further research on TDD

Benefits of TDD
How TDD affects external software quality?
How TDD affects productivity? What is the overall cost of TDD?
How TDD affects internal code quality in terms of complexity and maintainability?
How should TDD be used in practice to maximize the benefits?
Are the found benefits related to test-first approach or time used for unit testing?
Other practical questions
Is upfront design needed with TDD?
How can the challenges with test code size be managed?
What are the limitations in applicability? In which context TDD is (or is not) applicable or beneficial?
What kind of skills are needed for effective use of TDD?
What kind of practical challenges (and solutions) are there in adoption of TDD?

Second, there are several other possible issues, limitations and challenges that have been related to TDD in the literature. The following main themes of this discussion were introduced in this paper.

- lack of design
- test code size
- applicability of TDD
- required skill level
- challenges in adopting TDD

The aim of this paper is not to question TDD as a software development practice, but find the critical questions that have be studied for better understanding on TDD in practice. There are some promising results of using TDD, but we need a lot of further research in order to understand the actual benefits, limitations and challenges of TDD. Table 4 includes some concrete research questions that are listed based on the critical issues discussed this paper.

References

1. Abrahamsson, P., Hanhineva, A., Jäälinoja, J.: Improving Business Agility Through Technical Solutions: A Case Study on Test-Driven Development in Mobile Software Development. In: Business Agility and Information Technology Diffusion. IFIP International Federation for Information Processing, vol. 180, pp. 227–243. Springer, Boston (2006)
2. Ambler, S.W.: Test-driven development of relational databases. IEEE Software 24(3), 37 (2007)
3. Astels, D.: Test Driven Development: A Practical Guide. Prentice Hall, Upper Saddle River (2003)
4. Beck, K.: Aim, fire [test-first coding]. IEEE Software 18(5), 87 (2001)
5. Beck, K.: Extreme Programming Explained: Embrace Change, 1st edn., p. 224. Addison-Wesley Professional, Reading (1999)

6. Beck, K.: Test-Driven Development: By Example. The Addison-Wesley Signature Series. Addison-Wesley, Reading (2003)
7. Bhat, T., Nagappan, N.: Evaluating the efficacy of test-driven development: industrial case studies. In: ISESE 2006: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, pp. 356–363. ACM, New York (2006)
8. Boehm, B., Turner, R.: Balancing Agility and Discipline - A Guide for the Perplexed. Addison-Wesley, Reading (2004)
9. Larman, C., Basili, V.R.: Iterative and incremental developments. a brief history. Computer 36(6), 47–56 (2003)
10. Canfora, G., Cimitile, A., Garcia, F., Piattini, M., Visaggio, C.A.: Evaluating advantages of test driven development: a controlled experiment with professionals. In: ISESE 2006: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, pp. 364–371. ACM, New York (2006)
11. Canfora, G., Cimitile, A., García, F., Piattini, M., Visaggio, C.A.: Productivity of test driven development: A controlled experiment with professionals. In: Münch, J., Vierimaa, M. (eds.) PROFES 2006. LNCS, vol. 4034, pp. 383–388. Springer, Heidelberg (2006)
12. Crispin, L.: Driving software quality: How test-driven development impacts software quality. IEEE Software 23(6), 70–71 (2006)
13. Damm, L.-O., Lundberg, L.: Quality impact of introducing component-level test automation and test-driven development. In: Abrahamsson, P., Baddoo, N., Margaria, T., Messnarz, R. (eds.) EuroSPI 2007. LNCS, vol. 4764, pp. 187–199. Springer, Heidelberg (2007)
14. Desai, C., Janzen, D.S., Clements, J.: Implications of integrating test-driven development into cs1/cs2 curricula. In: SIGCSE 2009: Proceedings of the 40th ACM Technical Symposium on Computer Science Education, pp. 148–152. ACM, New York (2009)
15. van Deursen, A.: Program comprehension risks and opportunities in extreme programming. In: Proceedings of the Eighth Working Conference on Reverse Engineering (2001)
16. Deursen, A.V., Moonen, L., Bergh, A., Kok, G.: Refactoring test code. In: Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP 2001), pp. 92–95 (2001)
17. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review. Inf. Softw. Technol. 50, 833–859 (2008), http://portal.acm.org/citation.cfm?id=1379905.1379989
18. Dybå, T., Dingsøyr, T.: Strength of evidence in systematic reviews in software engineering. In: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2008, pp. 178–187. ACM, New York (2008), http://doi.acm.org/10.1145/1414004.1414034
19. Edwards, S.H.: Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance. In: Proc. Int"l Conf. Education and Information Systems: Technologies and Applications, EISTA 20 03 (2003)
20. Edwards, S.H.: Using software testing to move students from trial-and-error to reflection-in-action. In: SIGCSE 2004: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, pp. 26–30. ACM, New York (2004)
21. Erdogmus, H., Morisio, M., Torchiano, M.: On the effectiveness of the test-first approach to programming. IEEE Transactions on Software Engineering 31(3), 226–237 (2005), doi:10.1109/TSE.2005.37

22. Flohr, T., Schneider, T.: Lessons learned from an XP experiment with students: Test-first needs more teachings. In: Münch, J., Vierimaa, M. (eds.) PROFES 2006. LNCS, vol. 4034, pp. 305–318. Springer, Heidelberg (2006)
23. George, B., Williams, L.: An initial investigation of test driven development in industry. In: SAC 2003: Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 1135–1139. ACM, New York (2004)
24. George, B., Williams, L.: A structured experiment of test-driven development. Information and Software Technology 46(5), 337–342 (2004)
25. Geras, A., Smith, M., Miller, J.: A prototype empirical evaluation of test driven development. In: METRICS 2004: Proceedings of the Software Metrics, 10th International Symposium, pp. 405–416. IEEE Computer Society, Washington, DC, USA (2004)
26. Grenning, J.: Applying test driven development to embedded software. IEEE Instrumentation Measurement Magazine 10(6), 20–25 (2007)
27. Gupta, A., Jalote, P.: An experimental evaluation of the effectiveness and efficiency of the test driven development. In: First International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, pp. 285–294 (September 2007)
28. Hamill, P., Alexander, D., Shasharina, S.: Web service validation enabling test-driven development of service-oriented applications. In: Proceedings of the 2009 Congress on Services - I, pp. 467–470. IEEE Computer Society, Washington, DC, USA (2009), http://portal.acm.org/citation.cfm?id=1590963.1591598
29. Hedin, G., Bendix, L., Magnusson, B.: Teaching extreme programming to large groups of students. Journal of Systems and Software 74(2), 133–146 (2005)
30. Huang, L., Holcombe, M.: Empirical investigation towards the effectiveness of test first programming. Information and Software Technology 51(1), 182–194 (2009)
31. Janzen, D., Saiedian, H.: Does test-driven development really improve software design quality? IEEE Software 25(2), 77–84 (2008)
32. Janzen, D.S., Saiedian, H.: On the influence of test-driven development on software design. In: CSEET 2006: Proceedings of the 19th Conference on Software Engineering Education & Training, pp. 141–148. IEEE Computer Society, Washington, DC, USA (2006)
33. Janzen, D.S., Saiedian, H.: A leveled examination of test-driven development acceptance. In: 29th International Conference on Software Engineering, ICSE 2007, pp. 719–722 (May 2007)
34. Janzen, D.S., Turner, C.S., Saiedian, H.: Empirical software engineering in industry short courses. In: 20th Conference on Software Engineering Education Training, CSEET 2007, pp. 89–96 (July 2007)
35. Kaufmann, R., Janzen, D.: Implications of test-driven development: a pilot study. In: OOPSLA 2003: Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 298–299. ACM, New York (2003), doi:10.1145/949344.949421
36. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report (2007), http://www.dur.ac.uk/ebse/resources/Systematic-reviews-5-8.pdf
37. Kollanus, S.: Test-driven development - still a promising approach? In: Proceedings of the 7th International Conference on the Quality of Information and Communications Technology, pp. 403–408 (2010)

38. Kollanus, S., Isomöttönen, V.: Test-driven development in education: experiences with critical viewpoints. In: Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, pp. 124–127. ACM, New York (2008)
39. Kollanus, S., Isomöttönen, V.: Understanding tdd in academic environment: Experiences from two experiments. In: Pears, A., Malmi, L. (eds.) 8th International Conference on Computing Education Research, Koli Calling 2008, pp. 25–31 (2009)
40. Lui, K.M., Chan, K.C.: Test driven development and software process improvement in china. In: Extreme Programming and Agile Processes in Software Engineering, pp. 219–222 (2004)
41. Madeyski, L.: Preliminary analysis of the effects of pair programming and test-driven development on the external code quality. In: Proceeding of the 2005 Conference on Software Engineering: Evolution and Emerging Technologies, pp. 113–123. IOS Press, Amsterdam (2005)
42. Madeyski, L.: The impact of pair programming and test-driven development on package dependencies in object-oriented design — an experiment. In: Münch, J., Vierimaa, M. (eds.) PROFES 2006. LNCS, vol. 4034, pp. 278–289. Springer, Heidelberg (2006)
43. Madeyski, L.: The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment. Inf. Softw. Technol. 52(2), 169–184 (2010)
44. Madeyski, L., Szała, Ł.: The impact of test-driven development on software development productivity — an empirical study. In: Abrahamsson, P., Baddoo, N., Margaria, T., Messnarz, R. (eds.) EuroSPI 2007. LNCS, vol. 4764, pp. 200–211. Springer, Heidelberg (2007)
45. Maximilien, E.M., Williams, L.: Assessing test-driven development at ibm. In: Proceedings of the 25th International Conference on Software Engineering, pp. 564–569 (May 2003)
46. Muller, M.M., Hagner, O.: Experiment about test-first programming. Software, IEE Proceedings 149(5), 131–136 (2002)
47. Nagappan, N., Maximilien, E.M., Bhat, T., Williams, L.: Realizing quality improvement through test driven development: results and experiences of four industrial teams. Empirical Softw. Engg. 13(3), 289–302 (2008)
48. Pancur, M., Ciglaric, M., Trampus, M., Vidmar, T.: Towards empirical evaluation of test-driven development in a university environment. In: EUROCON 2003. Computer as a Tool. The IEEE Region 8, vol. 2, pp. 83–86 (September 2003)
49. Rahman, S.M.: Applying the tbc method in introductory programming courses. In: 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, FIE, pp. T1E–20–T1E–21 (October 2007)
50. Rasmussen, J.: Introducing xp into greenfield projects: lessons learned. IEEE Software 20(3), 21–28 (2003)
51. Rendell, A.: Effective and pragmatic test driven development. In: Conference on AGILE 2008, pp. 298–303 (August 2008)
52. Sanchez, J.C., Williams, L., Maximilien, E.M.: On the sustained use of a test-driven development practice at ibm. In: AGILE 2007, pp. 5–14 (August 2007)
53. Sangwan, R.S., Laplante, P.A.: Test-driven development in large projects. IT Professional 8(5), 25–29 (2006)

54. Siniaalto, M., Abrahamsson, P.: A comparative case study on the impact of test-driven development on program design and test coverage. In: First International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, pp. 275–284 (September 2007)
55. Siniaalto, M., Abrahamsson, P.: Does test-driven development improve the program code? Alarming results from a comparative case study. In: Meyer, B., Nawrocki, J.R., Walter, B. (eds.) CEE-SET 2007. LNCS, vol. 5082, pp. 143–156. Springer, Heidelberg (2008)
56. Slyngstad, O.P.N., Li, J., Conradi, R., Rønneberg, H., Landre, E., Wesenberg, H.: The impact of test driven development on the evolution of a reusable framework of components - an industrial case study. In: ICSEA 2008: Proceedings of the 2008 The Third International Conference on Software Engineering Advances, pp. 214–223. IEEE Computer Society, Washington, DC, USA (2008)
57. Steinberg, D.H.: The effect of unit tests on entry points, coupling and cohesion in an introductory java programming course. In: XP Universe Conference 2001 (2001)
58. Tinkham, A., Kaner, C.: Experiences teaching a course in programmer testing. In: ADC 2005: Proceedings of the Agile Development Conference, pp. 298–305. IEEE Computer Society, Washington, DC, USA (2005)
59. Vodde, B., Koskela, L.: Learning test-driven development by counting lines. IEEE Software 24(3), 74–79 (2007)
60. Vu, J.H., Frojd, N., Shenkel-Therolf, C., Janzen, D.S.: Evaluating test-driven development in an industry-sponsored capstone project. In: Sixth International Conference on Information Technology: New Generations, ITNG 2009, pp. 229–234 (April 2009)
61. Williams, L., Maximilien, E.M., Vouk, M.: Test-driven development as a defect-reduction practice. In: 14th International Symposium on Software Reliability Engineering, ISSRE 2003, pp. 34–45 (November 2003)
62. Xu, S., Li, T.: Evaluation of test-driven development: An academic case study. In: Software Engineering Research, Management and Applications 2009. Studies in Computational Intelligence, vol. 253, pp. 229–238 (2009)
63. Ynchausti, R.A.: Integrating unit testing into a software development team's process. In: Intl. Conf. eXtreme Programming and Flexible Processes in Software Engineering, pp. 79–83 (2001)
64. Zhang, L., Akifuji, S., Kawai, K., Morioka, T.: Comparison between test driven development and waterfall development in a small-scale project. In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) XP 2006. LNCS, vol. 4044, pp. 211–212. Springer, Heidelberg (2006)

On the Difficulty of Computing the Truck Factor

Filippo Ricca[1], Alessandro Marchetto[2], and Marco Torchiano[3]

[1] DISI, Universitá di Genova, Italy
filippo.ricca@disi.unige.it
[2] FBK-irst, Trento, Italy
marchetto@fbk.eu
[3] Politecnico di Torino, Italy
marco.torchiano@polito.it

Abstract. In spite of the potential relevance for managers and even though the Truck Factor definition is well-known in the "agile world" for many years, shared and validated measurements, algorithms, tools, thresholds and empirical studies on this topic are still lacking.

In this paper, we explore the situation implementing the only approach proposed in literature able to compute the Truck Factor. Then, using our tool, we conduct an exploratory study with 37 open source projects for discovering limitations and drawbacks that could prevent its usage.

Lessons learnt from the execution of the exploratory study and open issues are drawn at the end of this work. The most important lesson that we have learnt is that more research is needed to render the notion of Truck Factor operative and usable.

Keywords: Truck Factor, Collective Code Ownership, Exploratory study.

1 Introduction

The Truck Factor (TF) of a project is defined as "*the number of developers on a team who have to be hit with a truck before the project is in serious trouble*"[1]. Clearly, "to be hit with a truck" is an extreme thought that can be substituted with more realistic ones such as, for example, to go on vacation, to become ill, to be out of the office or to leave the company for another. Ideally, to avoid potential problems, as advocated by the Extreme Programming (XP) principle of "collective code ownership" [1], the TF of a project should be as high as possible. Indeed, if all the knowledge of a system is in the hands of few developers only (the key contributors, also called Heroes [2]) and they decide to leave the project, then the same project could suffer the consequences.

The TF can be used as: (i) a quick metric that will highlight potential problems in a project ("*Having heroes on your team can be very beneficial but only if you don't become dependant on them. Truck factor is one metric that will highlight your dependencies*"[1]), (ii) an indicator of how expensive it will be to

[1] http://www.agileadvice.com/archives/2005/05/truck_factor.html

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 337–351, 2011.
© Springer-Verlag Berlin Heidelberg 2011

replace some developers in a project, and (iii) a way to measure the system's knowledge distribution in a team of developers.

In the best of our knowledge, in spite of the potential relevance for managers, neither shared and validated measurements nor tools implementing algorithms devoted to measure the TF exist. Moreover, no empirical studies exist on this topic.

In this paper, we started filling this gap. First, we extensively consulted the literature (Sect. 2) learning that few works exist about this topic. In particular, only Zazworka et al. [3] propose an approach to compute the TF using information about code ownership (Sect. 3). Second, following the approach by Zazworka et al., we implemented a tool (already used in [4] but detailed here for the first time) able to compute the TF starting from the code repository of a project (Sect. 4). Finally, we used our tool to conduct an exploratory study analyzing a large set of free and open source projects (Sect. 5) with the purpose of testing the Zazworka et al.'s approach to find possible limitations and problems.

2 Related Works

This work investigates the TF measurement. It is hence related to the notion of code ownership, given that the TF represents a way to evaluate code ownership. Non-ownership [5] for the code of a software system exists when the system and its sub-parts have no specific owners (i.e., developers responsible to implement and maintain the code), and thus every developer can potentially change each part of the system. Even if several projects de-facto apply such a non-ownership model, it is well-recognized (e.g., [5] and [6]) that non-ownership can lead to: (i) poor or missing documentation, (ii) unreadable source code due to mixtures of styles and inconsistencies, and (iii) long cycles of bug fixing and code maintenance. To face such problems, often, managers try to intentionally control the code ownership during the software development and maintenance. Nordberg III [5] identifies four models of code ownership: (1) product specialist, (2) subsystem ownership, (3) chief architect and (4) collective ownership. In (1) and (2), a developer is respectively the owner of the whole system or of a sub-part of it. In (3) there is a developer responsible of the system helped by several assistants. In (4) the code is owned by several developers and a developer that owns a piece of code can be also a contributor for another piece of the software by collaborating with other developers. The choice of the ownership model to apply is currently a source of debate for the scientific community. Two major trends can be identified. Some works (e.g., [6]) support strong ownership (by applying models such as (1), (2) and (3)) for motivating developers to produce high quality code and specializing the code knowledge of each developer. Other works (e.g., [7]) support collective code ownership (by applying the model (4)) for distributing software knowledge and responsibility among the team developers.

To monitor, analyze and thus change the ownership policy during the software life-cycle, adequate information and tools are needed. The major source of information for computing and evaluating code ownership is, often, the code

repository (e.g., SVN, CVS) and the activities (i.e., *commits*) performed by developers (i.e., *committers*) on code files and tracked in log files. These log files mainly contain the list of activities performed by developers on the code and, for each activity, some additional information (e.g., the version number of the system in which the activity has been executed). Different approaches have been proposed to measure the code ownership by analyzing these repositories. For instance, Weyuker et al. [8] propose to compute the ownership at file level while Girba et al. [9] propose a measure based on the percentage of source code lines modified by a specific developer.

Several works (e.g., [10] and [11]) analyze code ownership and developer activities tracked by code repositories for automatically identifying experts for a particular portion or aspect of a system. These recommender systems are often used to assign a developer (i.e., the more suitable) to a change request. These tools make the implicit assumption that a code change performed by a developer indicates the knowledge of the involved system portion for that developer. Fritz et al. [11] perform an analysis of developer activities comparing them with the actual knowledge of code achieved by developers. Their analysis confirms that the frequency of the interaction of developers with the code mainly indicates which portions of the system/code a given developer known.

In a previous short paper [4], we applied our tool to 20 open source projects to answer two research questions concerning the notions of Heroes (i.e., tireless developers) and TF. During that experiment, we discovered some TF usage problems addressed in this paper with new research questions (see Sect. 5.1). Preliminary results of [4] showed that: Heroes are common in open source projects and that the TF is in general low. In another paper [2], we used the same dataset of this work to discover existing relationships between the presence of Heroes and the time required to implement change requests. Preliminary results showed that the presence of Heroes in a project seems to be beneficial because reduce the time to implement change requests.

3 Truck Factor Usage Problems

3.1 No Shared and Validated Measurements

One of the biggest problems with TF is that there are not shared and validated measurements/metrics to compute it. That constitutes a huge limitation on its usage. The unique proposal, at least to the best of our knowledge, has been carried out by Zazworka et al. [3] who proposed an initial idea in which the TF is computed using the code ownership information inferred from a code repository. Moreover, they applied that approach to five small projects written by students to check the difference between the TF of XP/non-XP projects. That preliminary experiment provides the first evidence that non-XP projects have significant lower (i.e., worse) TFs than XP projects.

The core concept underlying their proposal is that a file in the repository is considered collectively owned by all the developers who worked on it. The assumption is that developers who edited the file have knowledge about it. In

Table 1. Jfreechart example

File (.java)	Dev.Set	File (.java)	Dev.Set
SWTUtils	{mungady, nenry}	DefaultKeyedValues	{mungady, taqua}
ChartColor	{mungady}	PolarChartPanel	{mungady}

Table 2. Truck Factor example

	TF: number of missing developers							
File (.java)	0	1	1	1	2	2	2	3
		{mungady}	{taqua}	{nenry}	{mungady, taqua}	{mungady, nenry}	{nenry, taqua}	{mungady, taqua, nenry}
SWTUtils	+	+	+	+	+	-	+	-
DefaultKeyedValues	+	+	+	+	-	+	+	-
ChartColor	+	-	+	+	-	-	+	-
PolarChartPanel	+	-	+	+	-	-	+	-
File Coverage %	100	50	100	100	25	25	100	0
Min. File Coverage %	100	50			25			0

this way, for each file 'f' of a project, the *developer set* for 'f' consists of the set of developers that did at least one commit on 'f'.

To better explain their proposal we consider some files of Jfreechart as an example. Table 1 and Table 2 show respectively details about the four considered files (i.e., SWTUtils.java, DefaultKeyedValues.java, ChartColor.java and PolarChartPanel.java) and their TF computation.

For example, columns "File" and "Dev.Set" of Table 1 show that the set of developers that did at least one commit on the file "SWTUtils" is {mungady, nenry}. The first rows of Table 2 present the possible sets of developers potentially "hit by a truck". In particular, the second row of that Table indicates how many developers would be missing (from 0 to 3), while the third row details exactly who. In each Table cell, the sign "+" means that the remaining developers (i.e., those developers in the Dev.Set that have not been "hit by a truck") have the knowledge of the corresponding file reported in the first column. Instead, the sign "-" means that the developers "hit by a truck" were the only to know that file. The penultimate row of the Table reports the preserved file coverage (or residual knowledge) measured as percentage, precisely, the number of files known by the remaining developers divided by the total number of files in the project * 100. Finally, the last row reports the minimum file coverage per number of missing developers. Let us consider, for example, the scenario in which the developer *mungady* leaves the project (see the third column of Table 2). When losing *mungady* the knowledge of "ChartColor" and "PolarChartPanel" is lost. This implies that in such a case the remaining file coverage corresponds to 50% (i.e., 2/4*100).

To identify the TF, a target threshold (e.g., 50%) that represents the critical file coverage for a project has to be defined. Then, considering the minimum file coverage (i.e., the worst case, where the set of developers with the most exclusive knowledge leaves) computed for set of missing developers, it is possible

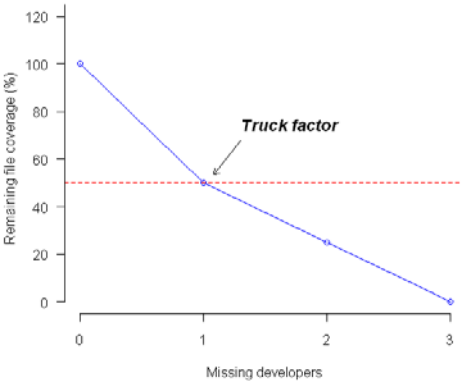


Fig. 1. Truck Factor chart

to plot a curve and identify the actual TF for the project. Indeed, the TF can be deduced from this plot by finding the intersection of the selected threshold with the curve; if the obtained value is not an integer the point on the left must be selected. Figure 1 represents the TF chart corresponding to Table 2 where the threshold has been fixed at 50%, thus obtaining a TF of 1. Instead, considering $threshold = 70\%$ the TF will be 0 and with $threshold = 20\%$ it will be 2. The example shows that, with the approach proposed by Zazworka et al., the TF of a project can also be 0. It corresponds to a critical situation in which the developer with the most exclusive knowledge can not leave the project without heavy consequences.

We think that the Zazworka et al. approach is very interesting and promising but at this stage it has two main drawbacks that could prevent its usage. First, the proposal is based on a really strong assumption: at least a commit on file 'f' implies the knowledge of 'f'. In this way, a developer which has done one commit is considered to have the same knowledge of a file than one that has done more. This is clearly questionable; maybe the measurement should also account of this aspect somehow. Second, the proposed approach does not come with an algorithm describing the steps for computing the TF, nor with a tool able to compute it.

3.2 No Reliable Thresholds

In the TF computation, thresholds are fundamental. As we have seen in the approach proposed by Zazworka et al., the TF strictly depend on a threshold that represents the critical file coverage for the target project. To compute the TF such a threshold, given in percentage, has to be defined. In other words, the TF is a parametric measurement depending on a threshold X (i.e., TFX%).

Moreover, to establish whether the TF is high or low we need another fundamental threshold. Without it, we can not infer/understand whether or not the target project has low TF and thus risks to get in trouble.

So far, to the best of our knowledge, Siddharta Govindaray has been the only one that tried to identify/propose some thresholds in this context. On his blog[2], he claims that: "*Small teams of under 10 people usually target a TF of 4-5 for most parts of the system (that is around 40-50% of the team). Larger teams will probably target a TF of around 8 (which would probably be around 20-25% of the team). This means that should a couple of critical people go on vacation or leave the company, there are enough people in the team who can cover for them.*" Therefore, one can make the assumption that for a *small team* (i.e., ≤ 10 developers) the TF can be considered low if it is $\leq 40\%$ of the team size. Instead, for a *large team* (i.e., > 10 developers) the TF can be considered low if it is $\leq 20\%$ of the team size. Unfortunately, Govindaray on his blog remained vague on the other threshold (the threshold for file coverage) with the sentence "*for most parts of the system*".

The problem with those two thresholds is that we can not trust them. They appeared on a non reliable source of information (a blog post) and it is not clear how they have been deduced by Govindaray (expert opinion or by measurement data from a representative set of projects?). Moreover, it is not clear which TF measurement has used Govindaray to deduce the thresholds since the definition of TF could permit different interpretations and measurements (in another blog, we found the steps suggested by Govindaray to compute the TF: (1) Go through each class, file or component; (2) Calculate the number of people in the team, who understand that part of the system; (3) This give you the TF for that class or component; (4) The smallest of all the component TFs is the TF for the project). As a consequence, we have no guarantee that those thresholds can be safely used with the Zazworka et al. approach.

3.3 No Tools

To the best of our knowledge, no tools exist for computing the TF of a project (neither proprietary nor free tools) and no algorithms have been proposed for this purpose in literature. To overcome this lack, we have proposed a naive algorithm based on the Zazworka et al. approach and we have implemented it in a tool available on-line[3]. The tool and the algorithm are explained in detail in the next section.

3.4 No Empirical Evidence

To the best of our knowledge, no empirical studies exist on this topic. For example, an interesting research question not addressed in the literature is the following: *Does a low TF really increase the risk of project failure?*. This research question concerns the well-known XP principle of "collective code ownership" [1] that can be summarized with the following phrase: to decrease the risk of project failure the TF of a project should be high. Indeed, if all the knowledge of a system is only in the hands of few developers and they decide to leave the project,

[2] http://siddhi.blogspot.com/2005/06/truck-factor.html
[3] http://selab.fbk.eu/marchetto/tools/tf/tf.zip

then the same project could get in trouble. This reasonable principle has been advocated by XP several years ago but ever since nobody has never tried to empirically show its truth and adherence to reality. We believe that this is also due to the lack of measurements, tools and thresholds on this topic.

With the aim of partially filling this gap, we have conducted the first experimental study concerning the TF with the purpose of testing the approach proposed by Zazworka et al. on real projects (so far, it was tested only on students' projects). Given this state of uncertainty, we opted for an exploratory study. It is well-known that an exploratory study does not draw definitive conclusions. An exploratory study is undertaken to better comprehend the nature of a problem since very few studies have been considered in that area. The purpose of an exploratory study is: "*finding out what is happening, seeking new insights and generating ideas and hypothesis for new research*" [12].

4 Truck Factor Tool

The proposed tool takes in input a couple of values, the URL of the target SVN code repository and the critical file coverage threshold, and returns the TF. It is composed of three modules:

- `Extractor`. A script based on SVN client commands ("svn list" and "svn log") to traverse the code repository and extract logs for each code file;
- `Scanner`. A pattern-based string matching script able to analyze such logs for extracting information about the committers. The output of this module is: (1) the list of project committers, (2) the code files in the repository, (3) and for each file the list of developers that performed at least one commit on it;
- `Analyzer`. A Java program that analyze the information produced by the `Scanner` and compute the TF.

Figure 2 shows the pseudo-code of the `Analyzer`. The `Analyzer` takes in input (see the declarative part of Figure 2): (1) the critical file coverage threshold (X), (2) the list of project committers (*Developers*), (3) the files in the repository (*SourceFiles*), and (4) for each file the list of developers that performed at least one commit on it (function *DeveloperSet*). Basically, for each possible combination of committers ($comb \in Comb(j, Developers)$, with $1 \leq j \leq developers\_number$), the corresponding percentage of the remaining file coverage (*FileCoverage*) is computed dividing (see Table 2) the number of "+" (*tot*) by the total number of files in the repository (*SourceFiles.lenght()*). Even if this file coverage is calculated for each committers combination, only the minimum one (*minFileCoverage*) is stored in *minfilecoverageVector*. Finally, the project TF is j-1.

Let 'n' be the number of committers/developers of the target project and 'm' the number of files in that project. Then, the worst case time complexity of the above naive algorithm is given by:

$$T(n, m) = n * \sum_{i=1}^{n} \frac{n!}{i!(n-i)!} * m$$

```
-X: int
//Critical file coverage threshold, e.g., 50%, 60%
-Developers: List
//Project developers, e.g., {mungdy, nenry and taqua}
-SourceFiles: List
//Project files, e.g., {SWRUtils, ChartColor,...}
-DeveloperSet(f: File): Function
//It gives the list of committers for the file f,
//e.g., DeveloperSet(SWRUtils)= {mungdy, nenry}
-Comb(j: Number, Developers: List): Function
//It lists all the combinations of j developers,
//e.g., Comb(2, Developers)= {<mungady,taqua>, <mungady,nenry>, <nenry,taqua>}

for j:=1 to Developers.lenght()
 minfileCoverage=100;
 for each comb ∈ Comb(j, Developers)
  tot=0;
  for each f ∈ SourceFiles
   if (DeveloperSet(f)− comb) ≠ null then tot=tot+1;
  endFor
  FileCoverage=tot/(SourceFiles.lenght()*100);
  if FileCoverage≤ minFileCoverage then minFileCoverage=FileCoverage;
 endFor
 minfilecoverageVector[j]=minFileCoverage;
 if minfilecoverageVector[j]<X then break;
endfor
print(``TFX%='', j-1);
```

Fig. 2. Core algorithm for the Truck Factor computation

5 Exploratory Study

Using our TF tool, we conducted an experimental study with a selection of open source projects taken from SourceForge[4] and GoogleCode[5]. We randomly selected some non-trivial projects with the following characteristics: number of files > 35, number of committers ≥ 3 and number of commits > 40. We selected both small and large projects (i.e., with more than 10 developers). Finally, the objects of our study were 37 open source projects.[6] Large projects are 1/3 of all the projects.

This experimentation, conducted according to the guidelines proposed by Wohlin et al. [13], aims at answering the following three research questions.

RQ1: *Is the tool based on Zazworka et al approach applicable on real projects?*. This research question deals with the ability of the approach to find the TF in real projects. So far, the approach has been tested only on small projects written by students. Given the high complexity of the naive algorithm, we are interested to empirically test its scalability.

RQ2: *Are the obtained results sensible?*. This research question investigates whether the computed TFs are sensible or not. Picking randomly the open source

[4] http://sourceforge.net/
[5] http://code.google.com/intl/it-IT/
[6] Raw data are available at http://selab.fbk.eu/marchetto/tools/tf/tf.zip

projects we expect a more or less random distribution of the computed TFs. Moreover, we expect that: (i) large projects have greater TF than small projects and that (ii) diminishing the file coverage threshold of a project (e.g., from 60% to 50%) its TF increases.

RQ3: *Is the Govindaray's threshold consistent with the Zazworka et al approach?*. This research question investigates if the Govindaray's threshold can be applied to discriminate low/high TFs when the Zazworka et al. approach is used. Indeed, the approach and the threshold have been independently proposed and it is possible that a different measurement of TF has been used by Govindaray during the assignment process (or deduction) of the threshold (a shared and validated measurement to compute the TF does not exist, see Sect. 3.1).

5.1 Execution and Results

Table 3 details some information about the 37 analyzed free and open source projects. That Table reports: code repository (SourceForge or GoogleCode), application's name, number of files, lines of code (LOCs), number of committers of the project, estimated dimension (small or large) and total number of analyzed system revisions (i.e., commits). In the analyzed sample, 24 projects are classified as small and 13 as large (consistently with the Govindaray's blog, we considered as large the projects with more than 10 developers).

Is the tool based on Zazworka et al approach applicable on real projects? We applied our tool to the applications listed in Table 3 obtaining the results of Table 4 (the TF has been computed considering different values of file coverage, i.e., 50%, 60% and 70%). The results of these first runs show scalability problems of our tool. Computation time was acceptable for all the applications, it was in the range of few seconds/minutes on a desktop PC (i.e., with an Intel(R) Core(TM) 2 Duo CPU working at 2.66GHz and with 3GB of RAM memory) except for `V8` and `mantisbt` (the largest projects of our dataset, respectively 32 and 38 committers). In these last two cases the computation time took some days and this is not acceptable from our viewpoint. In conclusion, the tool implementing the idea proposed by Zazworka et al. seems applicable only when the number of committers of the target project is ≤ 30. Overall, the number of files of the project seems exert a smaller influence on the overall scalability.

Are the obtained results sensible? Table 4 reports the TFs computed considering different values of file coverage (i.e., 50%, 60% and 70%) and the Govindaray's threshold (column *threshold*). Apparently, all the TFs are low (see the complete distribution of TF50% in Figure 3 (a)) and in several cases the TF is equal to zero (TF=0). Considering $TR50\%$ this is true for 12 out of 24 small projects and for 6 out of 13 large projects. That high percentage of TF=0 in both the categories is suspicious to us. In particular, this is strange for large projects. In addition, observing Table 4 it is apparent that for several projects the computed TF is lower than the threshold derived following Govindaray (it means that those projects have low TF). Note that this is true even when we

Table 3. Randomly selected applications. From left to right: repository, selected application, number of files, Lines Of Code, number of committers, dimension of the project (large if > 10 committers) and number of commits.

Repository	Software	Files	LOCs	Committers	Project Size	Revisions
Google	choscommerce	69	11016	4	small	224
Google	closure-compiler	536	123660	4	small	200
Sourceforge	cppunit	378	22646	6	small	582
Google	cspoker	208	144111	4	small	1382
Google	easycooking-fantastici4	141	15460	4	small	203
Sourceforge	erlide	820	134753	5	small	3155
Google	gdata	344	46805	19	large	982
Google	gmapcatcher	77	7164	6	small	842
Google	googlemock	55	56410	4	small	300
Sourceforge	gtk-gnutella	723	248697	15	large	17424
Google	h2database	722	125944	6	small	2628
Sourceforge	htmlunit	3404	548666	10	small	5743
Sourceforge	httpunit	354	39232	3	small	1062
Sourceforge	jfreechart	1063	149727	3	small	2272
Google	jpcsp	672	355728	20	large	1504
Google	jstestdriver	438	32082	9	small	603
Sourceforge	jtidy	174	30487	5	small	115
Google	keycar	669	83936	4	small	465
Sourceforge	mantisbt	641	3399759	38	large	5752
Google	mobileterminal	43	8095	12	large	364
Google	moofloor	39	10152	4	small	169
Google	nettiers	267	108159	13	large	837
Google	pagespeed	297	39069	6	small	845
Sourceforge	phpwiki	537	118815	15	large	7370
Sourceforge	remotes	289	35646	3	small	812
Google	sqlitepersistentobjects	170	17727	8	small	138
Google	testability	126	7282	4	small	151
Google	toolbox	309	70805	4	small	344
Sourceforge	tora	703	198512	17	large	3523
Google	torrentpier	366	72781	8	small	447
Google	unladenswallow	2815	863242	13	large	1158
Google	v8	982	324042	32	large	4556
Sourceforge	winmerge	862	172366	16	large	7149
Sourceforge	xcat	156	145106	8	small	6257
Sourceforge	zk1	1552	217337	17	large	14151
Google	zscreen	627	98181	10	small	1685
Google	zxing	1119	81501	19	large	1393

consider a relatively low file coverage, e.g., 50%. Looking more in detail the Table, we have $TR70\% < threshold$ in 37 cases out of 37, $TR60\% < threshold$ in 37 cases out of 37 and $TR50\% < threshold$ in 33 cases out of 37. More precisely, considering file coverage equal to 50%, we have high TF (or better non-low TF) only in 2 small projects (`easycooking-fantastici4` and `testability`) and in 2 large projects (`V8` and `mantisbt`). That high proportion of low TF projects along with the high number of TF=0 projects is strange. For this reason we decided to look more in detail the SVN repositories of the selected projects. In this analysis, we observed a set of facts (in the following called anomalies) that could impact on the quality and reliability of the TF measurement (future works will be devoted to understand their real impact on TF).

– **Large commits** [14]: in some projects we found commits that include a large number of files. Such commits are usually devoted to: (i) update the project to a new version, e.g., the first version or a new one produced off-line; and

Table 4. Threshold to be overtaken to have non-low TF and the computed Truck Factor for 50%, 60% and 70% of file coverage

Software	threshold	TF50%	TF60%	TF70%
choscommerce	1.60	1	0	0
closure-compiler	1.60	1	1	1
cppunit	2.40	0	0	0
cspoker	1.60	0	0	0
easycooking-fantastici4	1.60	2	1	1
erlide	2	1	1	0
gdata	3.80	0	0	0
gmapcatcher	2.40	0	0	0
googlemock	1.60	1	1	1
gtk-gnutella	3	1	1	1
h2database	2.40	0	0	0
htmlunit	4	1	1	0
httpunit	1.20	1	1	1
jfreechart	1.20	0	0	0
jpcsp	4	0	0	0
jstestdriver	3.60	2	2	1
jtidy	2	0	0	0
keycar	1.60	0	0	0
mantisbt	7.60	9	7	5
mobileterminal	2.40	0	0	0
moofloor	1.60	0	0	0
nettiers	2.60	0	0	0
pagespeed	2.40	0	0	0
phpwiki	3	1	1	1
remotes	1.20	0	0	0
sqlitepersistentobjects	3.20	0	0	0
testability	1.60	2	1	1
toolbox	1.60	1	1	0
tora	3.40	2	2	2
torrentpier	3.20	1	0	0
unladenswallow	2.60	0	0	0
v8	6.40	7	5	3
winmerge	3.20	3	1	0
xcat	3.20	1	1	0
zk1	3.40	0	0	0
zscreen	4	0	0	0
zxing	3.80	1	0	0

(ii) update the code of third-party software. For instance, the initial version of the `Zxing` project (15% of the whole project files) has been uploaded into the repository by a developer in the revisions r2/6 (from 2 to 6). In the revision r1549 of `Zxing`, 20% of the repository files has been added (a new library has been introduced). Another example is in the revision r1 of `Nettiers` in which all the files of the first project version have been added by one developer.

– **Similar changes in several files:** in some commits a large number of repository files is modified with small and similar changes, often, related to: (i) maintenance operations of the repository structure; (ii) the management of software license and configurations (as highlighted in [15]); and (iii) the management of project documentation and code comments. For instance, in the revisions r1505/1506 of `Zxing` around 15% of the project files have been changed by a developer only to update the license information. Moreover, in the revisions r39/40 of `Zscreen` a developer imported in the repository

a set of files (32% of the whole project files) taken from the old project repository. `Zscreen`, in fact, was initially supported by Sourceforge but then it has been migrated to the Google code repository, thus its original code has been migrated in some revisions.

- **Developers with several accounts:** in some cases some developers are registered into the repository with more than one account using e.g., different email addresses. For instance, in the projects `Zxing`, `Cspoker` and `Jtestdriver` respectively 2, 1 and 2 developers seem to have two accounts each one. For example, in `Zxing` we have *dswit@gmail.com* and *dswit@google.com*.
- **Brief history:** some projects have a relatively brief history, and thus a quite limited number of SVN revisions. For instance, `Jtidy`, `Moofloor`, and `Sqlitepersistentobjects` have less than 200 SVN revisions.
- **Few developers:** in some projects few developers compose the working team. Often, in such cases, the owner of the project plays a relevant role also during the project evolution. For instance, `Cspoker`, `Moofloor`, and `Remotes` have less than 4 developers working, on average, on 20k LOCs per project. Another example is `Jfreechart`: one developer (*Mungady*) out of 3 modified exclusively 98% of the Jfreechart code (he is a Hero [2]).
- **Some developers are not committers:** in some projects only a limited number of developers, out of the whole working team, commit files on the repository. The repository is managed by only few persons, hence, it does not actually trace the activities performed by the project developers. For instance, in `Zscreen` around 80% of the 1685 commits have been performed by only two developers (*flexy123* and *mcored*) out of 10 of the team. We believe that *flexy123* and *mcored* play the role of repository managers (i.e., persons allowed to commit on the repository). Similarly in the project `H2database` only one developer (*thomas.tom.mueller*), out of 6 of the project team, performed around 85% of the commits.

Is the Govindaray's threshold consistent with the Zazworka et al approach? Here we limit ourselves to a visual comparison (see scatter plot in Figure 3 (b)) between the Govindaray threshold (straight blue line) and the empirical TF values ($TR60\%$) obtained in this work (circular points). Precisely, each circular point in the scatter plot corresponds to the TF of a project in our dataset. A regression line of all the 37 TFs (dotted red line) is shown to facilitate/ease the comparison; in some sense, it summarizes the trend of all the points. From the Figure 3 (b) it is apparent that there is a huge difference between the two curves. It could be the consequence of one of the following two facts (also both): (1) the threshold suggested by Govindaray is overestimated with respect to the Zazworka et al. approach or (2) data are not appropriate (they contain too many anomalies, see the results of the previous research question). Given that the presence of "anomalies" could constitute a possible confounding factor, we decided to compute the regression line deleting the projects with TF=0 (i.e., the more suspect values corresponding to the projects containing more

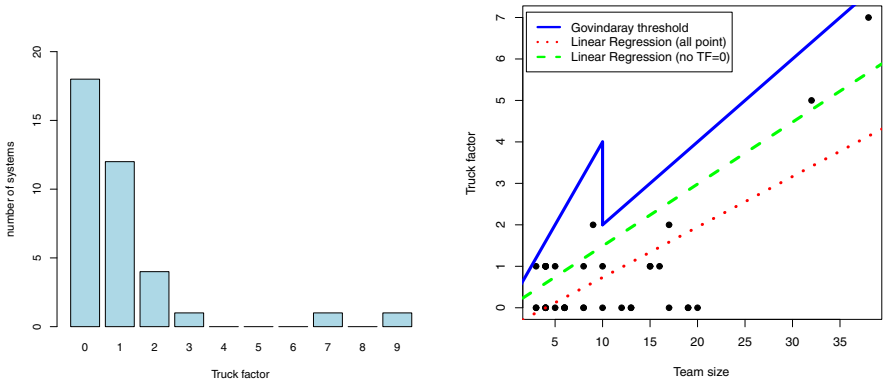


Fig. 3. TF50% distribution (left) and Visual comparison between Govindaray threshold and empirical values (rigth)

"anomalies"). The new regression line is represented with a green dashed line in Figure 3 (b). Also in this case, there is a big difference between the two curves. It lead us to believe that the threshold suggested by Govindaray is overestimated with respect to the Zazworka et al approach.

5.2 Lesson Learnt and Open Issues

Here we summarize the lessons learnt from the execution of the exploratory study and the open issues.

TF measurement. The TF measurement proposed by Zazworka et al. [3] is sensible and practical even if based on a really strong assumption (developers who edited at least once the file have knowledge about it). The measurement does not account for the fact that a developer which has done a lot of commits should have more knowledge of a file than one that has done very little. We believe that more precise measurements keeping account of that should be devised/proposed. In addition, we believe that the proposed measurement should be adequately validated, for example using a framework for evaluating metrics as proposed in [16].

TF algorithm. The conducted experiment revealed that the naive algorithm based on the Zazworka et al. approach is applicable with small and medium-sized projects but has scalability problems with real large projects (in particular when the number of committers is > 30). Even if projects with 30 or more committers are not so common, the algorithm should be improved in future.

Code repositories. The experiment has highlighted a possible threat to validity to the applicability of the approach. The code repositories (in our case SVN repositories), which constitute the input of the approach, should be carefully

analyzed and filtered to avoid unreliable TF results due to "anomalous" commits (e.g., updating the GPL license in every file's header comment [15]) and other problems (e.g., same committer with two or more nicknames/accounts). An open issue is how determining whether a commit is "anomalous" (and then unusable in the TF computation) or not. Clearly, manually inspecting the target repository is not possible: a taxonomy of "anomalous" commits should be proposed and some automatic strategies able to find anomalies should be devised. We think that more empirical investigation is needed in this context.

Govindaray's threshold. The study have experimentally shown, even if further experiments should confirm it, that the threshold suggested by Govindaray and used to decide if the TF is low or not is overestimated with respect to the Zazworka et al. approach. While no solid conclusions can be drawn, it is apparent that more investigation is needed to infer a reliable threshold. A supplementary difficulty could also be that the threshold could depend on the context of the applications (e.g., free vs. proprietary software). A possible method to derive the threshold could be driven by measurement data (and not by expert opinion) from a representative set of systems as proposed in [17].

6 Conclusion

In this paper, we have empirically investigated the well-known notion of TF. First, we have extensively consulted the literature and discovered that there exists only one proposal to compute the TF. Second, we have implemented it in a tool. Third, using our tool, we have conducted an exploratory study with 37 open source projects to answer to three research questions concerning the proposed approach.

The result of this study is that: the algorithm based on the approach proposed by Zazworka et al. is applicable but with some heavy limitations that should be resolved to render the notion of TF really operative and usable. Summarizing, the TF measurement has not been validated and it based on a strong assumption. The naive algorithm based on the Zazworka et al. approach is applicable only with small and medium sized projects but has scalability problems with real large projects. Moreover, the result of the tool strongly depends on the quality of the SVN code repository given in input. Finally, the Govindaray's threshold, useful for the managers to understand that a project is getting in trouble, should be used with great care, given that seems overestimated with respect to the Zazworka et al. approach.

Future works will be devoted to continue the empirical validation and enhancement of the tool presented in this work. We plan to extend the actual dataset and to conduct additional experiments to infer a reliable threshold alternative to the Govindaray's threshold. We are also working on alternative measurements of TF.

References

1. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading (1999)
2. Ricca, F., Marchetto, A.: Heroes in FLOSS projects: an explorative study. In: IEEE International Working Conference on Reverse Engineering, pp. 155–159 (2010)
3. Zazworka, N., Stapel, K., Shull, F., Basili, V., Schneider, K.: Are developers complying with the process: an XP study. In: IEEE Symposium on Empirical Software Engineering and Measurement (2010)
4. Ricca, F., Marchetto, A.: Are Heroes common in FLOSS projects? In: IEEE International Symposium on Empirical Software Engineering and Measurement (2010)
5. Nordberg III, M.: Managing code ownership. IEEE Software, 26–33 (2003)
6. Nagappan, N., Murphy, B., Basili, V.: The influence of organizational structure on software quality: an empirical study. In: IEEE International Conference on Software Engineering (ICSE), pp. 521–530 (2008)
7. Beck, K.: Embracing change with extreme programming. IEEE Computer 32(10), 70–77 (1999)
8. Weyuker, E.J., Ostrand, T.J., Bell, R.M.: Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models. Empirical Software Engineering 13(5), 539–559 (2008)
9. Girba, T., Kuhn, A., Seeberger, M., Ducasse, S.: How developers drive software evolution. In: International Workshop on Principles of Software Evolution (IWPSE), pp. 113–122 (2005)
10. Anvik, J., Hiew, L., Murphy, G.C.: Who should fix this bug? In: International Conference on Software Engineering (ICSE), pp. 361–370 (2006)
11. Fritz, T., Murphy, G.: E.Hill: Does a programmer's activity indicate knowledge of code? In: Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE), pp. 341–350 (2007)
12. Robson, C.: Real world research. Blackwell Publishing, Malden (2003)
13. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering - An Introduction. Kluwer, Dordrecht (2000)
14. Hindle, A., German, D., Holt, R.: What do large commits tell us?: a taxonomical study of large commits. In: International Working Conference on Mining Software Repositories (MSR). IEEE, Los Alamitos (2008)
15. Alali, A., Kagdi, H., Maletic, J.I.: What's a typical commit? a characterization of open source software repositories. In: 16th IEEE International Conference on Program Comprehension, pp. 182–191 (2008)
16. Kaner, C., Bond, W.: Software engineering metrics: What do they measure and how do we know? In: 10th International Software Metrics Symposium (2004)
17. Alves, T., Ypma, C., Visser, J.: Deriving metric thresholds from benchmark data. In: 26th IEEE International Conference on Software Maintenance. IEEE, Los Alamitos (2010)

Author Index