Rory V. O'Connor
Nathan Baddoo
Kari Smolander
Richard Messnarz (Eds.)

Communications in Computer and Information Science 16

Software Process Improvement

15th European Conference, EuroSPI 2008
Dublin, Ireland, September 2008
Proceedings

Springer

Communications
in Computer and Information Science 16

Rory V. O'Connor Nathan Baddoo
Kari Smolander Richard Messnarz (Eds.)

Software Process Improvement

15th European Conference, EuroSPI 2008
Dublin, Ireland, September 3-5, 2008
Proceedings

Springer

Volume Editors

Rory V. O'Connor
School of Computing
Dublin City University
Dublin, Ireland
E-mail: roconnor@computing.dcu.ie

Nathan Baddoo
University of Hertfordshire
Hatfield, Hertfordshire AL10 9AB, UK
E-mail: n.baddoo@herts.ac.uk

Kari Smolander
Lappeenranta University of Technology
Lappeenranta, Finland
E-mail: kari.smolander@lut.fi

Richard Messnarz
ISCN LTD, Bray, Co.
Wicklow, Ireland
E-mail: rmess@iscn.com

Library of Congress Control Number: 2008934046

CR Subject Classification (1998): D.2, D.1, D.3, D.2.9, I.7

ISSN 1865-0929
ISBN-10 3-540-85934-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-85934-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12517060 06/3180 5 4 3 2 1 0

Preface

This textbook is intended for use by SPI (Software Process Improvement) managers and researchers, quality managers, and experienced project and research managers. The papers constitute the research proceedings of the 15th EuroSPI (European Software Process Improvement, www.eurospi.net) conference in Dublin, Ireland, 3–5 September 2008.

Since the first conference, held in Dublin in 1994, EuroSPI conferences have been held in 1995 in Vienna (Austria), in 1997 in Budapest (Hungary), in 1998 in Gothenburg (Sweden), in 1999 in Pori (Finland), in 2000 in Copenhagen (Denmark), in 2001 in Limerick (Ireland), in 2002 in Nuremberg (Germany), in 2003 in Graz (Austria), in 2004 in Trondheim (Norway), in 2005 in Budapest (Hungary), in 2006 in Joensuu (Finland), and in 2007 in Potsdam (Germany).

EuroSPI has established an experience library (library.eurospi.net), which will be continuously extended over the next few years and was made available to all attendees.

EuroSPI has also started an umbrella initiative for establishing a European Qualification Network in which different SPINs and national ventures can join mutually beneficial collaborations (EQN - EU Leonardo da Vinci network project).

With a general assembly on 15.-16.10.2007 through EuroSPI partners and networks, in collaboration with the European Union (supported by the EU Leonardo da Vinci Programme), a European certification association has been created (www.eu-certificates.org) for the IT and services sector to offer SPI knowledge and certificates to industry, establishing close knowledge transfer links between research and industry.

A general assembly of the ECQA (European Certification and Qualification Agency) took place as an associated event of EuroSPI 2008 on September 3, 2008.

The greatest value of EuroSPI lies in its function as a European knowledge and experience exchange mechanism for SPI know-how between research institutions and industry.

Since its beginning in 1994 in Dublin, the EuroSPI initiative has outlined that there is not a single silver bullet to solve SPI issues, but it is necessary to understand a combination of different SPI methods and approaches to achieve concrete benefits. Therefore each proceedings volume covers a variety of different topics, and at the conference we discussed potential synergies and the combined use of such methods and approaches. These proceedings contain selected research papers for six topics each having three research papers:

Section I: Organizational Issues
Section II: Productivity, Effort Estimation and Metrics
Section III: Standards and Reference Models 1
Section IV: Standards and Reference Models 2
Section V: Documentation and Knowledge Management
Section VI: Project Issues.

Section I presents three studies that approach software development and process improvement from an organizational viewpoint. Ma et al. recognize conflict as an unavoidable issue in organizational settings. They present a mediation model that can be used for conflict resolution in requirements engineering. O'Donnell and Richardson study the implementation of agile methods in a small software organization. The conclusion they come to is that many of the problems in implementation relate to the management of the organization. The paper by Valtanen and Sihvonen makes empirical observations about a small company and its SPI efforts. In their analysis they identify factors that have a positive impact on the motivation of SPI efforts. The most important factors they identify are top-down commitment, shared best practices, resources and bottom-up initiatives.

Section II, "Productivity, Effort Estimation and Metrics", combines the results of three studies in this area. The first paper, by Chua et al., builds an empirical model for estimating effort on requirements changes. The paper shows the importance of gathering enough data to develop a cost estimation model. The paper by Pietinen et al. compares two cases and draws conclusions on the productivity of pair programming in a distributed environment. As a result, they suggest that pair programming is good for raising confidence and sharing tacit knowledge. The third paper (Ozkan et al.) introduces diverse uses of functional size measures and investigates how functional size measures can be incorporated into project management practices.

In Section III the focus moves to the relation of improvement efforts to proposed reference models and standards. Hauck et al. introduce process reference guides that can be used for mapping reference models and standards to improvement efforts. To implement such mapping they propose using an organizational WIKI. The paper by Landaeta et al. approaches SPI from the project management perspective. They present a procedure for planning, monitoring and closing an SPI program that uses PMBOK's process areas as a reference. Chen et al. focus on CMMI. They investigate practice dependencies within CMMI process areas and present a set of graphs that show the dependencies. Their model provides more information about the dependencies of CMMI for SPI researchers and practitioners.

Section IV focuses on reference models and standards and places more emphasis on IT services. Magdalena et al. use graph theory to represent the existing dependencies among the ITIL v2 processes. The result helps in determining the implementation priority of the service processes. Barafort et al. continue in the area of IT services. They approach the subject through industrial cases and integrate capability assessment with process modeling. As a conclusion, they present an enhanced framework for process improvement. Finally in this section, Laporte et al. propose a software engineering lifecycle standard for very small enterprises. They survey small enterprises and emphasize the importance of recognizing the contribution of small enterprises. They note that small enterprises require further guidance in order to integrate standards into their practices.

In Section V the focus shifts to managing SPI knowledge. Stapel et al. conclude that a document-centric process is an impossibility in pre-development phases in the automotive sector. Therefore they propose light-weight concepts for these phases and incorporate them in a semantic Wiki. Calvo-Manzano et al. provide an organizational tool, a process asset library that organizes the SPI-related knowledge of an organization. The tool is targeted especially for small organizations and acts as an organizational

repository for process improvement. Montoni et al. present an approach to support the execution of SPI implementation initiatives based on SPI strategies. Their aim is to capture the knowledge related to critical success factors of SPI initiatives. They implement this with a set of tools in a process-centered knowledge management environment.

Finally, Section VI concentrates on project management issues. The first paper of this section, by Majchrowski and Deprez, makes an interesting connection from open source to project management. It presents a process for selecting open source components for a software development project. Hole and Moe combine agile methods and global software development in their action research study of three distributed projects. They wish to find out whether it is possible to combine Scrum with global software development. They conclude that trust is important for getting the benefits of agile development, Scrum requires certain organizational adjustments, and that it is important to solve existing communication problems. Finally, Martins and da Silva present an SPI methodology that aligns processes and projects. They also propose a metric to analyze the alignment.

Recommended Further Reading

In [1] we integrated the proceedings of 3 EuroSPI² conferences into one book which was edited by 30 experts in Europe. In [2] you will find the EuroSPI² research proceedings published by Springer and based on EuroSPI 2005. In [3] you will find the EuroSPI research proceedings published by Springer and based on EuroSPI² 2006. In [4] you will find last year's EuroSPI² research proceedings published by Springer.

References

1. Messnarz, R., Tully, C. (eds.): Better Software Practice for Business Benefit – Principles and Experience, 409 pages. IEEE Computer Society Press, Los Alamitos (1999)
2. Richardson, I., Abrahamsson, P., Messnarz, R. (eds.): Software Process Improvement. LNCS, vol. 3792, p. 213. Springer, Heidelberg (2005)
3. Richardson, I., Runeson, P., Messnarz, R. (eds.): Software Process Improvement. LNCS, vol. 4257, pp. 11–13. Springer, Heidelberg (2006)
4. Abrahamsson, P., Baddoo, N., Margaria, T., Messnarz, R. (eds.): Software Process Improvement. LNCS, vol. 4764, pp. 1–6. Springer, Heidelberg (2007)

July 2008

Rory V. O'Connor
Nathan Baddoo
Kari Smolander
Richard Messnarz

Organization

Board Members

EuroSPI board members represent centres or networks of SPI excellence with considerable experience with SPI. The board members collaborate with different European SPINs (Software Process Improvement Networks).

The following six organizations have been members of the conference board in the last 8 years:

- ASQ, http://www.asq.org
- ASQF, http://www.asqf.de
- DELTA, http://www.delta.dk
- ISCN, http://www.iscn.com
- SINTEF, http://www.sintef.no
- STTF, http://www.sttf.fi

EuroSPI Scientific Program Committee

EuroSPI has established an international committee of selected well-known experts in SPI who are willing to be mentioned in the program and to review a set of papers each year. The list below represents the research program committee members. EuroSPI² also has a separate industrial program committee responsible for the industry/experience contributions.

- Ambriola, Vincenzo, Università di Pisa, Italy
- Aurum, Aybke, University of New South Wales, Australia
- Baddoo, Nathan, School of Computer Science at the University of Hertfordshire, UK
- Biffl, Stefan, Technische Universität Wien, Austria
- Biro, Miklos, Corvinus University of Budapest, Hungary
- Calvo-Manzano Villalón, Jose A., Universidad Politécnica de Madrid, Spain
- Ciolkowski, Marcus, Fraunhofer IESE, Germany
- Coughlan, Ray, Cork Institute of Technology, Ireland
- Dalcher, Darren, Middlesex University, UK
- Daughtrey, Taz H., James Madison University, USA
- Desouza, Kevin C., University of Washington, USA
- Dingsoyr, Torgeir, SINTEF ICT, Norway
- Duncan, Howard, Dublin City University, Ireland
- Dyba, Tore, SINTEF ICT, Norway
- Gabor, Andras, Corvinno Technology Transfer Center Ltd, Hungary
- García Guzmán, Javier, Universidad Carlos III de Madrid, Spain

- Gorschek, Tony, Blekinge Institute of Technology, Sweden
- Gresse von Wangenheim, Christiane, Universidade do Vale do Itaja - UNIVALI, Brazil
- Kreiner, Christian, Graz University of Technology, Austria
- Landes, Dieter, Fachhochschule Coburg, Germany
- Mäkinen, Timo, Tampere University of Technology, Finland
- Mac an Airchinnigh, Michael, Trinity College Dublin, Ireland
- Mc Caffery, Fergal, University of Limerick, Ireland
- McQuaid, Patricia, Orfalea College of Business, USA
- Müller, Matthias, EnBW Systeme Infrastruktur Support GmbH, Germany
- Münch Jürgen, Fraunhofer IESE, Germany
- O'Connor, Rory, Dublin City University, Ireland
- Oivo, Markku, University of Oulu, Finland
- Ovaska, Päivi, South Karelia University of Applied Sciences, Finland
- Peisl, Thomas, University of Applied Sciences Munich, Germany
- Pries-Heje, Jan, Roskilde University, Denmark
- Rejas, Ricardo, Universidad Francisco de Vitoria, Spain
- Richardson, Ita, Universtiy of Limerick, Ireland
- Ruhe, Günther, University of Calgary, Canada
- Siakas, Kerstin, Technological Educational Institute of Thessaloniki, Greece
- Sillitti, Alberto, Free University of Bolzano-Bozen, Italy
- Smolander, Kari, Lappeenranta University of Technology, Finland
- Stålhane, Tor, Norwegian University of Science and Technology, Norway
- Tiron Tudor, Adriana, UBB-University Babes Bolyai, Romania
- Vajde Horvat, Romana, University of Maribor, Slovenia
- Varkoi, Timo, Tampere University of Technology, Finland
- Winkler, Dietmar, TU Vienna, Austria

All four editors have quite a complementary and interesting profile. Dr. Messnarz works in close collaboration with Austrian research institutions (universities of applied sciences) and large German automotive companies. Dr. Nathan Baddoo is a professor at the University of Hertfordshire, UK, and he has published scientific articles about human factors in SPI and has performed studies at major European organizations, applying motivation techniques in SPI. Dr. Rory O'Connor is a senior lecturer at Dublin City University and a senior researcher with Lero, the Irish Software Engineering Centre. His main research interests centre on software processes and SPI in relation to small and very small organizations. And finally, Dr. Kari Smolander has studied software development organizations extensively, and he is a professor of software engineering at Lappeenranta University of Technology. The experience portfolio of the chairs covers different market segments, different sizes of organizations, and different SPI approaches. This strengthens the fundamental principal of EuroSPI² to cover a variety of different markets, experiences, and approaches.

Table of Contents

Organizational Issues

Productivity, Effort Estimation and Metrics

Standards and Reference Models

Documentation and Knowledge Management

Project Issues

Building a Narrative Based Requirements Engineering Mediation Model

Nan Ma[1], Tracy Hall[2], and Trevor Barker[1]

[1] School of Computer Science, University of Hertfordshire,
College Lane, Hatfield A10 9AB, UK
{N.ma,T.1.barker}@herts.ac.uk
[2] Department of Information Systems & Computing, Brunel University,
Uxbridge, Middlesex UB8 3P
Tracy.Hall@herts.ac.uk

Abstract. This paper presents a narrative-based Requirements Engineering (RE) mediation model to help RE practitioners to effectively identify, define, and resolve conflicts of interest, goals, and requirements. Within the SPI community, there is a common belief that social, human, and organizational issues significantly impact on the effectiveness of software process improvement in general and the requirements engineering process in particularl. Conflicts among different stakeholders are an important human and social issue that need more research attention in the SPI and RE community. By drawing on the conflict resolution literature and IS literature, we argue that conflict resolution in RE is a mediated process, in which a requirements engineer can act as a mediator among different stakeholders. To address socio-psychological aspects of conflict in RE and SPI, Winslade and Monk (2000)'s narrative mediation model is introduced, justified, and translated into the context of RE.

Keywords: Conflict, Method Tailoring, Narrative Mediation, Conflict Resolution, Requirements Negotiation.

1 Introduction

In this paper we present a narrative-based Requirements Engineering Mediation Model (NREMM). Conflict is a common phenomenon in everyday life [1]. It also has been recognized as an inevitable part of the RE process, as RE is both a social and technical process involving extensive interactions among different stakeholders (e.g. customers, users, developers and testers) from different backgrounds and with different individual and organizational goals [2]. However, in the current RE literature, conflict is consistently considered as a technical issue that may lead to inconsistency in the requirements specification (e.g. [3] [4] [5] [6] [7]). Much work in this area focuses on presenting technical methods or techniques for modelling, analyzing, and managing conflict or inconsistency e.g. KAOS [5], Problem Frames [6] and I* [7] or tools for automating conflict identification and resolution e.g. Oz [8], Synoptic [3], or prompting groupware systems for remote negotiation e.g. Win-Win [9]. Little attention is given to the socio-psychological aspect of the conflict. Furthermore, the term "

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 1–12, 2008.

requirements negotiation" is prevalent in the RE literature where the resolution of conflict in RE is considered as a purely negotiation-based process (e.g. [3] [4] [8] [10] [11]) in which a requirements engineer acts as a representative of a developer site and negotiates with users.

This paper adopts a complementary viewpoint and differentiates itself from previous work by recognizing conflict as a social, human, and organizational issue. We adopt Barki and Hartwick's definition of conflict as "a *phenomenon that occurs between interdependent parties as they experience negative emotional reactions to perceived disagreements and interference with the attainment of their goals.* [12]" Furthermore, we also view the process of resolving conflict in RE is a mediated process, in which a requirements engineer acts as a mediator among different stakeholders.

It is often possible to borrow relevant theories from other disciplines to improve RE practice. Resolving the human aspects of conflict and reaching an agreement in RE can thus be sought by applying relevant approaches that have proved successful in the mediation and conflict resolution discipline. In doing this we borrow the original narrative mediation theory from Winslade and Monk [13] and translate it into the context of RE. This paper aims to describe the rational of why we have built such a model, the methodological approach of how we built it, and finally what a narrative-based RE mediation model is.

This paper is organised as follows: Section 2 gives a review of the relevant literature to justify the rational of building the NREMM model. Section 3 provides an overview of the original narrative mediation model, and justifies its applicability to the context of RE. Section 4 presents our methodological approach of translating the original narrative mediation model into the context of RE, and also presents our NREMM model. Finally, section 5 concludes the paper with some plans for future research.

2 Conflict Resolution in RE

In this section, we argue that conflict resolution in RE is a mediation process rather than a negotiation process, in which a requirements engineer acts as a mediator to assist different stakeholders from different backgrounds with different individual and organizational goals to resolve conflicts. The fundamental difference between negotiation and mediation is that, negotiations often only involve conflicting parties themselves reaching an agreement. Mediations then involve a third party as a mediator to lead the process and help parties to reach an agreement.

Most of the RE literature argues that the process of resolving conflict is a purely negotiation-based process, in which a requirements engineer acts as a representative of a development site to "negotiate" with a users' site to make trade-offs (e.g. [3] [4] [8] [10] [11]). However, evidence from the IS discipline suggests that conflicting interests and goals are not only between the users' site and the developers' site, but are often between different user groups. For example, Robertson et al. describe a case where the decision to develop a new production management system was predominantly led by manufacturing and production department specialists who decide to invest heavily in a new manufacturing resources planning system (MRP2) [14]. However, in this case, stakeholders from other functional departments (e.g. purchasing and marketing) had different ideas about the problems they were facing and did not believe the new MRP2 to be the solution. Eventually the new system failed due to poor

management of such conflicting interests and goals between two users groups [14]. This negotiated form of conflict resolution is seriously questioned in the above situation. It is apparent in the above situation that a requirement engineer needs to play a mediator's role to facilitate the two users groups to reach an agreement on requirements. Our field study of 10 RE practitioners also indicates that RE workshops are the most widely used method of requirements elicitation, and he/she is often required to play the role of a mediator in a RE workshop [15].

The facilitative role of a requirements engineer has been documented in the RE literature. However, there are many diverse views on the facilitators' role in the RE literature ([11] [16]). The role of a requirements engineer as a mediator has not been explicitly identified in the previous RE literature. Few techniques, models, and guidelines have been developed to guide a requirements engineer to resolve conflicting viewpoints in RE practice. In the next section, I will provide a brief overview of the original narrative mediation approach and particularly focus on justifying its applicability and importance to RE.

3 A Brief Overview of Narrative Mediation

The narrative perspective is that people tend to organize their experiences in story form. In narrative mediation, the process of mediation is thus viewed as a story-telling process [13]. It has been recognized as an innovative conflict resolution paradigm that encourages conflicting parties to reach understanding and resolution through a deep understanding of the shared personal and cultural narratives underlying the conflict. In this section, we provide an overview of the original narrative mediation model, and justify it's applicability to the context of RE.

The narrative approach involves a simple and yet profound departure from commonly held assumptions about the conflicts that embroil people [13]. Its underlying assumption is that people live their lives according to stories rather than according to inner drives or interest. It thus privileges stories and the meanings within stories over facts and causes. In the story, people seek to establish coherence and produce lives, careers, relationship, and communalities [13]. Therefore, when they work with others to overcome the divisiveness of a conflict, they will find it "more productive to work with the stories in which the conflict is embedded than to pursue objective reality" [13]. The original narrative mediation model contains three sub-models [13]:

- **Engagement.** In this phase, the mediator focuses on establishing a relationship and identifying the problems with the conflicting parties. To achieve a workable relational context, the mediator needs to attend to the physical setting in which the mediation is to take place, to the non-verbal behaviour displayed by all parties, and to the relational moves made by the mediators and the parties. In the case of resolving conflicts in RE, we can refer this phase as conflict identification phases.
- **Deconstructing the conflict-saturated story.** This phase of the process involves the mediator developing a supportive relationship and listening respectfully to their own stories. The mediator works actively to separate the parties from their conflict-saturated story. The mediator seeks to undermine the certainties on which the conflict feeds and invites the participants to view

the plot of the dispute from a different viewpoint. In the case of resolving conflicts in RE, we can refer this phase as conflict definition phase.

- **Constructing the alternative story.** In this phase, the mediator is occupied with crafting alternative, more preferred story lines with people who were previously captured by a conflict-saturated relationship. This phase thus may lead to a resolution that takes the form of an agreement between parties. In the case of resolving conflicts in RE, we can refer this phase as conflict solution phase.

3.1 Narrative Mediation's Applicability to RE

We justify the applicability of the original narrative mediation to RE based on the following four aspects:

A process-oriented perspective
Narrative mediation model adopts a process-oriented perspective. As Winslade and Monk [13] state:

> *"We have deliberately called this approach a process because we think the word process focuses on the dynamic, shifting, and changing elements of mediation rather than on abstraction, facts, or structures. By concentrating on process, the mediator is invited to think about and work with the responses of the conflicting parties rather than follow some static, preconceived plans."*

This process-oriented perspective matches particularly well with the process aspect of RE practice. RE process is a set of activities that should be systematically followed to derive, validate, and maintain a systems requirements document [2]. The RE literature has presented many different process models, which can range from linear or iterative in structure (e.g. [2] [16]).

Although theses models are explicitly defined in the RE literature, the empirical studies have indicated that the systematic and incremental RE models presented in the RE literature do not really reflect the reality of RE process in real practice. For example, Hofmann et al., indicate that most companies regard RE as an ad hoc process, with only some using an explicitly defined RE process model or customising a company standard model [17]. Nguyen and Sawtmann also indicate that RE processes do not appear in a systematic, smooth and incremental way, but are "opportunistic, with sporadic simplification and restructuring of the requirements models when points of high complexity are reached" [18].

One reason for this chaotic and dynamic RE process is due to requirements changes [19]. It is apparent that the business environment in which software is deployed continually changes. Even if the environment is constant, people's perceptions and understandings are dynamic [20]. As a result, the process of resolving conflicts in RE is a dynamic and complex process. It does not involve discrete stages, and does not follow a tidy sequence of events. Rather, the process moves back and forth in a seemingly dynamic manner when necessary. In this sense, the narrative mediation model which focuses on the dynamic, shifting, and changing elements of mediation seems particularly applicable for the context of RE.

A storytelling process

Narrative mediation particularly builds on this storytelling metaphor, and provides a mediator with a way of incorporating stories into the resolution of conflict. In narrative mediation, narratives are interactively developed, modified, and contested as parties elaborate portions of their own and each other's conflict stories [13]. This approach thus assumes that conflicts are rooted in conflict-saturated stories that parties have developed through the course of their relationship. As Winslade and Monk state "conflict is likely because people do not have direct access to the truth or the facts about any situation. [13]"

In RE, the way of gathering user requirements fundamentally can be viewed as a storytelling process. New software development methodologies are increasing exploiting to storytelling aspect of RE process (e.g. user stories in XP practice) [21]. Viewing requirements elicitation as a storytelling process not only emphasizes the final outcome – "user stories", but also highlights the importance of verbal communication and interactions between users and developers, which can potentially minimize the ambiguity of requirements specification [22]. In this sense, the original narrative mediation model which builds on the storytelling metaphor seems well-matched with the fundamental nature of RE elicitation process.

Outsider-in perspective

The context in which RE takes place is a complex "human activity system"; eliciting and analysing requirements thus can not be performed in isolation from the organizational and social context in which any new system will have to operate [19]. This view stresses a good understanding of the social, political and cultural changes caused by new systems. Moreover, as shown in the Curtis et al.'s classic field study of software engineering process, conflicts result from a wide range of interrelated factors, from change in the organisational setting and business context, to the fact that software will be used by different people with different goals and different backgrounds [23].

In narrative mediation, Winslade and Monk argue an "outsider-in" perspective, which looks at conflict as produced in the socio-culture context, where meanings are contested within the social fabric of community [13]. The narrative mediation approach is based on the idea that people construct conflict from their narrative description of events, and concentrates on developing a relationship that is incompatible with conflict and that is built on stories of understanding, respect, and collaboration. The narrative mediation approach recognizes that the mediation context is filled with strong cultural, social, and organizational narratives that form around ethnicity, gender, class, education, financial background, organizational structure and strategies. The narrative mediation approach with an "outsider-in" perspective, which helps mediators and their conflicting parties make sense of the complex social contexts that produce conflicts is thus applicable for the social and organizational aspects of RE.

4 NREMM

In this section, we present our NREMM model. The first part of this section explains our models translation approach. Although many existing RE studies present their novel methods or models by borrowing and translating theories from the other disciplines, there is very little in the RE literature that directly and explicitly explains their

methodological approach of how their model is systematically and rigorously borrowed and translated. We believe that providing such a methodological approach will benefit further researchers who also seek to translate relevant theories from other disciplines to improve RE practice. However, here we only briefly present our methodological approach and NREMM model. For the detail, please refer to [15].

4.1 Model Translation Method

To ensure a rigorous and systematic model translation process, I follow three translation activities (See figure-1):

1. **Activity-1:** In the first activity, each element of the original narrative mediation model (defined as Model version-V0) is mapped onto the context of RE according to its relevance to the RE literature. This means that all irrelevant elements will be removed from the original model. The outcome of this activity is model version V1, which will retain the structure of the original model but only contain elements relevant to RE. To give a reasonable and subjective assessment of each element's relevance of RE, a scoring scheme was developed and used. A Cohen's Kappa measure of inter-rater reliability has been carried out, and indicates an acceptable level of agreements (0.68) between two individual raters.
2. **Activity-2:** A RE specialised mediation model essentially requires the integration of contemporary specialised RE techniques. In the second activity, model version V1 thus will be improved by adding specific RE techniques. The outcome of this activity will be defined as model version V2, which contains specific RE techniques from the RE literature.
3. **Activity-3:** The original mediation model itself contains a certain degree of overlap. Activity 3 will re-structure the model version V2.

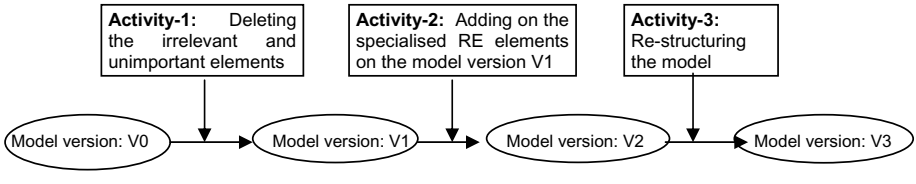


Fig. 1. Three activities of model translation

4.2 NREMM

As mentioned in section 3, the original model contains three sub-models, which are also translated into the context of RE: conflict identification (See figure-2), conflict definition (see figure-3), and conflict resolution (see fugirue-4).

4.2.1 Sub-Model-A: Conflict Identification

The aim of this phase is to establish a workable relationship with the conflicting parties and initially identify conflict between them. The major activities in this phase include selecting meeting settings, relationship practice, dialogical practice, and stakeholders

modelling. The new model below retains majority elements of relational practice, and dialogical practice from the original model, and is complemented by the feature of stakeholder modelling and preparing an RE meeting setting.

Selecting RE meeting setting

Mediation is a meeting based activity. It is important to ensure a RE meeting take place in the right place, with the group of right stakeholders, and with the facilitation of right artefacts. Therefore, selecting meeting setting in RE focuses on the meeting layout and the use of artefacts. In this research, good practice guidelines (e.g. [16]; [24]) from the existing RE literature are integrated with the original model.

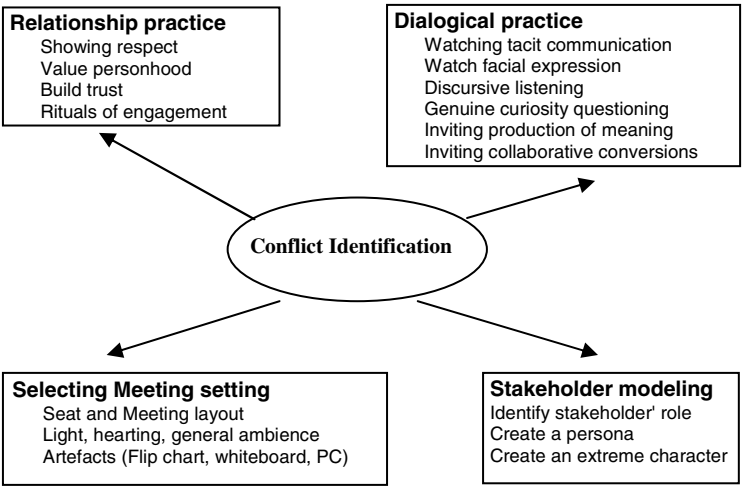


Fig. 2. A model of conflict identification

Stakeholder modelling

Identifying and involving the right stakeholders is of paramount importance in RE. In particular, stories in RE are interactively written through the collaborations between different stakeholders. Consequently, it is essential to identify the right stakeholder's role and personas prior to listening to his/her conflict story. The disciplines of user-centred design and interaction design provide the theories and techniques for identifying and modelling stakeholders as an initial step towards a successful RE mediation meeting. In this research, we will follow Constantinue and Lockwood's recommended practice to identify and model a useful set of stakeholder roles [25].

Dialogical practice

Dialogical practice provides a set of questioning and listening technique to develop a dialogue between parties. The key part of dialogical practice in this sub-model is about inviting and listening to the telling of their conflict stories. Narrative mediation requires the mediator should be more interested in learning the story from which the person is operating, not just with the story the parties are telling. The mediator should learn and listen to people as experts on their own lives. Winslade and Monk [13:140] introduce discursive listening techniques and defined it as:

> *"Careful listening involves hearing not just what has happened but also what necessary constructs are at work in this particular account to make sense of what has happened. This is what we call discursive listening, or listening to the discourses at work in a particular account and to the position calls that are issued within each discourse."*

The discursive listening aims to hear the stories as a version or construction of events rather than a set of facts. It does not merely listen for a definable problem, which is some facts that form the basis of the conflict, or the underlying interests of the parties that are being expressed in the conflict. Most importantly, discursive listening involves learning and listening for the intersection of narrative in a discursive context.

Relationship practice

Mediation is a cooperative practice in which the parties to the conflict are viewed as partners in mediation. Thus, at the very beginning, narrative mediation is very much about creating a relational climate. To achieve this relational climate, the original narrative mediation model recommends that a mediator should "show respect to the parties involved, value their personhood, and invite collaborative conversation" Winslade and Monk [13:120]. In the case of RE, it is apparent all these good practices should also be followed by a requirements engineer.

4.2.2 Sub-Model-B: Conflict Definition (Figure-3)

The aim of this sub-model of mediation is to gain an accurate understanding of conflict. The original narrative mediation model refer to this phase as "deconstructive" in that it gently seeks to undermine the certainties on which the conflict feeds. The sub-model-B thus retains the two elements from the original narrative mediation model: dialogical practice and relationship practice. In addition, the sub-model-B is complemented by adding a new activity: writing a good story.

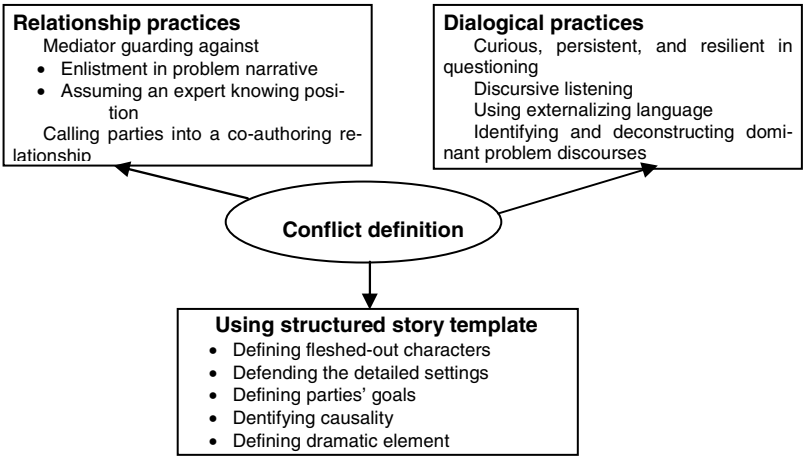


Fig. 3. Conflict definition

Dialogical practice
In this phase, the mediator needs to ask questions that will open up space for reconsideration of the conflict story and eventually separate the people from the conflict. Developing an externalizing conversation and questioning curiously play important roles to achieve this. Careful inquiry into the meanings of the elements of the stories that the parties tell seeks to avoid taking any particular meaning for granted. Curious inquiry sometimes needs to be pursued persistently for its best effect. For example, if a developer team speaks about misunderstanding a user's interpretation on software requirements as a result of what has happened in a conflicting situation, it might be productive to inquire about the word "misunderstanding" and what it means rather than assume we know what is being referred to. Using this type of questioning technique can break up our sense of certainty that we know all that can be known about what we mean, or even more dangerously, that we know what someone else means[13].

Continuing with above example, we now look how externalizing conversation might be used in the conflicts situation between a user group and a developing team. The mediator might look for some description of the conflict that includes both parties' perspectives. Such a description might need to include notions like betrayal or interference. It might even be called simply the argument. In this case, such a description can be viewed as misunderstanding between users and developers. Then the mediator might speak about the misunderstanding as the cause of two parties' problem, rather than speaking about two parties as the cause of the argument. Such linguistic play, done skillfully, might lead to a new perspective on the conflict, and eventually shifts focuses away from personalities, or blame, and focuses attention on the problematic features of the conflict itself.

Relationship practice
In this phase of mediation, the relationship established with the parties in the previous preparation phase needs to be continued. In fact, the mediation can proceed only if the mediator is able to continue to demonstrate respect and compassion to the parties. The mediator thus should be "encouraging, affirming trust, having courage to engage with the fullness of the story, and showing impact of conflict story on mediator" [13:80].

Writing a good story
The original model aims to undermine the fundamental causes of conflicts by adopting unique linguistic techniques such as discursive listening, curiosity questioning, and externalization conversation. The original model strongly emphasises the importance of verbal communication, but overlooks the importance of writing a good story document. The elements added on this activity are adopted from the fields of social science in which the concept and theory of narrative first emerged [26]. Those works recommend the basic practice on writing a good story such as using structured story template, defining fleshed-out characters, defending the detailed settings, defining parties' goals, identifying causality, and defining dramatic element [26].

4.2.3 Sub-Model-C: Conflict Solution (See Figure-4)

Once the relational issues are addressed in a positive way and the conflict itself is clearly defined, traditional problem-solving based mediation approach can become effectively in this phase. In this sense, a mediator then can begin to invent solutions. The original model asks the mediator to invite parties to identify with their preferred alternative to the conflicting relationship. In the context of RE, this can be understood

as the requirements engineer inviting the conflicting stakeholders to propose their preferred solutions as the alternatives for the conflicting situation. As a result, this phase will lead to a solution that takes the form of an agreement. The sub-model-C (see figure-4) retains two activities from the original model: dialogical practice and relationship practice. In addition, to help parties reach a fairly objective decision, a semi-quantitative RE prioritization technique is integrated with the original model.

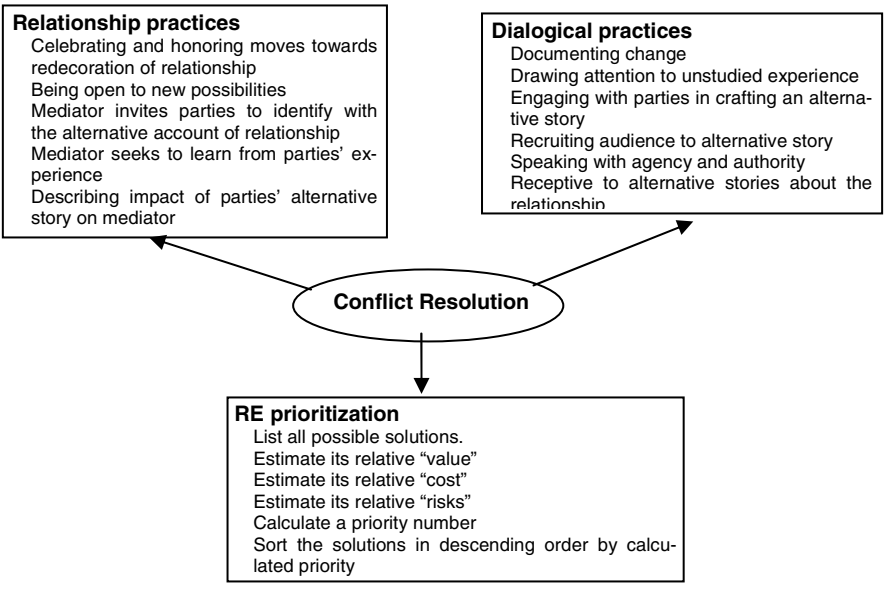


Fig. 4. Conflict resolution

Relationship practice and dialogical practice

Although relationship practice and dialogical practice is consistently recognised as two most important parts in the previous two phases of narrative mediation, in this phase of narrative mediation they may not play a most important role comparing with the newly added activity: RE prioritization. This is because that the primary focus of the previous two phases is on identifying and defining conflict. It is inevitable to involve a great deal of verbal communications and relationship practice. However, this phase of narrative mediation focuses on inventing resolution to conflict. It is a problem-solving process, which focuses more on brainstorming, selecting, and evaluating possible solutions. This does not imply that the relational and dialogical practice will be removed from this phase. Instead, all good practices recommended by the original model will be continually retained, but, are considered as less important than RE prioritization.

RE prioritization

RE prioritization is widely used to determine the relative necessary of the requirements [27]. Whereas all requirements are mandatory, some are more critical than others. Davis [28] points outs that it particularly aims to resolve conflicts when

customer expectations are high, timelines are short, and resources are limited. Indeed, conflicts more likely emerge from those situations. As people naturally have their own interests at heart and they aren't always willing to compromise their needs for someone else's benefit. In the context of conflict resolution in RE, RE prioritization can be used to help a mediator to evaluate their preferred solutions and eventually make a win-win decision. In this paper, we will use a semi-quantitative spreadsheet technique based on prioritization of solutions' *Value*, *Cost*, and *Risk*, which is developed by Weigers [29] and described in the figure-4.

5 Conclusion and Future Work

This paper presents a RE specialised narrative mediation model. We examined the importance of conflict resolution in RE and argued that the fundamental nature of conflict resolution in RE is a mediation process. Winslade and Monk (2000)'s narrative mediation model is described, justified, and translated into the context of RE. In the future, the newly developed model is about to be tested in the real-world contexts.

References

1. Pruitt, D.G., Kim, S.H.: Social conflict: Escalation, stalemate, and settlement, 3rd edn. McGraw-Hill, New York (2004)
2. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. Wiley, Chichester (1997)
3. Easterbrook, S.M.: Resolving Requirements Conflicts with Computer-Supported Negotiation. In: Jirotka, M., Goguen, J. (eds.) Requirements Engineering: Social and Technical Issues, pp. 41–65. Academic Press, London (1996)
4. Nuseibeh, B.: To Be And Not To Be: On Managing Inconsistency in Software Development. In: Proceedings of 8th International Workshop on Software Specification and Design (IWSSD-8), Scloss Velen, Germany, 22-23 March 1996, pp. 164–169. IEEE CS Press, Los Alamitos (1996)
5. Van Lamsweerde, A.: Requirements Engineering in the Year 2000: A Research perspective. In: Invited Paper for ICSE 2000 - 22nd International Conference on Software Engineering, Limerick, June 2000. ACM Press, New York (2000)
6. Jackson, M.: Problem Frames: Analysing and Structuring Software Development Problems. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (2001)
7. Yu, E., Mylopoulos, J.: Why Goal-Oriented Requirements Engineering. In: Dubois, E., Opdahl, A., Pohl, K. (eds.) Fourth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 1998), Pisa, Italy (1998)
8. Robinson, W.N.: Negotiation Behaviour During Multiple Agent Specification: A Need for Automated Conflict Resolution. In: Proceedings of 12th International Conference on Software Engineering (ICSE-12), Nice, France, March 1990, pp. 268–276. IEEE Computer Society Press, Los Alamitos (1990)
9. Boehm, B., Grünbacher, P., Briggs, R.: Developing Groupware for Requirements Negotiation: Lessons Learned. IEEE Software 18(3) (2001)
10. Damian, D.E., Shaw, M.L.G., Gaines, B.R., Zowghi, D.: A multi-disciplinary approach to the study of distributed requirements negotiations. In: Proc. of the 5th Australian Workshop on Requirements Engineering, Brisbane, Australia, December 8-9, pp. 91–100 (2000)

11. Damian, D.E.: A research methodology in the study of requirements negotiations in geographically distributed software system. In: Proceedings of 11 the IEEE International Requirements Engineering Conference (2003)
12. Barki, H., Hartwick, J.: Interpersonal conflict and its management in information system development. MIS Quarterly 25(2), 195–228 (2001)
13. Winslade, J., Monk, G.: Narrative Mediation: A New Approach to Conflict Resolution Jossey-Bass (2000)
14. Robertson, M., Swan, J., Newell, S.: The role of networks in the diffusion of technological innovation. Journal of manage studies 33(3), 335--26 (1996)
15. Ma, N., Hall, T., Barker, T.: Using an expert panel to validate a Requirements Engineering Mediation Model. In: EASE 2008 conference, Italy, 26 (June 2008) (submitted, 2008)
16. Macaulay, L.: Requirements engineering, London, UK. Springer, Heidelberg (1996)
17. Hoffmann, O., Cropley, D., Cropley, A., Nguyen, L., Swatman, P.: Creativity, Requirements, and Perspectives. Australian Journal of Information Systems 13(1), 159–175 (2005)
18. Nguyen, L., Swatman, P.: Promoting and Supporting Requirements Engineering Creativity. In: Dutoit, A.H., McCall, R., Mistrik, I., Paech, B. (eds.) Rationale Management in Software Engineering, ch. 10. Springer, Heidelberg (2006)
19. Nuseibeh, B., Easterbrook, S.: Requirements Engineering: A Roadmap. In: Proceedings of International Conference on Software Engineering (ICSE-2000), Limerick, Ireland, 4-11 June (2000)
20. Ian, S.: Integrated Requirements Engineering: A Tutorial. IEEE Software 22(1): 16-23, 2005
21. Beck, K.: Extreme Programming Explained: Embrace Change. Addison Wesley, Reading (2000)
22. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley, Boston (2004)
23. Curtis, B., Krasner, H., Iscoe, N.: A field study of the software design process for large systems. Communications of the ACM 31(11), 1268–1287 (1988)
24. Maiden, N., Bright, B.P.: Recurrent communication patterns in requirements engineering meetings. In: WETICE 1996, pp. 208–213 (1996)
25. Constantine, L.L., Lockwood, L.A.D.: Software for Use: A Practical Guide to the Essential Models and Methods of Usage-Centered Design. Addison-Wesley, Reading (1999)
26. Burroway, J.: Writing Fiction: A Guide to Narrative Craft. Addison-Wesley, Reading (1999)
27. Fellows, L., Hooks, I.: A Case for Priority Classifying Requirements. In: Eighth Annual International Symposium on Systems Engineering, Seattle, Washington, International Council on Systems Engineering (1998)
28. Davis, A.: The Art of Requirements Triage. Computer 36(3), 42–49 (2003)
29. Wiegers Karl, E.: First Things First: Prioritizing Requirements. Software Development 7(9) (1999)

Problems Encountered When Implementing Agile Methods in a Very Small Company

Michael J. O'Donnell[1] and Ita Richardson[2]

[1] Department of Computer Science and Information Systems,
University of Limerick, Limerick, Ireland
[2] Lero – the Irish Software Engineering Research Centre,
University of Limerick, Limerick, Ireland
michael.odonnell@coolnadeed.com, ita.richardson@lero.ie

Abstract. This paper presents a case study carried out in a very small company in Ireland, Sporting Software Ltd. The authors had access to Sporting Software Ltd. while the company was implementing eXtreme Programming when developing a new product. We discuss how the implementation was carried out and what went wrong; with the company itself ultimately declaring that the implementation was a failure. We present the problems and benefits experienced, and why these indicate that the implementation of agile methods within a very small company located remotely are not easily achievable While some of the outcomes are not surprising, for example, management did not wholly support the implementation of agile methods, others, such as pre-existing contractual employment arrangements, should be noted.

Keywords: Agile methods, eXtreme Programming, Very Small Software Development Company.

1 Introduction

While many development techniques, methods and processes have been successful in improving the quality and cost of software products, there is still a necessity for software development to become more effective and efficient. Agile is the latest technique introduced by the industry with the aim of achieving that increase in effectiveness and efficiency. According to Mikael Lindvall et al. "the use of, interest in and controversies surrounding agile methods have all increased dramatically" [1] and that this increase has been attributed to small organizations finding existing methodologies "too cumbersome, bureaucratic, and inflexible".

Given the positive reports (for example [2] , [3], [4]), regarding the implementation of agile methods, and having observed and experienced low developer morale and project failure while employing traditional based methodologies, the authors were interested in studying the implementation of agile methods in a very small company. To understand this, we investigated the problems and discuss how these could be avoided by a small company. Therefore our objective was to study an Irish SME as they implemented a project using eXtreme Programming (XP) and to examine the

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 13–24, 2008.

problems which they encountered, the solutions that they derived, and the overall result of the implementation. Our study, presented in this paper, demonstrated that the implementation of eXtreme Programming (XP), an agile method, within such a company was unsuccessful.

2 Why Agile Methods?

The primary goal of software development has changed from "conforming to plan" to "satisfying customers - at the time of delivery, not at project initiation" [5]. Therefore, the software industry must either adapt its existing development methodologies, or formulate new methodologies which deal with rapidly changing requirements. It is these changing requirements that are blamed for software companies' inability to deliver software within budget, on time, and to acceptable quality. Furthermore, these rapidly changing requirements are forcing developers to cut quality in order to incorporate the changing needs of the client. This subsequently has a significant impact on the developer's ability to respond rapidly and cost effectively, to changing requirements in future releases [6].

Traditional methods, such as the waterfall model, V-model and Rational Unified Process (RUP), assume that it is possible, early in the project, to identify all user requirements. This in turn, should reduce the probability of requirements uncertainty. However, the fact is that requirements change. In this situation, using traditional methods hinders the company's ability to meet customer needs. It has been argued that agile methods are optimised for dealing with changes and responding to customer requests in a cost effective and timely manner. According to Highsmith and Cockburn [7], unlike traditional approaches, agile development stresses quality in design. [Agile] methods appeal "to our sense of priorities and provide a better way of articulating ideas previously classified as "doing-the-right-thing", "being on the right side of the 80:20 rule", or "common-sense" [8]. Research carried out by Shine Technologies [2] with select companies in regards to employing agile methodologies found that:

- 93% experienced a productivity increase;
- 88% produced software with better or significantly better quality;
- 83% encountered an increase in business satisfaction.

On the other hand, research by Lindvall et al. [1] indicated that many companies – in particular large companies – are approaching agile methods with a high degree of scepticism due to the conflicting data regarding "in what environments and under what conditions agile methods work" but that "the use of, interest in, and controversies surrounding agile methods have all increased dramatically" [1]. Agile methods have been derided as nothing more that cowboy coding and undisciplined hacking [9] or as a "fad development methodology" [10]. Arguments have been made by Rakitin [11] and Paulk [12] about the software industry's inability to realistically compare projects developed using agile and traditional based methods. However, according to Lindvall et al. [1], the key to making agile methods work appears not to be the blind adoption of all the techniques and practices, rather it is the intelligent integration of the practices most suited to the environment and the project.

With Agile methods, working software is valued over documentation, as it allows all the parties to determine the current stage of development, as opposed to where it may be in theory. Secondly, it allows developers and project managers to more accurately determine the velocity of the overall project, thus allowing more effective estimation of the project's completion date. Individuals and interactions are afforded greater emphasis as it "facilitates sharing information and changing the process quickly when it needs changing" [7]. By collaborating with the client, developers can minimise the possibility of producing software that does not meet the client's actual needs. This increases the probability of repeat business in the long term. Collaboration enables the developers to identify new requirements and thus "change directions quickly so that they can produce more appropriate results and less expensive designs" [7].

2.1 Employing Agile Methods

Employing Agile methods "is a challenging task demanding a great deal of adjustment from all the stakeholders involved in the software development process" [13]. Therefore companies need assistance "to support systematic selection, deployment, and tailoring of agile practices to fit the firm's software development context" [14]. Methods that fall under the agile umbrella include eXtreme Programming (XP), Test Driven Development (TDD), SCRUM, and the Dynamic System Development Method (DSDM).

Extreme Programming. The agile development method, eXtreme Programming (XP), was implemented within Sporting Software Ltd[1]., the company which was studied. XP is an agile method that is built around 12 related practices. One of the keys to XP is the replacement of physical documents with face-to-face and informal communication between the various partners with the aim of "creating executable code and automated test drivers". The objective of XP's practices is to increase productivity while ensuring the same level of quality. Therefore they are grouped according to 3 key areas [15]:

> *Customer's Satisfaction* - a high quality system is of no practical use if it does not meet the basic needs of the client;
> *Software Quality* – the aim is to produce a system of extremely high quality;
> *Project Management* – XP aims to reduce management overhead while still considering the customer.

3 Research Methodology

The purpose of this research project was to examine the problems encountered and benefits derived by a very small Irish company when implementing an agile methodology – in this case eXtreme Programming (XP). Given the objectives and limited resources available, it was determined that a qualitative approach should be employed. Of the qualitative approaches available [16], the case study approach was chosen. Research data was collected using a number of techniques that included observation, interviews and informal conversations. Interview questions and conversations firstly focused on the

[1] Sporting Software Ltd. is a pseudonym.

perceived benefits and drawbacks that both developers and management expected to experience with employing the new methodology. This information was then compared and contrasted with data released by the Shine Technology organization [2] and other industrial experts such as Schwaber [6], Highsmith and Cockburn [7]. Once the perceived issues – such as the 40 hour week, increased morale and productivity – had been identified, the questions focused on the actual benefits and drawbacks as experienced by the participating parties, for example, actual hours worked and effect of limited documentation. This information was then compared and contrasted with all other collected data.

Further information was extracted from the system documentation such as project specifications, bugs and issues list, emails sent between those involved in the project from the technical and business areas, and documentation distributed with information about the agile methodology which was employed. The purpose of reviewing and analyzing such documentation was to determine the exact level of compliance with the principles of the selected AGILE method. Reviewing the previously mentioned documentation also allowed for the verification of the data collected during interviews and informal conversations.

Business oriented techniques - SWOT[2] and PESTEL[3] analyses - were also employed for the purpose of making the results of the research applicable to the business by identifying the issues that could potentially have either a positive or negative effect on the overall methodology implementation and on the success of the project itself. It should be noted that the project was a full commercial venture with the objective of generating capital for the company in question.

4 Case Study

Sporting Software Ltd., is located remotely close to the west coast of Ireland, and develops bespoke software applications using a mixture of Microsoft technologies (e.g. MS SQL SERVER (2000), MS VISUAL STUDIO .NET (2003)), COBOL, and open-source technologies. The company provides consultancy, technical support and network solutions on-demand. They have a total workforce of 10 employees, consisting of 4 software developers, 1 quality manager, 2 network support technicians, 2 sales personnel, and 1 administrator. All developers and network support technicians are responsible for the provision of technical support to customers, and for answering queries from potential customers.

Quality standards are an important marketing factor for Sporting Software Ltd. They employ the PRINCE2 process based approach to project management [17] - the de facto standard used by Irish Government bodies and the private sector, both Irish and international. They have ISO 9001:2000 certification in order to meet specific customer requirements. Sporting Software Ltd. has also enhanced its Quality Management System (QMS) by incorporating the principles of other standards such as GAMP[4], IEEE Standard 829-1998[5] and BS7925-2[6].

---

[2] SWOT – Strengths, Weaknesses, Opportunities and Threats.

[3] PESTEL – Political, Economic, Social-Cultural, Technology, Environmental, and Legal.

[4] *GAMP* (Good Automated Manufacturing Process) – standard outlines a transparent development, test and installation procedure.

One of the company's products, COMMS, is a web based application aimed primarily at sports clubs, in particular those associated with the Gaelic Athletic Association (GAA), the Football Association of Ireland (FAI), and Irish Rugby Football Union (IRFU). By employing a web browser, or mobile phone, the system allows managers to send SMS text messages, manage membership details and history, and track all messages sent by the system from their account. The system also allows purchase of additional message credits using credit cards or cheque book.

In late 2006, a decision was made to migrate to the MS VISUAL STUDIO .NET (2005) development environment and MS SQL SERVER (2005) database. This required that all code be upgraded from ASP.NET 1.0 to version 2.0, and was carried out using automated tools provided as part of MS VISUAL STUDIO .NET (2005).

4.1 Project Instigation

At the initial project meeting to introduce the requirements for the new system, managers informed the developers that they would be employing a new development methodology, the eXtreme Programming (XP) approach. This was to improve productivity and quality while reducing costs. Developers were given a 5 page document outlining XP, its key advantages, the 12 key principles, and a brief comparison with existing techniques such as RUP and UML. Emphasis was placed on the production of high quality code using the principles of pair-programming and code ownership. Management also presented the benefits of the new methodology from the developer's point of view, especially on the reduced development time and the 40 hour week.

Given the small size of the company, there were only 2 developers working on the project, therefore only one XP pairing was possible. The Quality manager (QM), who was part-time on this project, was responsible for reviewing documentation and sign-offs. The QM was also used as an 'independent person' if there was disagreement over such requirements as usability. The original Project manager (PM), who was also involved in sales, was involved in setting requirement priorities. This PM was replaced about half-way through the project by a senior developer who had been used as a consultant in the early stages of the project. The client was the managing director of the company. He was responsible for setting new requirements and aligning and prioritizing existing requirements for the system. He spoke regularly to customers and had the final decision making power.

5 Effect of Implementing Agile Methods

When implementing XP within Sporting Software Ltd, management was surprised at the number of unexpected problems experienced. This section discusses both the problems and benefits, and why these indicate that the implementation of agile methods within a very small company located remotely is not easily achievable.

---

[5] *IEEE Standard 829-1998* – outlines a standard for test documentation.

[6] *BS 7925-2* – defines the process for software component testing using specified test case design and measurement techniques.

5.1 Problems Encountered

Pair Programming. Pair programming involves two individuals sitting side by side working on a single machine. Using this configuration, one developer (driver) is responsible for driving the development/implementation while the second developer (navigator) is responsible for identifying more effective approaches and any flaws in the current approach. In some cases, the navigator is responsible for writing test specifications. It is advised that the role of driver and navigator should be swapped every couple of hours in order to prevent the onset of boredom. Williams et al. [18] have indicated that although pair programming can increase the effort taken to implement requirements by 15%, it can significantly increase the quality of software produced.

Effective pair programming requires a significant level of coordination, cooperation and trust between those involved in the pairing and effective and efficient pairings should be identified by management in order to maximise productivity.

As it was understood to be a core principle of XP, pair programming was supported by management. Due to the limited number of developers, no evaluation occurred into the suitability of pairing the developers together. Luckily, both developers had an excellent working relationship at the start of the project, and had conferred with each other on technical matters relating to previous projects that they had been involved in separately. During the initial project meeting, the developers decided that the driver would be responsible for implementing an entire requirement rather than changing periodically. An entire requirement could range from implementing a function that allows a user to add, edit, and delete a member to simply changing the message printed out on a label in response to a particular event. While the driver was implementing the requirement, the navigator monitored the code and prepared testing data. This was not a good decision as the requirement implantation could range from 1 minute to 40 hours, thus causing boredom on behalf of the navigator.

The second problem experienced when implementing pair programming related to the size of the available computer monitor – a 15 inches laptop screen. This significantly reduced the working area, and also caused problems when switching between different applications. Requests were submitted for either a larger or second monitor, but the tight financial restrictions within which small companies operate came to the forefront, and this solution was not implemented.

Forty-hour week. XP calls for a sustainable development approach whereby developers avoid working excessively long hours during a standard working week. By restricting the developer's working hours to 40, companies reduce the risk of introducing poor quality code due to reasons of tiredness and lapses of attention. Management want developers "to be fresh and eager every morning, and tired and satisfied every night" and at the start of every week they want developers "to come in full of fire and ideas" [19].

Sporting Software Ltd. employees have contractual flexible working arrangements. When traditional development methodologies were employed, differences in daily start times had a minimal effect on the work of individual developers. However, this flexibility, when combined with the requirement to employ the pair programming concept, and the remote location of the company, resulted in significant problems being encountered when trying to achieve the 40 hour working week. Although

developers informally discussed the possibility of a common daily start and finish time, this arrangement rapidly collapsed due to the contrasting family commitments of individual personnel. Given that it was a management requirement that pair programming be employed, differences in start times resulted in one developer working additional hours. For that one developer, his working week amounted to 47-50 hours per week. The difference in start time also resulted in the lunch hour being adjusted on a daily basis – usually in deference to the senior developer.

The increased working week combined with the lack of consistency in working hours, significantly affected the morale of individual developers. Lower morale was further affected by the remote location of the company, as access to retail outlets, stores and other facilities was limited. This drop in morale rapidly spread and began to have an affect on others involved in the development project.

Long term maintenance. Opponents to agile methods point to the long term maintenance of the system and argue that the lack of documentation as espoused by agile methods make this task extremely difficult if not impossible: "With no design documentation or requirements to refer to, you're left with only the code. I contend that even incomplete or outdated design documentation is better than no documentation" [11]. However, advocates argue that the working design speaks louder than the documentation and provides a more accurate representation of the system, as documentation can be used to hide problems and design flaws.

In relation to the COMMS system, the lack of up-to-date documentation resulted in numerous problems between those involved in selling the product and those involved in its development. Often, sales personnel would declare at meetings that the system once supported a feature but no longer did so. They were inclined to blame developers for randomly dropping features and sloppy source control. Should the developer reply that the system, as it stood, represented the requirements as specified by the sales personnel, meetings were likely to disintegrate into arguments and shouting matches. During one such meeting, developers threatened to start bringing a voice recorder and camera to future meetings to protect themselves from verbal abuse and misinformation. Furthermore, developers recognised that the lack of documentation effectively left them without any protective cover when dealing with the client. This lack of protection or shield had a significant impact on the level of morale among developers, especially in the lead up to project meetings and lead to calls for a return to previously employed traditional methods.

Rapidly changing requirements. Agile methods welcome changing requirements as they allow developers to produce a system that eventually meets the actual needs of the customer. This is in contrast to traditional techniques which often produce systems that fail to meet many of the user's basic needs. Once an iteration has been completed, the developers sit down with the client to identify and analyse any new requirements that may exist. It is the responsibility of the client to allocate a priority rating to each requirement. The client is then asked to either identify those requirements that are most essential for the iteration (DSDM), or alternatively the developers indicate that based on the timescale available that they will be able to complete the first 10 requirements (SCRUM).

In the case of COMMS, rather than wait for the iteration to complete, management were inclined in some cases to inject new requirements into the middle of the

iteration, while in other cases they re-prioritised requirements. On one occasion, developers spent nearly 2 working weeks changing the look of a data-grid in terms of fonts, colours, and data layout as the client and project manager declared that each new look was getting closer to what they wanted. Changing requirements, without adjusting the iteration time or without restarting the iteration, resulted in the workload of individual developers rapidly increasing. Consequently, developers cut corners in order to complete the work within the required timeframe. Additional bugs began to appear and the adaptability of the code was significantly compromised.

Problems experienced with the pair programming concept itself (as previously discussed), combined with these uncontrolled rapidly changing requirements, caused pair programming to be abandoned. Developers reverted to programming alone, agreeing to coordinate activities to maximise development resources.

Changing requirements and failure to restart iterations caused developers to exceed the 40 hour week on a more regular basis. The impact of this was that lower quality code was produced, with an increased number of bugs and implementation issues. Thus management and developers were able to observe first hand Schwaber's [6] comments regarding developers' willingness to compromise code quality in order to meet the demands and expectations of clients, and the additional problems that the compromised code caused.

Collective Code Ownership. With traditional approaches, often one individual is responsible "for the detailed design, the implementation and also for the maintenance" [20] of modules and components in the overall system. Collective ownership, or no ownership [21], advocates that nobody is allocated responsibility for any particular piece of code. This allows anyone to make a change to any piece of code at any stage in the development process. This concept gives companies the flexibility to redeploy staff to new tasks as and when needed.

In the case of COMMS, due to the abandonment of pair programming and the 40 hour week, developers reverted to the traditional approach of code ownership and became extremely territorial about 'their' code. In some cases, one developer would automatically blame the other for bugs in the overall code – this allocation of blame had the effect of significantly complicating relations between individuals, especially when ownership of the offending code could not be fully determined. Other problems were encountered in regards to the source control environment employed – whereby one developer would check out an essential class or module for editing, and maintain the class as checked out for long periods of time. This long term check-out prevented other personnel from accessing the most up-to-date version of the code or adding their own methods and performing code alterations.

In order to prevent further deterioration in morale and relations between developers, developers themselves came to an arrangement whereby a number of comments would be added to each method when created and edited. When a new method was created the author was required to include their name, date/time and method description. When a method or class was to be edited, the editor, date/time and reason for edit were added.

Furthermore, should a developer wish to add a new class or module, verbal confirmation was required from the second developer. Although the addition of these comments, and the need to seek approval for new classes and functions, resulted in

the code moving from an agile type approach back to a traditional based one and added to the workload of developers, it did allow everybody take a step back from a mentality where everything that the other individual did was considered poor and degraded the quality of the software.

Lack of insulation from clients by project manager. Although agile methods recommend that the developers have full time access to a representative from the client company, in cases this can prove problematic. By having full time access, the developers can refer to the representative – who is considered to be a domain expert – to clarify any requirements that are poorly specified. However, such a level of access can raise the client's expectations of the system and to the level of influence that they will have in future project, especially if the developers decides not to proceed with such an arrangement again.

In the case of the system been examined, the client was the developer's boss (project manager). This meant that they had a greater understanding of what was going on in development and a greater degree of access to information than would usually be afforded to the client. Furthermore, the project manager was heavily involved in the setting of system requirements and was just as likely to inject new requirements during an iteration. The project manager's involvement in setting requirements also prevented the individual from viewing new requirements in an objective fashion, and in carrying out their job of insulating the developers from the demands and activities of the client. This issue was raised on a number of occasions before management decided to act and replace the original project manager with a developer who was previously involved with the project. The replaced project manger continued to play a significant role in specifying new requirements as they were allocated the responsibility of acting as the client's representative to customers.

5.2 Benefits Derived

Upgrading of junior programmer's skills. Where pair programming can especially be beneficial is when a firm is trying to improve the skills of junior developers by pairing them with senior staff. This will also introduce them to the coding style of the company while ensuring that all relevant procedures are fully understood and followed.

Although the principle of pair programming was eventually abandoned, its early use allowed for the junior developer on the project team to learn new skills and to identify the style used by the company when programming. By learning the company's development style, other developers from within the company were quickly and easily able to read any code written by the new developer. Furthermore, the pair programming concept allowed the junior developer to explore new development concepts and approaches while having direct access to someone with whom they could discuss tactics and approach with. Also by having a senior developer present, the junior developer could quickly be pulled up on mistakes and bad approaches, effectively learning good practices on the job.

Pushed developers to rapidly adapt to a new environment and challenges. Species that fail to adapt to changes in their environment eventually die off, the greatest example of this is the dinosaurs. What the introduction of XP for the COMMS project did was to force developers to adapt to a new development methodology and to

learn new skills and languages. By adapting and learning these new skills, developers increased their employment opportunities and overall experience.

5.3 So What Went Wrong?

Although Agile methods were introduced with great intention in Sporting Software Ltd., their application are considered a failure in relation to the COMMS project due to the developers having to abandon many of the core principles such as pair programming, code ownership, and the 40 hour week. Regardless, many valuable lessons have been learned by those involved in relation to existing problems within the company, and on poor practices being employed by developers and management. With the identification of these problems, corrective action can now be planned for and implemented with the ultimate aim of possibly retrying agile methods or other alternative methods again in the future.

There were a number of issues that were prevalent during the implementation of agile methods within Sporting Software Ltd. These can be summarized as:

- Software Development group was too small;
- Sales force / project manager was the 'client';
- Lack of documentation caused problems;
- Management paid lip-service to the implementation of XP;
- Prior employment agreements interfered with XP implementation.

Software Development group was too small. Maurer and Martel [15] state that agile methods are best employed by teams of 5 to 15 developers. This was bourn out in the case of this very small company. There was no flexibility as to what developers would be grouped together, and, even though both worked very well together prior to this project, the pairing did not work out.

Sales force / project manager was the 'client'. The project manager in the early stages of the project had much control over the requirements, which in many cases were what he felt would benefit the customer. As this person had a dual role within the organization, this led to many additional requirements being added to iterations. Even in these circumstances, iterations were not re-started nor were time schedules changed as would have been expected.

Lack of documentation caused problems. With agile methods, documentation is not normally used. Sporting Software Ltd. experienced many problems with this, as the developers did not feel that the decisions that were made at meetings were maintained. This caused a further breakdown in relations between the developers and the project manager.

Management paid lip-service to the implementation of XP. While requirements changed, management did not ensure that XP issued were dealt with correctly. At one level they required that developers would work a 40-hour week, but they did not tackle the issues that arose due to employment contracts. They did not ensure that requirements did not 'creep' between iterations. Management support is important for change to be implemented [22], and in this case lack of such support contributed to the failure of XP implementation.

Prior employment agreements interfered with XP implementation. Sporting Software Ltd. had flexible working agreements in place with their employees. Having people work in pairs directly affected this as now employees not only had to take their own lives into account when coming into work, but also, on a consistent basis, consider another developer.

At the time of writing, the COMMS system was still undergoing development, with new features being added on a regular basis. Significant refactoring of existing code was also occurring on a weekly basis in order to improve the usability of the system, and to remove low quality and problematic code.

6 Conclusion

In the case study presented, the implementation of eXtreme Programming did not work. While many of the problems experienced were caused by poor planning and management support, implementing a 40-hour week within a very small company where pre-existing contractual arrangements existed also had an effect.

Therefore the question is: how should firms approach agile methods? Lindvall et al., [1] argue that companies should identify and examine those aspects of agile methods that are relevant to themselves. However, they should also examine what can go wrong when agile methods are implemented. By carrying out this examination, and by applying limited agile practices such as pair programming in a controlled environment, companies will be in a better position to decide for themselves as to the usefulness of such methods and practices. Furthermore, those that carry out such an examination may be surprised to learn that they have been using practices similar to those espoused by agile methods. The implementation of a limited number of successful agile practices may place them in a position to exploit competitor's weaknesses, and to increase the probability of repeat business with customers.

Acknowledgments. This research was partially supported by the Science Foundation Ireland funded project, Global Software Development in Small to Medium Sized Enterprises (grant number 03/IN3/1408C) within Lero - the Irish Software Engineering Research Centre (http://www.lero.ie) and by the Higher Education Authority through the M.Sc. in Software Engineering, University of Limerick. We would also like to thank employees and management of Sporting Software Ltd.

References

1. Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., Kahkonen, T.: Agile Software Development in Large Organisations. Computer 37(12), 26–34 (2004)
2. Shine Technologies, `http://www.shinetech.com/display/www/Extreme+success+with+Agile`
3. Schatz, B., Abdelshafi, I.: Primavera Gets Agile: A Successful transition to Agile Development. IEEE Software 22(3), 36–42 (2005)
4. Leszak, M., Perry, D., Stoll, D.: A Case Study in Root Cause Defect Analysis. In: 22nd International Conference on Software Engineering (June 2000)

5. Cockburn, A.: http://alistair.cockburn.us/index.php/Agile_software_development:_the_business_of_innovation
6. Google, http://video.google.com/videoplay?docid=-7230144396191025011&q=Scrum+et+al.&total=2&start=0&num=10&so=0&type=search&plindex=0
7. Highsmith, J., Cockburn, A.: Agile Software Development: The Business of Innovation. Computer 34(9), 120–122 (2001)
8. Griffiths, M.: Crossing the agile chasm: DSDM as an Enterprise Friendly Wrapper for Agile Development. Quadrus Development Inc., 1–13 (2003)
9. Infinity-IT, http://www.infinityit.co.il/index.aspx?id1405
10. The Great Pyramid of Agile – The Daily WTF, http://worsethanfailure.com/Articles/The-Great-Pyramid-of-Agile.aspx
11. Software Quality Consulting Inc., http://www.swqual.com/newsletter/vol2/no7/vol2no7.html
12. Paulk, M.C.: Extreme Programming From a CMM Perspective. IEEE Software 18(6), 19–26
13. Cohn, M., Ford, D.: Introducing an Agile Process to an Organization [Software Developer]. IEEE Computer Society 36(6), 74–78 (2003)
14. Pikkarainen, M., Salo, O., Still, J.: Deploying Agile Practices in Organizations: A Case Study. In: Case Study, EuroSPI 2005 (2005)
15. Maurer, F., Martel, S.: Extreme Programming: Rapid Development for Web-based Applications. IEEE Internet Computing 6(1), 86–90 (2002)
16. Association For Information Systems, http://www.qual.auckland.ac.nz
17. The Office of Government Commerce, http://www.ogc.gov.uk/prince2/about_p2/about_intro.html
18. Williams, L., Kessler, R.R., Cunningham, W., Jeffries, R.: Strengthening the case for Pair Programming. IEEE Software 17(4), 19–25 (2000)
19. Beck, K.: Extreme Programming Explained: Embrace Change. Addison Wesley Longman, Reading (2000)
20. XP Exchange, http://www.xpexchange.net/english/intro/collectivecodeownership.html
21. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison Wesley, Reading (1999)
22. Richardson, I., Varkoi, T.: Managing Change when Implementing Software Process Improvement Initiatives. In: EuroSPI 2005 (2005)

A Process Asset Library to Support Software Process Improvement in Small Settings

Jose A. Calvo-Manzano, Gonzalo Cuevas, Tomas San Feliu, and Ariel Serrano

Faculty of Computer Science, Polytechnic University of Madrid (UPM)
Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain
{jacalvo,gcuevas,tsanfe}@fi.upm.es,
aserrano@mpsei.fi.upm.es

Abstract. A main factor to the success of any organization process improvement effort is the Process Asset Library implementation that provides a central database accessible by anyone at the organization. This repository includes any process support materials to help process deployment. Those materials are composed of organization's standard software process, software process related documentation, descriptions of the software life cycles, guidelines, examples, templates, and any artefacts that the organization considers useful to help the process improvement. This paper describes the structure and contents of the Web-based Process Asset Library for Small businesses and small groups within large organizations. This library is structured using CMMI as reference model in order to implement those Process Areas described by this model.

Keywords: PAL, process library, process assets, OPD, CMMI.

1 Introduction

The Process Asset term was introduced for the first time by the Software Capability Maturity Model, (SW-CMM v1.1.) [1]. The key process area *Organization Process Definition (OPD)* describes the process asset concept. Its main objective is to develop and maintain a set of process assets that improve process performance across the organization projects and provide a basis for cumulative knowledge database process.

The SW-CMM defines process assets as a collection of entities for use by projects in developing, tailoring, maintaining, and implementing their software processes [1]. Also a process asset could be any entity that the organization considers useful in performing the activities of process definition and maintenance. These process assets provide the foundation to institutionalise the processes in any organization [1]. Although, SW-CMM introduces for the first time the process asset concept, it does not refer to the term *Process Asset Library*.

In the SW-CMM, the PAL concept was loosely translated into the "library of process-related documentation". As software process improvement became more common, the term *Process Asset Library* started to be used more frequently than the term used by SW-CMM [2].

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 25–35, 2008.

The successor of the SW-CMM the *Capability Maturity Model Integration better known with the acronym* CMMI [3, 4] takes all the concepts from its predecessor and develop the term *Process Asset Library (PAL)*, giving the importance that it have and dedicating a specific practice to create and maintain a PAL.

The Process Asset Library (PAL) is a repository of information used to keep and make available all process assets that are useful to those who are defining, implementing, and managing processes in the organization [3, 4]. These process assets typically include:

- organization's standard defined software process,
- software process-related documentation,
- descriptions of the software life cycles,
- guidelines for tailoring the organization's standard software process,
- process implementation aids,
- checklists and templates,
- lessons-learned documents,
- standards, procedures, plans,
- training materials, and other artefacts that can be used by a project.

Apart from these, any entity that the organization considers useful in performing the activities of process definition and maintenance could be included as a process asset.

Other definition describe the PAL as an organized, well-indexed, searchable repository of process assets that is easily accessible by anyone who needs process guidance information, examples, data, templates or other process support materials [2]. PAL development is very important for an organization, because it keeps in the same repository all artefacts that could be used by process. The PAL is more than a single repository of things, it is the live component of the organization and is itself the database that contains all organizational process patrimony [2].

A Process Asset Library is a very important source of guidance to an organization. A well designed PAL facilitates standardization and process improvement in organizations, and is the key to achieve higher capability maturity levels. A poorly designed PAL degenerates into an attic of process that discourages and frustrates process users [5].

The first step in a process improvement effort for an organization is to have well documented policies, process definitions, procedures, project plans, quality plans, process guides, and lessons learned. The second one is to have an electronic repository to keep all this information and maintain available for all process users. A well designed and implemented PAL reduces planning, implementation and training costs throughout whole organization, especially in those processes that only are partially executed [6].

In every organization, a PAL provides the key infrastructure element that is required to support the process improvement effort. The PAL allows all the process information that it is needed when a new project commence to be public for all organization, [6].

2 Process Asset Library Goals and Implementation Benefits

This section explains goals and benefits that could be obtained with the implementation of a Process Asset Library.

2.1 The Process Asset Library Goals

The main Process Asset Library (PAL) goal is to provide an organized, indexed, searchable repository of process assets and make it easily accessible to anyone who needs process guidance information. The PAL provides to the organization a central database for acquiring, defining, and disseminating the knowledge about processes related to the software development and maintenance [2]. Other PAL goals include but are not limited to the following:

- Reduce unnecessary duplication of process assets.
- Provide mechanisms for sharing knowledge about the process assets and how they will be used by the process owners and users.
- Support an effective learning environment to train new employees related have to use the organization's processes.
- Provide a basis for making decisions about developing and tailoring all organization's processes.
- Improve the consistency of content related to process assets.

In addition a PAL should be containing the lessons learned of those organization's projects that have been successful, in order to increase the knowledge database with each project's best practices.

2.2 Process Asset Library Implementation Benefits

For small settings, a PAL is a key infrastructure element that reduces training time, and helps to guide a process focused culture within the organization [2]. Also, a PAL is a key element to support the reduction in time needed for planning new projects. Other benefits could include:

- Increasing the participation of the people involved to the process in making suggestions for changes to process assets.
- Reducing the cost of project start-up, both from the point of view of less training time needed to prepare people in the way of the processes to be used, and from the point of view of reusing the existing assets.
- Making easy process standardization because the organization's projects use the same type of assets and facilitating the adaptation of those assets that are not compliant with some kind of projects.
- Providing information related to projects that are useful to develop an Integral Balanced Scorecard (BSC). The balanced scorecard is a strategic planning and management system to align business activities to the vision and strategy of the organization, improve internal and external communications, and monitor organization performance against strategic goals.
- Facilitating the implementation of new processes and allows that entire organization takes advantage of process assets stored in the PAL.

Note that the value of the process assets depends on its availability and accessibility, and with the PAL implementation is easier to gain this value. "The ability to rapidly deploy and use processes to serve the needs of their marketplace is a critical attribute of an organization experiencing hypergrowth" [7].

3 Web-Based Process Assets Library Application Structure

Considering the benefits that a PAL implementation brings to the success of a process improvement effort, this research work focuses on the development of a Web-based Process Assets Library easily accessible by the process users. This Web-based PAL has been building in the Polytechnic University of Madrid by the Research Group of Software Process Improvement for Spain and Latin American Region. The name of this research project is PAL-SS (*Process Asset Library for Small Settings*).

The International Process Research Consortium (IPRC) uses the term Small Settings to define organizations that are small and medium size enterprises, small groups within large companies, or small projects. The Software Engineering Institute (SEI) is the sponsor and organizer of the IPRC [8].

Small Settings are special challenges of process improvement. In the United States, small businesses account for 99 percent of all employer firms, employ 50 percent of all private sector employees, and hold 41 percent of all high technology jobs. Small businesses are recognized as a critical component of the USA economy. In other countries, small business is the economy [8].

The PAL-SS was developed initially as a prototype an currently is accessible by the Word Wide Web. Its main objective is to maintain available the knowledge of the process assets that has been developed by this research group for Small Settings organizations.

This paper describes the main structure of the PAL-SS that is based on the CMMI model components (Figure 1). The PAL-SS is designed using a Web-based platform, which in turn, allows the users to exploit it without any complex installation. Additionally PAL-SS could be used from any software and hardware platform. It can be accessed using any type of web browser (e.g Mozilla Firefox, Microsoft Internet Explorer, Netscape, or Opera).

The design of a PAL-SS include in their structure the three critical dimensions that organizations typically focus on to improve its process: "People", "Procedures and Methods", and "Tools and Equipment"[9].

The PAL-SS is structuring in intuitive way and its architecture allows adding process assets following the CMMI process model. In order to maintain the CMMI structure, each asset links with their respective subpractice, and each subpractice belongs to a generic or specific practice. The implementation of these practices will allow the goals achievement of a process area (Fig. 1).

The structure of the PAL-SS is based on standard process components, these components are classified by process areas, practices, subpractices, and products. Components are called standard because they will be used by different projects. Those standard components are grouped into patterns. A pattern keeps a cluster of standard components that are going to be used by a unique type of project but this pattern could be used by another equivalent project.

For example, on the one hand, the pattern "A" contains a set of process components 1, 3 and 5, on the other hand, the pattern "B" contains a set process components

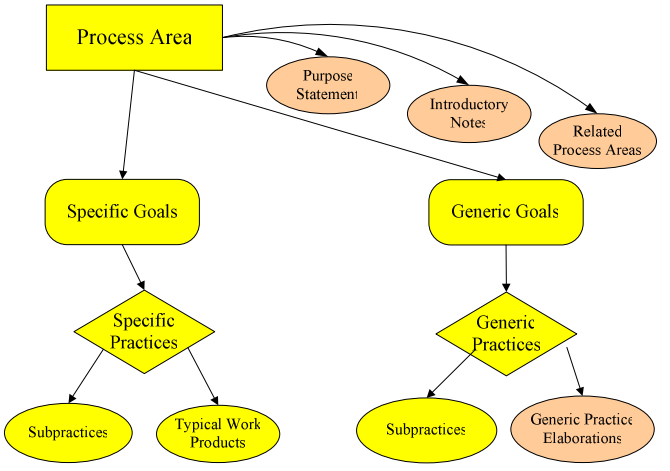


Fig. 1. CMMI model components

1, 4 and 8, in such a way all projects that need using process components 1, 3, 5 use Pattern "A" and all projects that need using process components 1, 4, 8 use Pattern "B". With the use of patterns it will be able to cluster standard process components in n different types of projects, so a pattern previously defined could be used by other projects of the same characteristics.

The use of patterns will allow having a knowledge database of those projects that are used with more frequency and in this way it will facilitate the use of defined assets into equal or similar projects (Fig. 2).

In order to facilitate the inclusion and the search of assets, the PAL-SS has been structured in two standard parts, on the one hand, "Organizational Standard Definitions" which maintain all the process components such as (process areas, goals, practices, subpractices and products) and, on the other hand, "Organizational Standard Metrics" which preserve all the metric that will be used to measure to each standard process component. Moreover those standard and metric are cluster into "Patter Statement Definitions" (Fig. 2).

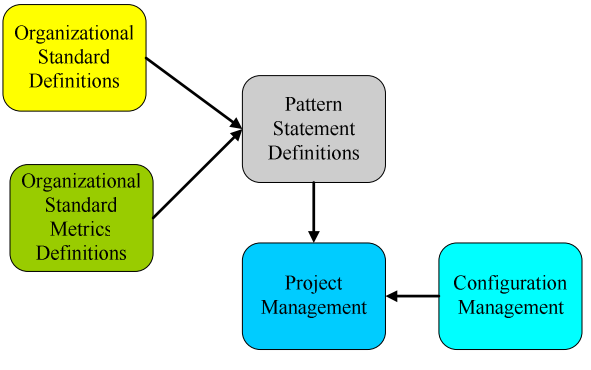


Fig. 2. Functional structure of PAL-SS

3.1 CMMI Model Components Descriptions

The PAL-SS is structured using the CMMI model components as reference, and the database design include each component.

The description of CMMI model components are the following [9]:

Process Areas. A process area is a cluster of related practices in an area that, when implemented collectively, satisfy a set of goals considered important for making improvement in that area. CMMI for Development v1.2 has divided into 22 Process Areas.

Specific Goals. A specific goal describes the unique characteristics that must be present to satisfy the process area. A specific goal is used in appraisals to help determine whether a process area is satisfied.

Generic Goals. A generic goal describes the characteristics that must be present to institutionalize the processes that implement a process area. Generic goals apply to multiple process areas.

Specific Practices. A specific practice is the description of an activity that is considered important in achieving the associated specific goal. The specific practices describe the activities that are expected to result in achievement of the specific goals of a process area.

Subpractices. A subpractice is a detailed description that provides guidance for interpreting and implementing a specific or generic practice.

Typical Work Products. The typical work products section lists sample output from a specific practice. These examples are called typical work products because there are often other work products that are also useful but are not listed.

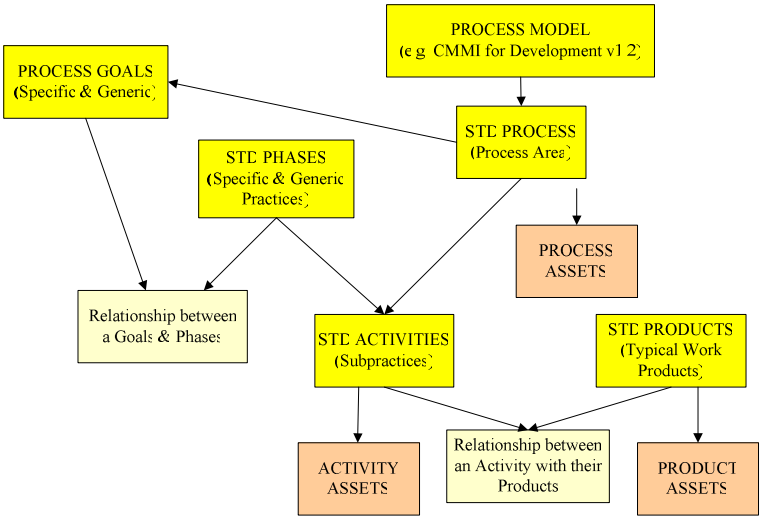


Fig. 3. Functional components in a PAL-SS relational database

Generic Practice Elaborations. A generic practice elaboration appears after a generic practice in a process area to provide guidance on how the generic practice should be applied uniquely to the process area.

3.2 Process Assets Library Structure Using CMMI Model Components as a Reference

Using the model components described at section 3.1 the PAL-SS has been structured taking these components as reference (Fig. 3).

The PAL-SS takes each CMMI Process Area and keeps up a correspondence with "Standard Process". The Specific and Generic Goals keeps up a correspondence with "Process Goals". And each Specific and Generic Practices keeps up correspondence with "Standard Phases" (Fig. 4).

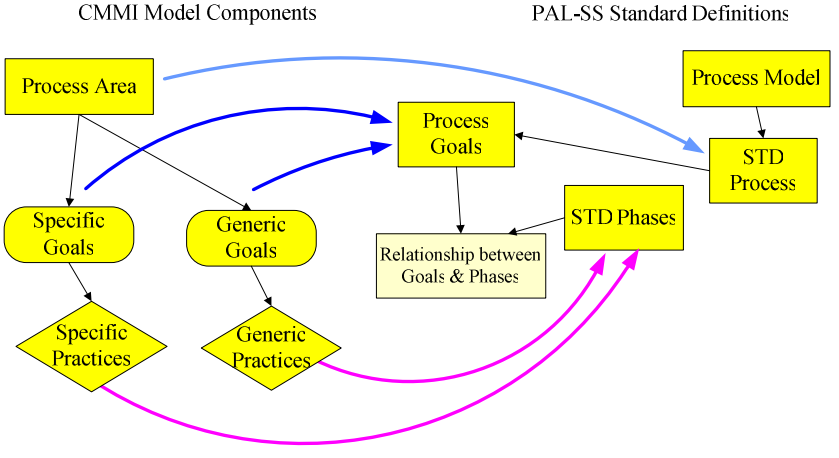


Fig. 4. CMMI model components match with PAL-SS

At the same time, and following the previous structure, the tasks needed to perform a process are represented at CMMI model components with the name of "*Subpractices*" and keep up a correspondence in the PAL-SS structure with "Standard Activities". When a task is performed the resulting products are named by CMMI as "*Typical Work Products*" and keep up a correspondence in the PAL-SS structure with "Standard Products" (Fig. 5).

Finally, the informative model components of CMMI, that is additional information that helps to understand a process area, are included at the PAL-SS as "Assets". These assets include a process area and they will be used for the implementation of the process. The PAL-SS structure classifies the assets into "Process Assets", "Activity Assets", and "Product Assets" (Fig. 6).

Process assets are divided by Process Area. Each Process Area contains the organization's policy for it, a process definition, and other supporting information.

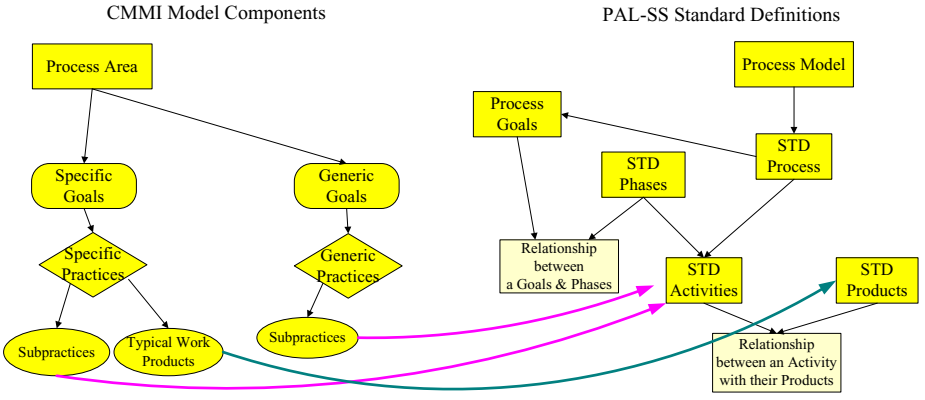


Fig. 5. Matching Subpractices, TWP with PAL-SS activities

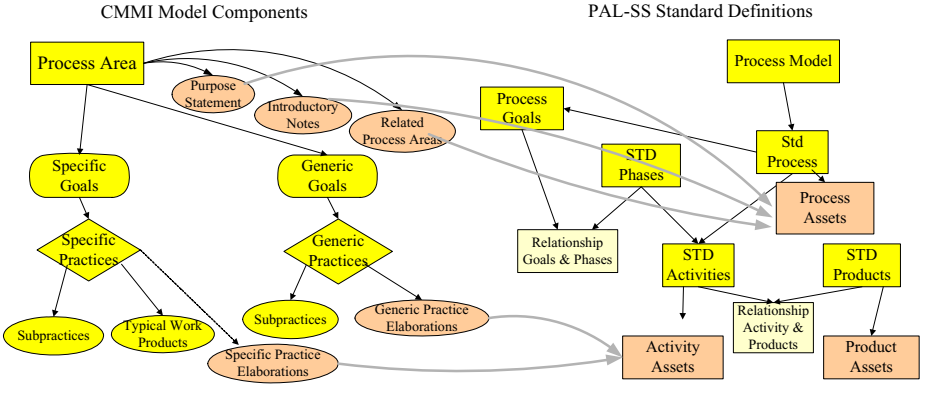


Fig. 6. CMMI informative model components are process assets in PAL-SS

Other information included as "Assets" are sample plans, templates, and other documents from our organization and others Small Settings. Internal processes used in the day-to-day process operations are also included.

This way the PAL-SS will allow making on a support tool for the process improvement implementation. It will let having an agile mechanism that contains all process components to be implemented.

The process area will include all the components and the assets that are required for the process implementation in a Small-Setting, such as process description, activities that are required to perform the process, tailoring guides and templates that will be due to use. Similarly, it will be due to include essential metrics that are required to measure the process and their products, which are necessary to be able to control the process.

3.3 Components of the Web-Based Process Assets Library for Small Settings

The PAL-SS web tool has been divided into five principal components (Fig. 7):

- Project management
- Organizational Standard Definitions
- Measurement Repository Definitions
- Configuration Management, and
- Pattern Statement Definitions

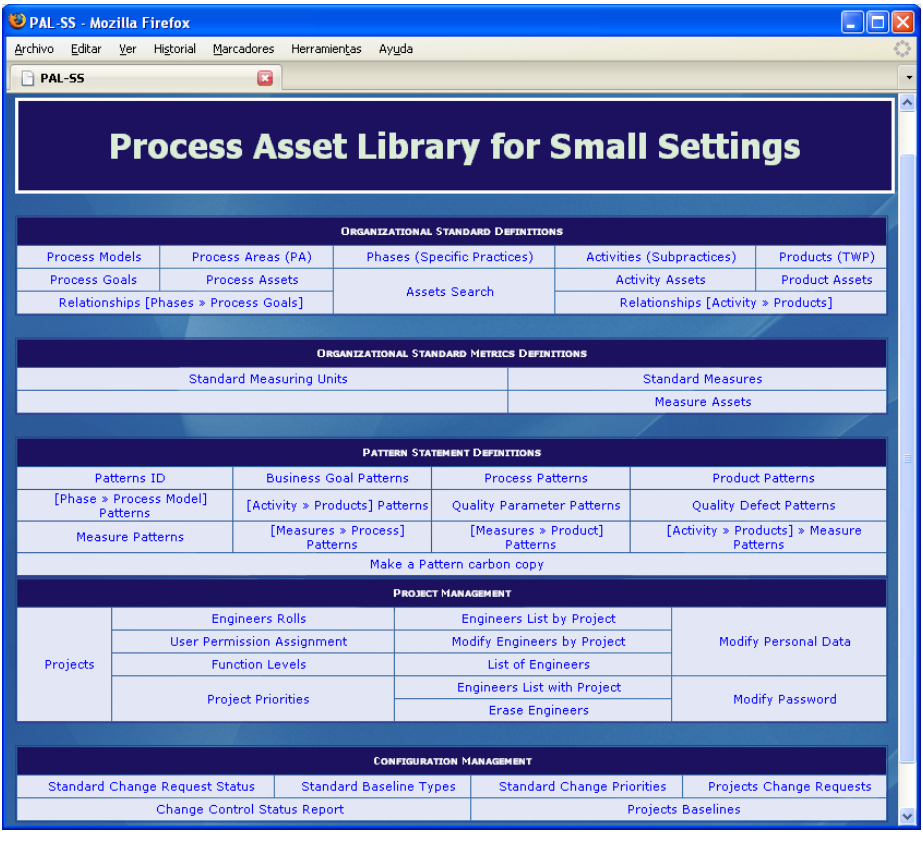


Fig. 7. Management Menu of web-based Process Asset Library for Small Setting

The functionality of each component is the following:

Organizational Standard Definitions. This component specifies all the relations that exist between a process their objectives, activities, tasks and products.

Measurement Repository Definitions. This component defines each metric and that it will be used to measure the process, tasks and products.

Pattern Statement Definitions. This component specifies a set of processes, phases, activities and products that were used by similar projects. Meaning that pattern for a project could only be used in projects of the same type, but it can be registered as many patterns as necessary.

Project Management. This component manages all projects and their data. It is important to say that each project must be match with a Pattern previously defined.

Configuration Management. This component manages the changes that have each project, such as requirement change, control baselines, project change management.

4 Conclusions

The PAL is a collection of assets, maintained by an organization, for use by projects in developing, tailoring, maintaining, and implementing their software processes an is important resource that can help reduce the effort in using processes [3, 4]. The CMMI establish that in order to advance at maturity level 3 a structured and well implemented Process Asset Library is required. Most of the organizations that are in advanced levels of maturity coincide on having a Process Asset Library and argue that is the key to have a culture focused to the maturity of the processes.[2]. A Process Asset has no value if it is not easily accessible by the users when they need it. Every matured organization has to implement a PAL but, it is generally one of the least concepts that an organization has considered for its implementation.

The Process Asset Library presented in this paper was developed by Research Group of Software Process Improvement for Spain and Latin American Region and it was structured to be used by Small Settings in a Web-based environment easy to use. Currently the PAL-SS has assets of Requirement Management (REQM), Project Planning (PP), and Project Monitoring and Control (PMC) process areas. Those assets will be used at the implementation those activities that are needed to perform the goals of a specific process area. The PAL-SS has been used by students software projects and is currently being validated it in a pilot project within a Small Setting. Future research includes developing assets from others CMMI process areas like as (PPQA) as well as the incorporation of other models like TCPi, CMMI-ACQ and ITIL.

Acknowledgments. This work is sponsored by Endesa, everis Foundation, Sun Microsystems, and Polytechnic University of Madrid through the Research Group of Software Process Improvement for Spain and Latin American Region.

References

[1] Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability Maturity Model for Software, Version 1.1. Pittsburgh, PA. Software Engineering Institute, Carnegie Mellon University (1993)

[2] Garcia, S.: What is a Process Asset Library? Why Should You Care, Aimware Professional Services, Inc., Boston, MA, USA (2004)

[3] CMU/SEI-2002-TR-011, Capability Maturity Model Integration (CMMI), Version 1.1, Continuous Representation," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., SEI March 2002 (2002)
[4] CMU/SEI-2002-TR-012, Capability Maturity Model Integration (CMMI), Version 1.1, Staged Representation, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, SEI March 2002 (2002)
[5] Fogle, S., Loulis, C., Neuendorf, B.: The Benchmarking Process: One Team's Experience. IEEE Software 18, 40–47 (2001)
[6] Groarke, B.: Web-Based Software Process Improvement Repository. CrossTalk The Journal of Defense Software Engineering 13, 24–25 (2000)
[7] Moore, G.A.: Inside the Tornado: Strategies for Developing, Leveraging, and Surviving Hypergrowth Markets. HarperCollins, New York (2004)
[8] Cepeda, S., Garcia, S., Langhout, J.: Is CMMI Useful and Usable in Small Settings? CrossTalk The Journal of Defense Software Engineering 21, 14–18 (2008)
[9] CMU/SEI-2006-TR-008, Capability Maturity Model Integration for Development (CMMI-DEV) version 1.2, in CMMI-DEV, V1.2, C. P. Team, Ed. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, p. 560 (2006)

Criteria for Estimating Effort for Requirements Changes

Bee Bee Chua[1], Danilo Valeros Bernardo[1], and June Verner[2]

[1] Faculty of Information Technology
University Of Technology, Sydney, Australia
bbchua@it.uts.edu.au, dbernard@ozonline.com.au
[2] National ICT Australia
june.verner@nicta.com.au

Abstract. IT practitioners realize that poor scheduling can cause project failure. This is because schedule overruns may be caused by the effort involved in making requirement changes. A software process improvement challenge is to better estimate the cost and effort of requirements changes. Difficulties with such effort estimation is partially caused by lack of data for analysis supported with little information about the data types involved in the requirements changes. This research is an exploratory study, based on change request forms, in requirements change categorization. This categorization can be used to develop an empirical model for requirements change effort as input into a cost estimation model. An empirically based estimation model will provide IT practitioners with a basis for better estimation of effort needed for requirements changes.

Keywords: Requirements Changes, Change Request, Rework Effort.

1 Introduction

Our objective is to address the information in change request forms that is usually insufficient to achieve a better understanding of the effort required for requirements changes during software development. Rework effort estimation should be more accurate if we are to categorize requirements changes on a number of dimensions related to the effort they require. The practicalities of using an approach to minimize software estimation risk are described in [1]. Prior to designing the framework, a preliminary investigation [2] and an empirical research study is needed to show why estimating rework effort for requirements changes is difficult and usually inaccurate. The cost of requirement changes after software development is also very expensive [3], and includes the cost of rework [4] during pre-maintenance [5], maintenance [6] and post-maintenance [7]. However, there are numerous requirements changes, which may be desired, but which cannot be implemented during a software development due to: 1) every requirement change is not equal in terms of its size, type and impact; 2) requirement changes are found at different points of the software development life cycle, resulting in difficulties in understanding the complexity of the change required; 3) major or minor requirement changes occur at the same time; 4) users and testers do not clearly describe requirement changes due to lack of understanding of the nature of requirements; 5) the assertions of IT practitioners regarding requirement changes

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 36–46, 2008.

are bounded by conflicting interests, especially when they are trying to prioritize which change to implement first, and 6) there are different perceptions of requirement changes by the user and tester with respect to implementation; for example, a typical user submits a change request form proposing a modification to an existing requirement because the system does not meet business needs. This kind of requirement changes is considered "non-error correction". In contrast a software tester may submit a change request form to report defects found during testing solely related to corrections required in a) data, b) software, c) telecommunication and d) hardware.

All in all, it is difficult for IT estimators to provide accurate effort estimation. The role of an IT estimator is not just planning for resources but also to estimate a project schedule precisely and concisely. Hence, there are several challenges when dealing with time, cost and effort estimation. Many cost estimation techniques are based on past data and not present projects and there is lack of appropriate data on which to base the calculation of a project implementation timeframe. Any change in requirements is driven by many factors such as: socio-economic, technology, business, politics and legal that impact on the actual software development effort required. These challenges increase an IT estimator's difficulties in meeting the following conditions for projects: 1) no time and budget overruns, 2) no unacceptable quality and non-agreed functionality, and 3) no delivery of empty promises of benefits to the intended users.

The remainder of this paper is organized as follows: In section 2, we present an update of the literature on requirements changes and their importance, where we found at present no effective cost estimation models are able to estimate rework effort. In section 3 we discuss our pilot study where we evaluate thirty-six change request forms for both government and non-government projects. Our proposed framework is illustrated and discussed in section 4 and the last section provides our conclusion and outlines future work on framework validation.

2 Literature Review

Requirement changes may occur during: 1) software design, 2) programming, 3) testing, 4) implementation, and 5) documentation. Each requirements change is regarded as being either due to a defect in the original requirements or caused by a change in the requirements [8, 9, 10, 11, 12, 13]. Therefore, there is a need to have a process in place to: 1) measure the number of requirements changes in order to balance them against the cost and effort required for the entire software development, and 2) to ensure requirements changes are appropriate given the project resources. According to Yashwant et al [14], the cost of fixing a defect is expensive because a large amount of time and effort is required to analyze the causes of a defect before actually fixing it. However, if a large amount of time and effort is spent fixing a defect, it can seriously affect project duration and project cost. Poor requirements analysis at the early stages of software development makes the problem worse [15] and can lead to project failure because of the number and cost of changes ultimately required to satisfy customers [16, 17].

No project is developed in a vacuum and even projects with the best initial set of requirements at the start, can still expect changes as the project progresses. The larger the project and the longer it takes, and the more requirements changes can be

expected. An understanding of requirements characteristics and their attributes can be confusing for IT practitioners and this makes it difficult to schedule rework accurately. In addition, a great deal of time and effort may be invested in meetings between change management and software development teams discussing every aspect of users' change requests.

Sommerville [18] noted that requirements changes are unavoidable, unpredictable and unanticipated because of changes in software project circumstances. Early risk mitigation for requirement changes in a software project helps ensure a project is completed on time and within budget. Hence, having a good understanding of requirements changes is necessary for estimating accurately the effort required to rework them. In addition, a categorization of requirements changes must be incorporated into a cost estimation model to expedite estimating "person effort" more accurately. To do so, it is important not only to understand the types of requirement changes but also to be aware of the characteristics of each type of change and the likely effort to make the change will avoid inaccurate effort estimation.

Analyzing patterns of requirement changes is difficult because we need not only understand the types of defect or omissions we must deal with, but also how much rework must be done. Parametric models estimate overall budget, provide cost and schedule breakdown by component, stage and activity, and can show the cost and schedule sensitivity of software project decisions. Other research tools for managing requirements have been developed. These tools assist in detecting and correcting poor requirements and helping to identify requirements defects [19, 20]. However, such tools do not provide any support in determining the cost of making changes.

There is a substantial body of research in software cost estimation tools for software project development effort including those based on expert judgment, parametric models, and analogy [21, 22, 23, 24, and 25]. However, estimation results may not always be accurate because there is too little data for analysis and/or inappropriate data types are used as the basis for estimating effort for requirements changes and integration. The difference between them is that a cost model uses one factor as its primary input, such as a size driver, and secondary cost driver factors, whereas constraint models are based on demonstrating the relationship over time between two or more parameters of effort, duration or staffing levels. However, since both types of models rely on mathematical equations, they may not estimate a project accurately because the time and effort for requirement changes (i.e. requirements rework effort), has not been considered. This explains why accurate project effort estimation is difficult. The aim of cost estimation models is to provide estimates for overall effort. Numerical data input is used to generate cost and time estimates but fails to illuminate the cost and schedule sensitivity of software project decisions. Hence, IT practitioners still face the same dilemma, that is the difficulty of understanding and interpreting the results produced, because the scientific and mathematical equations used to derive the final results are complicated [19, 20].

To estimate accurately the effort required during rework for any requirements change is a difficult task for many experienced cost and schedule estimators. This is because there is normally not enough data available on which to base a model, and there is no useful categorization of the various requirements types to help in estimating the amount of effort. Effort rework should be distributed into 1) re-investigation, 2) re-analysis, 3) re-identification, and 4) re-estimation, to trace the cause and cost of

the change, i.e. where do the changes lie? Where does the correction or modification fit in the lifecycle? from design to construction? or from construction to testing? or from testing to implementation etc? Effort estimation for rework is usually based on analogy. However, cost estimation models can help project managers by providing appropriate effort multipliers to help with better effort estimates. A cost estimation model such as COCOMO 2.0 [21], is useful for estimating effort based on appropriate personnel effort and schedules.

Verner et al [26] note that poorly managed change control, is a risk to project success. In addition, these researchers also report that poor estimation is frequently based on poor requirements. Chua et al [27] suggest that failing to consider requirements characteristics and attributes when making changes during software development also is a risk.

Change control forms designed by industrial practitioners are not based on any theoretical framework that categorizes requirement changes in a logical, economical and structured manner. Practitioners often design change control forms based on the types of requirements outlined in a project without real understanding of the requirement's actual characteristics and attributes. Until now, there has been no standard method for categorizing requirements changes able to provide an understanding that will enable IT practitioners to categorize requirements changes at both the project level (effect on the project) and the requirements level (effect on other requirements). We provide such a categorization and use this as the basis of a rework effort estimation model. In the following section, we present our findings based on a collection of thirty-six change request forms that we have gathered and explained why the change request information on them is not explicit enough to enable IT practitioners to gain a good understanding of what the change really involves.

3 Empirical Evidence Based on a Collection of Change Request Forms

Thirty-six change request forms are categorized by project sizes and types. Variables in each form are compared and analyzed and tabulated in a matrix. The limited sample size of organizations these variables are not generalizable. The purpose of evaluating thirty-six change request forms from different organizations is to present the change request criteria covered in the forms. The research method used for this study is qualitative. Observation is the preferred strategy for data collection, as it provides a good understanding that real-world data can be used as a basis of discussion and possible inclusion in the proposed framework.

Data in table 1 is based on the thirty-six change request forms, gathered from government and non-government organizations, ranging from different sizes of projects from small to large. To reduce data conflict validity threat, we focused on one vital concern, and that is project and company characteristics; our approach is to choose the same type of project from the same industry sectors of government and non-government. Projects are based on telecommunication systems, billing systems and transportation systems. Three change request forms from each of the two telecommunication systems are reviewed, similarly for the billing systems and transportation systems.

Evaluating change request criteria is very subjective because not all change request forms are the same. Some change request forms consist of open-ended questions, while others consist of closed questions. This study aims not at criticizing change request forms, but rather at suggesting ways to improve the change request forms. Not every change request form received for this study is acceptable, because each form must satisfy the following terms and conditions: 1) Every IT change request form submitted to us must be in English, and 2) change request forms may be from either government or non-government organizations. Change requests were found during the following three broad stages: 1) pre-maintenance, 2) maintenance and 3) post-maintenance (see table 1, note: PL: Project level and RL: Requirement level).

In the context of pre-maintenance, requirement changes normally refer to issues related to environment in the non-production stage. Particularly, requirements changes will occur during the later phases of the software development lifecycle, i.e. design, coding, development and testing, after the pre-release functional specification has been formally signed off. In the context of software testing, a requirements change is regarded either as an error correction or non-error correction. Both types of requirements changes are noted as either urgent or non-urgent. They can also be trivial or non-trivial [22]. An example of a trivial requirements change is one that does not affect overall system performance. For example, a requirements change in relation to a spelling correction. On the other hand, a non-trivial requirements change is one that disrupts and affects overall system performance. In this case, adding new functionality related to business value.

The types of requirements changes reported in change request forms found in the maintenance and post-maintenance environments present a major contrast to pre-maintenance requirements changes. One absolute challenge is that requirements changes under maintenance and post-maintenance often seem complicated but rework effort to implement the changes is critical as a response to tactical decisions that relate to system integration, new configuration files, new technology interfaces and backend databases integration.

However, in the post-maintenance environment, requirements changes reported by users are focused more on adding new functionality to the system (this change is not related to development of new source code from a programming language perspective but rather related to technology interfaces, and unfamiliarity with other new technology platforms)

The separation of government and non-government projects is to check whether the criteria in government change request forms are any different from non-government forms. For example, is the design of government change request forms better in the context of use and do they provide good content for IT practitioners' understanding? Are they informative enough to be used as a basis for estimating project schedule, cost and risk? Which kinds of information are most valuable and important to IT practitioners?

In table 1, our projects are categorized into government and non-government criteria. While project sizes are classified into three types: small, medium and large. Distinguishing projects type and size are important because we are interested in finding out 1) if the project size matters with respect to understanding the effect of requirement changes? and 2) what effect does a change of requirements, in each project type, have at the project level and requirements-level? Hence, categorizing project size is

one way of assessing the threats to validity of the data. Project size is determined by the number of people employed in the software development team and the cost of the project. For instance, small project have ten software developers in the team with a project cost less than five hundred thousand dollars ($500K), medium sized projects have a team size of more than ten people but less than fifty, and its cost is more than five hundred thousand dollars ($500K) but less than one million (1M); big projects are classified as more than fifty software development team members with a cost greater than one million dollars (1M).

Table 1. Matrix for evaluating change request criteria on IT change request forms

CR Criteria			Pre-maintenance						Maintenance						Post Maintenance					
Project A: Government Project B: Non-Government			A1			B1			A2			B2			A3			B3		
Project Size (2 CRS for each size)			S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L
Section I Description of a requirement change																				
Description of a RCS (title, type, rationale, scope)			*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Description of Requirement attributes and characteristics			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Section 2 Impact Assessment																				
PL and RL	Impact of making a requirement change	Scope	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
		Schedule	*	*	*	-	-	*	*	*	*	-	-	-	*	*	*	-	-	*
		Effort	*	*	*	-	-	*	*	*	*	-	-	-	*	*	*	-	-	*
		Risk	*	*	*	-	-	*	*	*	*	-	-	-	*	*	*	-	-	*
		Business value	*	*	*	-	-	*	-	-	-	-	-	-	-	-	-	-	-	*
		Project value	*	*	*	-	-	*	-	-	-	-	-	-	-	-	-	-	-	*
		Requirement Value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Section 3 Effort Estimation																				
PL	Estimate Start time	Lifecycle Stage	*	*	*	-	-	-	*	*	*	-	-	-	*	*	*	-	-	-
	Estimate finish time		*	*	*	-	-	-	*	*	*	-	-	-	*	*	*	-	-	-
RL	Estimate effort	New requirement	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		Existing requirement	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Examining IT change request forms is the first step in designing a framework. The aim is to discover, a) which change request criteria in change request forms are informative enough to allow IT practitioners to understand the requirements changes; b) what change request criteria are not explicitly addressed when producing IT change request forms; and c) which specific change request criteria (presented in IT change request forms) are relevant as a basis on which to effectively estimate project duration when the impact of the change is both at the requirements and project level.

Governments wish to streamline their processes to provide effective communication by having their policy, procedures and forms online and available to the public. Government projects we reviewed are mainly developed with internal software development teams – usually called ‘in house software development’ and rarely outsource their projects.

Data shown in table 1 of large government projects under the pre-maintenance stage focus on an impact assessment of cost, risk, time and business value when they are compared with large non-government projects. A never neglected variable in CR

form is scope. All requirements are documented in order to help project managers and users identify what is required to bring about project objectives. There are two types of scope, business scope and technology scope. The CR forms submitted by large non-government projects do not indicate that impact on cost, time and business value is important because 1) the projects have short duration times, 2) projects are not newly developed, and 3) the project sponsor is not really interested in the breakdown of project cost. Because large government-based projects that require audit checks criteria on CR forms are structured to give a detailed breakdown of project cost, time and business value.

Of the thirty-six change requests forms presented in table 1, interestingly, two levels of hierarchy are observed. One includes non-production (equivalent to SDLC stage of pre-maintenance) change request forms concerned with the impact on a project, particularly, effort, functionality, scheduling, risks, cost and overall quality. The second includes the production (equivalent to SDLC defined stage of maintenance and post-maintenance) change request forms, with non-standardized concerns on the impact on the business; consequences of not implementing a particular change, testing procedures, risks associated with change and business priorities etc.

Risk of requirements changes for government projects is perceived as more important than for non-government projects, based on the change requests forms we collected. Government change request forms show well-structured risk criteria that address impact assessment and effort estimation at the project level. This could be due to the fact that the governing style is bureaucratic, and software project management must be highly organized and controlled. No project should be fallible.

Of all the criteria found in change request with respect to understanding requirements changes, and its impact assessment in relation to effort estimation, there is no enough information for IT estimators to help them estimate rework effort.

At the project level, scope, effort, cost and time are the criteria used for determining overall impact. Conversely, at the requirements-level, attributes and characteristics are the criteria to assess the nature of a requirement for its completeness.

The first section of the table outlines what a requirement change is. Of the thirty-six change requests we observed, the criteria used for describing a requirements change is focused at a project level only, and not at the requirements-level. The description of a requirements change is very limited in helping an IT estimator or change management team understand how much time is really needed for rework when implement a requirements change. Ideally speaking, the depth of knowledge and understanding of a requirements change must be beyond just an understanding of its rationality, that is, what constitutes a bad or poor requirement and its associated attributes and characteristics. None of the forms we checked asked for information identifying other requirements characteristics or attributes regarding what would be needed for and understanding of its impact on other requirements, particularly during 1) pre-maintenance, 2) maintenance and 3) post-maintenance.

By highlighting the requirements attributes and characteristics criteria, it assists in making changes understandable to IT estimators and change management teams. This enables them to appreciate the added value of information with respect to effort estimation for new requirements and rework on existing requirements.

The second section of the change requests addresses impact assessment for requirement changes that is also directed only at the project level instead of the requirements level. IT practitioners of twenty-seven government projects of varying

sizes, and nine large-scale non-government projects did not consider or appear interested in knowing the outcome of the implemented requirements changes, whether they were of value to businesses, projects or requirements. One reason for project failure is rushing to reach a deadline without considering the full value of a project or the actual requirements for the overall business.

The third section examines specific change request criteria that are relevant to change management teams, IT practitioners, project managers, and users, in order for them to effectively schedule project duration when change impact is at both requirements level and project level. Of the thirty-six change requests forms we examined, only four of the twelve for pre-maintenance change requests dealt with "estimated effort in hours" and "estimate hours required to implement a change." The remainder of maintenance IT change request forms have a detailed breakdown for estimated effort and actual time required for the software development stages. However, software development effort can be classified into two main types - new effort for new requirements and effort for rework on existing requirements. None of the change request forms that we saw include this information. Estimated project schedule inaccuracy can arise when effort for new requirements and effort for the rework of existing requirements have not been taken into account. Estimating hours of effort for new requirements should be added on to the original estimates, while rework time for existing requirements should be included in the existing project schedule. The proposed solution is to incorporate these two factors into change request forms, so that IT practitioners can use this information as a base when revising project schedules and thus avoiding over or under estimating projects.

4 Proposed Framework

Our approach to developing a framework for estimating person-effort for requirements changes consists of three stages (see figure 1). The first is to identify the different kinds of requirements changes listed on the change request forms and analyze the causes of the requirements changes; this can help us better understand the reasons for the changes. The second stage requires us to distinguish between the different types of changes and allows us to classify them both horizontally and vertically. The third stage includes considering requirement changes as one of the cost drivers in COCOMO 2.0 and enhancing the model.

The rationale for the development of this framework is two fold. Firstly, we wish to improve the current process of reviewing and approving requirements changes by change management committees. Secondly, we wish to assist project managers in better planning for requirements changes through estimating more accurately the person effort required for these changes.

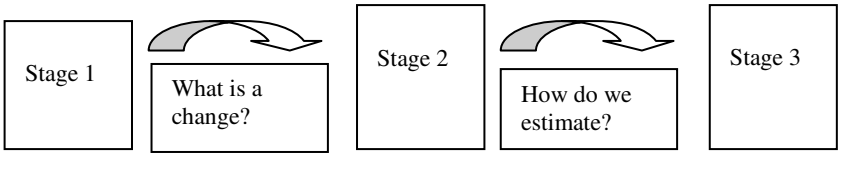


Fig. 1. Proposed framework for estimating person effort on requirements changes

The objectives for the development of our framework are to 1) improve the level of accuracy of schedule estimation with respect to software development effort when there are requirements changes, 2) to increase an understanding of requirements changes by incorporating specific criteria in change request forms that are not widely discussed in industry, 3) to integrate a category of requirement changes into parametric cost estimation models, 4) to understand from case studies of real world project failure why the project failed rather than focus on requirements changes or inadequate estimation, 5) to provide detailed explanation of specific reason for the project failure and to use this information to prevent from future failure. This framework should enable us to more accurately estimate the person-effort required to implement a proposed change. In addition, the framework may help reduce rework through analyzing patterns of requirements changes that assist in terms of assessing individual and group person-effort for each requirements change. We believe that estimating person effort through understanding the requirements change process and using variables that have a direct effect on changes, may be a more effective way of estimating requirements change effort rather than simply using effort estimation models developed mainly for initial schedule and effort estimation for a proposed software system.

Note that qualitative and quantitative types of data such as requirements changes (past or present) and effort for rework were collected from newly developed software development projects as well as existing ones. These data were considered as inputs to our framework on stage one. It was important to gain direct access to one change management database from the software industry that deals with pre-maintenance, maintenance, post maintenance changes to help us validate our framework in stage 2.

5 Conclusions and Future Work

The main contribution of this paper is not to propose new criteria to be applied in change request forms but rather to emphasize the risk of not gathering enough data for developing an effective cost estimation model. If the stream of requirement changes is not resolved through software process improvement, cost and effort estimation models become not very effective when estimating rework effort without taking into account requirement changes categories integrated into cost estimation models. Prior to designing our framework, the inclusion of new criteria in change requests would be beneficial to IT practitioners and IT estimators in increasing the turnaround time for rework effort estimation.

The next step of our research is to propose set of criteria to incorporate into change request forms to companies interested in including them in their IT change request forms so that requirements changes can be accurately estimated. Meanwhile, the framework developed will be tested for validation. Ideally, we will validate the framework through large and small industrial cases, in before production, during production and after production environments. This will give us the opportunity to calibrate a cost estimation driver with real-world data.

References

1. Chua, B.B., Verner, J.M.: Estimation of rework effort based on requirements categorization. In: 5th Proceeding of Software Measurement European Forum (SMEF) (in press, 2008)
2. Chua, B.B., Verner, J.M.: A Framework for predicting Person Effort on Requirements Changes. In: Proceeding of 5th International Conference on Software Methodologies Tools and Techniques (2006)
3. Firesmith, D.: Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them., Software Engineering Institute. Journal of Object Technology 6(1) (January-February 2007)
4. Gopal, A., Mukhopadhyay, T., Krishnan, M.S.: The Role of Software Processes and Communication in Offshore Software Development. Communications of the ACM 45(4) (2002)
5. Bianchi, A., Caivano, D., Lanubile, F., Rago, F., Visaggio, G.: An Empirical Study of distributed software maintenance. In: Proceeding of International Software Maintenance (IEEE) (2002)
6. Wiegers, K.E.: Lessons from Software Work Effort Metrics. Software Development (1994), http://www.processimpact.com/articles/metrics.pdf
7. Boehm, B.W., Clark, B., Horowitz, C., Westland, R., Madachy, S., Selby, R.: Cost Models for Futre Software Life-Cycle Processes: COCOMO 2.0. Annals of Software Engineering Special Volume on Software Process and Product Measurement, Vol. 1, pp 45-60 (1995)
8. Shooman, M.L., Bolsky, M.: Types, Distribution, and Test and Correction Times For Programming Errors. In: Proceedings of the International conference on reliable software, Los Angeles, California, USA (1975)
9. Bell, T.E., Thayer, T.A.: Software Requirements: Are They Really a Problem? In: Proceedings of 2nd IEEE Iinternational confernce on software engineering, San Francisco, California,United States (1976)
10. Basili, V.R., Weiss, M.: Evaluation of a Software Requirements Document By Analysis of Change Data. In: Proceedings of the 5th International conference on Software Engineering, San Diego, Califonia, United States (1981)
11. Basili, V.R., Perricone, B.T.: Software Errors and Complexity: An Empirical Investigation. In: Proceedings of Communciations of the ACM, vol. 27(1), pp. 42–52 (1984)
12. Stark, G.E., Skillicorn, A., Oman, P.O., Ameele, R.: An examination of the Effects of Requirements Changes on Software Releases. Journal of Software Maintenance: ReseaRCsh and Practice 5(11), 293–309 (1999)
13. Stark, G.E.: Measurements for Managing Software Maintenance. In: Proceedings of the International conference on Software Maintenance, Monterey, California, pp. 152–161 (1996)
14. Yashwant, K., Denton, M.J.: Requirements Volatility and Defect Density. In: Proceedings of the 10th International Symposium on Software Reliability Engineering, SA (1999)
15. Melonfire: Five common errors in requirements analysis (and how to avoid them)
16. Yardley, D.: Successful IT Project Delivery. Addison-Wesley, Reading (2002)
17. Standish Group : The Standish Group Report, Chaos (2006), http://www.cs.nmt.edu/~cs328/reading/Standish.pdf
18. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. John Wiley and Sons. Inc., New York (1997)

19. Tran, X.: Improving the Recognition and Correction of Poor Requirements. In: Proceedings of the Systems Engineering, Test & Evaluation Conference, SETE 2005 – A Decade of Growth and Beyond, Brisbane, Queensland (2005)
20. Kasser, J.E.: A Prototype Tool for Improving the Wording of Requirements. In: Proceedings of the 12th International Symposium of INCOSE, Las Vegas, NV (2002)
21. Chulani, S.D.: Incorporating Bayesian Analysis to Improve the Accuracy of COCOMO II and Its Quality Model Extension, P.h.D. Qualifying Exam Report, University of Southern California, USA (1998)
22. Boehm, B.W.: Software Engineering Economics. Prentice Hall, Englewood Cliffs (1981)
23. Albrecht, A.: Measuring Application Development Productivity. In: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83–92 (1979)
24. Putman, L.H.: A general empirical solution to the maCRsosoftware sizing and estimating problem. Proceeding of IEEE Transactions on Software Engineering 4, 345–361 (1978)
25. Fenton, N., Plfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach. PWS Publishing Company (1975)
26. Verner, J.M., Cerpa, N.: Software Project Failure. Accepted by CACM, Communication of ACM (in press, 2008)
27. Chua, B.B., Verner, J.M.: IT Practitioner's Perspective on Australian Risk Management Practices and Tools For Software Development Projects: A Pilot Study. In: The 4th International Conference proceeding on Software Methodologies, Tools, and Techniques (SoMET) (2004)

Productivity of Pair Programming in a Distributed Environment – Results from Two Controlled Case Studies

Sami Pietinen, Vesa Tenhunen, and Markku Tukiainen

University of Joensuu, Department of Computer Science and Statistics,
PL 11, 80110 Joensuu, Finland
{Sami.Pietinen,Vesa.Tenhunen
Markku.Tukiainen}@cs.joensuu.fi
http://www.joensuu.fi/tkt/english/

Abstract. Several methods and techniques have surfaced to address the ongoing concerns of quality and productivity of software development. Among these is the Pair Programming (PP) method, which has gained a lot off attention through being an essential part of an agile software development methodology called the eXtreme Programming (XP). In this paper, we present the results of two controlled case studies that investigate the possible productivity improvement through the incorporation of PP over solo programming. The main focus is on implementation task, more specifically in programming, although PP is suitable for other tasks too. Our results show that very high level of PP use might be difficult to achieve in a very tightly scheduled software development project, but some of the benefits can be seen to come true even with proportional use of PP. In our case, PP added the additional effort of 13% over solo programming.

Keywords: Pair programming, virtual teams, productivity.

1 Introduction

Experimental methods have been used in most of the Pair Programming (PP) studies. It might be that all of the advantages or disadvantages of PP are not visible in those kinds of controlled, but simplified settings with short programming tasks, so we decided to use a controlled case study method to study quantitatively and qualitatively the productivity of PP.

The present paper describes a research in which a complex cognitive environment is set up for two two-month-long software development projects where two programmers worked as a pair within a larger, geographically distributed project team. The development was done using a single computer display screen, but also a second workspace was provided for an opportunity to work individually if needed. This paper's main contribution is to the empirical knowledge of PP productivity.

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 47–58, 2008.

2 Research Background

2.1 SoPro Project

This research was conducted during a project called SoPro (Software Productivity Increase). The general aim of the project was to increase software engineering productivity. The project was carried out in three phases. First, an assessment of the software development processes of the participating companies was conducted, and then the project continued by identifying problems and needs in implementation phase. Finally, we aimed at creating a pair programming method as one of the possible solutions.

A combination of agile development process together with Pair Programming is regarded as a cost-effective method [25]. The current multinational software development often happens in geographically distributed environments. In the SoPro research project, we planned to enhance PP to suit it for the geographically distributed project teams. To this end, the method has been, and will further be, developed and piloted in real software projects.

At the moment, there is not enough knowledge available about the aspects of distributed PP processes. We therefore implemented two pilot cases that were focused on the data collection aspects of the study of PP process. It was done as a joint project with University of Joensuu's Department of Computer Science and Statistics and VTT Technical Research Centre of Finland in Oulu. Only the product parts that were produced in Joensuu's end are under evaluation.

2.2 Pair Programming

Pair programming is a method where two persons work together with an algorithm, design or programming task using one computer [31]. According to the literature, PP is said to yield several benefits: code has less errors, designs are simpler, problem solving is more efficient, co-operation and communication increases, productivity is better, and integrating newcomers into teamwork is faster [32,25,29].

Still, there have been only few empirical PP studies, and their results have been controversial about the claimed benefits; in particular, the alleged better quality and productivity has been questioned [12]. On the other hand, some of the claims have been scientifically confirmed, including the better problem solving and learning. However, PP does not benefit all software development tasks [29]. It is not clear, for example, what phases of software development would better be conducted by a single programmer, or a pair, or a team of programmers.

2.3 Productivity

The primary reason that software productivity is such a critical problem is that the demand for new software is increasing faster than our ability to supply it. Increased demand for software is the result of pressure throughout the economy to improve commercial, industrial, and service productivity [7]. With the recent

push to downsize or outsource, most of us are trying to look for ways to cut down our software costs. Today, the majority of improvement strategies being pursued try to either reduce the inputs (i.e. people, time, equipment) needed to do the job, or increase the outputs generated per unit of input [22].

In addition to changing programming related activities in search of productivity improvement, nonprogramming related options can be used. According to Boehm, all the factors influencing software productivity is not under our control, but one of the primary controllable factors is the number of instructions we choose not to develop, either by deferring the development of marginally useful features or using already available software like commercial software packages, adaptation of existing in-house software or program generators [7].

No indications towards the superior productivity of PP over solo programming could be detected based on a multiple case study in [12]. PP effort levels reported in various studies have been from -28% to 100% [2,4,9,16,18,19,23,27,32].

3 Research Design

3.1 Research Method

Two controlled case studies were made in order to study the productivity of PP. By using controlled case study method, the resulting information is applicable in a wider context compared to using general case study method in academic settings. More elaborate description of the method can be found from [24]. Using this method, the development of a software product is justified, because it produces industrially useful end-products in a acceptable time frame.

3.2 Data Collection

Several protocols were recorded in order to be able to accurately measure and analyze multiple aspects of the development process and product. An overall video with sound was recorded for being able to afterwards find out, what really happened in the development environment as a whole. Video with sound from monitors were also recorded to see, what task was done when and how much time was spend on each task. Task effort was also recorded by the used software development process, but in order to make more fine-grained analyses, both of the developers also kept a diary about ongoing activities and those were checked against the recorded monitor and overall video. Notes were written in releases to record the number of issues found.

3.3 Research Setting

Before the empirical part of this study was started, all the subjects were trained to Mobile-D, an agile software process. It has been development by our partner in this study and been used for several years in the ITC industry. Iterations in Mobile-D consist of three types of days: planning day, working days and release

day. The day length is six hours and there is only four days in a week that are used for development, others are either free of work or used in planning or release. Mobile-D approach is based on Extreme Programming, Crystal methodologies and Rational Unified Process [1].

First project used PHP and HTML as an implementation language and the second project was done using Java2ME platform. The use of different implementation languages is a result of the fact that both of the projects were real world projects with real business needs and pressure. The main tool used in both of the projects was Eclipse, an integrated development environment.[1]

3.4 Projects and Developers

The number of developers at Joensuu was two in both projects. During the first project, Joensuu's programmers developed an application for querying from and storing data in a database with a mobile device's browser and were also involved with respective web application implementation. The project team consisted of six persons in two geographically separated locations, so the team should be called a virtual team. The second project consisted of seven persons with same geographic distribution, where Joensuu's developers build an mobile database centric client-server application, but this project is used as solo programming reference for the low amount of PP used. The low amount of PP use was result from the client-server division, where dedication to different sides were formed between developers, and because of the high business pressure, it was very difficult to detach from own work for pairing.

Developers programming experience has an effect to the generalizability of this study. First developer had about 1.5 years of programming experience in multiple academic software projects and the second had 6 months of academic and 2 years of industrial programming experience.

4 Productivity Factors

The traditional productivity metrics, like dividing LOCs or Function Points by person-months, have apparent limitations. They disregard inherent variety in both software projects and software developers and they are especially ill-suited for estimating the outcome of future projects [21]. Therefore, both the LOC and Function Point counts are used for measuring productivity. Only information of past development efforts is needed at this case and also the developers under investigation has remained the same. According to Brooks [8], the number of statements that programmer can write in fixed time is constant between different implementation languages. It is also still unknown if the programming paradigm, i.e. object oriented paradigm vs. procedural programming, has some kind of effect on productivity.

It should be noted that in the long run, PP method is probably more productive than in the first project, because of the pair jelling time [30], but as a

[1] More info available at http://www.eclipse.org/

balancing factor, we used PP also in second project about 6.1% off the development effort. To which way this has an effect of advantage (solo or PP), is just the matter of evaluating the required pair jelling time which might be from few hours to several days. Also learning to use PP very efficiently, in our experience, might take several days or even longer.

Software Process. Much of the process model's methods and techniques regarding implementation (i.e. Test Driven Development) were not used for being able to keep the two studies comparable and to be able to say in higher confidence that the effect under consideration is the result of using PP and not by some other factor. The task under investigation is mostly just coding, so the used software process should not raise a thread of concern.

Lines of Code. Counting lines of code (LOC) is probably the oldest way to count program size and it can be done in multiple ways. We could count only executable lines; executable lines and data definitions; executable lines, data definitions and comments; executable lines, data definitions, comments and job control language; lines as physical lines on an input screen; or lines as terminated by logical delimiters [21]. The best solution is the one that minimizes the effect of different coding styles between different developers and languages, so the number of executable lines of code (lines as physical lines on an input screen) and the comment lines was selected leaving out blanks. Same indentation, code block start character position and line length rules were used in both projects. Deviations from the coding standard, which contained mostly the positioning of block start markings and code line length exceeds, were afterwards corrected.

More lines of code produced in a constant time frame does not necessary mean better productivity. It is the functions or features delivered to customer that bring the real value, so we also used function points to size the end-product. We were not able to avoid the use of modifiers when counting the product size based on LOC, so we also present the analysis figures alternatively without any modifiers.

Function Points. Function points have been used to calculate and estimate program size for almost three decades. It was first created by Albrecht in 1979 [10]. The basic idea is that the function point value is a universal metric in which all different kinds of programs, irrespective of the implementation language and platform, can be compared with each other. According to Dreger [10], using function point analysis one can evaluate project size, cost and development time reliably within 10% error margin for existing systems and 15-20% error margin for systems in planning phase. Hours per function point is a unit-of-work measure that is industry-accepted [11], but not so much used as LOC. It measures the number of hours required to develop one function point and is that way directly linked to productivity. The distinct counting method used in this study is based on the book by Garmus and Herron ([11]), which uses International Function Point Users Group's (IFPUG) counting rules version 4.1.

Function point analysis (FPA) is based on counting data functions and transactional functions, more precisely, inputs, outputs, queries and files that are

used by the computer system. The data functions, internal logical files (ILFs) and external interface files (EIFs), relate to the logical data stored and available for update or retrieval (or both). The transactional functions, external inputs (EIs), external outputs (EOs), and external inquiries (EQs), perform the processes of data maintenance, retrieval, output, etc. Each has it's own unadjusted function point weight based on its unique complexity matrix and the weighting will give the number of Unadjusted Function Points (UFPs). Then we need to calculate the Value Adjustment Factor (VAF), which is based on identification of 14 General System Characteristics (GSCs). The 14 GSCs are totaled to calculate a Total Degree of Influence (TDI). The value adjustment factor (VAF) is calculated from equation (1):

$$VAF = (TDI * 0.01) + 0.65, \tag{1}$$

where the already given numerical values are constants. VAF adjusts the Unadjusted Function Point count by ±35% and produces the Adjusted Function Point (AFP) count [11]. There are several possible types of function point counts: development project counts, enhancement project counts and application counts. In this paper existing applications are sized, therefore the count is an application count resulting in AFPs.

Code Reuse. Code reuse has been proven to be a productivity increasing factor [7,15]. Even inside a one single project, one can reuse some parts of the modules, although this sometimes indicates that the duplicate like code parts should be generalized into a single code part. In our projects the reuse has not been in its pure form, because it is better described as usage of example code. These are not counted as real reuse and do not affect to the counting of code lines. The kind of code reuse, where most of the code is unchanged, will this case have more effect to the total number of lines of code produced. The effect of this kind of pure reuse in our project is handled using a modifier in which 1:3 of the reused code lines are included in the total number of lines of code. This is based on the case example in [14], where 25% reuse resulted in decrease of coding effort from 3 calendar months to 2.3 months. The amount of reused code compared to the total size of code base in first project is 13.3% and 9.6% in second project.

Debug Code. The number of produced debug code lines where significantly higher in Java based project (2nd). It is much easier to debug HTML based applications using a web browser that it was found to be with mobile Java applications. The debug code consists of print statements in both projects and this kind of statements are, in our experience in these projects, much faster to produce than other parts of the code on average. The difference in number of debug lines between the two projects has to be counted in with another modifier in which 1:3 of the debug code lines are included in the productivity model. There doesn't seem to be any previous studies about debug code effort so the given rough estimate needs to be used. The amount of debug code compared to the total size of code base (code + comments - blanks) in 2nd project is 6.1%, which is big enough for not giving us a change to ignore it. The respective value in first project is 1.7%.

Effort. According to [20], for most projects the biggest component of cost is effort. Counting effort has to be done in categorized way using the following classes: project management, it-support and development. Project management includes all the communication and other management activities that might be different between projects. IT-support is related to setting up and managing the development environment and it has to be also separated from the productivity count. Last, but the most important factor, is the development effort which contains all the effort marked into task cards.

The effort used for design documentation is also left out, because the requirements for documentation where different in each project, although very light documentation level was kept. Coarse UI mock-ups where drawn in both projects and other more detailed documents where produced as post-implementation, so in this sense, documentation did not affect differently to the actual implementation phases of the projects.

Although there where only 2 developers under investigation, it is very important to keep the effort the same for both developers in both projects, because the productivity between developers can vary by a factor of 10 [8,20,21]. The total effort needed per projects where different, but the ratio of each developer's effort compared to total effort is the same in both projects.

Quality. The quality of the software can be presented using for example the number of defect that remains in the end-product in subsequent releases and most importantly after the final release. This can be presented as defect density i.e. the number of defects identified across one or more phases of the development project lifecycle compared to per thousand lines of code. In the second project, in which solo programming was mostly used, the delivery rate of features was too low to make any comparison between the projects concerning subsequent releases but the final releases were comparable. There was no long term observation of defects after the final release. Couple of defects were also found when going through the code base when fixing coding standard deviations.

The amount of code reuse affects increasingly in the number of produced lines of code and to product quality by decreasing the defect density [15]. In this way, the code reuse could mask the effect of PP usage related to quality. This is not a factor, because reused code is excluded from the quality analysis. Amount of code commenting can also be considered as one indicator of quality.

5 Analyses

As can seen from table 1, solo programmers produced more code than pair programmers (+13%). Even the quality seems to be better, including number of defects relative to product size (1.6 vs. 1.16 defects/KLOC) and the amount of commenting relative to product size (14.8% vs. 24.6%).

The severity of defects where mostly cosmetic, only one major defect was recorded for each project. No meaningful difference in productivity calculated with function points where found. It might be the result of using high level features of the program to size it rather than module level analysis, which might

Table 1. Effort, Code, Quality and Function Point Analyses

	Proj1 (Solo)	**Proj2 (PP)**
Complexity	easy	easy
Effort analysis		
Task (hours)	245	275
Total (hours)	365	386
Velocity	67.1	71.2
Total Adjusted function point count (AFP)	73.7	83.7
Code analysis		
Lines of code (code, comments, blanks, reuse)	7509	10215
Reused lines of code (code)	991	660
Debug lines of code	127	621
Number of comment lines	1114	2516
Number of blank lines	842	1200
Lines of code (code, comments, fixed debug, fixed reuse)	5760	7353
2 person team LOC per hour	23.51	26.74
Quality analysis		
Defects after final release	7	5
Defects per KLOC	1,6382	1,1609
FPA analysis		
Total unadjusted function points (UFP)	81	92
Total degree of influence (TDI)	26	26
Value adjustment factor (VAF)	0.91	0.91
Total Adjusted function point count (AFP)	73.7	83.7
FPUs per hour	0,3008	0,3044

reveal the underlying complexity better. The high amount of commenting in Java-based project is at least partly the result of the code consisting higher number of functions than the first PHP/HTML-based project. The lack of commenting e.g. the need for commenting is more visible and obvious in the method interfaces than inside the methods, because of the Javadoc [2] style commenting, and therefore in our case, results in a higher amount of commenting than a code with fewer methods.

6 Results

Productivity. In our case the productivity of PP is lower compared to solo programmers. When the used software development process is more heavy than the one we used, difference of even 15% in productivity might be acceptable due to the additional review phase needed with solo programmers [6]. But with our partial use of PP, an additional review phase would also be needed, so roughly less than 10% would be acceptable and therefore the limit is overrunned.

The FPU analysis still leaves some hope, but the use of high level program features might lead into results that are not inside the expected accuracy of the

[2] More info available at http://java.sun.com/j2se/javadoc/

sizing method. There are still other benefits of using PP, like the increased learning, which might be considered highly relevant especially concerning tacit knowledge. But what comes to productivity, the results show clearly an negative effect.

Pair Programming Method. The developers enjoyed very much using PP. Particularly the following two factors might be easily disregarded at least in the beginning when pairs do not have very much experience of using the PP method. When one reaches a very deep level of concentration, it is hard to keep track of time. As the developers are working as a pair, the other partner might not be in a very excited mental state after couple of hours. There are many obvious reasons for getting tired and one of them is just the fact that there are individual differences in how much brainwork a person can do in one go without breaks. Pairs could even make some effort to monitor each others mental state of vigor to recognize the points where roles should be switched or it's time for a break. According to [29], breaks should be taken on hourly basis at the minimum.

Most of the bugs that was found during the coding, where typo errors, but other potentially more dangerous bugs was found too. There was an automatic syntax checking in the IDE, so typo errors were mostly catched on the fly by the controller anyway. As a consequence, the navigator should give enough time for the driver to correct the error on one's own initiative before mentioning it, or otherwise in the long run it will be very irritating for the driver.

Another factor noticed in our study, and also mentioned by [29], was the mental support provided by the partner seemed to have a positive effect on how efficiently developer starts to actually implementing something when the problem at hand seems difficult. In other words, there is less planning required to feel confident to proceed from planning to implementation, when you have the partner giving fresh ideas or confirmation on how to proceed.

As the use of PP was based on voluntariness, the effect of business pressure probably lowered the amount of PP use, which was considered the be the most influencing factor. In [26], ratio as low as ca. 10% was reported, but the problems there were related to organizing and infrastructure for PP. Our case the problems were more affective, the feeling of need for hurry and the prejudice of PP being less productive than solo programming.

6.1 Validity

Internal Validity. Both developers had prejudice against PP in the sense that it was not felt to be more productive than solo programming. This has an effect at least on how much PP is actually used over solo programming. As the first project was also the first time when the developers developed software together, the pair jelling time might have decreased the productivity a little. The productivity differences between the Java and HTML/PHP cannot be accurately quantified at the moment, but based on the FPU analysis, the combination of PHP and HTML in our case produced a function point with ratio of 9:10 compared to Java based lines of code.

External Validity. Threads concerning generalizability of this study include the experience of the developers. In general, developers working only in industrial software development companies have probably more experience in average than the developers in this study. The developed systems had low difficulty level, which improves the external validity when the developers are not experienced enough for ranking them as industrial developers. Another evident thread is the small team size in Joensuu, because with bigger team sizes, the PP method should be used differently i.e. by incorporation of pair rotation.

7 Discussion

The accommodation of PP into a company is not trivial, because many of the industrial managers do not believe PP methods superiority over solo programming. It also does not help that in PP studies the results have been contradictrary. Fortunately, most of the people that have tried the method, have liked it, although it is not for everyone. According to Beck [5], XP should be adopted one problem area at a time. So, if the problem at hand is related to product quality, re-enforcing the software process with continuous reviews by taking PP in use might be just the way to go. For example in [26], accommodation tactics of PP included: PP guidelines, PP champion, voluntariness, motivation and a separate PP room. PP can be excercised in a more structured way by using Collaborative Software Process (CSP) desribed in [30].

Another interesting area where to apply PP is the assimilation of new people to an on-going software project. Brooks says that adding people to a late project makes it even later [8]. There is very anecdotal evidence, that adding manpower to a late project will yield productivity gains to the team more quickly, if PP is used [28].

Some of the PP benefits, like knowledge sharing, could be still improved by consciously interacting more with other developers and discussing about possible problem solutions openly without the fear of critique. In our second project, which mostly consisted of solo programming, a great deal of information was still exhanged between the developers, due to the effect of PP in the first project, which clearly teached the developers to communicate better.

8 Conclusions and Future Work

PP seems to be a very good method for raising the workers confidence in their work, team building, improving communication, sharing of tacit knowledge and not to mention the feel-good factor. But there is a price in getting all these benefits and that is the increased effort needed to bring the project to enclosure. PP might not be productive enough for continuous use (used all the time) compared to solo work, but many of the benefits can be seen to come true also in partial use. Then again in our study, the proportion of effort, where PP was used, might not be big enough to see all the benefits of PP to come true.

When the use of PP is proportional, it has to be complemented with additional quality verification methods, because the preview process is otherwise incomplete. One of these methods is peer reviews. There is evidence that, peer reviews and the following error fixing task should be done in pairs and by the pair programmers, because according to [17], quality assurance (QA) made by pairs was over two times faster than solo QA, which might be for the reason of reviewing familiar code or otherwise inherent to PP.

Further analyses of the recorded data is still needed in order to find out, if there is a visible pattern in where the PP seems to be the most efficient. We are also investigating the possible productivity implications of PP based on eye movements that where recorded during this study.

References

1. Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O.: Mobile-D: an agile approach for mobile application development. In: OOPSLA 2004, Vancouver, Bc, Canada, pp. 174–175. ACM, New York (2004)
2. Arisholm, E., Gallis, H., Dyb, T., Sjraberg, D.: Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. IEEE Transactions on Software Engineering 33(2), 65–86 (2007)
3. Arisholm, E.: Design of a Controlled Experiment on Pair Programming. In: Proceedings, ISERN 2002 (2002)
4. Baheti, P., Gehringer, E., Stotts, D.: Exploring the Efficacy of Distributed Pair Programming. In: Proceedings, XP/Agile Universe 2002, New York, pp. 208–220. Springer, Heidelberg (2002)
5. Beck, K.: Extreme Programming Explained: Embrace change. Pearson Education, London (2000)
6. Boehm, B.W., Turner, R.: Balancing Agility and Discipline: A Guide for the Perplexed. Addison Wesley, Reading (2003)
7. Boehm, B.W.: Software Engineering Economics. 1st. Prentice Hall PTR, Englewood Cliffs (1981)
8. Brooks Jr., F.P.: The Mythical Man-Month: Essays on Software Engineering. Addison Wesley, Reading (1975)
9. Ciolkowski, M., Schlemmer, M.: Experiences with a Case Study on Pair Programming. In: Workshop on Empirical Studies in Software Engineering (2002)
10. Dreger, J.B.: Function Point Analysis. Prentice-Hall, Englewood Cliffs (1989)
11. Garmus, D., Herron, D.: Function Point Analysis: Measurement Practices for Successful Software Projects. Addison-Wesley, Boston (2001)
12. Hulkko, H., Abrahamsson, P.: A multiple case study on the impact of pair programming on product quality. In: Proceedings of the 27th international conference on Software engineering (ICSE), pp. 495–504 (2005)
13. Jensen, R.: A Pair Programming Experience. CrossTalk 16(3), 22–24 (2003)
14. Jones, C.: Programming Productivity. McGraw-Hill, New York (1986)
15. Lim, W.C.: Effects of Reuse on Quality, Productivity, and Economics. IEEE Software, 23–30 (September 1994)
16. Lui, K., Chan, K.: Pair programming productivity: Novice-novice vs. expert-expert. International Journal of Human-Computer Studies 64(9), 915–925 (2006)

17. Müller, M.M.: Are Reviews an Alternative to Pair Programming? Empirical Software Engineering, vol. 9, pp. 335–351. Kluwer Academic Publishers, Dordrecht (2004)
18. Nawrocki, J., Wojciechowski, A.: Experimental Evaluation of Pair Programming. In: Proceedings, ESCOM 2001, pp. 269–276 (2001)
19. Nosek, J.: The Case for Collaborative Programming. Communications of the ACM 41(3), 105–108 (1998)
20. Pfleeger, S.L.: Software Engineering: Theory and Practice, 2nd edn. Pearson Education, London (2001)
21. Putnam, L.H., Myres, W.: Five Core Metrics: The Intelligence Behind Successful Software Management. Dorset House, New York (2003)
22. Reifer, D.J.: Practical Software Reuse, 1st edn. John Wiley & Sons, Inc, Chichester (1997)
23. Rostaher, M., Hericko, M.: Tracking Test-First Pair Programming - An Experiment. In: Proceedings, XP/Agile Universe, pp. 174–184. Springer, New York (2002)
24. Salo, O., Abrahamsson, P.: Empirical Evaluation of Agile Software Development: A Controlled Case Study Approach. In: Bomarius, F., Iida, H. (eds.) PROFES 2004. LNCS, vol. 3009. Springer, Heidelberg (2004)
25. Succi, G., Marchesi, M.: Extreme Programming Examined. Pearson Education, London (2001)
26. Vanhanen, J., Lassenius, C., Mntyl, M.V.: Issues and Tactics when Adopting Pair Programming: A Longitudinal Case Study. In: Proceedings of the Second International Conference on Software Engineering Advances (ICSEA) (August 2007)
27. Vanhanen, J., Lassenius, C.: Effects of Pair Programming at the Development Team Level: An Experiment. In: Proceedings of International Symposium of Empirical Software Engineering (ISESE 2005) (2005)
28. Williams, L., Shukla, A., Anton, A.I.: An Initial Exploration of the Relationship Between Pair Programming and Brooks' Law. In: Proceedings of the Agile Development Conference, June 22 - 26, 2004, pp. 11–20. ADC. IEEE Computer Society, Washington (2004)
29. Williams, L., Kessler, R.: Pair Programming Illuminated. Pearson Education, London (2003)
30. Williams, L.A.: The collaborative software process. PhD Dissertation. Department of Computer Science. Salt Lake City. University of Utah (2000)
31. Williams, L., Kessler, R.: All i really need to know about pair programming i learned in kindergarten. Communications of the ACM 43, 108–114 (2000)
32. Williams, L., Kessler, R., Cunningham, W., Jeffries, R.: Strengthening the case for pair programming. IEEE Software 17, 19–25 (2000)

Software Functional Size: For Cost Estimation and More

Baris Ozkan*, Oktay Turetken, and Onur Demirors

Informatics Institute, Middle East Technical University
06531, Ankara, Turkey
{bozkan,oktay,demirors}@ii.metu.edu.tr

Abstract. Determining software characteristics that will effectively support project planning, execution, monitoring and closure remains to be one of the prevalent challenges software project managers face. Functional size measures were introduced to quantify one of the primary characteristics of software. Although functional size measurement methods have not been without criticisms, they have significant promises for software project management. In this paper, we explore the contributions of functional size measurement to project management. We identified diverse uses of functional size by performing a literature survey and investigating how functional size measurement can be incorporated into project management practices by mapping the uses of functional size to the knowledge areas defined in project management body of knowledge (PMBOK).

Keywords: Software Project Management, Functional Size Measurement.

1 Introduction

Software project managers require knowledge on software product for effective management. Size is one of the key attributes for engineering projects and size measurement supports project management in many processes such as scope determination, cost and duration estimation, performance and quality measurement, and contract management. Software size has been associated with several attributes of software artifacts, documents and deliverables and software development practitioners have measured size using a wide range of metrics and methods. Fenton explains software size as a multidimensional attribute and describes it with the dimensions of length, functionality and complexity [1].

Unlike other engineering disciplines in which size is measured from the project initiation to the closure phase and measurements are effectively used for different purposes each project management process have, software size measurement has been mostly practiced in software cost and effort estimation and remained unfruitful in other project management processes.

* This study is supported by The Scientific and Technological Research Council of Turkey.

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 59–69, 2008.

Among the various approaches developed to software size measurement, the measures and methods on quantifying the 'functionality' attribute have been widely accepted in practice. Software functional size measures the amount of functionality provided to the users. Functional Size Measurement (FSM) methods are utilized frequently in estimation of effort and cost for software development and maintenance projects. The need for well established estimation models is so imperative that the relation between FSM and software cost, effort and time estimation can easily cause the misinterpretation of FSM methods as estimation models. However, software project management essentially requires reliable size measurements for many other purposes including controlling and monitoring project scope and risks, assessing process performance and establishing organizational level standard measures.

In this study, we focused on the software functional size and explore its usability in various software project management practices. We investigated how methods and approaches based on the functional size can address the concerns and requirements of software project management, by going through the specific process knowledge areas (KAs) defined in the project management body of knowledge (PMBOK) [2] and by reviewing the literature for studies that specifically utilize FSM methods for various project management tasks. We selected the PMBOK to identify the project management tasks, as it is a well-known project management guide that provides process definitions along with inputs, outputs, relevant tools and techniques and the presentation of the interactions between processes. This representation and structure eased the identification of potential uses of functional size measures. In mapping the process KAs to FSM uses, process descriptions in each KA are reviewed and relevant applications of FSM in the industry and research studies are surveyed.

The remainder of the paper is structured as follows: Section 2 discusses the related research on the functional size measures, related measurement methods, and the uses of functional size in project management processes. Section 3 investigates how project management KAs in PMBOK can be supported with functional size measures. Section 4 concludes by presenting the uses of functional size that span across the project management processes.

2 Related Research

The research study involved the identification of potential uses of FSM methods in software project management practices. For this purpose, we have used PMBOK as a reference that is representative of the state of the art collection of project management practices and combines them in a well-defined structure. In the exploration of the uses of FSM, we selected three well known ISO/IEC conformant FSM methods, Mk II FPA [3], IFPUG FPA [4] and COSMIC FSM [5].

Project Management Body of Knowledge (PMBOK), published by Project Management Institute, is an internationally recognized collection of processes and KAs often accepted as a best practices reference. It is supported by an international standard, IEEE 1490-2003, published by the IEEE Standards Association [6]. The knowledge in PMBOK is organized into five project management process groups; initiating, planning, executing, controlling and monitoring, and closing. The processes are categorized into nine KAs; project integration management, project scope management, project time management, project cost management, project quality

management, project human resource management, project communications management, project risk management, and project procurement management. PMBOK describes each process in three sections; the process inputs, outputs, and tools & techniques applied for the process. The knowledge area processes interact with each other and processes from other KAs directly or indirectly.

Function Point measure has gained considerable interest since it has been first introduced by Allan Albrecht in 1979 [7]. The motivation was to quantify the software functional requirements independent from the development method, people and the implementation language. Later, Albrecht and Gaffney improved this method [8]. During the 1980s and 1990s, several new counting methods were developed that intended to improve the original FPA or extend its domain of application.

In 1986, the International Function Point Users' Group (IFPUG) was set up as the design authority of Albrecht's FPA method. Since then, IFPUG has been clarifying FP counting rules and expanded the original description of Albrecht.

Mk II FPA method was developed by Charles Symons in 1988 to solve the shortcomings of the regular FPA method [3]. Currently, the Metrics Practices Committee (MPC) of the UK Software Metrics Association is the design authority of the method.

COSMIC FSM method was published by Common Software Measurement International Consortium (COSMIC) in November 1999 [5]. This group has been established to develop this new method as the one which would measure the functional size of software for not only business information systems, but real-time systems and hybrids of both.

Due to the proliferation of the methods, a workgroup was initiated by International Organization for Standardization (ISO) in 1996, with the purposes of identifying the fundamental concepts of a measurement process, clarifying the conceptual basis and establishing an international standard for functional size measurement. ISO/IEC joint committee first published ISO/IEC 14143-1:1998 International Standard, which defines the fundamental concepts of FSM methods [10]. This standard defines the core concepts of FSM, such as functional user requirements (FURs)[1], Functional Size[2], Base Functional Component (BFC[3]) and the FSM requirements that should be met by a candidate FSM method. In the following years, other ISO/IEC standards were published [11]-[15].

Currently, MkII FPA [3], IFPUG FPA [4], COSMIC FSM [5] and NESMA FSM [16] are accepted as international standards for functional size measurement. All these FSM methods measure the functional size of a software product; however, they use different metrics and rules during the measurement process [9].

The uses of FSM in software project management have been studied by other researchers. These studies focus mainly on estimation processes emphasizing cost and effort prediction for a software project [17], [18], [25], [26] ,[28], [32]. Few of these studies exploit FSM in other project management processes. Muller in [19], describes how function point metrics can be adopted to earned value technique used in cost control process. Dekkers [20] explores use of FSM in IT Governance processes.

[1] FURs: "a sub-set of the user requirements. The FURs represent the user practices and procedures that the software must perform to fulfill the users' needs".

[2] Functional Size: "a size of the software derived by quantifying the FUR".

[3] BFC: "an elementary unit of FUR defined by and used by an FSM Method for measurement purposes".

Abran, Meli, Buglione [21] discusses how the use of FSM can strengthen an ICT Balanced Scorecard (BSC). They show how BSC, as a performance measurement driven by business strategies, is supported by FSM through establishing normalized performance indicators. Rispens and Vogelezang [22] present a case study demonstrating the use of functional size in application portfolio management of a bank. ISO/IEC 14143-1 [10] includes an informative annex section about the uses of FSM in project management. In [23], Symons addresses various aspects where functional size can contribute, including scope and procurement management.

3 Uses of Functional Size Measures in Project Management Knowledge Areas

We explored the use of functional size measure for each applicable project management knowledge area (KA). Each KA requires different techniques, skills and expertise, and constitutes a distinguishable aspect of project management practices such as cost, quality and scope management. In the mapping, we also discuss how functional size relates to the techniques and practices used in software project management.

We identified seven of the KAs defined in PMBOK, where functional size offers direct opportunities to contribute. In the exploration of uses, the capabilities of the three ISO compliant FSM methods, MkII FPA, IFPUG FPA, COSMIC FSM are taken into consideration and the findings about the extend of each method concerning the capabilities are identified.

Project Integration Management. Project integration is primarily concerned with integration and harmonization of processes among the project management process groups. Functional size can be used to estimate preliminary scope, cost, and schedule. The applicability of functional size at the early stages of a project can enable the project managers and sponsors with quantitative analysis capability on feasibility and project selection studies.

Portfolio management evaluates the individual projects and their relations to maximize the benefits and ensure alignment with strategic objectives of the organization. Functional size measures can support portfolio management by providing the functional size for the portfolio; hence functional size of completed, maintenance, ongoing and potential projects. Quantifying project's functional size, the portfolio or program managers can compare projects with each other, give investment and project selection decisions based on the functional size to be developed or to be delivered for each software project. Functional size can support the definition of quantitative criteria and measurement against the criteria when selecting among candidate projects. In risk/reward ratio calculations, reward for each project can include the functionality offered by the software that will be developed. Similarly, estimated costs based on functional size can be included in risk calculations. Functional size can help the allocation of project costs to lines of business by measuring the software functionality delivered to each line. Measures, such as cost per functional size unit can be used to value software assets.

Monitoring and controlling the project work essentially needs measurement capabilities and performance information. Using functional size, project work in each phase can be continuously monitored against the functional size implemented and the

remaining size planned to be developed. Depending on this performance rate, estimation of project cost, scope and schedule can be performed to update forecasts. In [19], Muller explains the adoption of Earned Value technique, where values are calculated in terms of earned software functional size units as an alternative to value approximations to costs.

FSM methods size the software project at BFC level and integrated change control process can utilize this to evaluate the impact of a change that can be represented in BFC granularity level to the scope, cost and schedule. Automated tools can help to collect performance data.

Scope Management. FSM methods supports the breakdown of project work into work packages such that the FURs of a software project that are included in work packages can be measured individually. Among the three FSM methods, COSMIC FFP is fully scalable, thus functional size of FURs can be aggregated to get the total size. In scope definition, the set of BFCs that are included in the initial scope can be measured as the baseline and can be used to estimate initial cost and schedule. Scope can be monitored by comparing the functional size at different development phases and product baselines. Changes in the scope, whether they are additions, removals or alterations for the available functionality, can be quantified independently. The impact of the change on project cost and schedule can be estimated, so that a change request can be evaluated and negotiated with the project sponsors and stakeholders on a quantitative basis.

Time Management. Development of a project schedule requires estimation of activity durations in work packages defined in project work breakdown structure (WBS). The decomposition of FURs into BFCs can be used for estimating the effort and duration for each activity. In activity resource estimation process, -using a parametric resource estimation technique- the effort required for each activity can be estimated over the functional size using productivity rates, which are based on the actual values for the functional size and the development effort for completed projects. Constructive Cost Model (COCOMO) [24] breaks the software development schedule down into pre-determined software life-cycle phases/activities, where functional size is taken as a primary size input to the model. The most recent version of the model uses only IFPUG function points.

Cost Management. Cost estimation is the process of developing the approximate costs of resources to complete the project activities. Effort has been the primary resource for software development projects. There are many researches in the literature that study the relationship between the software functional size and effort [17], [18], [24], [25]. Other studies discuss the productivity and the factors - called as cost drivers [24], [26] - that influences the size-effort relationship.

Software cost estimation techniques can be categorized into two; heuristic and parametric approaches. Either a bottom-up or top-down method can be selected when applying these techniques [1], [33]. Bottom-up methods estimate each individual component and combine all components to give the overall, complete estimation of the project. On the other hand, top-down approaches estimate the size of the project as a whole, considering the overall characteristics of the system to be developed. When using functional size in cost estimation, bottom-up estimation needs detailed WBS

and FURs to come up with accurate estimations. Top-down approach can be applied at early phases of a project, where software requirements are not detailed. Among the techniques defined in PMBOK Cost estimation process, functional size can support cost-rate based and parametric estimations. Functional size based organizational or benchmark productivity rates from international databases such as The International Software Benchmarking Standards Group (ISBSG), can be used as cost rates to calculate resource cost per schedule activities [27]. ISBSG [28] works through software metrics groups in major countries of the world and defines standards for recording data about software projects, collects software project data, manages a software project database, and publishes periodic reports. The ISBSG repository is richer for benchmarking projects measured by IFPUG than relatively new methods such as COSMIC FSM and MK II FPA.

Regression models can be built with organizational project data that includes functional size, effort or other costs [29]. COCOMO [30], as a parametric and top-down cost model, takes functional size as a model input and converts it to source lines of code and then calculates the development cost based on these values. Other models that take functional size as input for effort and cost calculations are SEER SEM, Putnam's SLIM, Albrecht Gaffney, Kemerer, Matson-Barret-Meltichamp, [32], [33], [34], [29].

IFPUG and MkII FPA adjust the total functional size of a software product with weights given for non-functional characteristics of the software. The adjusted function points can be compared with past project data to estimate the cost more precisely [31].

Cost estimation can be performed throughout the project to update forecasts and increase accuracy as more detailed project information is available. FSM methods can be used to estimate the functional size starting from earlier project phases where software requirements are high-level. Accurate measurement is possible starting from the time requirements are specified to the later phases of software design and implementation. Language and platform independence enable size measurement before such technical details are decided. The organization can keep detailed productivity rates in functional size units with respect to different implementation languages, hardware platforms and use these rates on cost estimations for combinations of selections.

In cost control process, baselines can be set by determining functionality to be developed in each time-phase and calculating cost baseline by applying relevant estimation techniques. Cost variance analysis and cost performance measurement can be performed by comparing the planned and actual cost values based on implemented and remaining functionality. Earned value management, as a performance management technique used in cost control process, can be performed based on earned function points [19].

Quality Management. Software functional size is one of the most important attributes that enables comparisons of many process and project measures between projects of different sizes and utilizing different developments methods and implementation languages. Functional size values can be used in the normalization of several process, product and resource base measures. Derived measures such as defect density (number of defects detected in a software component per functional size unit), defect detection efficiency (number of defects detected in qualification tests per functional size unit), and productivity rate (effort spent for a specific activity per functional size unit) can be constructed only when software size values, such as functional size, is

available as the denominator for normalizations. This makes functional size a good candidate to construct a measurement framework distributed across project management processes and to maintain organizational project data to be based on.

In order for functional size to be measured correctly and accurately, ISO qualified FSM methods decompose the functional user requirements into a set of coherent and well-structured functional components with respect to the techniques and rules defined for each method. This practice also contributes the establishment of quality criteria for requirements specification by reducing the ambiguity and improving the rigor in requirements documentation.

Risk Management. Functional size can support quantitative risk analysis at the early phases of the software supporting methods for risk impact calculations with rates based on functional size, such as productivity, defect density and penalty cost per function points that are delivered late. It can provide the project managers with a mechanism to turn risks from scope creep into controllable and negotiable software scope changes with estimated impacts on cost and duration.

Project costs and schedule can be estimated early in project's life through use of functional size based effort estimation models. In risk response planning, using these models can be considered as an alternative risk mitigation strategy for such risks as schedule and budget overruns.

Project Procurement Management. Software acquisition and purchase planning is one of the most challenging processes that needs tools and techniques to handle problems of unstable requirements and relevant uncertainties. The identification and distribution of risks among acquirer and supplier parties becomes more difficult particularly when the software development is iterative. The use of functional size measures in different KAs, such as cost estimation, quality, scope and project integration management, is also valid and applicable in procurement management, since processes in procurement management is closely linked to processes in other KAs.

In addition, FSM based quantification can be further utilized by introducing functional size as the purchasing unit in bids, statement of works and contract documents. Cost and delivery rates can be agreed over functional size units [45]. The procurement items can be identified in terms of the amount of software purchased; scope changes can be mutually adjusted without breaking the contracts and sustaining a degree of risk avoidance for acquirer and supplier parties. Standardized FSM methods can improve contracts, increasing independency and objectivity in the quality and delivery terms. Thus, the functional size of the software delivered can facilitate the establishment of a consensus among acquirer, supplier and other stakeholders. Focusing on functionality rather than technical items (lines of code, number of components, and etc.) can be more meaningful in contracts since from an acquirer's perspective, services that will be provided by the software application are highly correlated to the use value -benefits to the user- of the software [35].

Applicability and Limitations. Project managers should be aware of the difficulties and limitations of FSM methods as well. Pros and cons of specific FSM methods and their limitations should be taken into consideration when using the functional size and a related measurement method. The studies by Lother & Dumke [36], Symons [23], Meli [38], Kitchenham[39], Kitchenham & Fenton [37], and Gencel & Demirors [40] discuss the criteria for evaluating FSM methods and explore the challenges.

In practice, the functional domains where FSM methods are effectively applicable are limited to business applications and real-time systems. All three FSM methods are used effectively in data-strong application domains and COSMIC FSM explicitly covers real-time software measurement in detail with examples and illustrations. Thus, the FSM method selection can be different in different projects. The functional size based comparison of projects depends on certain factors with respect to the purpose of the comparison. For instance, technical and environmental factors and productivity rates should be taken into account in comparisons of cost and duration of projects. Although there are studies and formulations for converting functional size measured by different FSM methods, it may not possible to convert them in every case since FSM methods have different purposes and measures functionality over different software elements or at different levels of detail. The measurement artifacts may be not detailed to support feasibility studies where accurate size measurement has the most value and early estimation techniques can be preferred at early project phases [41], [42], [43]. ISO/IEC 14143-3 [13] was developed to provide assistance to the FSM evaluators in determining an appropriate method, by providing a process for verifying the certain properties of an FSM method. This standard part can provide guidance in selection of the best method that satisfies the project management's needs.

4 Conclusions

In our study, we explored how functional size can contribute to effective execution of the project management processes. We addressed the potential opportunities for using functional size in project management areas in addition to cost estimation, which gained the focus in majority of FSM research and practices. While mapping the potential uses of the functional size to project management KAs, we observed that, although functional size can be used for different purposes in each, several uses are prevailing and they span across many of the project management and other organizational processes.

Input for cost, effort and schedule estimation. One of the significant purposes for measuring the software functional size is to be able to make accurate estimations of the development effort by incorporating the size into an estimation model. The relationship between software functional size and effort has been explored in many cost estimation models and techniques. Leung and Fan [44] discuss both the strengths and weaknesses of these models. Although their applicability is limited in scope and to specific functional domains, successful applications of functional size based cost estimation are reported in many studies [25], [34].

Support for monitoring scope change. Functional size provides a means to quantify change in project scope. Tracking the functional size at each baseline – requirements, design, implementation and maintenance - supports monitoring and controlling scope creep. Based on the functional size of the change, the impact on project cost and schedule can be estimated.

Enables size measurement at early project phases. As defined in ISO/IEC 14143-1, FSM methods can be applied to estimate size as soon as any functional user requirement is elicited. Estimating size early in the life-cycle is significant not only for

deciding feasibility of the project and related risks but also estimating its cost and effort at a point where such measurement information is actually required. As project progresses, size is (re) measured to obtain more accurate values. Early estimation methods, such as EFPA [41], Function Points Simplified [42], and Early & Quick COSMIC FFP [43] proposes techniques for estimating size before software requirements are specified in detail.

Using as a purchasing unit for software acquisition. Functional size can be used as a unit to be negotiated between supplier and acquirer. As a unit for sizing software projects and managing the size of a software project during the development, it can help managing and decreasing the risks for both sides. Having been ensured that an appropriate level of functionality will be delivered, the acquirer can be more ready to accept the risk for a given size of software project. Similarly, the supplier can be more willing to accept the risk for the cost of production (the cost per functional size unit).

Normalizing performance and quality measures. Functional size provides a significant value for normalizing several base measures as an indication of how well the organizational processes are performed. Such measures demonstrate general trends and progress and help to identify problem areas. For example, inconsistent productivity rates between projects can be considered as an indication that a standard process is not being followed. Similarly, varying defect densities in software products can be signs for inconsistencies in performing standard quality assurance activities. Unpredictable values for such measures are expected to stabilize as the project teams conform to standard organizational processes. Measures normalized with size not only help monitoring process performance but also assist comparing products and projects at the organizational level in terms of productivity, reliability, and other quality attributes. Defects per function size unit, maintenance effort per functional size altered, functional size per calendar month, cost per functional size unit are examples of measures that can be monitored at the organizational level to indicate trends and progress in performance levels.

In addition to the direct usage of functional size to support processes, as the project management processes in PMBOK are directly or indirectly linked to each other within and between KAs, the utilization of functional size in a KA process can implicitly contribute to all processes that are linked to that process. In relation to that, the prevalent uses can be a good selection of areas to incorporate functional size measurement in project management, where their application can show themselves in immediate returns and accelerate the spread of functional size measurement usage in project management processes.

In our exploratory study, we observed that the recognition of functional size measurement in software industry is due to its support for software estimation and the concentration of FSM research is on the cost and effort models. Despite concerns and limitations, functional size has many uses to answer software project managers' requirements beyond estimating cost and effort. In the development and improvement of models, tool and techniques that measures functional size or uses functional size to support project management processes, these requirements should be taken into account to increase the usability and applicability, thus realizing the opportunities in many management processes.

References

1. Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. International Thomson Computer Press, Boston (1996)
2. Project Management Body of Knowledge, 3rd edn. Project Management Institute (2004)
3. ISO/IEC IS 20968:2002: Software Engineering - MK II Function Point Analysis - Counting Practices Manual (2002)
4. ISO/IEC IS 20926:2003: Software Engineering - IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual (2003)
5. ISO/IEC 19761:2003: Software Engineering - COSMIC-FFP: A Functional Size Measurement Method (2003)
6. IEEE Std. 1490-2003 Adoption of PMI Standard - A Guide to the Project Management Body of Knowledge –Description (2003)
7. Albrecht, A.J.: Measuring Application Development Productivity. In: Proc. Joint SHARE/GUIDE/IBM Application Development Symposium (1979)
8. Albrecht, A.J., Gaffney, J.E.: Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Trans. Software Eng. 9(6), 639–648 (1983)
9. Gencel, C., Demirors, O.: Functional Size Measurement Revisited. ACM Transactions on Software Engineering and Methodology (July 2008) (to be published)
10. ISO/IEC 14143-1:1998 Information Technology - Software Measurement - Functional Size Measurement - Part 1: Definition of Concepts (1998)
11. ISO/IEC 14143-2:2002: Information Technology - Software Measurement - Functional Size Measurement - Part 2: Conformity Evaluation of Software Size Measurement Methods to ISO/IEC 14143-1:1998 (2002)
12. ISO/IEC TR 14143-3:2003: Information Technology - Software Measurement - Functional Size Measurement - Part 3: Verification of Functional Size Measurement Methods (2003)
13. ISO/IEC TR 14143-4:2002: Information Technology - Software Measurement - Functional Size Measurement - Part 4: Reference Model (2002)
14. ISO/IEC TR 14143-5:2004: Information Technology- Software Measurement - Functional Size Measurement - Part 5: Determination of Functional Domains for Use with Functional Size Measurement (2004)
15. ISO/IEC FCD 14143-6:2005: Guide for the Use of ISO/IEC 14143 and related International Standards (2005)
16. ISO/IEC IS 24570:2005: Software Engineering - NESMA functional size measurement method Ver.2.1 - Definitions and counting guidelines for the application of FPA (2005)
17. Abran, A.: Estimation Models for Software Maintenance Based on Functional Size. DoD SoftwareTech News 9(3) (2006)
18. Abran, A., Silva, I., Primera, L.: Field studies using functional size measurement in building estimation models for software maintenance. Journal of Software Maintenance: Research and Practice 14(1) (2002)
19. Muller, R.J.: Earning Function Points in Software Projects. In: SM/ASM Conference (1999)
20. Dekkers, T.: IT Governance requires performance measurement. In: PSQT/PSTT North Conference (2004)
21. Buglione, L., Abran, A., Meli, R.: How Functional Size Measurement supports the Balanced Scorecard Framework for ICT, FESMA-DASMA (2001)
22. Rispens, M., Vogelezang, F.: Application Portfolio Management - The Basics - How much software do I have? In: Software Measurement European Forum-SMEF (2007)

23. Symons, C.: Come Back Function Point Analysis (Modernized) – All is Forgiven. In: Proc. of the 4th European Conference on Software Measurement and ICT Control, FESMA-DASMA 2001, pp. 413–426 (2001)
24. Boehm, B.W., Abts, C., Brown, A.W., Chulani, S., Hall, B.K.: Software Cost Estimation with Cocomo II. Prentice Hall, NJ (2000)
25. Abran, A., Ndiaye, I., Bourque, P.: Contribution of Software Size in Effort Estimation. Research Lab. In: Software Engineering, École de Technologie Supérieure, Canada (2003)
26. Kitchenham, B., Mendes, E.: Software Productivity Measurement Using Multiple Size Measures. IEEE Transactions on Software Engineering 30(12), 1023–1035 (2004)
27. Forselius, P.: Benchmarking Software-Development Productivity. IEEE Software 17(1), 80–88 (2000)
28. ISBSG, International software benchmarking standards group, http://www.isbsg.org/au
29. Tran, C., Levesque, G.: Maintenance Effort and Cost Estimation Using Software Functional Sizes. In: IWSM (2003)
30. Rollo, T.: Functional Size Measurement and COCOMO – A Synergistic Approach. In: Proc. of Software Measurement European Forum (SMEF), Rome, Italy, pp. 259–267 (2006)
31. Lokan, C.J.: An empirical analysis of function point adjustment factors. Information & Software Technology 42(9), 649–659 (2000)
32. Jensen, R.: A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models. In: Proc. Int'l Society of Parametric Analysis, pp. 96–106 (1984)
33. Nasir, M.: A Survey of Software Estimation Techniques and Project Planning Practices. In: SNPD, pp. 305–310 (2006)
34. Matson, J., Barrett, B., Mellichamp, J.: Software development cost estimation using function points. IEEE Transactions on Software Engineering 20(4), 275–287 (1994)
35. Meli, R.: The Software Measurement Role in a Complex Contractual Context, Software. In: Measurement European Forum, Rome, pp. 28–30 (2004)
36. Lother, M., Dumke, R.: Points Metrics - Comparison and Analysis. In: International Workshop on Software Measurement (IWSM 2001), Montréal, Québec, pp. 155–172 (2001)
37. Kitchenham, B., Fenton, N.: Towards a Framework for Software Measurement Validation. IEEE Transactions on Software Engineering 21(12) (1995)
38. Meli, R.: Functional Metrics: Problems and Possible Solutions, FESMA, Antwerpen (1998)
39. Kitchenham, B.: The Problem with Function Points. IEEE Software 14(2), 29–31 (1997)
40. Gencel, C., Demirors, O.: Conceptual Differences Among Functional Size Measurement Methods. In: Empirical Software Engineering and Measurement, ESEM (2007)
41. Conte, M., Iorio, T., Meli, R., Santillo, L.: E&Q: An Early & Quick Approach to Functional Size Measurement Methods. In: Proc. of Software Measurement European Forum (SMEF), Rome, Italy (2004)
42. Bock, D.B., Klepper, R.: FP-S: a simplified function point counting method. Journal of Systems and Software 18, 245–254 (1992)
43. Meli, R., Abran, A., Ho, V.T., Oligny, S.: On the Applicability of COSMIC-FFP for Measuring Software Throughout Its LifeCycle. Escom-Scope (2000)
44. Leung, H., Fan, Z.: Software Cost Estimation. Handbook of Software Engineering. Hong Kong Polytechnic University (2002)
45. Demirörs, O., Karagoz, N.A., Gencel, C.: Acquiring Innovative Software Systems: Experiences from the Field. In: EUROMICRO-SEAA, pp. 393–400 (2007)

Process Reference Guides – Support for Improving Software Processes in Alignment with Reference Models and Standards

Jean Carlo R. Hauck[1,2], Christiane Gresse von Wangenheim[1,2], Richard H. de Souza[1], and Marcello Thiry[2]

[1] UFSC – Universidade Federal de Santa Catarina. Campus Universitário, Trindade, 88040.900 Florianópolis - Santa Catarina, Brazil
[2] UNIVALI – Universidade do Vale do Itajaí. Rodovia SC 407, Km 4, 88102.280 São José - Santa Catarina, Brazil
jeanhauck@egc.ufsc.br,
{gresse,richardhenrique,marcello.thiry}@gmail.com

Abstract. Software process improvement in small organizations in alignment with reference models or standards remains complicated. In this paper, we enhance an approach for software process improvement and introduce the concept of process reference guides as a way to explicitly map reference models/standards and potential solutions in order to systematize and facilitate the process definition in improvement initiatives. Experiences provide a first indication that such reference guides can be useful in this context and may help to reduce the effort for process definition.

Keywords: Software Process Improvement, Process modeling, CMMI, ISO/IEC 15504, Alignment.

1 Introduction

Many small organizations have problems in improving effectively and efficiently their software processes [1]. Although, today, exist a variety of widely accepted reference models for various processes, such as, CMMI [2], ISO/IEC 15504 [3], ITIL [4], still only a small number of small organization manages to successfully systematize their software process in alignment with those models. For example, a survey run by the ISO/IEC JTC1/SC7 Working Group 24 on Life Cycle Processes for Very Small Enterprises (VSEs) [5], found out that less than 18% of very small organizations (with less than 25 employees) are certified and among the 82% of VSEs not certified, only 25% claim to use standards.

One of the predominant reasons for this is that reference models or standards are often perceived as difficult and bureaucratic, not offering adequate guidance for small business environments. These models are typically developed focusing on larger companies and do not simply "scale down" to small organizations, especially to those with a low capability level [1]. Consequently, compliance with such standards or reference models is difficult, if not impossible for them to achieve.

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 70–81, 2008.

Another aspect is that these reference models or standards define requirements in relation to software processes. They do not intend to nor provide detailed support on how these requirements can be satisfied. In this context, process models, such as RUP [6] or ICONIX [7] for example, provide generic process descriptions. Yet, such process models also cannot simply be deployed in an organization, as in order to be effective and operational, processes need be defined based on the actual processes in place in the organization taking into consideration its specific characteristics, needs and limitations [8].

Thus, in order to assure an effective adoption of a defined process, the process should be defined in a balanced way by eliciting the actual process in place and only improving or completing the existing process, where necessary [9]. In this context, there exist a variety of approaches, which deal on different levels of formalism with the descriptive modeling of software processes (e.g., [10] [11] [12] [13] [14]).

However, considering high-level reference models on one side and software process modeling approaches on the other side, in practice, we observed a need for intertwining descriptive and prescriptive process modeling in order to come up with a defined process aligned with reference models. Therefore, we need to integrate descriptive and prescriptive process modeling activities as well as provide guidance, which in a more refined way presents possible solutions for process shortcomings in alignment with reference models or standards.

In this paper, we describe such an extension to the process improvement approach ASPE/MSC (Approach for Software Process Establishment in Micro and Small Companies), which has been developed to ASPEI/MSC (Approach for Software Process Establishment and Improvement in Micro and Small Companies) specifically for software process improvement (SPI) in small companies. As part of this, we introduce the concept of process reference guides, which can be considered a flexible collection of alternative processes, techniques and tools mapped to practices required by reference models and standards. Such process reference guides can facilitate the improvement of existing processes by indicating various alternative solutions to be tailored to the specific needs of the organization. An example process reference guide for project monitoring & control is presented. We also summarize first experiences and feedback from the application of the approach in practice.

2 ASPEI/MSC

Few approaches have been described specifically for the establishment and improvement of software processes in small organizations [13], [14]. In this context, we are defining ASPEI/MSC by integrating and adapting existing approaches (including [10], [11], [12], [13], [14]) to the characteristics of small software companies [1], [15]. We do not intend to develop a new method, but rather aim at the integration and tailoring of existing approaches to the context of small software companies.

The principal phases of the approach are Instantiation, Diagnosis, Strategic Analysis, Definition and Deployment to be executed in an iterative and incremental way in order to establish and improve step-by-step one or more process(es) within an organization (Figure 1). During the **Instantiation** phase, the software process improvement initiative is prepared and the necessary infrastructure and pre-conditions are created (e.g., allocating personnel for SPI). During the **Diagnosis** phase, a process assessment of the actual software process is performed, identifying its strengths and weaknesses

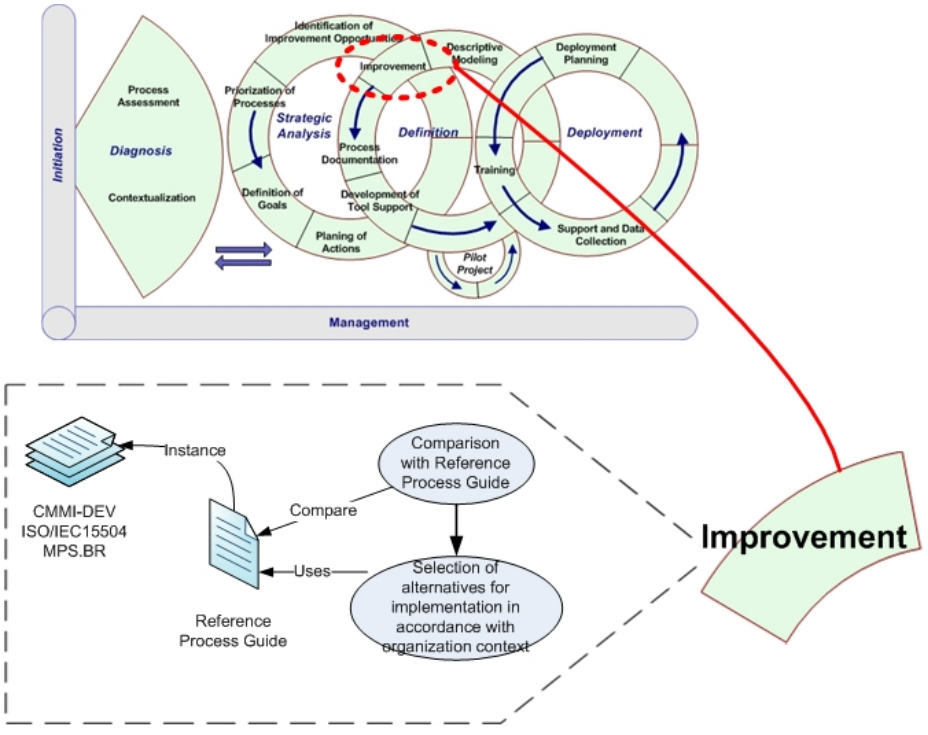


Fig. 1. ASPEI/MSC – enhancing the improvement of the process definition

and establishing a target process profile to be achieved. Such an assessment can be done on different levels of scope and detail, ranging from an overview assessment to a focused assessment of a set of processes in alignment with one or more reference models or standards, using, for example, the MARES method [16]. Based on the assessments results, during **Strategic Analysis**, processes to be established and improved are prioritized in accordance to the organization's business and improvement goals. As a result, improvement cycles and the process(es) to be improved in each cycle are defined. During the **Process Definition** phase, each process chosen is modeled, improved and documented in form of an organizational process guide. This is done by eliciting the actual process in place creating a descriptive process model, which in alignment with relevant reference models or standards is being improved, where necessary. The standard process being defined is applied and evaluated in pilot-projects and then, during **Deployment,** institutionalized throughout the organization.

In addition, the approach also covers the management of the process establishment, including planning, monitoring & control and post-mortem.

We developed a first version of this approach (called ASPE/MSC) in 2005, which has since then been applied successfully in several SPI initiatives in small organizations [13], [15], [16], [17]. However, while adopting this approach, we also perceived a specific difficulty in finding solutions for weaknesses identified especially in order to satisfy requirements of reference models or standards. And, although, today, exists

a variety of alternative solutions to satisfy requirements of reference models or standards, there basically does not exist a more systematic support or know-how on which alternative processes, techniques or tools could be adopted in a specific environment and how they may need to be tailored to fit the specific context. Thus, the identification of potential solutions to improve an organizational process still remains a complex task, which requires high level of expertise and is typically performed without reuse and no explicit, specific know-how. This represents especially a problem to small organizations, which typically cannot count with experienced software process engineers, which have a profound understanding of the reference models or standards and the available processes, techniques or tools. To fill this gap, we enhanced the improvement step in the ASPEI/MSC approach by introducing the concept of process reference guides, which map existing alternatives with required practices in reference models and standards and, thus, facilitate the identification of possible solutions in order to improve the current process in place (Figure 1).

3 Introducing Process Reference Guides

By process reference guides we understand a model, which maps requirements of reference models or standards (e.g., outcomes or base practices) with a broad variety of processes, techniques and/or tools to satisfy these requirements. Process reference guides are expected to provide detailed information on these processes, techniques and tools as well as information on their applicability in certain contexts and on how to tailor them to suit specific needs and characteristics. They are not a single, prescriptive process description, but rather a collection of diverse alternative solutions, from which potential improvement solutions can be selected and tailored to a specific organization's standard process.

Table 1. Process reference guide structure

Introduction	Visualization of the scope of the respective process and its relation to other processes (Figure 2).
Basic concepts	Description of basic concepts and terminology in relation to the respective process.
Assessment	Set of assessment indicators derived from the considered reference models and standards (e.g. CMMI-DEV and ISO/IEC 15504) with a mapping of potential processes, techniques and tools, which may help to the improve the process (Figure 3).
Typical activities	Set of typical activities executed as part of the respective process, describing for each activity its purpose, objective, steps, working products and templates, roles as well as relationships between activities.
Practices	Presentation of practices required by reference models or standards, such as, CMMI-DEV or ISO/IEC15504.
Techniques	Description of relevant techniques, which can be used in order to establish required practices and achieve the required outcomes.
Tools	Comparison and reviews of relevant tools in relation with the support they provide for the specific process in alignment with the considered reference models and standards.

As one of the purposes of the process reference guides is to provide guidance for process improvement in consistency with reference models and standards, the guides are based on relevant reference guides and present detailed information on how the requirements of the models can be satisfied. A structure for such a process reference guide is shown in Table 1.

Such process reference guides can be developed initially based on a literature research, compiling and comparing well-accepted processes, techniques and tools as well as consolidate experiences on their application in practice. This includes also a detailed analysis of relevant reference models and standards.

However, the principal focus of the development of such reference guides has to be on their continuous evolution. Such an effort, in practice is only viable through collaborative knowledge management. In this context, various types of tools can support the management of such process reference guides. For example, process modelling tools, such as, SPEARMINT [18] or Wagner [19]; general modeling tools, such as, Enterprise Architect [39], BPMN Designer (free) [40] or tools specifically designed for process framework development, such as the Eclipse Process Framework Composer (EPF Composer) [41] as well as general tools for collaborative knowledge management, including semantic web and ontologies [46], WIKIs [20] and, especially, semantic WIKIs [47].

Currently, we are initiating the development of process reference guides for various processes focusing on the context of small organizations. An example is a process reference guide on Project Monitoring and Control [17] being developed in alignment with CMMI-DEV [2], ISO/IEC 15504 [3] and the Brazilian Software Process Improvement Model MPS.BR [21] based on literature in the area of project monitoring & control, including [2], [23], [24], [25], standards [26], [27] and guides [28], [29],

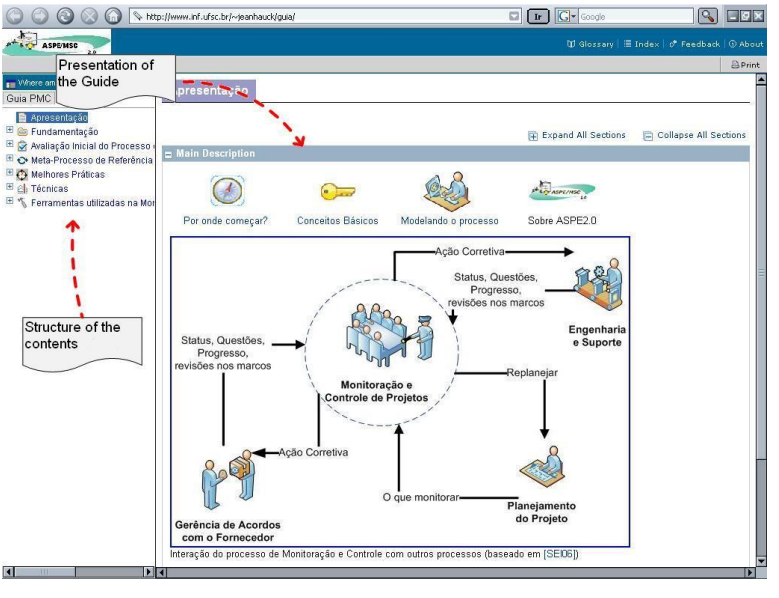


Fig. 2. Introduction of the project monitoring & control guide [42]

[30], as well as experience reports in the context of small organizations [1], [31], [32], [33] and our experiences on establishing this process in practice.

For example, Figure 2 shows the introduction of the process monitoring & control guide. Starting from this page, the guide can be used in different ways depending on the level of expertise of the process engineer, either by directly accessing an aspect of interest or by guiding the engineer through an assessment questionnaire, which explicitly links potential solutions to identified shortcomings (Figure 3).

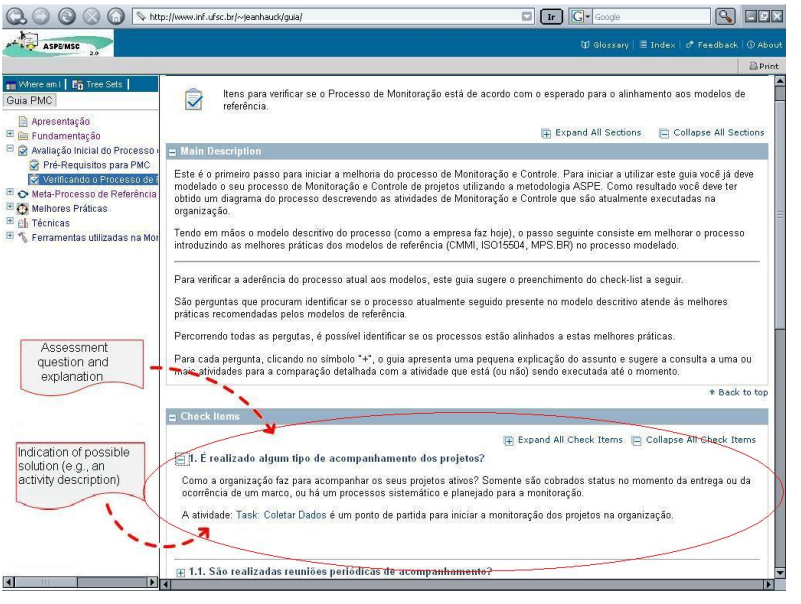


Fig. 3. Support provided for process assessment [42]

In this way, process reference guides provide a structured overview on existing processes, techniques and tools mapping these alternatives to reference models and standards. Offering such a process reference guide as support, is expected to reduce the related effort and time as well as to improve the quality and adherence of defined processes.

Software processes can be defined on various levels of detail, ranging, typically, from concrete instantiated processes in a specific project, organizational standard processes to high-level and generic reference models or standards. Using the Software Process Engineering Metamodel (SPEM) [34], a standard for software process modeling, we can express these levels and place the concept of process reference guides. Therefore, we enhance the SPEM architecture of four to six levels of abstraction, refining level M1 in order to separate organizational process models from generic process models, similar to the proposal by [35]. In Figure 4, these different levels of processes are shown with respect to the four-layered organization of SPEM. Layer 3 describes the process modeling metamodel. Layer M2 defines the ASPEI/MSC notation based on the SPEM notation as model for the definition of process models.

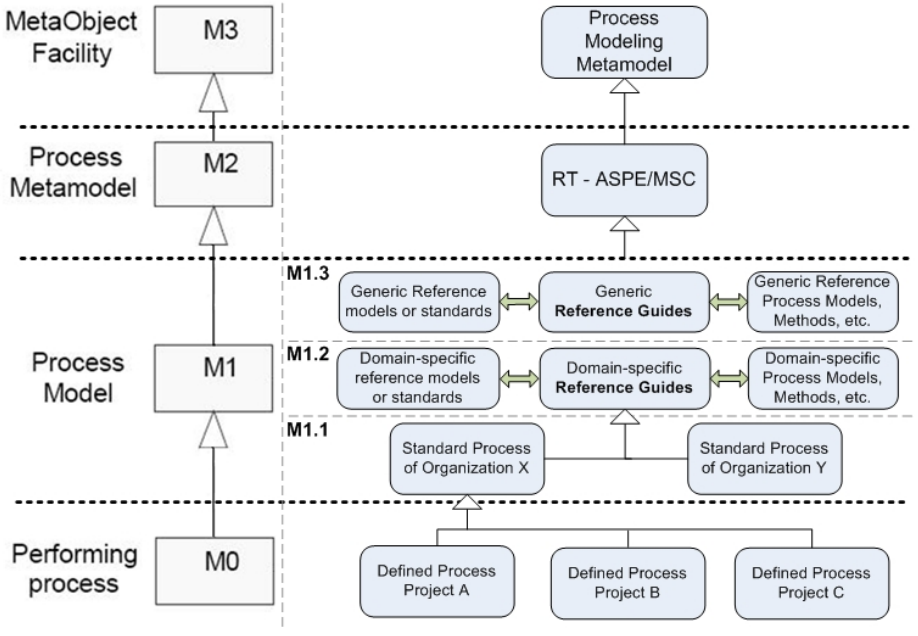


Fig. 4. Definition of process levels in alignment with SPEM

Following the original definition of SPEM, Layer M1 contains process models. Yet, this representation does not explicitly express different levels of process modeling and, thus, we suggest dividing layer M1 in three sub-layers:

- Layer M1.1 represents organizational standard processes, which are sets of definitions of the processes that must be incorporated into the defined processes that are implemented in projects across the organization.
- Layer M1.2 represents references models or standards for a specific domain or sector, such as, e.g., S4S (SPICE for Space [36]), AutomotiveSPICE [43] as well as domain-specific process models or reference process guides, such as, for example, a reference process guide for project monitoring & control in small organizations.
- Layer M1.3 represents generic reference models or standards, such as CMMI, ISO/IEC 15504 as well as generic process models, such as RUP, and generic reference process guides.

Layer M0 represents the defined process, which is an instantiated process in a specific project that is managed (planned, monitored and adjusted), tailored from the organization's set of standard processes according to the organization's tailoring guidelines. It provides a basis for planning, performing, and improving a project's tasks and activities.

4 First Experiences

So far, we have started to apply the enhanced approach ASPEI/MSC using the project monitoring & control guide in two process improvement initiatives. The first application was run in parallel to the development of the guide in the software R&D group CYCLOPS [44] at the Federal University of Santa Catarina/Brazil. A second application has been run in a small software company in Florianópolis/ Brazil.

The focus of the first improvement initiative was focused on project management, including monitoring & control as well as on requirements development and management. The process improvement was coordinated full-time by a junior process engineer and weekly supported by external senior consultants. We followed the ASPEI/MSC approach and during the Definition phase elicited the processes in place through process workshops with the process performers. A gap analysis was done in order to identify shortcoming of the processes actually in place in relation with the respective reference models. Based on an initial version of the process reference guide, process engineers started to improve the processes in cooperation with process performers. The organization's standard process has been documented in form of an Electronic Process Guide (EPG) as part of the organizational WIKI. We, then, started to implement the process in pilot projects, identifying aspects in which it remained inefficient or ineffective. Once an operational standard process had been achieved, we started to train and deploy the process organization-wide.

The total effort spent during the process improvement, so far, is about 265 person-hours, with about 80 person-hours spent on the definition and deployment of the project monitoring & control process, which represents about 2% of the total effort spent by the R&D group during this period. Today, the standard process is used in all projects of the CYCLOPS Group. More than 60 percent of the specific practices of the project monitoring & control process area of CMMI-DEV are characterized at least as largely implemented in the majority of the software projects of the organization.

In this first application, we could not yet identify a significant advantage through the usage of process reference guides, principally due to the fact that the project monitoring & control guide was being developed in parallel.

Using the organizational WIKI for the documentation of the process description was considered helpful, especially as it made the review and collaborative evolution easy. However, we also observed some weaknesses, especially when compared to other process modeling tools, which provide more support for structuring EPGs and linking elements automatically as well as offering functionality for the graphical visualization of process elements. As a consequence, we decided to use the Eclipse Process Framework Composer for the development of the process reference guides. The principal advantages are its support for various levels of abstraction and its comprehensive support to interrelate elements of the guide. Another substantial advantage is that the EPF Composer permits in a very simple way to construct an organizational standard process from a reference guide. Similar to other EPG tools, it also allows to publish the process guide on the web. Yet, a significant shortcoming at the moment is the inability to continuously evolve the framework easily in a collaborative manner. But, such a support is foreseen to become available as part of the next releases with EPF WIKI.

The second application took place after a first version of the process reference guide on project monitoring & control had been developed. We also applied the ASPEI/MSC approach focusing on the establishment of the project planning, monitoring & control as well as the requirements management. The improvement initiative has been realized by a part-time process engineer and external junior and senior consultants. Similar to the application at the CYCLOPS Group, we performed a high-level process assessment in the beginning and started to define the process in place. Based on a gap analysis in relation with the considered reference model, we identified improvement opportunities and indicated possible solutions using the process reference guide. The defined standard process has been applied in a pilot project and the company is now starting to use the process organization-wide.

During this application of ASPEI/MSC, we started to perceive an indication for a potential effort reduction during the process definition. In comparison also to other initiatives in small organizations where we improved the project monitoring & control process and spent an average of 23 person-hours on the process definition, the definition of the project monitoring & control process using the process reference guide was reduced to 12 person-hours. Yet, this may have happen also due to other reasons, as, e.g., the fact that the junior consultant got more experienced after having established the process in several organizations. Subjectively, the consultants perceived that the process reference guide can facilitate the process improvement by mapping weaknesses of the organization's process to expected practices of the reference models and standards. In this respect, especially, the association of practices and solution alternatives was considered valuable. So far, a principal shortcoming is that insufficient information on the applicability and tailoring of solution alternatives in certain contexts is available. In addition, we also observed that even using the EPF Composer, it remained difficult to navigate through the guide.

5 Discussion

Our experiences provide a first indication that such process reference guides can be useful in order to support the definition of a software process. In contrast to reference models or standards, such as CMMI or ISO/IEC 15504, they intend to provide more concrete information on how to implement the required practices by presenting various alternatives. In this way, they also go a step further than implementation guides, which are being published as part of some reference models, as, e.g. the implementation guide of the Brazilian Process Improvement Model MPS.BR [21] or [28].

A difference of the concept of process reference guides in comparison to process frameworks, such as, the Rational Unified Process (RUP) [6], ICONIX [7] or Cleanroom [37] is that the idea of reference guides is to offer a broader process vision, which may cover several different process frameworks and presenting them (or parts of these frameworks) as alternative solutions, including also information on when to apply which and how to tailor such frameworks to a specific organization.

Another similar concept are Process Patterns, which are collections of general techniques, actions, and/or tasks (activities) for developing object-oriented software [38] [45]. They describe a proven, successful approach and/or series of action for developing software. And, although, originally limited to object-oriented software

development, they also seem to be valuable using other methodologies. Ambler classifies three types of process patterns [38]: task process patterns, stage process patterns, phase process patterns. Each pattern is structured as Forces, Initial Context, Solution, and Resulting Context. But, as process pattern are understood to describe what should be done, they do not describe the exact details of how.

Thus, in comparison, we consider a principal strength of process reference guides their detailed description also on how to execute practices as well as their larger variety of potential alternative.

6 Conclusions

In this paper, we introduce the concept of process reference guides as a way to explicitly map reference models and potential solutions in order to systematize and facilitate the process definition in improvement initiatives, principally in small organizations. Experiences provide a first indication that such reference guides can be useful in this context and may help to reduce the effort for process definition. We are planning to continue the application of the approach in future improvement programs and the evaluation of the process reference guides as well as to amplify the definition of reference guides for other process areas, such as, project planning, requirements development, etc.

Acknowledgement

Our thanks to all involved in the improvement initiatives at CYCLOPS/UFSC and the small software company.

References

1. Richardson, I., Gresse von Wangenheim, C.: Why are Small Software Organizations Different? IEEE Software 24(1) (January/Febuary 2007)
2. CMMI Team. CMMI-DEV: CMMI for Development, version 1.2. Software Engineering Institute/Carnegie Mellon University, Pittsburgh (2006)
3. International Organization for Standardization. ISO/IEC 15504: Information Technology – Process Assessment: Part 1- Part 5 (2003 -2006)
4. ITIL v3 (2007), http://www.itil-officialsite.com
5. Laporte, C.Y., Alexandre, S., Renault, A.: Developing International Standards for Very Small Enterprises. IEEE Computer 41(3) (March 2008)
6. IBM Rational Unified Process (2008-05-06), http://www-306.ibm.com/software/awdtools/rup
7. Rosenberg, D., Collins-Cope, M., Stephens, M.: Agile Development with ICONIX Process. Apress (2005)
8. Acuña, S.T., et al.: The Software Process: modeling, evaluation and improvement. Handbook of Software Engineering and Knowledge Engineering, vol. 1. World Scientific Publishing Company, Singapore (2001)

9. Jacobson, I.: Does Process come from the top or from the bottom? Flexi Newsletter, 1 (2008-04-12) (2007), http://www.flexi-itea2.org/newsletter.php
10. Machado, L.F., Oliveira, K.M., Rocha, A.R.: Using Standards and Maturity Models for the Software Process Definition. Quality Week, Belgium (2000)
11. Scott, L., Zettel, J., Hamann, D.: Suporting Process Engineering in Pactice: An Experience Based Scenario. IESE Technical Report no. 033.00/E, Fraunhofer Institute for Experimental Software Engineering, Germany (2000)
12. Becker, U.: Towards Systematic Knowledge Elicitation for Descriptive Software Process Modeling. IESE Technical Report n° 036.01/E, Fraunhofer Institute for Experimental Software Engineering, Germany (2001)
13. Thiry, M., von Wangenheim, C.G., Zoucas, A.: Uma Abordagem para a Modelagem Colaborativa de Processos de Software em Micro e Pequenas Empresas. In: SBQS – Brazilian Symposium on Software Quality, Vitória (2006)
14. Dingsøyr, T., Moe, N.B., Dyba, T., Conradi, R.: A Workshop-Oriented Approach for Defining Electronic Process Guides. In: Juristo, N., Acuña, S.T. (eds.) Software Process Modeling. Kluwer Academic Publishers, Dordrecht (2005)
15. von Wangenheim, C.G., Weber, S., Hauck, J.C.R., Trentin, G.: Experiences on establishing software processes in small companies, vol. 48. Elsevier, Amsterdam (2006)
16. von Wangenhein, C.G., Anacleto, A., Salviano, C.F.: Helping Small Companies Assess Software Processes. IEEE Software 23(1) (January/ Febuary 2006)
17. Hauck, J.C.R., Wangenheim, C.G., Thiry, M.: Suportando a Modelagem de Processo de Monitoração e Controle em Micro e Pequenas Empresas, alinhado ao CMMI, MPS.BR e ISO/IEC15504. In: SBQS – Brazilian Symposium on Software Quality, Ipojuca (2007)
18. Becker-Kornstaedt, U., et al.: Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626. Springer, Heidelberg (1999)
19. Scott, L., Kurniawati, F.: WAGNER – Web-bAsed process Guide aNd Experience Repository. Centre for Advanced Software Engineering Research (CAESER), University of New South Wales, Sydney (2001)
20. Louridas, P.: Using wikis in software development. IEEE Software 23(2) (March/ April 2006)
21. SOFTEX, MPS.BR – Brazilian Software Process Improvement Model – Implementation Guide, version 1.2, Brasília (in Portuguese) (2007)
22. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 3. edn, PMI, Pennsylvania (2004)
23. Hughes, B., Cotterell, M.: Software Project Management, 3rd edn. McGraw-Hill, New York (2002)
24. Department of Defense & US Army. PSM - Practical Software and System Measurement, A foundation for Objective Project Management, Ver. 4.0c (2003)
25. ANSI, ANSI/EIA 748. A Standard for Earned Value Management Systems. ANSI Standard (1998)
26. International Organization for Standardization. ISO/IEC 10006: Quality Management – Guidelines to Quality in Project Management, 2 edn. (2003)
27. Jalote, P.: CMM in Practice: Processes for Executing Software Projects at Infosys. Addison Wesley Longman, Amsterdam (2000)
28. Kulpa, M.K., Johnson, K.A.: Interpreting the CMMI®: a process improvement approach. Auerbach Publications (2003)
29. Kerzner, H.: Project Management: A Systems Approach to Planning, Scheduling, and Controlling, 9th edn. Wiley, Chichester (2003)

30. Orci, T., Laryd, A.: CMM for Small Organisations Level 2. Suécia: Umeå University (2000)
31. Meneses, J.B.: Inspector: Um Processo de Avaliação de Progresso para Projetos de Software. Máster Thesis, Federal University of Pernambuco, Recife/Brazil (2001)
32. Otoyo, S., Cerpa, N.: An Experience: A Small Software Company Attempting to Improve its Process. Software Technology and Engineering Practice (1999)
33. OMG - Object Management Group. Software Process Engineering Metamodel Specification, Version 1.1 OMG Specification (2005)
34. Järvi, A., Makila, T.: Observations on Modeling Software Processes with SPEM Process Components. In: Proc. of 9th Symposium on Programming Languages and Software Tools, Estonia (2005)
35. Cass, A., et. al.: SPiCE for SPACE: A Method of Process Assessment for Space Software Projects.In: Proc. of the International SPICE Conference (2000)
36. Mills, H., Dyer, M., Linger, R.: Cleanroom Software Engineering. IEEE Software 4(5) (September 1987)
37. Ambler, S.: Process Patterns - Building Large-Scale Systems Using Object Technology. Cambridge University Press, Cambridge (1998)
38. Sparx Systems (2008-05-05), http://www.sparxsystems.com.au
39. Intalio, BPMN Designer (2008-05-06), http://www.intalio.com
40. Eclipse Process Framework (2008-05-07), http://www.eclipse.org/epf
41. Process Reference Guide (2008-05-06), http://www.inf.ufsc.br/~jeanhauck/guia/
42. AutomotiveSPICE (2008-05-06), http://www.automotivespice.com
43. CYCLOPS Research Group (2008-05-06), http://www.cyclops.ufsc.br/
44. The Process Patterns Resource Page (2008-05-06), http://www.ambysoft.com/processPatternsPage.html
45. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web, May 17, 2001. Scientific American Magazine (2001)
46. Schaffert, S., Baumeister, J., Bry, F., Wikis, M.K.S.: IEEE Software, July/August (2008)

Practical SPI Planning

José Francisco Landaeta, Javier García, and Antonio Amescua

Computer Science Department
Carlos III University
Avda. Universidad, 30, 28911, Leganés, Madrid (Spain)
100064802@alumnos.uc3m.es, jgarciag@inf.uc3m.es,
amescua@inf.uc3m.es

Abstract. This paper presents a practical procedure named P4SPI for planning, monitoring and closing an SPI. Planning activities are performed using PMBOK's process areas as a reference; monitoring activities using Six Sigma tools and techniques and closing activities using gathering qualitative and quantitative information about the SPI Implementation. These activities are supported by office templates.

Keywords: Software Process Improvement, Project Planning, Project Management, PMBOK.

1 Introduction

The knowledge area of this paper is Software Process Improvement (SPI) Project Management. According to Zhang [34], project planning is the third most important factor affecting project success and maybe the most time consuming. It is said that [32] the project planning process requires approximately 35% of the project manager's effort over the life of the project. Blair [5] reminds us that: "The success of a project will depend critically on the effort, care and skill you apply in its initial planning."

One of the problems identified is that some SPIs are not planned as projects, even when it is generally known that SPIs require a large quantity of effort and resources from the organization. This conclusion is supported because the average time needed to move up one level (out of five) is around 16-32 months [1] and, the cost of SPIs depends on the size of the organization where it takes place. As some studies can prove to be expensive [11], SPI must be considered as a Project [3] and planned carefully [1].

Another problem is that the lack of success in SPI implementation is not due to a lack of a standard or a model, but rather the lack of an effective strategy to successfully implement these standards or models [25]. In this study, two SPI models (IDEAL, DMAIC) were selected because they allow planning and implementing SPI Programs.

Finally, according to [25], in order to increase the SPI success probability it is necessary to offer guidance to SPI Managers on how to implement SPI activities, rather than suggesting what SPI activities are actually implemented. [19] has identified nine

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 82–93, 2008.

key functions or activities to ensure process improvement success. These activities are included and mapped to the two SPI models selected (IDEAL and DMAIC).

Having identified the problems: SPIs are not managed and planned as normal Projects and the lack of a practical procedure to implement an SPI, the focus of this paper is based on the following research questions:

1. What activities should be performed to plan, monitor and close an SPI? How are these activities and their sequences performed?
2. Which set of components are necessary to plan and monitor an SPI?
3. How can the success of SPI Implementation be ensured?
4. What results are obtained from using P4SPI in an organization?

These questions can be answered by is to answer these questions by:

- Defining a procedure to manage an SPI based on Project Management best practices.
- Developing a set of templates and tools to help SPI Managers plan and monitor an SPI and its restrictions: time, cost, scope.
- Providing a practical procedure to ensure the success of SPI Implementation.

The rest of the paper is structured as follows: Section 2 reviews the state of art in SPI planning. Section 3 describes P4SPI. Section 4 presents the steps needed to apply the P4SPI and finally, section 5 presents the conclusions of this research and future work.

2 State of the Art

P4SPI is founded on a practical view of managing an SPI Implementation. Among the different models for managing SPIs, the most famous are IDEAL and DMAIC. The IDEAL model defines and determines how to manage an SPI from inception to closure. However, when it comes to planning, its activities are very generic and are not easy to put into practice. The DMAIC model has a definition phase to determine the project scope, but does not specify how to create a project planning. A review of the IDEAL and DMAIC models and an analysis on using them for planning, monitoring and closing an SPI follow.

2.1 IDEAL Model

The IDEAL model provides guidelines on how to organize SPI initiatives. The model is based on five recommended phases: Initiating, Diagnosing, Establishing, Acting and Leveraging. The Initiating phase is where the initial improvement infrastructure is established. The Diagnosing phase lays the groundwork for the later phases through the creation of the SPI action plan. During the Establishing phase, the issues that the organization has decided to address are prioritized and strategies for pursuing the solutions are developed. In the Acting phase, solutions to address the areas for improvement discovered during the Diagnosing phase are created, piloted, and deployed throughout the organization. Finally, the objective of the Learning phase is to make the next step the IDEAL model more effective through reflection and learning [6]. These sequences represent the ideal scenario, but it is known that their sequential application is complicated because every organization must customize each step to its particular situation.

Additionally, the limits between steps are not clearly defined. The model does not provide much help in how to adapt it, what factors and interdependences must be taken to adapt the model to the vision, goals objectives and resources of the organization [12]. IDEAL is oriented to big companies and small settings cannot afford to implement it as proves too costly [9].

2.2 DMAIC Model

The Define-Measure-Analyze-Improve-Control (DMAIC) model is the most commonly used to achieve six sigma projects. It has five phases and 26 steps. According to [28], the DMAIC model can be used to find and solve problems in existing processes. It can also be used to expand the current capabilities of an existing process by identifying opportunities to improve current processes. Each phase of DMAIC is explained as follows:

- **Define phase** defines project goals aligned with business goals, project scope, customers with their requirements, project charter and project teams. A high-level map of the current process is also created.
- **Measure phase** collects data on current processes, and develops measurement systems to validate collected data. The current process performance is calculated based on measured data.
- **Analyze phase** identifies ways to narrow the gap between the current performance level and the desired goals. The project team analyzes collected data of current processes, and determines the root causes of the poor sigma performance of the processes.
- **Improve phase** identifies, evaluates and selects. Focusing on the root causes identified in the Analyze phase, the project team generates and selects a set of solutions to improve sigma performance.
- **Control phase** is to implement the final solutions and guarantee the maintenance of newly improved processes so that the improved sigma performance holds up over time.

2.3 Analysis of Project Management of SPI Models

Both models include/define project management activities, but IDEAL needs to be more explicit in order to be used and DMAIC must be included in the Project management tasks because it is a powerful tool [4] and can serve as an enabler for the successful implementation of domain-specific improvement models. On the other hand, according to [10], implementing an SPI based on the CMMI (and using IDEAL) model is directly related to the increase in project performance within an organization. The authors proposed including Six Sigma tools and techniques in the SPI phases to plan and monitor implementation, and to ensure the SPI success. Blending Six Sigma and CMMI is possible and their joint deployment is synergistic. The potential added value is the accelerated SPI adoption [24], [31]. P4SPI's goal is to make the planning activities of an SPI, based on the CMMI model, more practical and to incorporate some Six Sigma activities in its planning.

3 P4SPI Description

3.1 Overview

The P4SPI is a procedure used to plan an SPI as a normal project, to monitor its implementation using Six Sigma techniques and tools, and to evaluate the SPI implementation collecting its information and lessons learned. P4SPI allows:

- To connect business goals with the improvement goals.
- To connect improvement goals with CMMI Process Areas (PA).
- To perform some of the activities suggested by PMBOK.
- To monitor SPI implementation using Six Sigma techniques and tools
- To collect SPI postmortem information as a feedback for continuous improvement.

P4SPI life cycle is based on the project life cycle proposed by PMBOK, but it has 4 phases instead of 5. The execution phase is not included because is outside the scope of this study. The phases are: Initiation, Planning, Control and Monitoring, and Closure.

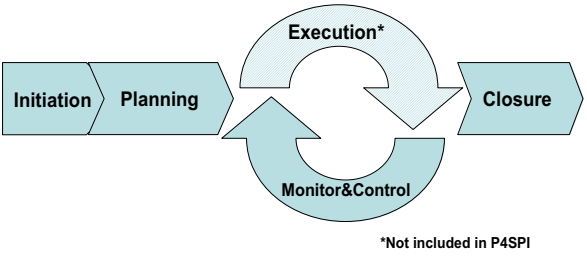


Fig. 1. P4SPI Life cycle

3.2 Initiation Phase

This phase defines an SPI based on the organization's needs, goals and its assessed maturity level. An Initial SPI Case is created using this information, as defined in [17], in order to reflect, at a high level, the organization and improvements goals, a summary of actual maturity level and the scope based on the improvement plan. The activities associated with this phase are:

- **SPI Formal Approach:** This activity is necessary to perform an SPI formal approach according to [14]'s proposal, to determine the organization's goals and improvement objectives. The improvement metrics are determined using the GQ(I)M Model [29].
- **Maturity Assessment:** During this activity an assessment (SCAMPI [2], ISO/IEC15504, etc.) must be performed in order to know the organization's maturity and capability level.
- **Develop Preliminary Project Scope:** This activity is necessary to create an Initial SPI Case based on the information obtained from the SPI formal approach and the deliverables generated by the assessment (Improvement Plan).

3.3 Planning Phase

This phase refines the SPI objectives and performs planning activities to determine an agreed scope. According to Cadle [7], a project is defined as "a managed environment that assures the delivery of a specific product defined in a Business Case". This means that a SPI process improvement must be done according to the agreed constraints: time, cost and resources. An element named Planning SPI Case is a fundamental part of this phase. This element is created and completed through the following activities:

- **SPI Preplanning:** This activity is necessary to perform an SPI Preplanning. According to [13], it suggests the need to determine a preliminary budget in order to justify the project plan, and a rough estimation of resource requirements based on previous management experiences. An agent, who performs an estimation based on queries to an SPI Case Database, obtains the preplanning information or estimation and then generates a Planning SPI Case as an output.
- **Scope Definition:** This activity is necessary to refine the scope generated by the previous activity. It includes modifying the Planning SPI Case, for instance the expected implementation degree of each CMMI PA depending on the specific model and/or the selected generic/specific practices.
- **Determine WBS:** This activity is necessary to determine a WBS based on Planning SPI Case information. A WBS is a hierarchical decomposition of work to be executed by the team in order to accomplish the project goals [30]. It organizes and defines the scope of the SPI. The WBS must include as many work packages as are necessary, having management, training and implementation activities [22].
- **Activities definition:** This activity is necessary to identify and give details of the work packages and activities to be executed. During this activity all the deliverables must be identified.
- **Activity sequencing:** This activity is necessary to identify and document the logical relations between SPI activities.
- **SPI Resource estimation:** This activity is necessary to identify the resources (persons, profiles, equipments or materials) needed and how much of each resource will be used.
- **Activity duration estimation:** This activity is necessary to estimate the amount of effort required to perform the scope of the SPI, based on available resources.
- **Cost estimation:** This activity is necessary to obtain the cost of resources that will execute the project activities.
- **Schedule development:** This activity is necessary to generate an SPI Project Planning by an agent that extracts information from the Planning SPI Case and creates the schedule. Once the file is created, it must be updated regularly. This includes modifying starting and finishing dates. This activity may require reviewing or correcting duration estimation or resources levels. The resulting plan/schedule should be approved by relevant SPI Stakeholders and become the SPI Baseline.
- **Quality Planning:** This activity is necessary for an agent, who uses the information contained in the Planning SPI Case, to generate a House of Quality named QFD4SPI. The QFD4SPI structure and functionalities are based on the model proposed in [16]. QFD4SPI allows control of an SPI, based on its organization goals, improvement goals, and monitors the institutionalization and project scope.

3.4 Control and Monitoring

This phase is necessary to monitor and control the SPI Execution and to report variances from a baseline of schedule, scope, cost, etc. During this phase the institutionalization grade is monitored. This value is very important because the SPI success is measured by the number of people using it [26]. The activities associated with this phase are:

- **SPI Execution Monitoring:** This activity is necessary to monitor the implementation grade of Business Goal, Improvement Goal and Process Area. The monitoring is performed by using questionnaires and the QFD4SPI.
- **Institutionalization Monitoring:** This activity is necessary to monitor the institutionalization grade (IG) for each project, and based on these, a summary for SPI, is determined. A Project's IG is determined by the technical advance of its scheduled activities. The IG monitoring is performed by answering a Questionnaire for each Project included in the SPI. This matrix is included in the QFD4SPI.

		SPI Sheduled Activities																	
Projects	PWeights	Project WBS Developed	Tasks Defined, Resources Assigned	Effort, Costs and Time Estimated	Project Plan Developed	Project Commitment Obtained	Project Plan Updated to Reflect Actual Status	Requirement Changes Validated	Requirement Changes Validated	Requirement Validated Based on Customer Needs	Requirement Changes Updated	Project Scope Statement Established	Project Scope Statement Updated	Accepted Contract Uploaded	Contract Requirements Baselined	Project Lessons Learned Documented	Target Value	Absolute Score	Project IG	Relative Weight
PROY1	7	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	105	35	33%	35%
PROY2	6	1	1	1	1	1	0	1	1	1	0	1	1	0	0	0	90	60	67%	30%
PROY3	5	1	1	1	1	1	0	1	1	0	0	1	0	0	0	0	105	40	38%	35%
SPI Activity Weights		5	5	3	5	5	3	5	5	5	3	5	2	5	5	3				
Target Value		15	15	9	15	15	9	15	15	15	9	15	6	10	10	9				
Absolute score		15	15	9	15	10	0	15	10	5	0	10	2	0	0	0				
Activities IG		100%	100%	100%	100%	67%	0%	100%	67%	33%	0%	67%	33%	0%	0%	0%				
Relative Weight		8%	8%	5%	8%	8%	5%	8%	8%	8%	5%	8%	3%	5%	5%	5%				

Fig. 2. SPI Institutionalization Matrix

- **Schedule Control:** This activity is necessary to monitor the SPI Earned Valued using the information contained in the SPI Planning file. The information is extracted and stored into a metrics database by an agent.
- **Information Distribution:** This activity is necessary to report information to the relevant stakeholders about the implementation grade, the institutionalization and/or SPI Earned value.

3.5 Closure Phase

This phase is necessary to close the SPI through a postmortem assessment and the collection of SPI information from various sources. During this phase a Final SPI Case is elaborated according to [17]. The only activity during this phase is:

- **SPI Closure:** This activity is necessary to extract the information contained in the QFD4SPI and SPI Planning, and then to store it in a specific database. During this activity an SPI postmortem analysis is performed through the development of an Final SPI Case. Once this element is finished, the information will be extracted and stored in a Database SPI Cases by an agent.

3.6 P4SPI Components

The P4SPI is made up of various components (elements, agents) that allow an SPI management throughout all its phases. One component can be used in various phases of P4SPI and are depicted in Figure 3.

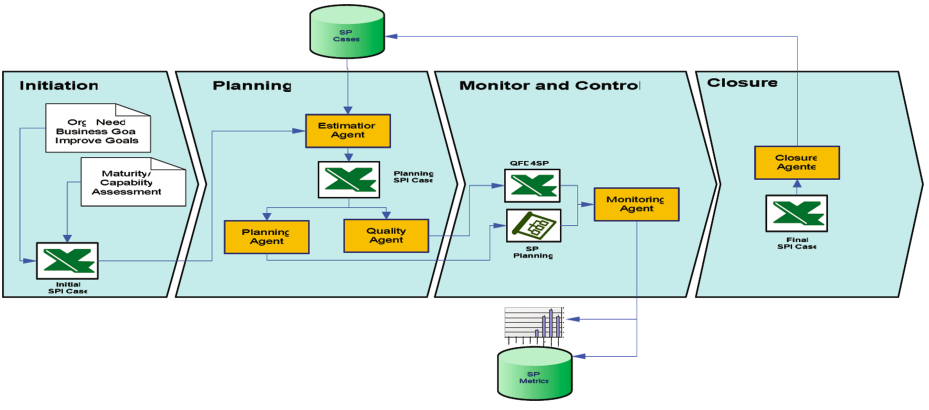


Fig. 3. P4SPI Components

3.6.1 P4SPI Elements

The P4SPI elements are SPI deliverables developed and managed using Microsoft Office tools. The description of each item is as follows:

- **SPI Case:** The SPI Case is a high level proposal to an investment initiative to meet its functional needs and business goals [23]. The purpose of SPI Case is to collect information about SPIs, depending on its phase (Initiation, Planning or Closure). The different structures of SPI Case are defined in [17]. The tool used for creating and updating the SPI Case is Microsoft Word.
- **QFD4SPI:** The QFD4SPI is a set of related houses of quality which implements a technique to monitor and control the technical advance of SPI programs. Its structure and operation are described in [16].This element must be updated regularly through the use of some Questionnaires that reflect the actual implementation and institutionalization grade (IG). This element includes an SPI Institutionalization Matrix, which allows evaluating the IG for each SPI project, and based on each one of them being able to determine a weighed global value of the SPI implementation. The IG of a project is determined by the technical advance degree of its planned activities. The activities are those proposed by the implemented SPI Process Areas that need to be performed on each project. The tool used for updating QFD4SPI is MS Excel.
- **SPI Planning:** The SPI Planning contains the schedule of the project with its planned activities, planned start and finish dates; it allows reflecting the resource level. This element must be updated regularly in order to communicate and control the SPI Earned Value. The tool used for updating the Planning SPI is Microsoft Project.

3.6.2 P4SPI Agents

The P4SPI agents are functions and/or activities that need to be performed by a role/person/function, in order to:

1. Extract information from different sources, including an SPI Cases database.
2. Create/Fill any of the P4SPI elements,

The description of each of the players is as follows:

- **Estimation Agent:** This agent performs an effort analogy estimation [20] from the attributes defined in the Initial SPI Case and the information contained in the SPI Case database (SPICDb) to generate a prediction of efforts that will be reflected in a Planning SPI Case.
  Inputs: Initial SPI Case, SPI Case Database.
  Outputs: Planning SPI Case.
- **Planning Agent:** This agent creates an SPI Planning file with the information contained in the Planning SPI Case, which includes: activities, Start Dates, Finish Dates, Resources, etc.
  Inputs: Planning SPI Case.
  Outputs: SPI Planning.
- **Quality Agent:** This agent creates a QFD4SPI file with the information contained in the Planning SPI Case, which includes: Business Goals, Improvement Goals, Process Areas, etc.
  Inputs: Planning SPI Case.
  Outputs: QFD4SPI.
- **Monitoring Agent:** This agent generates electronics notifications and/or reports based on information from SPI Planning and QFD4SPI
  Inputs: SPI Planning and QFD4SPI.
  Outputs: Electronics mail and/or Notification Reports.
- **Closure Agent:** The agent stores information in the SPI Case Database using the information contained in the Final SPI Cs.
  Inputs: Final SPI Case.
  Outputs: SPI Case Database (updated).

Table 1. The nine key activities mapped to P4SPI

Activities	Initiation	Planning			
	Initial SPI Case	Estimation Agent	Planning SPI Case	Planning Agent	Quality Agent
Data collection		X		X	X
Process Tailoring	X		X		
Process assessment	X				
Problem identification	X	X			
Problem analysis			X		
Process definition			X		
Solution identification			X		
Result measurement	N/A	N/A	N/A	N/A	N/A
Document Management	N/A	N/A	N/A	N/A	N/A

3.6.3 Validating P4SPI Components

Leung concludes [19], that there are nine "Key" activities that any tool that intends to support an SPI program must provide. The following table maps the P4SPI Components with the activities proposed by [19] to ensure the completeness of our proposal:

4 Case Study of CMMI Implementation Using P4SPI

Considering an IT organization that recognized the benefits of implementing CMMI, considering that they already know their strengths, weaknesses and Business Objectives, considering an SPI Project Plan that has been developed by the SEPG Group and that they were ready to start the SPI program, the project sponsors asked to about having some method to plan an SPI program, and know in advance their limitation on funding resources. P4SPI was proposed to plan a CMMI implementation in its small organization. Their organizational scope was limited to Web Development Group, and specifically to a 10-man group dedicated to new development of Internet Web Portals, the technology used for developing are: Macromedia, Java, JavaScript, etc. This organization chose to implement CMMI using the continuous model; they selected to improve the Project Planning, Measuring and Analysis, and Project Monitoring and Control Process Areas. This section describes the 5 steps proposed by P4SPI to cover the SPI Life Cycle.

Step 1: Perform Initial Analysis

Purpose: The objective of this step is to determine the actual needs and organization goals (OG) and to create an Initial SPI Case based on the SPI Plan resulting from assessment and the SPI formal approach.

Description: Once the SPI formal approach and the initial appraisal (Unofficial SCAMPI Class B) were performed in parallel, some needs were identified and some goals were chosen. The Initial SPI Case was created to gather the resulting information from previous activities.

Activities: (1) SPI Formal Approach, (2) Maturity Assessment and (3) Develop Preliminary Project Scope.

Process/Outputs: Organization Goals List, Improvement Goals List, Assessment Report, Improvement Plan, Initial SPI Case.

Step 2: Perform SPI PrePlanning

Purpose: The objective of this step is to determine the effort/cost of performing the SPI and adjusting its values and scope until the organization considers it affordable or cost effective.

Description: During this step an agent, who used the information contained in the Initial SPI Case and in SPICDb, created a Planning SPI Case. This step was repeated several times until the estimated SPI scope was almost similar to the planned SPI scope to be implemented.

Activities: (1) SPI PrePlanning and (2) Scope Definition.

Process/Outputs: Planning SPI Case.

Step 3: Perform SPI Planning

Purpose: The objective of this step was to refine the SPI objectives and perform planning activities to determine an agreed scope and an SPI schedule.

Description: During this step the Planning SPI Case was refined and updated. It included determining the SPI WBS, defining the SPI activities and their sequences, and determining the SPI resources. On completion of these activities, an agent generated two files (SPI Planning and QFD4SPI). The SPI Planning is a Microsoft Project file containing all the activities, starting and finishing dates, human and material resources, and a baseline of SPI. The QFD4SPI is a Microsoft Excel file that contains two houses of quality: (1) to monitor the implementation grade of the OG, Improvement Goals (IG) and Process Areas (PA) [16]; and (2) to monitor the institutionalization grade of the projects using the new SPI Process Areas. The SPI Planning and QFD4SPI were updated after they were created/creation.

Activities: (1) Determine WBS, (2) Activities Definition, (3) Activity sequencing, (4) SPI Resource Estimation, (5) Schedule development, and (6) Quality Planning.

Process/Outputs: SPI Planning and QFD4SPI.

Step 4: Perform SPI Monitoring

Purpose: The objective of this step is to perform different monitoring activities to review the earned value, the institutionalization grade and the implementation grade.

Description: This step was repeated until the SPI reached its closure phase. This step was performed before every project meeting or SPI reporting activity. Both files (SPI Planning and QFD4SPI) were updated to reflect the actual situation of SPI. After the data was updated, an agent extracted specific information from each file, and created several records in the Metrics Database. Electronics reports that were sent to SPEG were created using this information.

Activities: (1) SPI Execution Monitoring, (2) Institutionalization Monitoring, (3) Schedule Control and (4) Information Distribution.

Process/Outputs: SPI Planning (updated), QFD4SPI (updated) and Emails/reports

Step 5: Perform SPI Closure

Purpose: The objectives of this step were to perform a postmortem assessment and gather qualitative and quantitative information and store it in the SPICDb.

Description: During this step a postmortem assessment using a Final SPI Case was performed. Once the assessment was completed, an agent extracted the information and stored it in the SPICDb.

Activity: SPI Closure.

Process/Outputs: A Final SPI Case and the SPI Case Database (updated).

5 Conclusions

This paper presents a practical procedure to manage an SPI implementation as a normal project throughout its phases. The procedure determines the necessary activities to plan, monitor and close an SPI. The P4SPI Procedure has answered the research questions because its definition includes activities to perform planning, monitoring and closure of an SPI. The P4SPI life cycle is based on the projects Life Cycle presented in [PMI,2004]. The necessary components to apply P4SPI were developed with Microsoft office application to ensure its use and accessibility by the SPI Manager and the SPEG Group.

P4SPI was applied in an organization and it has demonstrated:

- That is possible to manage and plan an SPI as a normal project.
- That P4SPI serves as guidance for an SPI implementation.
- That SPI Implementation can be facilitated by providing tools to SPI Managers.
- That using P4SPI, the success of SPI is ensured by monitoring and controlling its implementation and institutionalization.

As a future research work, the integration of all components defined in different proposals ([14], [16], [17]) within a utility that provides an integrated SPI management framework is being considered.

References

[1] Aaen, I., Arent, J., Mathiassen, L., Ngwenyama, O.: A Conceptual MAP of Software. Process Improvement, Scandinavian Journal of Information Systems 13(1) (2001)
[2] Ahern, D.: CMMI SCAMPI Distilled. Addison Wesley, Reading (2005)
[3] Appleton, B.: Patterns for Conducting Process Improvement. In: PLoP 1997 conference (1997)
[4] Babacan, B.: An Application of 6 Sigma Methodology for Project Management Process Improvement. In: PICMET 2006 Proceedings (2006)
[5] Blair, G.: Planning a project. Engineering Management Journal (1993)
[6] Making Software Process Improvement Happen. Gothenburg Studies in Applied Information Technology. IT University of Gothenburg. Doctoral Dissertation (2006)
[7] Cadle, J., Yeates, I.: Project Management for Information Systems, 3rd edn. Pearson Education Limited, Sydney (2002)
[8] Clark, T.A.: Project Management for Planners: A Practical Guide. Planners Press, American Planning Association (2002)
[9] Cuevas, G., Gil, M.Á.: Mejora rápida en los procesos de desarrollo de software, II Congreso Nacional de Ingeniería de Telecomunicación. Madrid (1998)
[10] Jiang, J., Klein, G.: An exploration of the relationship between software development process maturity and project performance. Information & Management 41 (2003)
[11] Emam, K., Briand, L.: Costs and benefits of Software Process Improvement. International Software Engineering Research Network technical report (1997)
[12] Kautz, H., Hansen, W., Thaysen, K.: Applying and adjusting a software process improvement model in practice. In: ICSE 2000, pp. 626–633 (2000)
[13] Kwon, S., Shin, K.: PPSS: CBR System for ERP Project Preplanning. In: Kim, T.G. (ed.) AIS 2004. LNCS (LNAI), vol. 3397, pp. 157–166. Springer, Heidelberg (2005)

[14] Garcia, J.: Aproximación Formal para la mejora de procesos de software. Univesidad Carlos III de Madrid. Tesis Doctoral (2001)
[15] GartnerGroup, Moderate Process Rigor Is Faster (In the Long Run..). Stanford GartnerGroup (2000)
[16] Landaeta, J., Garcia, J., Amescua, A.: QFD4SPI: A Technique to Monitor and Control Software Process Improvement Programs. In: EuroSPI 2007 (2007)
[17] Landaeta, J., Garcia, J., Amescua, A.: Estimación Formal de un SPI. Universidad Carlos III de Madrid (2008)
[18] Leung, H.: Slow change of information system development practice. Software Quality Journal 8(3), 197–210 (1999)
[19] Leung, H., Liao, L., Qu, Y.: Automated support of software quality improvement. International Journal of Quality & Reliability Management 24(3), 230–243 (2007)
[20] Li, J., Ruhe, G.: Decision Support Analysis for Software Effort Estimation by Analogy. In: Third International Workshop on Predictor Models in Software Engineering. IEEE, Los Alamitos (2007)
[21] Malan, A., Pretorius, L., Pretorius, J.: A Framework for Increasing Project Maturity and Capability in Southern Africa. In: PICMET 2007 Proceedings (2007)
[22] McFeeley, R.: IDEAL: A User's Guide for Software Process Improvement. Software Engineering Institute. Carnegie Mellon University (1996)
[23] Miller, K.: Case Study: Simulation of the Call Center Environment for comparing competing call routing technologies for Business Case ROI Projection (1999)
[24] Murugappan, M., Keeni, G.: Blending CMM and Six Sigma to Meet Business Goals. IEEE SOFTWARE (2003)
[25] Niazi, M., Wilson, D., Zowghi, D.: A maturity model for the implementation of SPI. The Journal of Systems and Software 74, 155–172 (2003)
[26] Niazi, M., Wilson, D., Zowghi, D.: A framework for assisting the design of effective software process improvement implementation strategies. Journal of Systems and Software 78(2), 204–222 (2005)
[27] Li, J., Ruhe, G.: Decision Support Analysis for Software Effort Estimation by Analogy. In: Workshop on Predicto Models in Software Engineering (PROMISE 2007). IEEE, Los Alamitos (2007)
[28] Pan, Z., Park, H., Baik, J., Choi, H.: A Six Sigma Framework for Software Process Improvements and its Implementation. In: Software Engineering Conference. IEEE, Los Alamitos (2007)
[29] Park, R.E., Wolfhart, B., Geothert, W., Florac, A.: Goal-driven Software Measurement, software engineering institute, handbook CMU/SEI-96-HB-002 (1996)
[30] PMI: A Guide to the Project Management Body of Knowledge, 2000 Edition and Third Edition, 2004. Newtown Square: Project Management Institute, USA (2000 & 2004)
[31] Sivi, J., Penn, M., Harper, E.: Relationships Between CMMI and Six Sigma. Software Engineering Institute, Carnegie Mellon University (2005)
[32] Rehessar, H.: Project Management Success Factors. University New South Wales (1996)
[33] Zahran, S.: Business and cost justification of software process improvement (2000)
[34] Zhang, H., Kitchenham, B., Jeffery, R.: Planning Software Project Success with Semi-Quantitative Reasoning. In: Proceedings of the 2007 Australian Software Engineering Conference (ASWEC 2007). IEEE, Los Alamitos (2007)

Analysis of Dependencies between Specific Practices in CMMI Maturity Level 2

Xi Chen, Mark Staples, and Paul Bannerman

NICTA, Australian Technology Park, Eveleigh, NSW 2015, Australia
School of Computer Science and Engineering,
University of New South Wales, Australia
{xi.chen,mark.staples,paul.bannerman}@nicta.com.au

Abstract. CMMI contains a collection of Process Areas (PAs), each of which contains many Specific Practices (SPs). However, the CMMI specification does not provide any explicit recommendation about which individual SPs can or should be implemented before other SPs. In this paper we identify dependencies between CMMI SPs in PAs in maturity level 2, and between the PAs. We analyzed the text of the CMMI specification to identify every Work Product (WP) produced and used by every SP in maturity level 2. Our analysis was validated by independent researchers and comparison with an existing dependency analysis shown in CMMI training materials. Our results have significance as a reference model of SP and PA dependencies for both SPI researchers and practitioners. For researchers we have provided an explicit representation of SP and PA dependencies that were previously only implicit in the CMMI specification. For practitioners, our results may provide guidance on the order of implementation of SPs and PAs. Our dependency analysis has limitations in being derived from the text of the CMMI specification – we have no direct evidence that these dependencies are valid in practice.

Keywords: SPI, CMMI, Specific Practice, Work Products, Dependency.

1 Introduction

The CMMI [1] Software Process Improvement (SPI) and assessment model has been adopted and implemented in many companies [2, 3]. Some report great impact and benefit in helping reduce cost, increase productivity, and improve performance. Other researchers [4, 5, 6] claim that CMMI has limitations because of its complexity, time-consuming adoption, and costly services. Furthermore, others object that CMMI results in excessive documentation and interferes with developers' creativity [6].

Wilkie et al. [7] have investigated the use of the Specific Practices (SPs) of six Process Areas (PAs) in CMMI maturity level 2 within six small software development companies. They found that companies did not pursue all CMMI SPs equally, and proposed "perceived value" as an indicator of preferences of companies in adopting SPs. This suggests that a better understanding of CMMI SPs may improve appraisal and adoption approaches. It also implies that an alternative approach to using

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 94–105, 2008.

CMMI could be based on SPs, rather than the higher level of whole PAs or overall maturity levels.

In CMMI version 1.2 [1], 173 SPs are grouped under 22 PAs. Regardless of whether a company uses the Staged or Continuous Representation of CMMI to improve their process, they will be required to implement all of the SPs within a group of PAs. However, the CMMI specification [1] provides no explicit description of dependencies between SPs in any group of PAs (i.e. required or suggested order of implementation).

In this paper, we present an analysis of the dependencies between CMMI SPs in PAs in maturity level 2. Our analyses are based on the text of the CMMI specification [1] and are based on the assumption that a dependency exists between two SPs when one SP uses as an input a Work Product (WP) that is produced as an output from the other SP. We also derive a view of the dependencies between PAs by analyzing dependencies between their SPs. To validate our results we compare the relationships we identified to those presented in a CMMI tutorial [8].

The proposed dependency model has significance for both researchers and practitioners. Researchers may be able to use our results to qualify their understanding of the adoption of CMMI-based SPI, and to inform research regarding the structured representation of CMMI. Practitioners may be able to use our results as informative guidance in determining the order of implementation of SPs and PAs.

The remainder of this paper is organized as follows. In section 2, we discuss how WPs relate to SPs in CMMI, and how we classify WPs in this paper. In section 3, we describe how the relationship map of CMMI SPs in maturity level 2 is generated. In section 4 we compare our results to existing material from a CMMI tutorial. We conclude in section 5.

2 Work Products of CMMI Specific Practices

In CMMI version 1.2 [1], a Work Product (WP) is the result or output of a SP. Instead of merely tangible products or parts of products that are to be delivered to customers, WPs in the CMMI model can represents a much broader range of products, including files, documents, specification, process descriptions, as well as services. Some of these might not be part of the product package that the customer eventually receives.

SPs can not be implemented without sufficient resources, and a WP from one SP can be an input resource for another SP. This output-input relationship provides a basis for expressing dependency relationships between SPs in terms of WPs. We have classified WPs according to the role they play in SPs as described in the CMMI specification [1]:

- ***Input Work Product (InpWP)***
  An Input WP of a SP is not produced by this SP but is necessary to carry out the SP.
- ***Output Work Product (OWP)***
  An Output WP of a SP is produced when executing this SP. It is not part of any other WP and does not help produce another WP in the same SP.

- ***Internal Work Product (IntWP)***
  An Internal WP of a SP is produced when executing this SP. It is either part of another WP generated from this SP or it helps produce another WP in the same SP.

3 Work Product-Based CMMI Specific Practice Dependencies

3.1 Methodology

We divided our work into two stages. In stage 1, by using the categorization of WPs from Section 2, two of the researchers independently analyzed the specification of CMMI (v1.2) and identified Input WPs and WPs that are produced for each SP. Then, for WPs generated by each SP, the two researchers separated them into Internal WPs and Output WPs. When WPs were identified, they were recorded in a spreadsheet by category for each PA. The researchers compared their results and resolved differences in a joint meeting. In stage 2, we calculated the relationships between SPs by using the spreadsheet from stage 1 as input to the open source tool Graphviz, to draw the relationships graphs. These results were later validated as described in section 4.1 below.

3.2 Specific Practice Dependencies within Process Areas

Figures 1 to 7 show the mapping of dependencies for SPs in seven PAs in CMMI maturity level 2, namely: Configuration Management (CM), Measurement and Analysis (MA), Project Management & Control (PMC), Project Planning (PP), Process and Product Quality Assurance (PPQA), Requirement Management (REQM), and Supplier Agreement Management (SAM). Tables 1 to 7 identify the ID labels listing WPs produced by one SP that are used by another SP. IDs and WPs marked with a grey background mean additionally identified dependencies, compared to the CMMI tutorial material, which will be discussed in section 4.

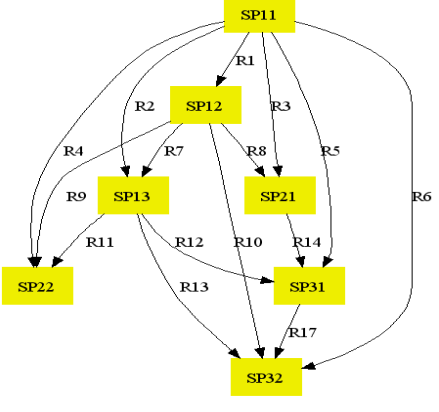


Fig. 1. SP dependencies in CM

Table 1. Dependent WPs in CM

ID	WP
R1	Identified configuration items
R2	Identified configuration items
R3	Identified configuration items
R4	Identified configuration items
R5	Identified configuration items
R6	Identified configuration items
R7	Configuration management system
R8	Change request database
R9	Configuration management system
R10	Configuration management system
R11	Baselines
R12	Baselines
R13	Baselines
R14	Change requests
R17	Configuration Management Records

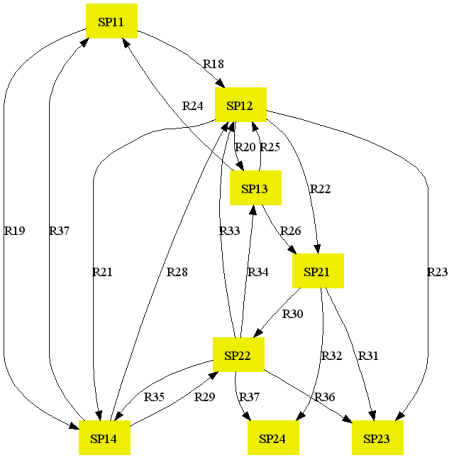


Fig. 2. SP dependencies in MA

Table 2. Dependent WPs in MA

ID	WP
R18	Measurement objectives
R19	Measurement objectives
R20	Specifications of base and derived measures
R21	Specifications of base and derived measures
R22	Specifications of base and derived measures
R23	Specifications of base and derived measures
R24	Data collection and storage procedures
R24	Updated measures & measurement objectives
R25	Updated measures & measurement objectives
R26	Data collection and storage procedures
R26	Data collection tools
R27	Updated measures & measurement objectives
R28	Updated measures & measurement objectives
R29	Analysis specifications and procedures
R29	Data analysis tools
R30	Base and derived measurement data sets
R31	Base and derived measurement data sets
R32	Base and derived measurement data sets
R33	Refined criteria for spec'n of measurement
R34	Refined criteria for spec'n of data collection
R35	Refined criteria for spec'n of data analysis
R36	Analysis results and draft reports
R37	Analysis results and draft reports

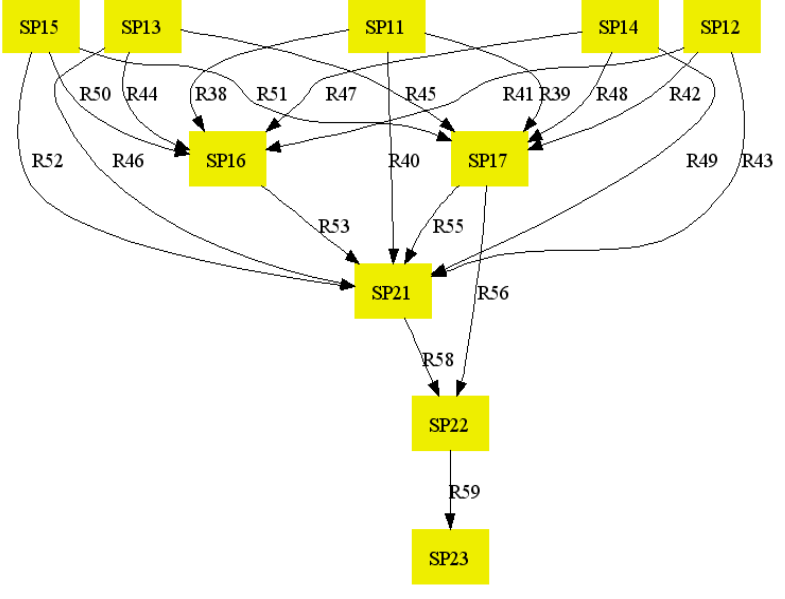


Fig. 3. SP dependencies in PMC

Table 3. Dependent WPs in PMC

ID	WP	ID	WP
R38	Records of project performance	R45	Records of project risk monitoring
R38	Records of significant deviations	R46	Records of project risk monitoring
R39	Records of project performance	R47	Records of data management
R39	Records of significant deviations	R48	Records of data management
R40	Records of project performance	R49	Records of data management
R40	Records of significant deviations	R50	Records of stakeholder involvement
R41	Records of commitment reviews	R51	Records of stakeholder involvement
R41	Records of commitments that are either unsatisfied or of significant risks of not being satisfied	R52	Records of stakeholder involvement
R42	Records of commitment reviews	R53	Documented project progress review results
R42	Records of commitments that are either unsatisfied or of significant risks of not being satisfied	R55	Documented milestone review results
R43	Records of commitment reviews	R56	Documented corrective action items
R43	Records of commitments that are either unsatisfied or of significant risks of not being satisfied	R58	List of issues needing corrective actions
R44	Records of project risk monitoring	R59	Corrective action plan

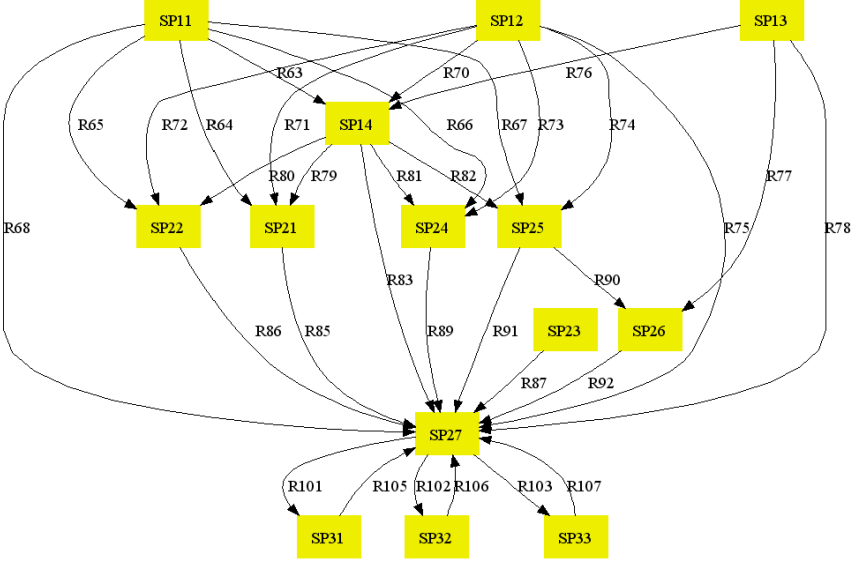


Fig. 4. SP dependencies in PP

Table 4. Dependent WPs in PP

ID	WP	ID	WP
R63	Task descriptions Work package descriptions Top level WBS Documentation of product or product components that will be externally acquired or reused	R83	Documentation of uniqueness of the project Project effort estimates Project cost estimates
		R85	Project schedules
R64	Top level WBS		Schedule dependencies
R65	Top level WBS		Project budget
R66	Top level WBS		Identified constraints on flexibility of management options
R67	Top level WBS		Corrective action criteria
R68	Task descriptions Work package descriptions Top level WBS Documentation of product or product components that will be externally acquired or reused	R86	Identified risks Risk impacts and probability of occurrence Risk priorities Agreement with relevant stakeholders on the completeness and correctness of the documented risks
R70	Technical approach Estimates of attributes of WPs and tasks	R87	Data management plan
		R89	WBS work packages WBS task dictionary Staffing requirements based on project size and scope
R71	Technical approach Estimates of attributes of WPs and tasks		Critical facilities or equipment list Process or workflow definitions and diagrams
R72	Technical approach Estimates of attributes of WPs and tasks		Program administration requirements list
R73	Technical approach Estimates of attributes of WPs and tasks	R90	Inventory of skill needs
R74	Technical approach Estimates of attributes of WPs and tasks		Staffing and new hire plans Databases for available and needed skills knowledge and training
R75	Technical approach Estimates of attributes of WPs and tasks	R91	Inventory of skill needs Staffing and new hire plans Databases for available and needed skills knowledge and training Mechanisms for providing needed knowledge and skills
R76	Project's lifecycle phases		
R77	Project's lifecycle phases		
R78	Project's lifecycle phases	R92	Stakeholder involvement plan
R79	Project effort estimates	R101	Overall project plan
	Project cost estimates	R102	Overall project plan
R80	Project effort estimates	R103	Overall project plan
	Project cost estimates	R105	Record of the reviews of plans that affect the project
R81	Project effort estimates Project cost estimates	R106	Revised overall project plan
		R107	Documented commitments from relevant stakeholders responsible for performing and supporting plan execution
R82	Project effort estimates		

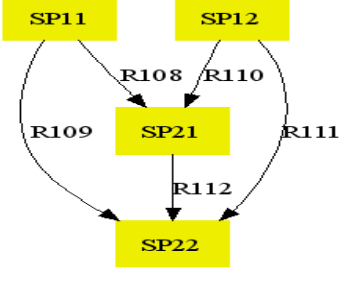


Fig. 5. SP dependencies in PPQA

Table 5. Dependent WPs in PPQA

ID	WP
R108	Process noncompliance reports
R109	Process evaluation reports Process noncompliance reports Process corrective actions
R110	WPs noncompliance reports
R111	WPs evaluation reports WPs noncompliance reports WPs corrective actions
R112	Corrective action reports Evaluation reports Quality trends analysis Documented noncompliance issues when they cannot be resolved within the project

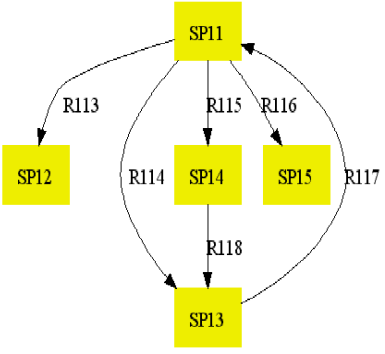


Fig. 6. SP dependencies in REQM

Table 6. Dependent WPs in REQM

ID	WP
R113	An agreed to set of requirements
R114	An agreed to set of requirements
R115	An agreed to set of requirements
R116	An agreed to set of requirements
R117	Revised requirement
R118	Requirements traceability matrix

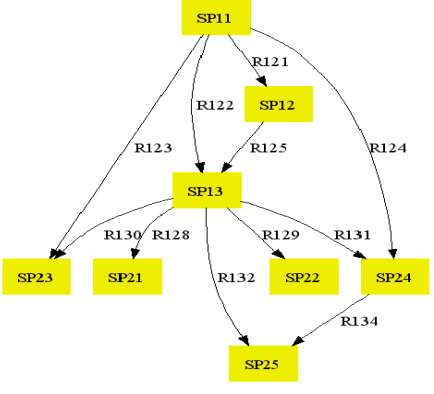


Fig. 7. SP dependencies in SAM

Table 7. Dependent WPs in SAM

ID	WP
R121	List of the acquisition types Supplier requirements
R122	Supplier requirements
R123	Supplier requirements
R124	Supplier requirements
R125	Preferred supplier list
R128	Supplier agreement document
R129	Supplier agreement document
R130	Supplier agreement document
R131	Supplier agreement document
R132	Supplier agreement document
R134	Accepted WPs

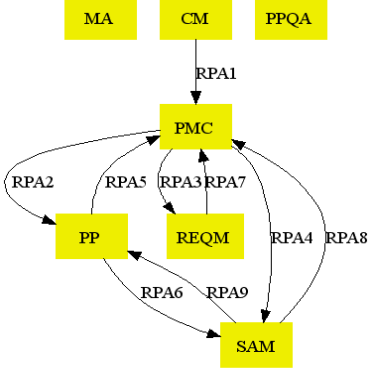


Fig. 8. Relationships between PAs in CMMI ML2

Table 8. Dependent WPs for SPs across PAs in CMMI ML2

ID	WP	ID	WP
RPA1	Change requests	PRA6	Documentation of product or product components that will be externally acquired or reused
RPA2	Documented project progress review results		Data management plan Stakeholder involvement plan Overall project plan
	Documented milestone review results Revised overall project plan	RPA7	Documentation of inconsistencies between requirements and project plan and WPs including sources conditions and rationale Corrective actions
RPA3	Revised requirement	RPA8	Supplier agreement document
RPA4	Changes to internal and external commitments		Document actions needed to addressed deficiencies during the evaluation
RPA5	Corrective action criteria	RPA9	Revision of project plans
	Overall project plan		

3.3 Specific Practices Dependencies Across Process Areas

Figure 8 shows the interrelationships between each PA in CMMI maturity level 2 based on the SP inter-dependencies. The identifier on each edge represents one or many dependencies between SPs. The labels are identified in Table 8.

4 Discussion

4.1 Interpreting the Model

Figures 1 to 7 suggest the order of adoption of SPs in each PA. The accompanying tables show the WPs that constitute the relationships, i.e. the WPs that are produced by one SP and used by another. When implementing each PA, one could start from those SPs that have no "parent" SP. Then, SPs could be implemented whose parents' SP have all been implemented. This can continue until all the SPs in a PA have been adopted.

Taking Figure 5 and Table 5 for example, in PPQA, (Product and Process Quality Assurance), the dependencies model suggests that practitioners should start by implementing PPQASP11, (Objectively Evaluate Processes), and PPQASP12, (Objectively Evaluate Work Products and Services). The "Process noncompliance reports" (R108 in Table 5) output from PPQASP11, and "WPs noncompliance reports" (R110) output from PPQASP12 will become the input WPs of PPQASP21, (Communicate and Ensure Resolution of Noncompliance Issues). Finally, PPQASP22, (Establish Records) will take "Process evaluation reports", "Process noncompliance reports", and "Process corrective actions" produced by PPQASP11 (R109), "WPs evaluation reports", "WPs noncompliance reports", and "WPs corrective actins" produced by PPQASP12 (R111), and "Corrective action reports", "Evaluation reports", "Quality rends analysis", and "Documented noncompliance issues when they cannot be resolved within the project" produced by PPQASP21 (R112), as input WP, and so PPQASP22 should be implemented last.

4.2 Validation

Consistency in the interpretation of independent researchers offers *prime facie* validation for the relationships found. However, we also compared the results to the intra-PA SP relationships identified in a CMMI (version 1.1) tutorial [8]. Our results showed some differences to the tutorial material. These differences are explained in Table 9.

Differences of type D1 (additional dependencies not recognized in the tutorial) are highlighted with a grey background in Tables 1-7. As there are slight differences between the PA of Supplier Agreement Management in CMMI version 1.1 and 1.2, we also regarded dependencies of newly added SPs as additionally recognized relationships. Table 10 shows the details of differences found for types D2 and D3.

The differences identified between our findings and the tutorial's dependencies do not invalidate our findings. Table 9 explains the differences found. The differences arise because the tutorial is intended for teaching rather than specification purposes, and because our analysis was systematic, based on the text of the CMMI specification. We compared our inter-PA relationships with those of the tutorial, and found no differences.

Table 9. Clarification of differences

Type	Number	Description	Explanation
D1	109	We recognized more dependencies than the tutorial.	The tutorial was intended to be instructional. It showed most important WP dependencies, but did not give exhaustive information.
D2	1	The tutorial recognized more dependencies than us.	The tutorial's extra dependency was not explicitly described in the CMMI specification but may have been implicit.
D3	14	Both analyses recognized the same dependency, but in a different direction.	The tutorial's converse dependency was not explicitly described in the CMMI specification. WPs modified by a SP were sometimes considered by the tutorial to depend on that SP, despite being able to exist in principle without the SP.

Table 10. Analysis of differences of type D2 and D3 in each Process Area

PA	Type ID	Dependencies	
		From	to
CM	D2	Configuration Management System	SP 3.1
		SP 3.1	Configuration Management System
	D3	Configuration Management System	SP 1.2
		Change Request Database	SP 1.2
		SP 1.3	Configuration Management System
		SP 2.1	Change Request
		SP 2.2	Change Request Database
MA	D3	SP 1.2	Measurement Repository
		SP 2.1	Measurement Repository
		SP 2.3	Measurement Indicator
		SP 2.4	Measurement Indicator
PP	D3	SP 3.3	Project Plans
REQM	D3	SP 1.1	Requirements
		SP 1.2	Requirements
		SP 1.4	Requirements

4.3 Problems with Completeness and Terminology in the CMMI Specification

When performing the textual analysis of the CMMI specification, one limitation we found in the specification was that some WPs were not explicitly stated for some SPs, but nevertheless would probably be required to perform those SPs. To guard the integrity of our findings, we excluded this group of WPs from the analysis. This suggests opportunities for improving the completeness of future versions of the CMMI specification, and implies that companies should use judgment in interpreting and applying CMMI to suit their development contexts.

Another difficulty encountered in this research concerned the ambiguity of terminology within the CMMI specification. Sometimes individual WPs were identified with different names in the CMMI specification. This ambiguity initially resulted in missing dependencies within the specification, which were later found and corrected during the review process in joint meetings. A more systematic solution to the problem of ambiguity would be to use a glossary of consistent nomenclature for WPs in the CMMI specification. The use of a process meta-model for the CMMI specification

could additionally describe relationships between different WPs and between WPs and SPs.

4.4 Discussion of Specific Practice Dependencies and Process Area Relationships

Figure 8 shows that PP and PMC are key PAs in CMMI Maturity Level 2, which closely interact with other PAs by providing and using shared WPs. From Table 8 we see that PP is important in directing the production of WPs in other SPs, while the practices of PMC revise and change WPs from other SPs. The Support PAs (CM, MA, and PPQA) are mostly isolated in our relationships graph. This is because these PAs take "Any WP" as inputs. We have not shown these as links explicitly in our graph. In practice they can interact with every other PA in CMMI.

The following brief observations are made about each PA, based on the SP inter-dependencies mapped in Figures 1 to 7:

- For CM (Figure 1) the order of SP is mostly linear, with no cyclic dependencies.
- MA (Figure 2) has many cyclic dependencies, especially in SG1, as WPs from each of the SPs can become criteria for the others to consider. The CMMI specification comments that companies often adopt all SPs in MA SG1 concurrently [1, p180].
- The graph for PMC (Figure 3) does not show any cyclic dependencies within the PA itself, but as discussed above, PMC is very important in interactions with other PAs, and there are cyclic dependencies with other PAs, as shown in Figure 8.
- In PP (Figure 4), we see that SP2.7 is an important SP. It takes outputs from many other SPs in the PA to produce the "Project Plan" WP, which is central in the project. The Project Plan is used and revised in many other SPs in other PAs.
- PPQA (Figure 5) has a very simple structure with no cycles.
- REQM (Figure 6) has a cycle between SP 1.1 and SP 1.3, because requirements change as the project evolves.
- SAM (Figure 7) has a key SP 1.3 which produces the "Supplier Agreement" WP as the basis for managing suppliers, and receives feedback from other PAs to maintain it, as shown in Figure 8.

5 Conclusion

In this paper, we identify SP dependencies in PAs in CMMI maturity level 2, by noting the WPs used and produced by each SP, as described in the CMMI specification [1]. We validated our dependency analysis by having two researchers independently perform the textual analysis, and by comparing the results to the relationships presented in a CMMI tutorial [8]. Our results provide more detail on dependencies than is shown in the tutorial.

The dependency model developed has significance for researchers and practitioners as a reference model. SPI researchers investigating the adoption and implementation of CMMI SPs will have a basis for understanding possible SP inter-dependencies. SPI practitioners may benefit by being informed about possible implementation

dependencies between PAs and between SPs within PAs. We have shown an example of how the dependencies model can be used to suggest an order for implementing SPs.

Future research will extend our analysis to PAs in CMMI maturity levels 3 to 5. It will also combine the relationships identified here with an outcome-based SP categorization [9] to propose a goal-oriented method for planning the implementation of CMMI SPs. Other research may investigate the structured representation of CMMI in a process meta-model, to derive a CMMI model to express SP dependencies rigorously and be interpreted consistently. Further work is needed to evaluate the utility of our dependency model for practitioners. For example, how can it be applied for practitioners to better understand the relationships between SPs and between PAs? Empirical studies are also required to determine if there are benefits to using this dependency model when implementing SPI.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI: Guidelines for Process Integration and Product Improvement, 2nd edn. Addison-wesley, Reading
2. Goldenson, D., Gibson, D.: Demonstrating the Impact and Benefits of CMMI – An Update and Preliminary Results. Technical Report, CMU/SEI-2003-SR-009, Software Engineering Institute, CMU (2003)
3. Gibson, D.L., Goldenson, D.R., Kost, K.: Performance Results of CMMI-Based Process Improvement. Technical Report, CMU/SEI-2006-TR-004, Software Engineering Institute, CMU (2006)
4. Reifer, D.J.: The CMMI: it's formidable. The Journal of Systems and Software 50, 97–98 (2000)
5. Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., Murphy, R.: An Exploratory Study of Why Organizations Do Not Adopt CMMI. Journal of Systems and Software 80(6), 883–893 (2007)
6. Turgeon, J.: CMMI on the Sly for the CMMI Shy - Implementing Software Process Improvement in Small Teams and Organizations. In: SEPG (2006)
7. Wilkie, F.G., McFall, D., McCaffery, F.: An Evaluation of CMMI Process Areas for Small-to Medium-sized Software Development Organizations. Software Process: Improvement and Practice 10(2), 189–201 (2005)
8. Phillips, M.: CMMI V1.1 and Appraisal Tutorial (February 16, 2005)
9. Chen, X., Staples, M.: Using Practice Outcome Areas to Understand Perceived Value of CMMI SPs for SMEs. In: Abrahamsson, P., Baddoo, N., Margaria, T., Messnarz, R. (eds.) EuroSPI 2007. LNCS, vol. 4764, pp. 59–70. Springer, Heidelberg (2007)

Appendix: CMMI Maturity Level 2 Specific Practice Glossary

Configuration Management (CM)	
SP11	Identify Configuration Items
SP12	Establish a CM System
SP13	Create or Release Baselines
SP21	Track Change Requests
SP22	Control Configuration Items
SP31	Establish CM Records
SP32	Perform Configuration Audits
Measurement and Analysis (MA)	
SP11	Establish Measurement Objectives
SP12	Specify Measures
SP13	Spec. Data Collection & Storage Procs
SP14	Specify Analysis Procedures
SP21	Collect Measurement Data
SP22	Analyze Measurement Data
SP23	Store Data and Results
SP24	Communicate Results
Project Management and Control (PMC)	
SP11	Monitor Project Planning Parameters
SP12	Monitor Commitments
SP13	Monitor Project Risks
SP14	Monitor Data Management
SP15	Monitor Stakeholder Involvement
SP16	Conduct Progress Reviews
SP17	Conduct Milestone Reviews
SP21	Analyze Issues
SP22	Take Corrective Action
SP23	Manage Corrective Action
Process and Product Quality Assurance (PPQA)	
SP11	Objectively Evaluate Processes
SP12	Objectively Evaluate Work Products and Services
SP21	Communicate and Ensure Resolution of Noncompliance Issues
SP22	Establish Records

Project Planning (PP)	
SP11	Estimate the Scope of the Project
SP12	Establish Estimates of WP and Task Attributes
SP13	Define Project Life Cycle
SP14	Determine Estimates of Effort and Cost
SP21	Establish the Budget and Schedule
SP22	Identify Project Risks
SP23	Plan for Data Management
SP24	Plan for Project Resources
SP25	Plan for Needed Knowledge and Skills
SP26	Plan Stakeholder Involvement
SP27	Establish the Project Plan
SP31	Review Plans that Affect the Project
SP32	Reconcile Work and Resource Levels
SP33	Obtain Plan Commitment
Requirement Management (REQM)	
SP11	Obtain an Understanding of Requirements
SP12	Obtain Commitment to Requirements
SP13	Manage Requirements Changes
SP14	Maintain Bidirectional Traceability of Reqs
SP15	Identify Inconsistencies between Project Work and Requirements
Supplier Agreement Management (SAM)	
SP11	Determine Acquisition Type
SP12	Select Suppliers
SP13	Establish Supplier Agreements
SP21	Execute the Supplier Agreement
SP22	Monitor Selected Supplier Processes
SP23	Evaluate Selected Supplier Work Products
SP24	Accept the Acquired Product
SP25	Transition Products

A Solution for Establishing the Information Technology Service Management Processes Implementation Sequence

Magdalena Arcilla[1], Jose Calvo-Manzano[2], Gonzalo Cuevas[2], Gerzon Gómez[3], Elena Ruiz[1], and Tomás San Feliu[2]

[1] Universidad Nacional de Educación a Distancia,
Escuela Técnica Superior de Ingeniería Informática
{marcilla,elena}@issi.uned.es
[2] Universidad Politécnica de Madrid, Facultad de Informática
{jacalvo,gcuevas,tsanfe}@fi.upm.es
[3] Universidad Autónoma de Tamaulipas, Unidad Reynosa Rodhe
ggomez@uat.edu.mx

Abstract. This paper addresses the implementation sequence of Services Management processes defined in ITIL v2, from a topological perspective. Graphs Theory is used to represent the existing dependencies among the ITIL v2 processes, in order to find clusters of strongly connected processes. These clusters will help to determine the implementation priority of the service management processes. For it, OPreSSD (Organizational Procedure for Service Support and Service Delivery) is proposed in order to identify the processes implementation sequence related to the Service Support (SS) and Service Delivery (SD) areas.

Keywords: ITIL, Service Delivery, Service Support, Service Management.

1 Introduction

Nowadays, organizations are depending more and more on IT services to satisfy their corporative objectives and cover their business needs [1] [2] [3]. This trend shows that Information Technology Services Management (ITSM) is becoming an important factor for the success of business in many organizations. When ITSM is not suitable for companies or does not work as it should be, it means a huge daily cost for the company or loss of productivity and new opportunities, and an increase in service costs [1].

Although this situation is not new, it has motivated the emergence of process models to manage IT since the 80's. One of these models was the Information Technology Infrastructure Library (ITIL) developed by the Office of Government Commerce of the United Kingdom (OGC-UK) [2] [3]. ITIL is a generic framework constituted by a library of best practices for ITSM focusing on Service Delivery [2] (SD) and Service Support [3] (SS).

ITIL is one of the most used and widely disclosed models at international level. It has been adopted by large companies, like IBM [5], Microsoft [6], SUN [8] and HP [8] among others, as the foundation for the development of their own models of ITSM. These companies apply their own experiences to implement ITIL processes.

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 106–116, 2008.

ITIL v2 does not explicitly describe the processes implementation sequence. This situation has created the need for a procedure in order to get the ITSM processes implementation sequence.

This research work addresses the implementation sequence of Services Management processes defined in ITIL v2, from a topological perspective. Graphs Theory is used to represent the existing dependencies among the ITIL processes, in order to find clusters of strongly connected processes. These clusters will help to determine the implementation priority of the services processes.

This paper is organized as follows. Section 2 shows a brief description on the organizational proposal of ITIL service management processes. Section 3 describes the OPreSSD (Organizational Procedure for Service Support and Service Delivery) procedure developed in this research work and the results obtained. Finally, section 4 establishes a brief summary and future study.

2 Organization and Structure of the ITSM Processes

The two main areas of Service Management are Service Delivery (SD) and Service Support (SS). ITIL v2 provides a set of best practices for ITSM, promoting a quality approach to obtain effectiveness and efficiency in the use of Information Systems. (Fig. 1 shows processes[1] that compose IT Service Management areas).

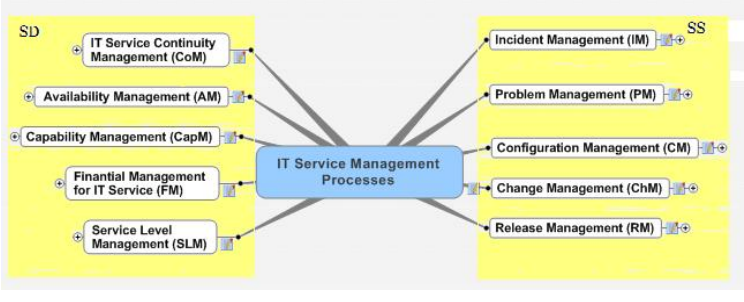


Fig. 1. IT Service Management Processes

The Service Delivery area looks at what service the business requires of the provider in order to provide adequate support to the business User. The Service Support area is concerned with ensuring that Customer has access to the appropriate services to support the business functions [2].

All processes shown in Fig. 1 have a common structure: objectives, scope, basic concepts and activities. Some processes also describe the related processes (although in ITIL official documentation there is a global chapter [10] [11] which indicates for each process their related processes), costs and implementation problems. As an example, the structure of the Configuration Management process in the Service Support area is shown in Fig. 2.

[1] ITIL official documentation considers Service Desk as a functional unit therefore it is not discussed in this research work.

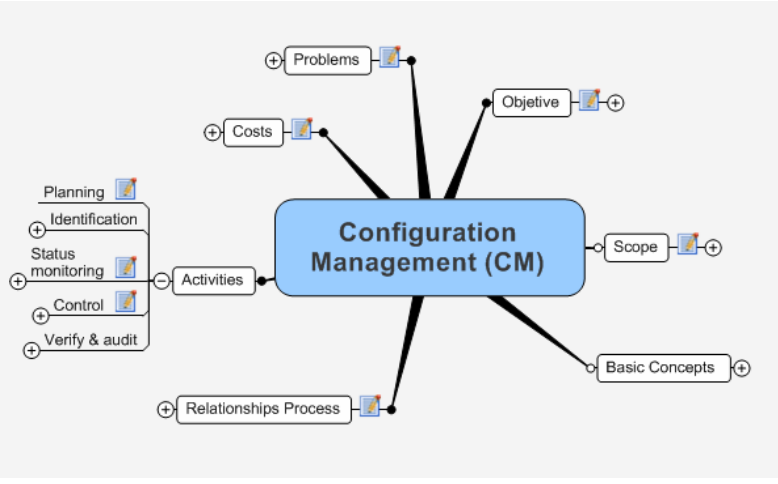


Fig. 2. Configuration Management process structure

3 OPreSSD: Organizational Procedure for Service Support and Service Delivery

The procedure described in this section, called OPreSSD (Organizational Procedure for Service Support and Service Delivery), identifies the processes implementation sequence related to the Service Support (SS) and Service Delivery (SD) areas [10] [11].

The main reason to elaborate this procedure was the need for a guideline to help us with the processes implementation sequence of SS and SD areas. OPreSSD is divided into two stages (see Fig. 3). In the first stage, dependencies among processes are identified in order to generate a dependency matrix. In the second stage, clusters of Strongly Connected Components (SCCs) are proposed in order to generate the implementation sequence.

3.1 Establish Dependency Relationships

The first stage of the procedure is to identify and represent the dependencies among processes by reviewing the ITIL official bibliography [10] [11].

3.1.1 Identify Dependencies

A review of the official books provides the information to determine how each process is related to each other. A matrix of dependencies is elaborated (see Table 1) taking into account the dependencies found among processes. This matrix represents all the existing dependencies among processes from both areas (SS and SD). In a dependency relation between processes there is a source process and a destination process. In Table 1, rows represent the source processes and columns the destination processes. Processes are named by their acronym as indicated in Fig. 1. In cell P_{ij}, the value 1 indicates there is a dependency between process i and process j ($P_i \cap P_j=1$). The value 0 indicates there is not a dependency between process i and process j ($P_i \cap P_j=0$).

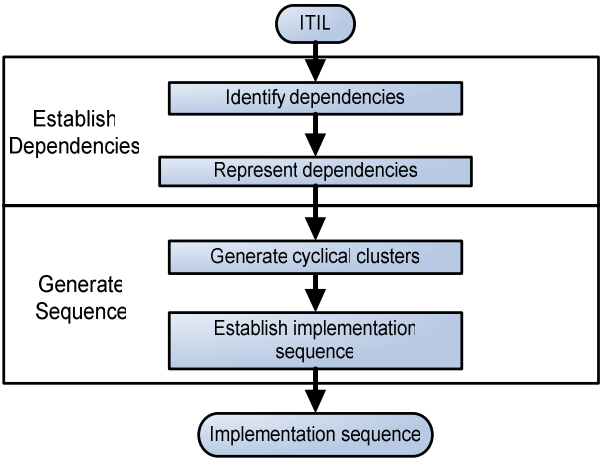


Fig. 3. OPreSSD Stage Diagram

Table 1. Matrix of dependencies

		SS						SD						
	Destination / Source	IM	PM	CM	ChM	RM	TDSS	SLM	FM	CapM	CoM	AM	TDSD	TS
	IM		1	1	1	0	3	1	0	1	0	1		6
	PM	1		1	1	0	3	1	0	1	0	1		6
SS	CM	1	1		1	1	4	1	1	1	1	1		9
	ChM	1	1	1		1	4	1	0	1	0	1		7
	RM	0	0	1	1		2	1	0	0	0	0		3
	SLM	1	1	1	1	1			1	1	1	1	4	9
	FM	0	0	1	0	0		1		1	0	0	2	3
SD	CapM	1	1	1	1	1		1	1		1	1	4	9
	CoM	0	0	1	1	0		1	0	1		1	3	5
	AM	1	1	1	1	0		1	0	1	1		3	7
	TD	6	6	9	8	4		9	3	8	4	7		

3.1.2 Represent Dependencies

The dependencies shown in Table 1 are represented by two types of graphs, a global graph (see Fig. 4) and two specific graphs for SS and SD respectively (see Fig. 5). Data provided by the matrix of dependencies are represented by a directed graph (digraph) where the processes are represented by vertices and dependencies by arcs. Each vertex is labeled with the process acronym as shown in Fig. 1.

The following definitions and proposition describe the elements of Graphs Theory used:

Definition 1: A digraph D consist of a non-empty finite set V(D) of elements called vertices (or nodes) and a finite set A(D) of ordered pairs of distinct vertices called

arcs (or edges). V(D) is the vertex set and A(D) is the arc set of D. D = (V, A) means that V and A are the vertex set and arc set of D, respectively [12].

Definition 2: A trail is walk in which all arcs are distinct. If the vertices of W are distinct, W is a path. If the vertices v_1, v_2 ...vk_{-1} are distinct, k≥3 and $v_1=v_k$, W is a cycle [12].

Definition 3: A digraph D (V, A) is strongly connected if each pair of vertices u and v (u ≠ v), has a path from u to v [13].

Proposition 1: Let D be a digraph and let x, y be a pair of distinct vertices in D. If D has an (x, y)-walk W, then D contains an (x, y)-path P such that A (P) ⊆ A (W). If D has closed (x, x)-walk W, then D contains a cycle C through x such that A (C) ⊆ A (W) [12].

The global graph in Fig. 4 is obtained by applying definition 1 to the dependency matrix.

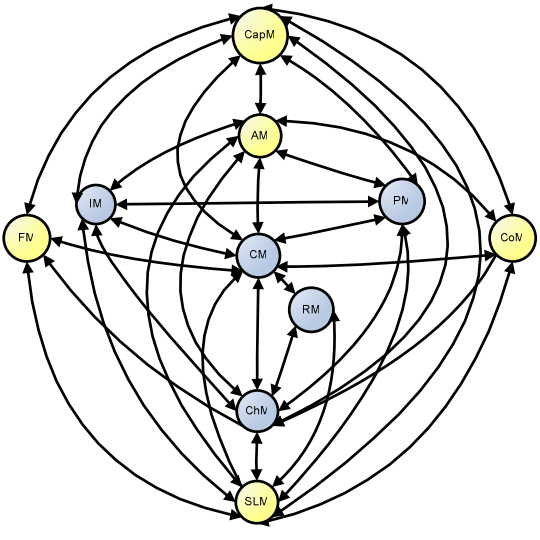


Fig. 4. Global graph corresponding to the SS and SD processes

The global graph in Fig. 4 shows the complexity of the dependencies among the processes. It is observed that it is not possible to determine a process implementation sequence. Moreover, in order to reduce the level of complexity, processes will be grouped by areas (SS & SD), to facilitate the elaboration of the implementation sequence.

The specific graphs for SS and SD (see Fig. 5) are obtained by applying definition 1 to the dependency sub matrix (the internal dependencies of each area are only taken into account).

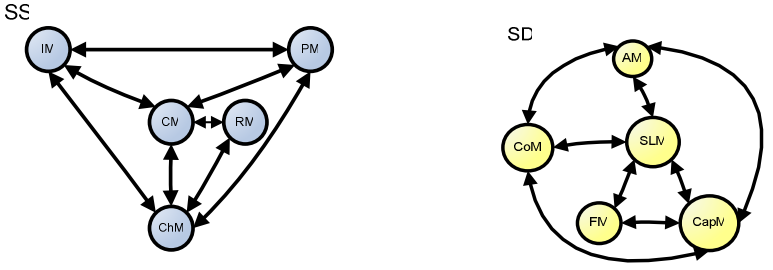


Fig. 5. Specific graphs of the SS and SD areas respectively

3.2 Generate Sequence

In the second stage of OPreSSD, the processes implementation sequence for each area is generated through the following steps.

3.2.1 Generate Cyclic Clusters

A cluster decomposition of a graph D is a partition of D into connected components [14]. The following steps are used to generate cyclic clusters:

- Step 1: Verifying that specifics graphs are Strongly Connected Components (SCCs).
- Step 2: Generating combinations for each SCC.
- Step 3: Obtaining cyclic clusters.

For each specific graph (see Fig. 5) a component that goes through all processes and relationships is elaborated in order to check the SCCs. These components are executed using a mathematical software tool [15] in order to know the SCCs.

Component SS: {IM→PM, PM→ChM, ChM→CM, CM→RM, RM→ChM, ChM→IM, IM→CM, CM→PM, PM→CM, CM→IM, IM→ChM, ChM→RM, RM→CM, CM→ChM, ChM→PM, PM→IM}
Component SD: {AM→CapM, CapM→SLM, SLM→CoM, CoM→AM, AM→SLM, SLM→FM, FM→CapM, CapM→CoM, CoM→CapM, CapM→FM, FM→SLM, SLM→AM, AM→CoM, CoM→SLM, SLM→CapM, CapM→AM}

After executing the mathematical software tool only one SCC (step 1) is obtained for each area:

SCC of SS: {IM, PM, ChM, CM, RM}
SCC of SD: {AM, CapM, SLM, CoM, FM}

Step 2 is applied to each SCC in order to get all the different combinations of 3-processes. Results are shown in Table 2. The minimal number of vertices in a cycle is 3 and this is the reason for selecting groups of 3-processes.

Cyclical clusters (step 3) are obtained by applying the Graphs Theory definition 2 to the resulting groups indicated in Table 2 (see Tables 3).

Table 2. SS and SD areas Combinations

SS Combinations	Processes	SD Combinations	Processes
A	CM, PM, RM	K	SLM, AM, CoM
B	CM, PM, ChM	L	SLM, FM, CapM
C	CM, PM, IM	M	SLM, CapM, AM
D	CM, IM, ChM	N	SLM, FM, CoM
E	CM, RM, ChM	O	SLM, CapM, CoM
F	CM, IM, RM	P	SLM, FM, AM
G	IM, PM, ChM	Q	CoM, CapM, AM
H	IM, PM, RM	R	CoM, FM, AM
I	IM, RM, ChM	S	CoM, FM, CapM
J	PM, RM, ChM	T	AM, FM, CapM

Table 3. SS and SD cyclical cluster

Cluster	SS Processes	Cluster	SD Processes
B	CM, PM, ChM	K	SLM, AM, CoM
C	CM, PM, IM	L	SLM, FM, CapM
D	CM, IM, ChM	M	SLM, CapM, AM
E	CM, RM,ChM	O	SLM,CapM, CoM
G	IM, PM, ChM	Q	CoM, CapM, AM

3.2.2 Establish Implementation Sequence

To identify the processes implementation sequence, the following steps are applied to the previous cyclical clusters.

- Step 4: Generate permutations for each cyclical cluster from Table 3.
- Step 5: Select valid permutations. A permutation is considered valid when is sorted by higher to lower number of dependencies (in this case in accordance with TDSS and TDSD columns).
- Step 6: Select permutations by the total number of source dependencies (column TS in Table 1).
- Step 7: Select valid permutations in accordance with Total row in Table 7.

The fourth step is applied in order to get all different permutation of the processes clusters from Table 3. Results are shown in Table 4.

The fifth step is applied in order to obtain valid permutation from Table 4. In this case, for each cyclical cluster, six permutations are obtained. In accordance with TDSS column in Table 1 an example of valid permutation is the B3 permutation where TDSS (ChM) is equal to 4, TDSS (CM) is equal to 4 and TDSS (PM) is equal to 3 dependencies, and an example of a non valid permutation is the B2 permutation where TDSS (CM) is equal to 4, TDSS (PM) is equal to 3, and TDSS (ChM) is equal to 4.

Results of valid permutations for SS and SD are shown in Tables 5 and 6. The Total TS is the sum of the number of dependencies that each process has as source with other processes.

Table 4. SS and SD areas permutations

Permutation	SS area Processes			Permutation	SD area Processes		
B1	CM	ChM	PM	K1	SLM	AM	CoM
B2	CM	PM	ChM	K2	SLM	CoM	AM
B3	ChM	CM	PM	K3	AM	SLM	CoM
B4	ChM	PM	CM	K4	AM	CoM	SLM
B5	PM	ChM	CM	K5	CoM	SLM	AM
B6	PM	CM	ChM	K6	CoM	AM	SLM
C1	CM	PM	IM	L1	SLM	FM	CapM
C2	CM	IM	PM	L2	SLM	CapM	FM
C3	PM	CM	IM	L3	FM	SLM	CapM
C4	PM	IM	CM	L4	FM	CapM	SLM
C5	IM	CM	PM	L5	CapM	FM	SLM
C6	IM	PM	CM	L6	CapM	SLM	FM
D1	CM	IM	ChM	M1	CapM	CoM	AM
D2	CM	ChM	IM	M2	CapM	AM	CoM
D3	IM	CM	ChM	M3	CoM	CapM	AM
D4	IM	ChM	CM	M4	CoM	AM	CapM
D5	ChM	IM	CM	M5	AM	CoM	CapM
D6	ChM	CM	IM	M6	AM	CapM	CoM
E1	CM	RM	ChM	O1	SLM	CapM	AM
E2	CM	ChM	RM	O2	SLM	AM	CapM
E3	RM	CM	ChM	O3	CapM	SLM	AM
E4	RM	ChM	CM	O4	CapM	AM	SLM
E5	ChM	RM	CM	O5	AM	CapM	SLM
E6	ChM	CM	RM	O6	AM	SLM	CapM
G1	ChM	IM	PM	Q1	SLM	CapM	CoM
G2	ChM	PM	IM	Q2	SLM	CoM	CapM
G3	IM	ChM	PM	Q3	CapM	SLM	CoM
G4	IM	PM	ChM	Q4	CapM	CoM	SLM
G5	PM	IM	ChM	Q5	CoM	CapM	SLM
G6	PM	ChM	IM	Q6	CoM	SLM	CapM

In order to obtain those permutations having the highest number of source dependencies we apply step 6 to Tables 5 and 6. Results are shown in Tables 7 and 8 for each permutation.

The Total row in Table 9 represents the global source dependencies. The Total row is obtained by adding TS and TD values of each process from Table 1 (matrix of dependencies).

Applying step 7 we will select from Tables 7 and 8 valid permutations taking into account the Total row of Table 9. Results are shown in Table 10. In the case of the SS area, B1 and D2 permutations are selected. And in the case of the SD area, O1 permutation is selected.

Table 5. Valid permutations for each SS cluster

Permutation	Implementation sequence			Total TS
B1	CM	ChM	PM	22
B3	ChM	CM	PM	22
C1	CM	PM	IM	21
C2	CM	IM	PM	21
D2	CM	ChM	IM	22
D6	ChM	CM	IM	22
E2	CM	ChM	RM	19
E6	ChM	CM	RM	19
G1	ChM	IM	PM	19
G2	ChM	PM	IM	19

Table 6. Valid permutations for each SD cluster

Permutation	Implementation sequence			Total TS
K1	SLM	AM	CoM	21
K2	SLM	CoM	AM	21
L2	SLM	CapM	FM	21
L6	CapM	SLM	FM	21
M1	CapM	CoM	AM	21
M2	CapM	AM	CoM	21
O1	SLM	CapM	AM	25
O2	CapM	SLM	AM	25
Q1	SLM	CapM	CoM	23
Q2	CapM	SLM	CoM	23

Table 7. SS area permutations sorted by TS Total

Permutation	Implementation sequence			TS Total
B1	CM	ChM	PM	22
B3	ChM	CM	PM	22
D2	CM	ChM	IM	22
D6	ChM	CM	IM	22

Table 8. SD area permutations sorted by TS Total

Permutation	Implementation sequence			TS Total
O1	SLM	CapM	AM	25
O2	CapM	SLM	AM	25

Table 9. Global source dependencies of SS and SD areas

	SS process area					SD process area				
	IM	**PM**	**CM**	**ChM**	**RM**	**SLM**	**FM**	**CapM**	**CoM**	**AM**
TS	6	6	9	7	3	9	3	9	5	7
TD	6	6	9	8	4	9	3	8	4	7
Total	12	12	18	15	7	18	6	17	9	14

Table 10. Implementation Sequence SS and SD permutation

Permutation	Implementation sequence		
B1	CM(19)	ChM(15)	PM(12)
D2	CM(19)	ChM(15)	IM(12)
O1	SLM(18)	CapM(17)	AM(14)

Table 10 shows the possibilities of implementation sequence; there are two for the SS area and one for the SD area. In the case of the SS area, where Total (IM) and Total (PM) have the same number of dependencies, the sequence is ordered taking into account their functionalities. According to ITIL–SS [3], a problem exists if there is a previous incident. So it is necessary to first implement the IM process, then the PM process.

Thus, in the case of the SS area the processes implementation sequence would be CM first, ChM later and finally IM (B1 permutation).

In the case of the SD area the processes implementation sequence would be SLM, CapM and AM (O1 permutation).

For each process area the full processes implementation sequence is obtained by taking into account the total dependencies of the remaining processes (see Total row in Table 9). Therefore, the processes implementation sequence for SS is CM, ChM, IM, PM and RM. For the SD area it is SLM, CapM, AM, CoM and FM.

4 Summary

It has been observed that all processes of Service Support (SS) and Service Delivery (SD) areas are strongly connected; the functional organization by areas confirms it. However, with respect to the structural organization it has been found that not all processes have the same structure, which makes difficult to identify the activities of each process, as well as the way to interrelate with each other. This situation opens the possibility of future studies to propose a standardized organization of the structure processes like other models have already done [16].

ITIL proposes a set of processes that will have to be established, but not its implementation sequence. This work has proposed a procedure that will allow initializing the sequence of implementation in each area related to SS and SD, through dependencies on processes. The SS area should begin with CM and the SD area should begin with SLM.

According to everis consultants, usually there are three typical scenes of ITIL implementation in the business field: 1) begin with Configuration Management, 2) begin

with Services Desk, and 3) begin with Service Level Management. The implementation sequence obtained by means of OPreSSD has been confirmed with the results of the experiences obtained by everis consultants, demonstrating that indeed the obtained results correspond to a present implementation sequence of ITIL processes.

The results obtained represent dependencies at a process level. In future studies the dependencies at activity level among processes of ITIL v3 will be considered.

Acknowledgements

This paper is sponsored by ENDESA, *everis* Foundation and Sun Microsystems through *"Research Group of Software Process Improvement for Spain and Latin* America", as well as by the Secretariat of Public Education (Mexico) with a scholarship PROMEP through the agreement with the Autonomous University of Tamaulipas.

References

1. Johnson, B.: (November 2006), http://www.ca.com/hk/event/itil2005/bjohnson.htm
2. Office of Government Commerce (OGC). ITIL Managing IT Service: Service Delivery. TSO, London (2001)
3. Office of Government Commerce (OGC). ITIL Managing IT Service: Service Support. TSO, London (2001)
4. Sarbanes, P., Oxley, M.: Sarbanes-Oxley Act (SOX/SORBOX), United States (July 2002)
5. IBM (November 2006), http://www-306.ibm.com/software/tivoli/features/ITIL/
6. Microsoft (November 2006), http://www.microsoft.com/technet/itsolutions/cits/mo/mof/default.mspx
7. SUN (November 2006), http://es.sun.com/services/itil/
8. HP (November 2006), http://www.hp.com/large/itsm/
9. Thu, T.D., Hanh, N., Bich Thuy, D.T., Coulette, B., Cregut, X.: Topological Properties for Characterizing Well-formedness of Process Components. Software Process Improvement and Practice 10(2), 217–247 (2005)
10. Office of Government Commerce (OGC). Relationship between processes en ITIL Managing IT Service: Service Delivery, ch. 2. TSO, London (2001)
11. Office of Government Commerce (OGC). Relationship between processes en ITIL Managing IT Service: Service Support, ch. 2. TSO, London (2001)
12. Bang-Jensen, J., Gutin, G.: Digraphs Theory, Algoritms and Applications. Springer, London (2006)
13. Diestel, R.: Graph Theory. Springer, New York (1997)
14. Lecture Notes in Theorethical Distributed Computing, Alexandro Panconesi (19-12-1997), http://www.nada.kth.se/kurser/kph/2d5340/wwwbook/wwwbook.html
15. WolfromResearch, Inc., Matemática 5.2 (November 2006), http://www.wolfrom.com/mathematica/functions/advanceDocumentationGraphPlot
16. CMMI for Services Team: CMMI for Services v0.5. Carnegie Mellon University SEI, Pittsburgh, PA (2007)

Modeling and Assessment in IT Service Process Improvement

Béatrix Barafort[1], David Jezek[3], Timo Mäkinen[2], Svatopluk Stolfa[3], Timo Varkoi[2], and Ivo Vondrak[3]

[1] Centre de Recherche Public Henri Tudor, Luxembourg
[2] Tampere University of Technology, Pori, Finland
[3] Technical University of Ostrava, Ostrava, Czech Republic

Abstract. This paper is based on the experiences of a research project with the aim to develop modeling and assessment readiness for IT companies. As a part of the project, process assessments for process improvement purposes were performed in some of the participating companies. This paper describes the background of applying process reference model based assessment and modeling of the processes for the same process instance. Some findings and experiences based on an industry case are documented. We also discuss how these approaches could be combined in an efficient way.

Keywords: Process improvement, assessment, process models, modeling.

1 Introduction

Process improvement approaches are means to develop an organization's processes to more effectively meet its business goals. Process assessments are used to find out the capability of the process to reach this goal. Assessments also point out opportunities for process improvements. A disciplined process assessment evaluates organization's processes against a process assessment model (PAM), which typically includes exemplar practices as assessment indicators. A process reference model (PRM) describes the processes typical for a specific domain.

One of the difficulties in process improvement is that, as the first step, the existing process must be understood correctly. One prominent approach is to integrate process modeling with assessments, which additionally is known to provide more accurate process ratings and higher quality process models.

A two year research project launched in 2006, Modeling and Simulation in Software Engineering (MoSSE), studied different methods and tools applicable for combined assessment and modeling. The project is a part of the Finnish national technology program on modeling and simulation (MASI), which is funded by Tekes, Finnish Funding Agency for Technology and Innovation.

The primary goal of the MoSSE project was to answer the question, "How to integrate software process assessment and process modeling?" In the project we refined our goal to develop a method, which integrates the techniques of software process assessment and process modeling. The approaches are applicable to similar domains, e.g. IT service management.

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 117–128, 2008.

This paper describes the background of applying process reference model based assessment and modeling of the processes in the IT service management domain. In section two we describe a method that aims to process improvement using process capability assessments. Section three explains background and development of a process assessment model for IT service management. In section four business process modeling is introduced as a basis for the modeling with appropriate techniques. Section five discusses the experiences of the industry case, in which the approaches were applied to. The main result of the study, the integration of assessment and modeling approaches, is presented in section six. Finally, section seven summarizes the work and proposes future developments.

2 Capability Assessments

Earlier projects at Tampere University of Technology contained expert and training services to support software engineering processes in companies [22]. One of the most important forms of these services was software process assessments. We have carried out altogether about 40 assessments in over 20 organizations. Most of the assessments are based on the international standard ISO/IEC 15504 [12, 13], or the preceding set of technical reports. The applied, experience based software process assessment and improvement practices have been modeled and resulted in the development of the Software Process Improvement initiation (SPINI) framework, which is similar to the results of some other research projects e.g. in Ireland [18] and in Brazil [1].

The SPI Initiation framework developed covers the process improvement start-up from the examination of an organization's needs to the development and support of the software process improvement (SPI) program. The framework consists of activities which are grouped from external advisors' points of view into the following three steps: understanding and deciding priorities, performing assessments, and supporting SPI. [21]

The ISO/IEC 15504 based process assessment has its origins in software development and an exemplar process assessment model, 15504-5, is based on ISO/IEC 12207 Software lifecycle process reference model. Nevertheless, the process capability dimension describes process attributes that are applicable to any process and to any domain.

The SPINI approach forms the methodological basis in this study. Modeling methods will be integrated into this approach producing an enhanced method for process improvement. The same capability assessment principles are applied for the IT service process assessments.

3 Process Assessment Models for IT Service Management

Nowadays, organizations are highly dependent on their Information Technology (IT) services and expect they not only support the organization but also bring new possibilities to realize its objectives. Contrary to the past when many IT organizations were centered on internal technical matters, today's businesses have high requirements regarding the quality of services and these needs are quickly evolving with time. IT

organizations have to meet these requirements and must pay attention to service quality and develop a more customer-oriented approach: cost issues are now high on the agenda as is the development of a more businesslike attitude to provision of service.

In that context, the IT Infrastructure Library (ITIL), produced by the CCTA (Central Computer and Telecommunications Agency in UK) in the late '80s, was the first comprehensive structured approach publicly available on providing IT services [14, 15]. The ITIL® rapidly became a worldwide de facto standard for IT Service Management. The core of ITIL® initially focused on delivering and supporting IT services that are appropriate to the organization's business requirements, whatever its type or size. ITIL® provides a comprehensive, consistent and coherent set of best practices for IT Service Management processes, promoting a quality approach to achieving business effectiveness and efficiency in the use of information systems.

In the Public Research Centre Henri Tudor in Luxembourg, the ISO/IEC 15504 standard, as well as the ITIL® de facto one, had been studied and used since the mid-nineties. Several experiences in R&D projects showed that many companies used process approaches in software development (CMM and/or ISO/IEC 15504 approach) and for operations (ITIL®), but with not many connections (separate departments and culture; project versus non-project approaches). Then the combined use of both standards appeared as an interesting matter of research. The AIDA® R&D project was defined and aimed at developing a common approach for IT process assessment and improvement. There were already existing process assessment models such as ISO/IEC 15504-5 and CMM, and more recently CMMI [20]. But there were not many initiatives linking assessment purposes and IT Service Management. So the AIDA R&D Project developed an IT Service Management Process Reference Model (PRM) and its associated Process Assessment Model (PAM) [2, 3, 4].

The AIDA® model was inspired by ITIL® best practices, with the goal to enable objective IT Service Management capability assessments. The references used to create the PRM and PAM were the Service Support and Service Delivery books published by the Office of Government Commerce (OGC). These inputs are considered as implementation best practices, and can be considered as a Process Implementation Model (PIM) to start with. The purpose of the PRM was to define, at a high level of abstraction (i.e. in term of Process purpose and Process outcomes), a set of processes that can be used as the process dimension for a PAM in the Service Management area. According to the maturity of the definition of these processes, the process list of the PRM was directly derived from the Service Support and Service Delivery ones. The ten processes from Service Support and Service Delivery were then selected without adding or removing any of them.

Using ITIL® best practices, the CRP Henri Tudor developed a Process Reference Model. The purpose of this Process Reference Model was to define a set of processes that can be used as the process dimension for a Process Assessment Model. [6]

The Process Assessment Model proposes additional information that can be requested when running a process assessment: base practices, inputs and outputs. The capability levels and process attributes are identical to those defined in ISO/IEC 15504-2. Fig. 1 describes the steps required to derive the models.

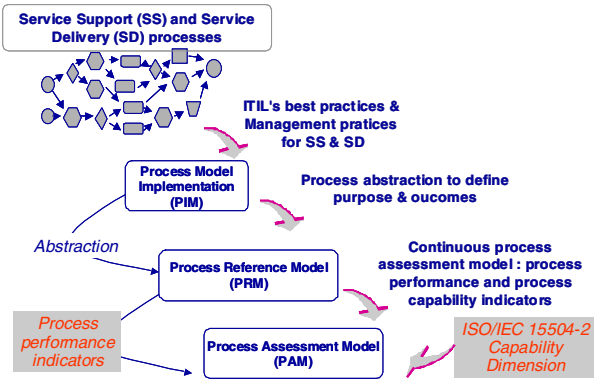


Fig. 1. Deriving the IT Service Management Process models

In addition to the Process Assessment Model, a specific questionnaire for each field of activity was developed to help assessors during the interviews. However this questionnaire is not mandatory and a competent assessor can conduct interviews using the Process Assessment Model without a questionnaire, as part of a more 'open' discussion.

As ITIL® has become more and more popular, the new version 3 of ITIL® was published in June 2007. The objective is to propose a consistent, modular and dynamic conceptual framework providing IT managers with on demand methodologies and tools. ITIL® v3 is not only more structured but also particularly focusing on services implementation strategy and services life cycle. Finally, ITIL® V3 tends to involve more the development actors within Service Support and Delivery processes, with particular attention to the management of change. This is a real challenge because both approaches generally reconcile poorly within companies.

Additionally the International standardization Organization (ISO) develops the ISO/IEC 20000 IT Service Management standard [7, 8]. It is aiming at certifying a service provider with a management system for IT Service Management Processes. The ISO/IEC 20000-1 standard, titled "Specification" promotes the adoption of an integrated process approach to effectively deliver managed services to meet the business and customer requirements. On the other hand, ISO/IEC 20000-2, titled "Code of practice" takes the form of guidance and recommendations.

The International standardization community recognized the benefits of using complementary approaches (New Work Item Proposals Accepted for defining a PRM and a PAM based on the ISO/IEC 20000 standard [9, 10]). Then, the extension of the AIDA R&D project enables to develop such a PRM and a PAM, based on the collection of requirements provided by the ISO/IEC 20000 Parts 1 and 2. Further works develop a methodology for transforming requirements into PRM and PAM. Some particular Requirements Engineering approaches and techniques (Goal Oriented Requirements Engineering) have been used as well as issues met by R&D projects in CRP Henri Tudor in Luxembourg for modeling PRMs and PAMs [19]. These RE techniques have shown how helpful they can be. The whole set of tools and techniques constitute a methodology, which is continually refined along with experimentations and standardization

works related to process models. In this context, the very structured way required by ISO/IEC 15504 for building process models enables process assessment with capability determination. In order to structure the methodology leading to the construction of a PRM-PAM and to organize components, a PRM has been drafted, aiming at engineering process models. The purpose of this Process Model Engineering PRM is to design and manage an ISO/IEC 15504 compliant process model (validation and traceability) fulfilling the stakeholders' requirements and needs, and to provide a knowledge base supporting uses of the model. This Process Model Engineering PRM draft provides the framework for the overall methodology and it will be further developed in next R&D activities within CRP Henri Tudor.

By using a rigorous and systematic approach for developing PRMs and PAMs, it provides a very structured and trusted basis for process improvement. Then it can be valuable inputs for combining process modeling and assessment with the help of a support tool, within an improvement approach contextualized to an organization.

4 Process Modeling

Process modeling ranges from informal to formal, where formal models are based on mathematical principals and are executable by computing systems. BPM (Business Process Modeling) method is a modeling instrument that combines UML-like tool based on the best practices with such formalized approach [23]. The main goal of the BPM is to balance clarity of visualization with the power of theory of Petri Net that strengthens correctness of the process specification. Consequently, the BPM method can be characterized by the following:

- BPM is a formalized and visual modeling tool. Formalization is employed to model a process uniquely and precisely enough to use a built model for simulation and enactment without any or limited changes. Visual approach enables to increase modeling capabilities and clarity to make all necessary communications easier.
- BPM enables structural analysis of the process, visual simulation and enactment of the process dynamics.
- BPM uses concurrency of process activities execution as a primary focus. This aspect of process definition provides key way how to get processes more effective.

BPM employs three types of models to capture the process as a whole. The main aim of the *functional model* is an identification of the process architecture, as well as the identification of process customers and products. The primary focus of the functional model is to find an answer to questions *what* processes are employed and what is their structure. A simple fictive process example process demonstrates notation of the functional model and how the process can be hierarchizised using so called contains relationship (Fig. 2). Obviously any sub-process can be expanded into the similar diagram with its own customers, owners and products.

Object model identifies static structure of all entities (objects) that are essential for the enactment of the process. In other words, the answer to the question by *whom and what* the process is realized is searched. This model tries to capture all active objects responsible for an execution of activities and passive objects that can be understood as

material, products or documents that are manipulated by the process. All these objects have a set of attributes associated. The notation used for this sort of models is similar to notation used by typical object-oriented method except that active and passive objects are represented by different icons to distinguish them (Fig. 3). Object models are created for every process identified during the functional modeling.

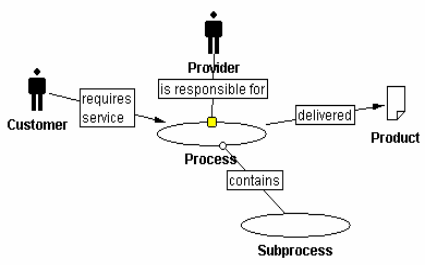


Fig. 2. Functional model of the process

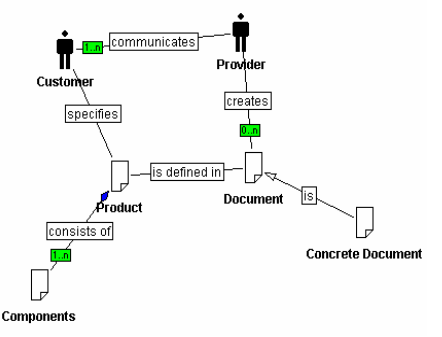


Fig. 3. Object model for the process

Coordination model is based on previous two models and its goal is to show *how* the process will be enacted. The coordination model specifies interactions among objects (active and/or passive) and defines the way *how* all these activities are synchronized based on principles used in Petri Nets. The coordination view is the most important because it enables to define the execution order of all activities, including conditions for their potential concurrency. It means that the correct order is defined, as well as sharing of used resources. Each activity can have more than one scenario

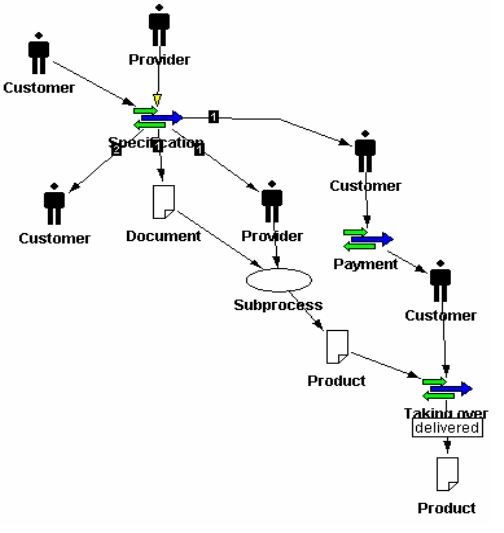


Fig. 4. Coordination model of the process

with the duration time and costs associated to provide necessary information for the analysis. Based on the architecture definition captured in a functional model, the atomic activities are accompanied by sub-processes icons that can be refined further into more detailed collaboration models again (Fig. 4).

The specified process models serve as a basis for testing and analysis. The analysis is based on a discovery of both structural and behavioral properties. The first kind of properties is encoded in the model itself while the second one is obtained from the process model via its simulation. The analysis based on a process simulation verifies model and provides user with the information on how long it takes to get from the initial request to the final product and what are the process costs.

The main and unique property of the presented methods is that the modeling, simulation and enactment are based on one common model that is used by Petri Net engine for the purposes of its execution. No compilations are needed and the process can be verified and validated during the development time.

5 Experiences of Combined Modeling and Assessment

The BPM method was used to evaluate utilization of formal based modeling and simulation for the assessment of service processes. The processes were provided by an international IT company with a large service business unit. The company has an ISO 9000 compatible quality system, it uses ITIL and will probably apply for the future ISO/IEC 20000 IT Service Management standard certification.

The main goal of the work carried out was to introduce and evaluate the way how both modeling and assessment could be hooked up together to generate some synergy effects. The combined approach that was used in this case study was concentrated on the assessment and evaluation of two main service support processes: incident and problem management.

The first step of the process modeling was based on the analysis of the materials and documents provided by the company, in comparison to an existing IT Service Management Process Assessment Model. The used model, AIDA PAM, is described in section 3. Although, the processes were thoroughly specified in these documents, the specification was mainly textual and suffered from the lack of formal description. The preliminary models were created and the unknown details were discussed with the company representatives responsible for the process facilitation. It took several sessions to refine the models. Each session consisted of the definition of the formalized models that were checked by customer first, and afterwards these verified models were used for the purposes of their assessment. The final result of the modeling was the descriptive process model.

The next step is the definition of the prescriptive processes. The combined output of the analysis and simulation of the descriptive processes and the improvement suggestions from the assessment activity provides the information for the creation of the final prescriptive process model.

From the simulation point of view, process models needed to be completed by the information set required by the BPM simulation tool. The information set comprises of duration and cost of each activity. Then, the simulation part of the BPM could be used effectively. The obtained results showed the total time and cost needed for the

execution of processes being modeled. The interpretation of the results led to the conclusions that were more or less known by the company, but the results of cost and time analysis could never have been achieved without the simulation. For example, one of the main disclosures was the confirmation of the theory that the existence of the good knowledge base is one of the main time and cost saving factors. Since the knowledge base is especially used during the incident management process and execution of the incident management was proved as significantly cheaper and less time consuming than problem management process, the existence of proper knowledge in the knowledge base is crucial to save company time and money and to achieve better efficiency.

Simulation and analysis based on simulation experiment are not the only benefits that could be gained by this approach. The main benefit could be achieved by the continuous refining, control and simulation of the company processes. Although, the initial assessment of the company's service processes showed the high process maturity, the next logical step should be the application of the systematic process improvement according to this approach.

From the improvement point of view, the BPM tool confirmed the potential to be an efficient modeling, simulation and control tool for the modeling and assessment of company processes. In this case study, modeling and simulation of the processes were performed manually. The implementation of the automated control and enactment tool that is already part of the BPM tool could bring more precise results. This enhanced approach is more complex and might be used not only for process assessment but also for purposes of continuous process improvement.

Generally, the combination of assessment methods and modeling, simulation and control tool fulfilled the expectations and fully satisfied the company representatives that were eager to find a way to proceed to achieve efficient process improvement.

6 Integration of the Approaches

Based on the presented experiences of using both assessment and modeling to gain input for process improvement, we now describe an integrated approach that, at least in theory, contains the elements needed to answer the research question: "How to integrate software process assessment and process modeling?" This section presents the integration steps and a new concept: the prescriptive process model.

Process assessment is a disciplined evaluation of an organization's processes against a Process Assessment Model. The essential part of an assessment output is a set process profiles, which are process attribute ratings for the assessed processes. The indicators of the process assessment model support assessors' judgment in rating process attributes. [11]

Assessment itself can also be considered as a process which transforms its inputs to outputs using resources. In an assessment process, Process Evidence stands for the inputs, Process Profiles for the outputs, and Assessment Model for the resources (Fig. 5). During the assessment the process evidence is classified by the indicators of the assessment model to assist the process ratings of the process profiles.

Descriptive process modeling uses a process meta-model that defines the process elements and their relationships, to classify the process evidence. Typical process

elements are tasks, work products, roles, and resources. There is similarity to the assessment process, while the output of the process is Process Model and its resource is Process Meta-model (Fig. 5).

A combination of process assessment and descriptive process modeling is represented in Fig. 5. The combined approach was proposed by Hamann in his dissertation [5]. During an assessment, the descriptive process modeling is performed by mapping process performance indicators of the assessment model to the existing process elements of the assessed organization. Later on, Mäkinen and Varkoi suggested the use of process capability indicators in the combined assessment [16, 17].

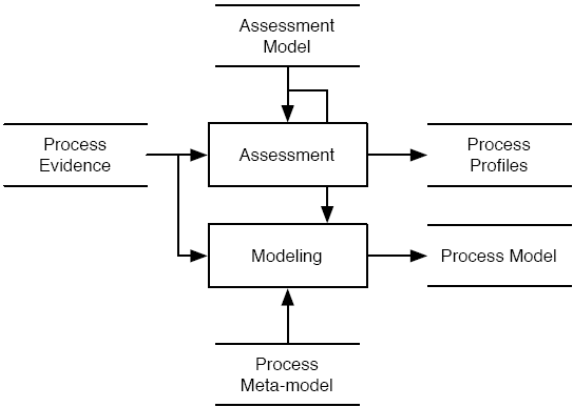


Fig. 5. Combination of process assessment and descriptive process modeling

In addition to the process profiles, the assessment usually produces a set of suggestions for process improvements. In the combined assessment, a prescriptive process model can be regarded as the counterpart for the suggestions. In this case, the prescriptive process model illustrates the organization's process after the implementation of the suggested improvements.

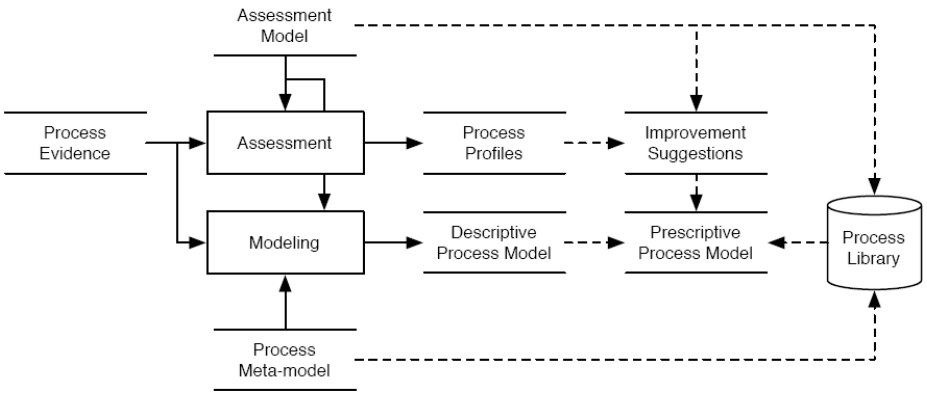


Fig. 6. Combination of descriptive and prescriptive process modeling, and process assessment

Fig. 6 depicts the extended combination of assessment and process modeling [17]. The extended combined assessment produces Improvement Suggestions and a Prescriptive Process Model in addition to Process Profiles and a Descriptive Process Model. The suggestions are derived from the process profiles and the assessment model. The prescriptive process model is based on the descriptive model and the improvement suggestions. The construction of the prescriptive process model is supported by a process library that includes the indicators of the assessment model as process elements.

7 Conclusions

This paper presents the work of a research project with international connections. Assessment models and methods applying process reference models were discussed, as well as business process modeling concepts. As an example, an advanced IT service process assessment model is described. An assessment and modeling exercise was carried out in an industrial case. The experiences confirm the need for a combined approach for modeling and assessment. The main result of the study is the integrated method for assessment and modeling.

The framework in Fig. 7, named SPINI+, uses assessment input to create a descriptive process model of the assessed processes without extra effort to elicit information for modeling. The input data is classified using an industry standard process metamodel to support a choice of modeling tools. A revised prescriptive model illustrates an organization's processes after the improvements, and it is created using a process library that is based on the indicators of the assessment model. The benefit of the approach is that the improvements can be clearly expressed in the process model to make the process changes more manageable. Modeling with a formal approach related to process assessment is effective in ensuring up-to-date process descriptions in an organization.

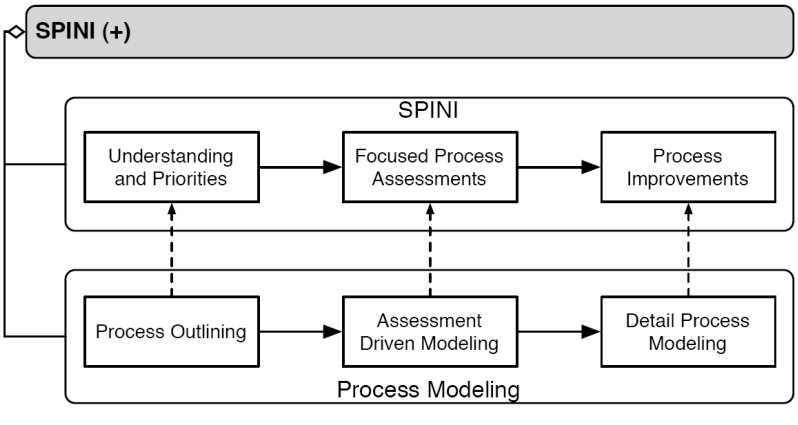


Fig. 7. Enhanced framework for process improvement

The method requires further validation. Expected strengths of the proposed method include:

- efficient use of resources; same input can be used both for modeling and assessment,
- improved assessment reliability,
- improved descriptive process model accuracy and
- precise and illustrative improvement suggestions

Further development of the SPINI framework will seek to solve needs for proficient process modeling and simulation using advanced process description approaches and methods. For instance, some of our earlier research projects have produced software measurement knowledge and process improvement experience items that can be related to the process improvements that have been identified using process assessments. The process improvement knowledge is collected as libraries.

Acknowledgments. This research has been partly funded by Academy of Finland and Tekes, Finnish Funding Agency for Technology and Innovation.

References

[1] Anacleto, A., Gresse von Wangenheim, C., Salviano, C.F., Savi, R.: A Method for Process Assessment in Small Software Companies. In: Proceedings of the 4th International SPICE Conference, Portugal (2004)

[2] Barafort, B., Di Renzo, B., Merlan, O.: Benefits resulting from the combined use of ISO/IEC 15504 with the Information Technology Infrastructure Library (ITIL). In: Proceedings of the International Conference PROFES 2002, Rovaniemi, Finland (2002)

[3] Barafort, B., Di Renzo, B.: Assessment and improvement integrated approach: combined use of the ISO/IEC 15504 (SPICE) and the Information Technology Infrastructure Library (ITIL). In: Proceedings of the National Conference SPIRAL 2004, Luxembourg (2004)

[4] Barafort, B., Di Renzo, B., Lejeune, V., Simon, J.-M.: ITIL Based Service Management measurement and ISO/IEC 15504 process assessment: a win – win opportunity. In: Proceedings of the 5th International SPICE Conference on Process Assessment and Improvement, Klagenfurt, Austria (2005)

[5] Hamann, D.: Towards an Integrated Approach for Software Process Improvement: Combining Software Process Assessment and Software Process Modeling. PhD dissertation. Technische Universität Kaiserslautern (2006)

[6] Hilbert, R., Renault, A.: Assessing IT Service Management Processes with AIDA – Experience Feedback. In: Proceedings of the 14th European Conference for Software Process Improvement EuroSPI, Potsdam, Germany (2007)

[7] ISO, ISO/IEC 20000-1: Information technology – Service management – Part 1: Specification (2005)

[8] ISO, ISO/IEC 20000-2: Information technology – Service management – Part 2: Code of practice (2005)

[9] ISO, ISO/IEC JTC1/SC7 3797: NWI Proposal - Information Technology - Service Management Process Reference Model (2007)

[10] ISO, ISO/IEC JTC1/SC7 3798: NWI Proposal - Information Technology - Process assessment - Part 8: An exemplar process assessment model for IT service management (2007)
[11] ISO, ISO/IEC 15504-1: Information technology - Process assessment - Part 1: Concepts and vocabulary (2004)
[12] ISO, ISO/IEC 15504-2: Information technology - Process assessment - Part 2: Performing an assessment (2003)
[13] ISO, ISO/IEC 15504-5: Information technology - Software Process Assessment - Part 5: An exemplar process assessment model (2006)
[14] IT Infrastructure Library – Service Delivery, The Stationery Office Edition, ISBN 011 3308930 (2001)
[15] IT Infrastructure Library – Service Support, The Stationery Office Edition, ISBN 011 3308671 (2000)
[16] Mäkinen, T., Varkoi, T., Soini, J.: Integration of Software Process Assessment and Modeling. In: Proceedings of the PICMET 2007 Management of Converging Technologies Conference, Portland, Oregon, USA (2007)
[17] Mäkinen, T., Varkoi, T.: Assessment Driven Process Modeling for Software Process Improvement. In: PICMET 2008 Technology Management for a Sustainable Economy Conference, Cape Town, South Africa (accepted for publication, 2008)
[18] McCaffery, F., Richardson, I., Coleman, G.: Adept – A Software Process Appraisal Method for Small to Medium-sized Software Development Organisations. In: Proceedings of the EuroSPI 2006 Conference, Joensuu, Finland (2006)
[19] Rifaut, A.: Goal-Driven Requirements Engineering for supporting the ISO 15504 Assessment Process. In: Richardson, I., Abrahamsson, P., Messnarz, R. (eds.) EuroSPI 2005. LNCS, vol. 3792, pp. 151–162. Springer, Heidelberg (2005)
[20] Rout, T.P., El Emam, K., Fusani, M., Goldenson, D., Jung, H.-w.: SPICE in retrospect: Developing a standard for process assessment. Journal of Systems and Software 80(9), 1483–1493 (2007)
[21] Varkoi, T., Mäkinen, T.: Software process improvement initiation in small organisations. In: Proceedings of the 3rd European Software Measurement Conference, FESMA-AEMES, Madrid, Spain (2000)
[22] Varkoi, T., Mäkinen, T.: Software process improvement network in the Satakunta region - SATASPIN. In: Proceedings of the EuroSPI 1999 Conference, Pori, Finland (1999)
[23] Vondrak, I., Kruzel, M., Szturc, R., Benes, M.: Unified Environment for Business Process Modeling and Workflow Enactment. In: Proceedings of the ECEC 2002 Conference, Ghent, SCS (2002)

A Software Engineering Lifecycle Standard for Very Small Enterprises

Claude Y. Laporte[1], Simon Alexandre[2], and Rory V. O'Connor[3]

[1] École de technologie supérieure, Montréal, Canada
[2] Centre d'Excellence en Technologies de l'Information et de la Communication, Charleroi, Belgium
[3] Lero, The Irish Software Engineering Research Centre and School of Computing, Dublin City University, Dublin, Ireland
Claude.Y.Laporte@etsmtl.ca, simon.alexandre@cetic.be, roconnor@computing.dcu.ie

Abstract. Industry recognizes that very small enterprises (VSE), that develop parts involving software components are very important to the economy. These parts are often integrated into products of larger enterprises. Failure to deliver a quality product on time and within budget threatens the competitiveness of both organizations. One way to mitigate these risks is to haveall suppliers of a product chain put recognized engineering practices in place. Many international standards and models such as ISO/IEC12207 or CMMI have been developed to capture proven engineering practices. However, these standards were not designed for very small development organizations, those with less than 25 employees, and are consequently difficult to apply in such settings. An ISO/IEC JTC1/SC7 Working Group has been established to address these difficulties by producing a software engineering standard tailored to VSE.

Keywords: ISO, Lifecycles, Very Small Enterprises, Standards.

1 Introduction

The ability of organizations to compete, adapt, and survive nowadays depends increasingly on software. By 2010, it is estimated that cellular phones will contain 20 million lines of code; one automobile manufacturer estimates that its cars will have up to 100 million lines of code [1]. Manufacturers depend increasingly on the components produced by their suppliers. A manufacturing chain, of large mass market products, often has a pyramidal structure. The pyramid is composed of a layer of dozens of main suppliers which are supplied by a layer of hundreds of smaller suppliers. This small suppliers layer may have thousands of very small suppliers. As an example, a large mass product manufacturer integrated a part with an unknown software error produced by one of its 6000 producers into one of its products, [2]. The defective part resulted in a multi-million dollar loss by the manufacturer. The need for international software engineering standards is thus clear.

There is evidence that the majority of small software organizations are not adopting existing standards as they perceive them as being orientated towards large organizations. Studies have shown that small firms' negative perceptions of process model

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 129–141, 2008.

standards are primarily driven by negative views of cost, documentation and bureaucracy. In addition, it has been reported that VSEs find it difficult to relate ISO/IEC 12207 to their business needs and to justify the application of the international standards in their operations. Most VSEs cannot afford the resources for, or see a net benefit in, establishing software processes as defined by current standards (e.g. ISO/IEC 12207) and maturity models such as the Capability Maturity Model Integration CMMI) developed by the Software Engineering Institute [3].

Accordingly there is a need to help these organizations understand and use the concepts, processes and practices proposed in the ISO/IEC JTC1/SC7's international software engineering standards. This paper presents a new project intended to facilitate access to, and utilization of, ISO/IEC JTC1/SC7 software engineering standards in very small enterprises.

This paper is divided into six sections. Section 2 presents the concept of a VSE and describes the characteristics that distinguish a VSE from other organizations. Section 3 presents a historical perspective on the events that led to an ISO/IEC JTC1 SC7 project proposal for VSEs and Section 4 presents the results of a survey that was developed to question VSEs about their utilization of ISO/SC7 standards. Section 5 explains the approach being taken by the VSE working group and finally Section 6 presents concluding remarks and discusses future actions.

2 Very Small Enterprises

The definition of "Small" and "Very Small" Enterprises is challengingly ambiguous, as there is no commonly accepted definition of the terms. For example, the participants of the 1995 CMM tailoring workshop [4] could not even agree on what "small" really meant. Subsequently, in 1998, in an SEPG conference panel on the CMM and small projects [5], small was defined as "3-4 months in duration with 5 or fewer staff." Johnson and Brodman [6] define a small organization as "*fewer than 50 software developers and a small project as fewer than 20 software developers*". Another definition for VSE introduced by Laporte et al [7] as "*any IT services, organizations and projects with between 1 and 25 employees*".

Taking a legal perspective, the European Commission [8] defines three levels of small to medium-sized enterprise (SME) as being: **Small to medium** –"*employ fewer than 250 persons and which have an annual turnover not exceeding 50 million Euro, and/or an annual balance sheet total not exceeding 43 million Euro*"; **Small** – "*which employ fewer than 50 persons, and whose annual turnover or annual balance sheet total does not exceed 10 million Euro*" and **Micro** – "*which employ fewer than 10 persons and whose annual turnover*".

To better understand the dichotomy between the definitions above it is necessary to examine the size of software companies operating in the market today. In Europe, for instance, 85% of the Information Technology (IT) sector companies have 1 to 10 employees[1]. In the context of indigenous Irish software firms 1.9% (10 companies), out of a total of 630 employed more than 100 people whilst 61% of the total employed 10 or fewer, with the average size of indigenous Irish software firms being

[1] http://www.esi.es/en/main/iitmark.html

about 16 employees [9]. In Canada, a survey of the Montreal area found that 78% of software development enterprises have less than 25 employees and 50% have fewer than 10 employees [10]. In Brazil, small IT companies represent about 70% of the total number of companies [11].

Therefore, for the purposes of this paper we are adopting the definition for VSE introduced in [7] as "*any IT services, organizations and projects with between 1 and 25 employees*".

2.1 Characteristics of a VSE

The unique characteristics of small entrepreneurial businesses as well as the uniqueness of their situations of necessity make their style of business different [12]. Some of the unique differences between very small and large businesses behavior are given in Table 1.

Table 1. Characteristic differences between large firms and small firms

Characteristic	Small firm	Large firm
Planning orientation	Unstructured/operational	Structured/strategic
Flexibility	High	Structured/strategic
Risk orientation	High	Medium
Managerial process	Informal	Low
Learning and knowledge absorption capacity	Limited	High
Impact of negative market effects	More profound	More manageable
Competitive advantage	Human capital centered	Organizational capital centered

Software VSEs are subject to a number of distinctive and intrinsic characteristics that make them different from their larger counterparts, therefore affecting the contents, the nature and the extent of the activities. We classify VSE characteristics according to four main categories: Finance, Customer, Internal Business Processes and Learning and Growth.

VSEs are economically vulnerable as they are driven by cash-flow and depend on project profits, so they need to perform the projects within budget. They tend to have low budgets which have many impacts, such as: lack of funds to perform corrective post delivery maintenance; few resources allocated for training; little or no budget to perform quality assurance activities; no budget for software reuse processes; low budget to respond to risks; and limited budget to perform Process Improvement and /or obtain a certification/assessment.

Typically the VSE's product has a single customer, where the customer is in charge of the management of the system and the software integration, installation and operation. It is normal practice for the customer not to define quantitative quality requirements and for customer satisfaction to depend on the fulfillment of specific requirements that may change during the project. A close relationship between all involved project members including the customer shows that software development in small and very small companies is strongly human-oriented and communication

between them is important. For example, in contrast to small companies, very small companies often do not have regular project meetings [13].

The internal business process of VSEs are usually focused on developing custom software systems, where the software product is elaborated progressively and which typically does not have strong relationship with other projects. Typically most management processes (such as human resource and infrastructure management) are performed through informal mechanisms, with the majority of communication, decision-making and problem resolution being performed face-to-face.

The learning and growth characteristics of VSE are typified by a lack of knowledge (or acceptance) of software process assessment and improvement and a lack of human resources to engage in standardization. It is usual for a negative perception of standards to exist in smaller organizations who consider they are made by large enterprises, for large enterprises [9].

3 History of the ISO/IEC Working Group for VSEs

The mandate of ISO/IEC JTC1/SC7 is the standardization of processes, supporting tools, and supporting technologies for the engineering of software products and systems. A description of SC7 and of the development of ISO/IEC JTC1/SC7 standards is presented in [14]. In this section, a brief history of the events leading to the creation of a new ISO/IEC JTC1/SC7 Working Group (WG) is presented. A detailed description of its history is available in [3].

At the May 2004 SC7 Plenary meeting in Brisbane, Canada raised the issue of small enterprises requiring standards adapted to their size and maturity level. The current software engineering standards target (or are perceived as targeting) large organizations. A meeting of interested parties was organized and a consensus was reached on general objectives for a future working group:

- To make the current software engineering standards more accessible to VSEs;
- To provide documentation requiring minimal tailoring and adaptation effort;
- To provide harmonized documentation integrating available standards:
  - Process standards
  - Work products and deliverables
  - Assessment and quality
  - Modeling and tools
- To align profiles, if desirable, with the notions of maturity levels presented in ISO/IEC 15504.

In March 2005, the Thailand Industrial Standards Institute (TISI) invited a Special Working Group (SWG) to advance the work items defined at the Brisbane meeting. A key topic of discussion was to clearly define the size of VSE that the SWG would target, consensus being reached on IT services, organizations and projects with 1 to 25 employees. The major output of this one-week meeting was a draft of the New Work Item (NWI) to be tabled at the next SC7 meeting.

In May 2005, a resolution was approved to distribute the NWI Proposal for the development of Software Life Cycle Profiles and Guidelines for use in Very Small Enterprises for ballot. Twelve countries voted in favor of the NWI Proposal [15]. As a

result of this vote, the Project was approved and the new working group, WG24, was established.

The Thailand Industrial Standards Institute (TISI) sent out a second invitation to participate in the SWG, to be held in September 2005 in Bangkok. The main objective of the meeting was to prepare material that would be presented to WG24 in order to facilitate the start-up of the working group that was scheduled for October 2005 in Italy.

In October 2005, Italy hosted ISO/IEC JTC1 SC7 Interim Meeting. The New Work Item was updated in order to take into account relevant comments received during balloting, and the requirements were validated by WG members. In addition, some VSE Business Models were identified, as was a strategy for creating profiles. Finally, WG24 decided to conduct a survey to collect relevant information from VSEs around the world.

4 Gathering VSE Requirements

In 1997, the Technical Council on Software Engineering responsible for the IEEE Software Engineering Standards conducted a survey to capture information from software engineering standards users in order to improve those standards [16]. They gathered 148 answers, mainly from the USA (79%) and large companies (87% of them having more than 100 employees). The main application domains of the survey respondents were IT (22%), military (15%) and aerospace (11%). (It should be noted that the purpose of this section is not to systematically compare the two sets of survey results.) Even though the IEEE survey objectives differ from those of the ISO/IEC survey, there are some interesting common findings. In response to the question concerning the reasons why their organization does not use standards, 37% said that the standards were not available in their facilities, while 37% explained that they use other standards. In fact, the IEEE survey underscores the fact that ISO/IEC standards, rather than the IEEE standards, are often used in organizations.

The IEEE survey underlined the difficulties regarding IEEE standards use reported by the respondents. The two main difficulties were a lack of understanding of the benefits (28%) and a lack of useful examples (25%). The survey also revealed how IEEE standards are used in organizations. Most of the organizations claimed to use IEEE standards for internal plan elaboration. The IEEE survey gathered several new requirements about IEEE standards being requested by the respondents. These were principally examples and templates of deliverables, support for metrics and measurement, help on life cycle process definition, a training course and support for small, rapid application development efforts.

The WG24 survey was developed to question VSEs about their utilization of ISO/SC7 standards and to collect data to identify problems and potential solutions to help them apply standards and become more competitive. From the very beginning, the working group drew up several working hypotheses regarding VSEs. The survey was intended to validate some of these hypotheses, such as the following:

- The VSE context requires light and well-focused life cycle profiles.
- Particular business contexts require particular profiles.

- There are significant differences, in terms of available resources and infrastructure, between a VSE employing 1 to 10 people and an Information Technology (IT) department of the same size in a larger company.
- VSEs are limited in both time and resources, which leads to a lack of understanding of how to use the standards for their benefit.
- Benefits for VSEs may include recognition through assessment or audit by an accredited body.

The survey questionnaire and an introductory text were developed by the WG24 and translated into 9 languages: English, French, German, Korean, Portuguese, Thai, Turkish, Russian and Spanish. The survey is made up of 20 questions structured in 5 parts: General information, Information about standards utilization in VSEs, Information about implementation and assessment problems in VSEs, Information about VSE needs and Information about justification for compliance to standard(s). Over 392 responses have been collected from 29 countries.

4.1 Categorization of the Sample According to the Size Criterion

Of the 392 responders, 228 (58%) are enterprises with 0 to 25 employees as illustrated in Figure 1. Note that responders of small organizations (<25 persons) that are a part of a larger enterprise are not included in these 228 responses. These 228 VSEs constitute the sample for this study. The following paragraphs present findings common to the 228 VSEs and identify correlations inside the sample, and findings that differ from those of the bigger companies that contributed to the survey.

This categorization and several studies underscore the differences between micro, small and medium enterprises in terms of available resources. Therefore, WG24 decided to focus on the first category (micro enterprises with 0-9 employees) and on a subpart of the small enterprise category (10-25 employees).

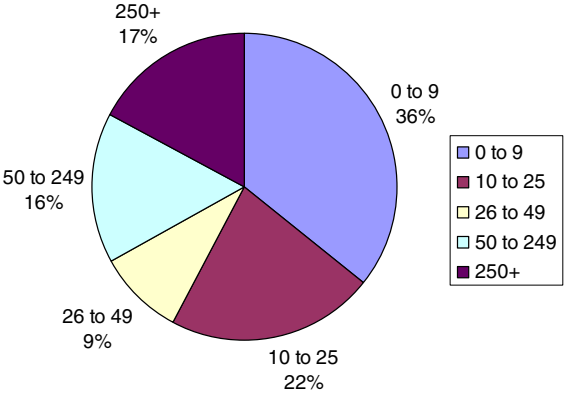


Fig. 1. Number of employees in the enterprises surveyed

4.2 General Characteristics

Here, we draw attention to some weaknesses of the sample itself. Since the survey was initiated through WG24 contacts without building a true random sample, the survey results may have been impacted. The first observation about the respondent sample, as illustrated in Table 2, is the geographical distribution of answers. We collected a high number of responses from Latin America (46%), mainly from Colombia and Brazil.

Table 2. Number of Survey Responses per Country

Country	No. of Responses	Country	No. of Responses
Argentina	2	Italy	2
Australia	8	Japan	3
Belgium	10	Korea (South)	4
Brazil	68	Mexico	20
Bulgaria	3	New Zealand	1
Canada	8	Peru	4
Chile	1	Russia	4
Colombia	88	South Africa	10
Czech Rep.	3	Spain	2
Ecuador	9	Taiwan	1
Finland	13	Thailand	52
France	3	Turkey	1
India	57	United Kingdom	2
Ireland	10	United States	3

At the same time, we received only a few responses from European countries (48), Japan (3) and the United States (3). Therefore, our results may only generalize to the broader populations of projects in each region to the extent that this sample represents them. Moreover, we have no evidence that participating companies are representative of the situation in their own countries.

4.3 Use of Standards

An interesting finding of the survey is the difference in the percentage of certified companies with regard to company size: less than 18% of VSEs are certified, while 53% of larger companies (more than 25 employees) claim to be certified. Furthermore, among the 18% not certified, 75% do not use standards. In larger companies using standards, two families of standards and models emerge from the list: ISO standards (55%) and models from the Software Engineering Institute (SEI) (47%).

WG24 anticipated the weak use of standards by VSEs by asking questions designed to provide a better understanding of the reasons for this. There are three main ones, as shown in Figure 2. The first is a lack of resources (28%); the second is that standards are not required (24%); and the third derives from the nature of the standards themselves: 15% of the respondents consider that the standards are difficult and bureaucratic, and do not provide adequate guidance for use in a small business environment.

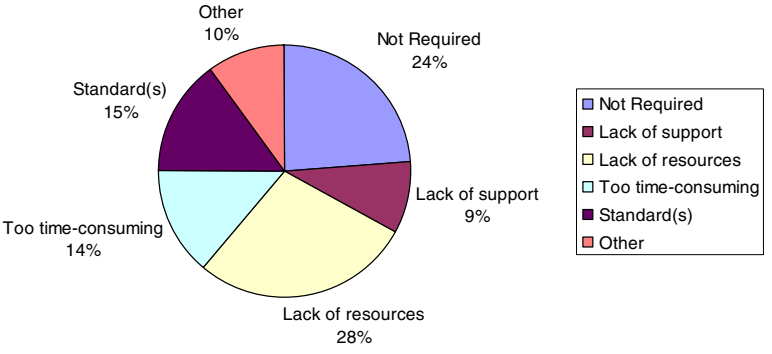


Fig. 2. Why VSEs do not use standards

For a large majority (74%) of VSEs, it is very important to be evaluated or certified against a standard. ISO certification is requested by 40% of them. Of the 28% requesting official market recognition, only 4% are interested in a national certification. From the VSE perspective, some benefits provided by certification are:

- Increased competitiveness
- Greater customer confidence and satisfaction,
- Greater software product quality
- Increased sponsorship for process improvement
- Decreased development risk
- Facilitation of marketing (e.g. better image)
- Higher potential to export

However, VSEs are expressing the need for assistance in order to adopt and implement standards. Over 62% would like more guidance with examples, and 55% are asking for lightweight and easy-to-understand standards complete with templates. Finally, the respondents indicated that it has to be possible to implement standards with minimum cost, time and resources. All data about VSEs and standards clearly confirm WG24's hypothesis and the requirements. Therefore, WG24 uses this information to help define its approach for the development of profiles, guides and templates to meet VSE needs.

5 The WG24 Approach

The approach used by WG24 had to take into account, as a starting point, the ISO requirements in terms of standard definition. Indeed, since an international standard dedicated to software lifecycle was already available (i.e. ISO/IEC 12207) [17], WG24 had to use the concept of ISO profiles (ISP – International Standardized Profile) in order to develop the new standard for VSEs. A Profile is defined as "A set of one or more base standards and/or ISPs, and, where applicable, the identification of chosen classes, conforming subsets, options and parameters of those base standards, or ISPs necessary to accomplish a particular function" [18]. From a practical point of

view, a Profile is a kind of matrix that identifies precisely all elements that are taken from existing standards from those that aren't.

The overall approach followed by WG24 to develop this new standard for VSE consisted of three steps:

- Select ISO/IEC12207 process subset applicable to VSEs of less than 10 employees
- Tailor the subset to fit VSE needs
- Develop guidelines

Firstly, since WG24 wished to prepare an initial set of software development standards as quickly as possible, WG24 analyzed international reference standards and models that could help subset ISO/IEC 12207 for low maturity VSEs. To achieve these initial products quickly, WG24 began a search for existing standards or models that could be tailored. Moprosoft, a Mexican standard developed to assist Mexican small and medium enterprises (SMEs) has been selected in order to achieve this objective [19].

Moprosoft uses ISO/IEC 12207 as a general framework. It borrows practices from ISO9001, the Capability Maturity Model Integration (CMMI) developed by the Software Engineering Institute, the Project Management Body of Knowledge (PMBOK) and the Software Engineering Body of Knowledge SWEBOK.

However, WG24 felt that Moprosoft was addressing the needs of organizations larger than targeted VSEs. Therefore, as a second step, WG24 decided to tailor Moprosoft in order to address key characteristics of low maturity VSEs. The tailoring approach lead to the development of incremental profile targeting as starting point, low maturity VSE of less than 10 employees and, in a second phase, those with 10 to 25 employees. Therefore, the first profile, developed by WG24, contains basic activities coming from project management and software development related processes. The idea was to concentrate on core activities that a low maturity VSE should perform.

The first document of the family of documents developed by WG24, titled "Overview", introduces the major concepts required to understand and use the suite of documents. It introduces the business aspects, characteristics and requirements of VSEs, and clarifies the rationale for VSE-specific profiles, documents, standards and guides. It also introduces basic process, lifecycle and standardization concepts, and the 29110 family of documents. It is targeted both at a general audience interested in these documents, and more specifically at users of these documents. The Overview is identified as technical report (TR) TR 29110-1.

The second set of documents; titled "Profiles" are defined to formally package references to and/or part of other documents in order to adapt them to the VSEs needs and characteristics. Preparing profiles is an ISO/IEC JTC1 defined process. It involves producing two types of documents: a framework and taxonomy and a profile specification:

- **Framework and Taxonomy** - The Framework and Taxonomy document (ISP29110-2) establishes the logic behind the definition and application of profiles. It specifies the elements common to all profiles (structure, conformance, assessment) and introduces the taxonomy (catalogue) of 29110 profiles. It is targeted at authors and reviewers of ISPs, authors of other parts, and authors of other

VSE-targeted profiles. The Framework and Taxonomy is applicable to all profiles and identified as TR 29110-2

- **Profile Specifications** - There is a profile specification document for each profile. Its purpose is to provide the definitive composition of a profile, provide normative links to the normative subset of standards (e.g. ISO/IEC 12207) used in profile, and provide informative links (references) to "input" documents (e.g. 90003, SWEBOK, PMI). It is targeted at authors/providers of guides, and authors providers of tools and other support material. There is one profile specification document for each profile, identified as 29110-4.x, where x is the number assigned to the profile.

The third set of documents, titled "Guides", contain implementation guidelines (domain specific) on how to perform the processes to achieve the maturity levels (e.g. recommended activities, measures, techniques, templates, models, methods ...). Guides are developed for the process implementation and for the assessment based on the domain's issues, business practices and risks. Guides are targeted at VSE, and should be VSE accessible, both in terms of style and cost. There are two guides: an assessment guide and a management and engineering guide:

- **Assessment Guide** - This guide describes the process to follow to perform an assessment to determinate the process capabilities and the organizational process maturity. This is, when an organization wants an assessment execution in order to obtain a process capability profile of the implemented processes and an organizational process maturity level. It is also applicable to the situation where customer asks for a third-party assessment execution in order to obtain a capability level profile of the implemented process by the software development and maintenance provider. It is also suitable for self-assessment. The Assessment Guide is applicable to all profiles and identified as TR 29110-3
- **Management and Engineering Guides** - The management and engineering guides provide guidance on its implementation and use or a profile. It is targeted at VSE (management and technical staff), VSE-related organizations (technology transfer centers, government industry ministries, national standards, consortiums and associations, academic use for training, authors of derived products (software, courseware, and acquirer and suppliers. There is one management and engineering guide document for each profile, identified as 29110-5.x, where x is the number assigned to the profile. This number matches the number assigned to the profile specification.

The third step of the approach consisted in defining guidelines explaining in more details the processes defined in the profile. These guidelines will be published as ISO Technical Reports which should be freely accessible to VSEs. These guidelines integrate a series of deployment packages. A deployment package is a set of artifacts developed to facilitate the implementation of a set of practices, of the selected framework, in a VSE. But, a deployment package is not a process reference model. The elements of a typical deployment package are: process description (e.g. activities, inputs, outputs, and roles), guide, template, checklist, example, presentation material, reference and mapping to standards and models, and a list of tools. Packages are designed such that a VSE can implement its content, without having to implement the complete framework at the same time. The first four deployment packages being

developed are: requirements analysis and management, change management, testing and project management. Future deployment packages are: architecture, issue tracking, unit testing and coding. The table of content of a deployment package is illustrated in table 3.

Table 3. Table of Content of a deployment package

1. Introduction
Purpose of this document
Key Definitions
2. Why this Process is important
3. Overview of Main Tasks
3.1 Tasks
3.2 Roles and artifacts
3.3 Activity Lifecycle and examples of lifecycles
Annex A Templates
Annex B Checklists
Annex C Coverage Matrices (ISO 12207, ISO 9001, CMMI)
Annex D Tools
Annex E Training Material
Annex F Deployment Package Evaluation Form

5.1 Recent Developments

At the Montreal meeting of WG24, in October 2007, the requirement analysis and management deployment package has been reviewed and received a broad support from the group members. The group decided to develop following deployment packages for its next meeting in Berlin: configuration management, project management, and testing.

Having profiles and guides for VSEs is not sufficient to ensure broad utilization and adoption: they have to be tested with real VSEs of a few countries. The Mexican delegation presented the result of the introduction, as pilot projects, of the first profile developed by WG24, in Latin American countries [20]. Also a new country, Colombia, and a new organization, the European Software Institute (ESI), joined WG24.

6 Conclusion and Future Work

Industry recognizes the value of VSEs in their contribution of valuable products and services. About 75% of software enterprises worldwide have fewer than 25 employees. ISO/IEC JTC1 SC7 standards are not easily applied in VSEs that generally find standards difficult to understand. Hence, VSEs require further guidance in order to integrate standards into their practices. ISO/IEC JTC1 SC7 decided to establish a new working group to address these issues.

With regard to future work, WG24 plan to invite VSEs to participate in the field trials before the standards get published by ISO Since a few WG24 delegates are already working closely with VSEs, they will play a key role in the coordination of the trials. Trials will help validate the approach and obtain feedback in order to improve the documents before going for ISO/IEC publication. WG24 is planning to produce a Final

Draft in 2009. Publication by ISO/IEC is scheduled for 2010. In the meantime, deployment packages will be made available, to VSEs, on public web sites.

Additional Information
The following Web sites provide more information as well as articles and eventually deployment packages, which members of WG24 will develop:

http://profs.logti.etsmtl.ca/claporte/English/VSE/index.html
http://www.cetic.be/indexEN.php3

References

1. Charette, R.N.: Why Software Fails, Spectrum, pp. 42–49. IEEE Computer Society, Los Alamitos (2005)
2. Shintani, K.: Empowered Engineers are Key Players in Process Improvement. In: The First International Research Workshop for Process Improvement in Small Settings, Software Engineering Institute, CMU/SEI-2006-SR-01, Pittsburgh, PA (2006)
3. Laporte, C.Y., April, A.: Applying Software Engineering Standards in Small Settings: Recent Historical Perspectives and Initial Achievements. In: Proceedings of the First International Research Workshop for Process Improvement in Small Settings. Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2006-Special Report-001, pp. 39–51 (January 2006)
4. Ginsberg, M., Quinn, L.: Process Tailoring and the Software Capability Maturity Model, Software Engineering Institute, CMU/SEI-94-TR-024 (November 1995)
5. Hadden, R.: Key Practices to the CMM: Inappropriate for Small Projects, Panel. In: Proceedings of the Software Engineering Process Group Conference, Chicago (1998)
6. Johnson, D., Brodman, J.: Applying the CMM to Small Organizations and Small Projects. In: Proceedings of Software Engineering Process Group Conference, Chicago (1998)
7. Laporte, C.Y., April, A., Renault, A.: Applying ISO/IEC Software Engineering Standards in Small Settings: Historical Perspectives and Initial Achievements. In: Proceedings of SPICE Conference, Luxembourg (2006)
8. European Commission, The New SME Definition: User Guide and Model Declaration (2005), http://europa.eu.int/comm/enterprise/enterprise_policy/sme_definition/sme_user_guide.pdf
9. Coleman, G., O'Connor, R.: Investigating Software Process in Practice: A Grounded Theory Perspective. Journal of Systems and Software 81(5), 772–784 (2008)
10. Laporte, C.Y., Renault, A., Desharnais, J.M., Habra, N., Abou El Fattah, M., Bamba, J.C.: Initiating Software Process Improvement in Small Enterprises: Experiment with Micro-Evaluation Framework. In: SWDC-REK, International Conference on Software Development, University of Iceland, Reykjavik, Iceland, May 27-June 1, 2005, pp. 153–163 (2005)
11. Anacleto, A., von Wangenheim, C.G., Salviano, C.F., Savi, R.: Experiences gained from applying ISO/IEC 15504 to small software companies in Brazil. In: 4th International SPICE Conference on Process Assessment and Improvement, Lisbon, Portugal (April 2004)
12. Mtigwe, B.: The entrepreneurial firm internationalization process in the Southern African context: A comparative approach. International Journal of Entrepreneurial Behavior & Research 11(5), 358–377 (2005)

13. Hofer, C.: Software Development in Austria: Results of an Empirical Study among Small and Very Small Enterprises. In: Proceedings of the 28th Euromicro Conference, pp. 361–366 (2002)
14. Coallier, F.: International Standardization in Software and Systems Engineering, Crosstalk, pp. 18–22 (February 2003)
15. New Work Item Proposal – Software Life Cycles for Very Small Enterprises, ISO/IEC JTC1/SC7 N3288 (May 2005), http://www.jtc1-sc7.org/
16. Land, S.K.: Results of the IEEE Survey of Software Engineering Standards Users. In: Software Engineering Standards Symposium and Forum, 1997. Emerging International Standards. ISESS 1997, Walnut Creek, CA, June 1-6, pp. 242–270 (1997)
17. ISO/IEC 12207:2008, Information technology – Software life cycle processes. International Organization for Standardization/International Electrotechnical Commission: Geneva, Switzerland
18. ISO/IEC TR 10000-1:1998, Information technology: Framework and taxonomy of International Standardized Profiles. Part 1: General principles & documentation framework
19. NMX-059-NYCE-2005, Information Technology-Software-Models of Processes and Assessment for Software Development and Maintenance. Part 01: Definition of Concepts and Products; Part 02: Process Requirements (MoProSoft); Part 03: Guidelines for Process Implementation; Part 04: Guidelines for Process Assessment (EvalProSoft), Ministry of Economy, Mexico (2005)
20. Oktaba, H., Felix, G., Mario, P., Francisco, R., Francisco, P., Claudia, A.: Software Process Improvement: The Competisoft Project. IEEE Computer 40(10) (October 2007)

Lightweight Process Documentation: Just Enough Structure in Automotive Pre-development

Kai Stapel[1], Eric Knauss[1], and Christian Allmann[2]

[1] FG Software Engineering, Leibniz Universität Hannover
Welfengarten 1, 30167 Hannover, Germany
[2] Audi Electronics Venture GmbH
Sachsstraße 18, 85080 Gaimersheim, Germany
{eric.knauss,kai.stapel}@inf.uni-hannover.de,
Christian.Allmann@audi.de

Abstract. Pre-development in the automotive sector is informally organized to support the engineers trying out new ideas and generally being creative. If feasibility studies reveal system's uncertainties or bad market opportunities and the development has to be discarded, all documentation attached is obsolete. As a result it is neither possible nor desirable to establish a document centric process in automotive pre-development.

However, without a defined development procedure it is hard to improve and validate development outcomes, respectively to repeat success strategies as well as to integrate new personnel. Furthermore, if new innovations do pass to series development, system characteristics and development activities certainly have to be documented.

In this contradictory situation we cannot apply traditional document centric process approaches. Instead we make use of our Information Flow Analysis. This way it is possible to document and analyze the pre-development activities. Based on our conclusions we developed lightweight concepts to systematically capture documentation from the engineers, without hindering their creativity. These concepts were incorporated in a semantic Wiki, in an effort to give a suitable starting point for comprehensive documentation in case of pre-development projects going into production.

1 Introduction

The role of automotive pre-development is to evaluate and demonstrate the technical feasibility of new ideas and technologies as the base for client and market studies. Generally speaking, pre-development is similar to research in general. The main difference between research and pre-development is the timeline. All pre-development activities focus on the contemporary (up to 3 years) vehicle deployment. Both have in common that the starting point can be described as a development on the "green field" [1, 2]. Meaning that the amount of requirements is indefinite, the technical realization is unknown, the identification of all relevant stakeholders is still ongoing, and the client usage can not be demonstrated. Therefore, the main goal of the development activities is to build up a system demonstrator regularly. In automotive pre-development this usually is a vehicle prototype. As in conventional systems engineering and software

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 142–151, 2008.

engineering, these prototypes prove whether a technical solution is reasonable or not. The main challenge assembling the vehicle prototypes is caused by the implementation of new technologies (e.g. sensors and image processing) and by networking formerly independent functions. Both rely on a growing number of increasingly powerful and highly integrated mechatronic components. The amount of functions and the extent of network interaction are only two indicators for system and thus pre-development complexity. Another dimension originates from the diversity and the overlapping of development domains where a growing number of systems are a compound of mechanical, electronic and software components. Traditionally, the activities within these disciplines are carried out separately, often within independent departments at the original equipment manufacturer (OEM) and the supplier. This further increases pre-development complexity and demands high efforts for the communication between all involved parties and the coordination of the system development.

In case a new technology and system development verified by a vehicle prototype does not demonstrate feasibility, the collected experience still has to be documented in order to avoid developers to try the same again. However, in such a case comprehensive documentation should be avoided, because it does not add value to the final product.

A difference of prototyping in the automotive context in contrast to prototyping in conventional software engineering is that developers responsible for the prototypes are not necessarily responsible for the series product development. Therefore, if an innovation is proved to be valuable for series deployment, comprehensive documentation of this technical invention is needed after all.

Because of these conflicting aspects (creative and innovative solution finding vs. good documentation of proved solutions) the process of pre-development is a challenging object of research. On the one hand these projects need to be organized, because they contribute to the company's success. Therefore, it is important to improve chances of success. Consequently, a process model is needed to make these projects' success repeatable. This is a typical goal of traditional process- or maturity models.

On the other hand traditional process modeling approaches are not designed to deal with the exchange of experiences and technical know-how in coffee-corners. Because of the creativity involved in pre-development a specification of a strict sequence of activities and process artifacts is undesirable. The main task is to make relevant information available to all project members, support exchange of information between projects, and help project leaders to cope with an informally organized process. We also strive to create an overview of pre-development projects, to give new project members some orientation.

In this context we applied our Information Flow Analysis [3, 4] in order to capture the information flows between project participants. We were able to derive a project map that shows which information flows are important for success in pre-development projects. It turned out that especially the documentation of project experiences is difficult in this context. As a result, we enhanced a Wiki-Web application to better support these crucial information flows (e.g. introducing templates for quick documentation of experiences and observations).

This paper is organized as follows: Section 2 gives some foundations and shows how our approach distinguishes from other approaches. Section 3 contains the design of our study as well as the foundations of Information Flow Analysis. The actual

analysis with its findings and highlights is presented in section 4. We give a short account of the direct benefits and consequences from the analysis in section 5. Finally, we draw our conclusions and give an outlook of open research questions and future activities in this area.

2 Related Work

Today, most of vehicle or system innovations arise from the intelligent integration of individual, complex (sub-)systems into the vehicle network which must fulfill vehicle constraints like real-time, safety and dependability aspects. On the other hand these (sub-)systems nowadays are developed by a network of suppliers together with the OEM. The manufacturer is in the role of the client and at the same time manager of the system to be developed. He defines the requirements and monitors the project through testing. One conflict arises because on the one hand the OEM is depended on the specific knowledge of the supplier when it comes to individual components (e.g. specific sensor devices). On the other hand the supplier depends on the OEM's knowledge of the environmental conditions he has to comply with, like construction space, physical ambient parameter, dependencies to other systems or the power supply. Therefore the interaction of OEM and the supplier has to be considered as a key factor for success in this kind of projects. Naturally both sides have a strong interest to protect their intellectual property. So an efficient development is not only difficult to realize due to system complexity, but also because of the gap of communication resulting from the diverging views of OEM and suppliers [5, 6].

At first glance applying a formal process model like the Rational Unified Process [7] or V-Modell XT [8] seems to be a good method to coordinate interactions between OEM and supplier. But, a strict process requiring a lot of documentation rather hinders creativity and innovation needed in pre-development instead of enabling them. Thus, our approach is a lightweight method to document the already working informal process. The method is called Information Flow Analysis [3, 4]. The resulting information flow map can be used as a base for process improvements. In this context lightweight means two things: First, the method to collect the information flow data and create the according model is easy to learn and easy to use. Furthermore, with the method it does not take a lot of time to get an overview of a process. Second, lightweight means process shaping without bothering the developers too much. Regular processes provide a lot of benefits but also demand a lot of duties. In the automotive pre-development context a lightweight process with fewer duties provides a better cost-benefit ratio.

3 Study Design

In this section we present the study design that we used to conduct the information flows in an automotive pre-development context. The basis for this is our Information Flow Analysis, which therefore will be introduced first. After that the actual study design will be presented. Finally, some notes on the execution of the elicitation are given.

3.1 Information Flow Basics

Information is the most important resource in development projects, this applies particularly to pre-development. The correct flow of information is essential for project

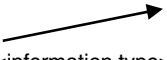
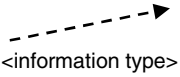
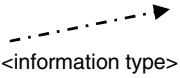
success and product quality. Especially in pre-development information is not only being passed on by documents but also by e-mails, phone calls, and direct communication like in meetings.

Information Flow Analysis is our approach to conquer the challenges with different ways of information exchange. Information flows are modeled, analyzed and optimized based on the following fundamental concepts:

- *Information appears in different states. Fluid* information is verbal or non-objectively reproducible information including e-mails and personal notes third parties cannot access or reproduce. *Solid* information refers to written or recorded information (like documents or videos) which is long-term accessible even to third parties.
- *Experience is a special kind of information* which is being modeled explicitly. It often influences activities and acts as a catalyst. Experiences made in one project can be of value in others.
- *Coarse modeling of information content.* Just the type of information is being modeled, e. g. product requirements, experiences, or development decisions, not the exact content.

In order to be able to note down the information flows of a given project a simple, easy to understand, and easy to use notation is needed. All relevant aspects of information and its flow need intuitive representation. The following Information Flow Notation was designed to fulfill these needs:

Table 1. Information Flow Symbols

information state	store	information flow	experience flow	activity
solid	 <identifier>	 <information type> (optional)	 <experience> (optional)	
fluid	 <identifier>	 <information type> (optional)	 <experience> (optional)	<activity>
combined solid-fluid	 <identifier>	 <information type> (optional)	 <experience> (optional)	

Especially these aspects were addressed:

- Means of expression for state of information for both, information stores and information flows. An information store can be *solid*, represented by a document symbol (since a document is the most prominent solid information store), or fluid, represented by a smiley symbol (fluid information is stored in peoples' minds).

Information flows can be *solid* and *fluid* as well. The state of an information flow is determined by its originating information store: If the originating store is solid the flow is also solid (represented by a solid line), if the originating store is fluid the flow is fluid (represented by a dashed line).

- Means of expression for experiences. To be able to distinguish experience flows from other information flows, experience flows and stores are depicted in a different color (e. g. gray).
- The ability to establish connections between information flow models and process models. The activity symbol is available in both notations and therefore acts as a connection point (all process notations have a symbol for activities/functions).
- Very easy understandable even for non computer scientists. Few, easy explainable symbols are used.
- Fast and Easy to use. Especially for pragmatic reasons the combined solid-fluid store and flow were introduced. In a few special cases it is not that important in what state certain information flows, it is just important that it flows at all.

With the Information Flow Notation and the basic principles of Information Flow Analysis in mind we designed the elicitation.

3.2 Elicitation Design

Besides the Information Flow Analysis concepts we incorporated the automotive pre-development context in our elicitation design. The following general assumptions express that.

General Assumptions

1. A fluid information culture is important in automotive pre-development.
2. Rework has to be done in case a new approach makes it to serial production because of sparse documentation in pre-development.
3. There are reoccurring project patterns indicating a certain process despite of the fluid information culture and the informal process.
4. Interviewees will be more open when interviewed by an intern staff member.

Based on these assumptions we designed the study. With the given time and resource constraints collecting data via interviews was most promising. As interviewees several project managers and engineers of different pre-development projects were selected.

Elicitation with interviews is a bottom up technique. Each interviewee gives a low level view on the project. In the following analysis step these local views are put together to build up a global pre-development view which then can be used as a base for further discussions and improvements.

Each interview consists of two parts:

1. A catalogue of general questions: employee background, project, involved roles and persons, general information flows, experiences, what works good, what does not, etc.
2. Information Flow data entry forms: For each work-task of an interviewee a form has to be filled. Data about the task, the executing role, required information, supporting information, and outgoing information is being collected.

To verify assumption 1 questions concerning the distribution of verbal and written communication or how the work result gets to its users were incorporated in the catalogue of general questions. Because of assumption 4 an intern staff member was used to conduct the interviews. A problem with that is that industry staff usually is not familiar with Information Flow Analysis concepts. Therefore we had to train the intern staff member.

3.3 Elicitation Execution

A widely accepted intern was selected as the intern staff member to conduct the interviews. We taught her the Information Flow Analysis concepts in a personal instruction. It turned out that the training of the new concepts only needed short time to be successful. Thus, we conclude that the Information Flow concepts are easy to understand. This adds to the fact that Information Flow Analysis is a lightweight method.

In total 6 interviews were held with staff members from 4 different pre-development projects. The intern scheduled and conducted the interviews herself and reported the results to us. We then analyzed each interview and built up the global information flow model.

4 Information Flow Analysis

After the elicitation phase the raw data from the interviews had to be analyzed to build up the global view. The results are presented in this section. First the unmodified answers from the interviews are presented. After that our interpretation and aggregation is given.

4.1 Results

The interviews confirmed assumption 1 (see section 3.2.). There is a mainly fluid information culture. The interviewees claimed that between 60% and 90% of the information being shared is fluid. Everybody stated that most information is exchanged in meetings. It was also stated that there is few required documentation in pre-development.

Assumption 3 could also be confirmed. Despite of working in different projects the interviewees mentioned many tasks that were common among the projects:

- Organizational and project management tasks
- Information search tasks
- Supplier analysis tasks
- Vehicle prototype construction tasks
- Coordination tasks with departments

Most interviewees summed up, that pre-development intern communication mainly works well, but coordination with departments often causes difficulties because of unknown competencies and missing documentation.

4.2 Aggregation

In this section we present the results of the interpretation and the aggregation of the raw interview data. This is the final and most important step in Information Flow Analysis.

First we created a communication network showing the staff and their direct communication paths without documents. Generally speaking, from such a communication network one can see who is talking to whom.

We then build a generic pre-development project model, because several tasks common to all projects could be identified (assumption 3). From this an employee new to pre-development can be instructed or a project manager setting up a new pre-development project can create an instance of the generic model and use it for e. g. project planning.

Experiences were not systematically used throughout all projects. Especially in frequently reoccurring tasks this leads to a lot of rework. A typical frequently reoccurring task in automotive pre-development is the construction of a vehicle prototype to test new approaches under real world conditions. A generic test vehicle construction process as derived from the interviews is depicted in figure 1.

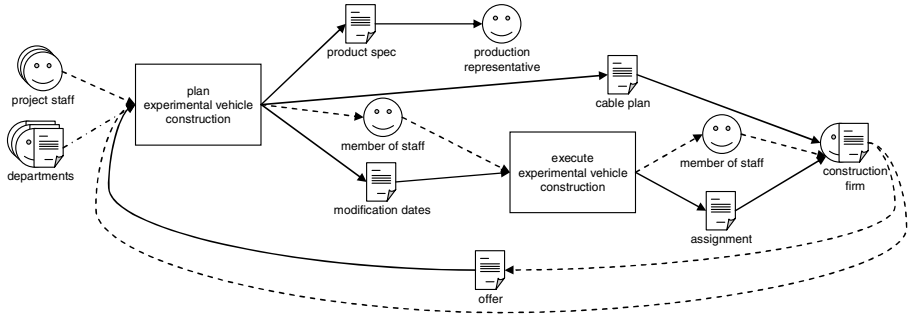


Fig. 1. Information Flow Model of Present Vehicle Prototype Construction Process

The vehicle prototype construction task is divided into two main activities: plan and execute. The planning activity incorporates information from project staff, departments and the construction firm. The outcome is a product specification, some calendar entries and a cable plan for the vehicle. The execution activity is mainly concerned with the assignment of the construction firm with the actual construction of the vehicle. The offer and other results from the construction firm are incorporated in the planning activity of the next iteration. In particular it is noticeable that no experiences are shared to be helpful in following iterations.

Based on these observations, we drew some direct consequences. On the one hand, we enhanced an existing Wiki-Web to better reflect the information flow demands. On the other hand we showed our results to the developers and started a discussion of the pre-development process model.

5 Drawing the Consequences

In the opinion of the analyzed pre-development department clear documentation between the roles on the side of the pre-development team, adjacent departments and involved suppliers is needed. Methods and tools like a customized Wiki are supposed

to map both the horizontal and the vertical information flows to avoid that communication becomes the bottleneck of development.

In this context the horizontal flow represents the organizational hierarchy (development departments) and the vertical depicts the supplier chain. Therefore, all relevant stakeholders, developers as well as manager decisions, process steps, etc. can be mapped to the actual development state. Hence, the Information Flow Analysis revealed communication bottlenecks, knowledge drains and loosely coupled documents. This unbalanced relationship between stakeholders, documentation and system development activities call for suitable methods facilitating the communication, knowledge and project experience acquisition in pre-development projects. Scenario based development like it is described in [13-16] can be used to bridge the gap between less documentation and documentation centric approaches.

For future development in highly innovative fields, like hybrid technologies, x-by-wire or sensor fusion, documented project experience is inevitable. The mentioned distributed information flow is very sophisticated to organize and to lead. Passing experience on to further development means sharing knowledge, the knowledge about the actual developed system as well as the project experience each team member has collected over the years. The illustrated clash of development "philosophies" requires some kind of cooperation platform to organize the different development activities, determine the development chain and teams responsibilities. *Time-shared* and *artifact-shared* development is mandatory concerning the effort for system's development under the pre-development circumstances.

All mentioned implications can be subsumed to one essential point: a common platform for planning, documenting, and representing pre-development projects is crucial. To avoid misunderstandings this platform comprehends solutions for arising communication overhead, distributed access to the different knowledge basis, project management, and tools. A Wiki has proved to be a pragmatic technology for such a platform and has already been used in some projects at the analyzed pre-development department.

This specific Wiki is designed around scenarios. It is based upon an extension of the MediaWiki [9] and a semantic Wiki extension [10]. With the help of this extension it is

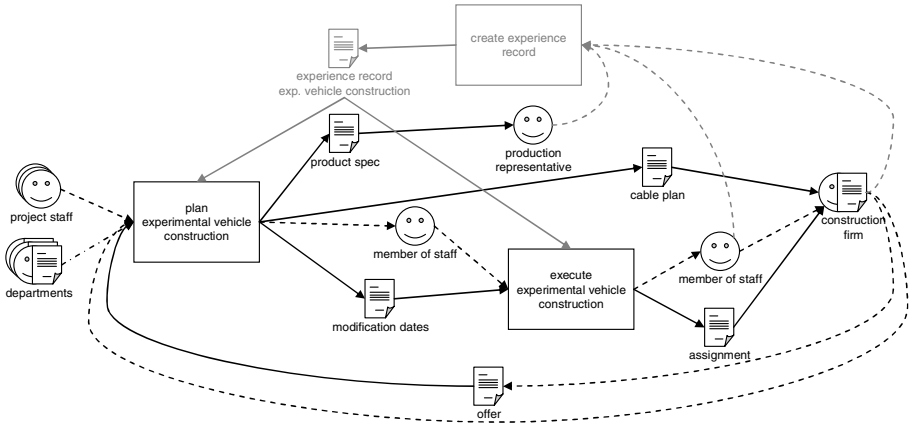


Fig. 2. Information Flow Model of Improved Vehicle Prototype Construction Process

possible to define templates as well as relations between pages in the Wiki (e.g. templates that allow scenario-based requirements engineering, project-management activities, and reports).

Based on our findings we decided to improve this Wiki: For example, the Information Flow Analysis revealed improvement potential for documentation of project experiences.

Figure 2 shows which information needs to be included in experience reports. Consequently we introduced links to observation and experience report templates at the according positions in the Wiki. For example, if the production representative reads the product specification, she has a direct facility to leave some experience-related remarks on that specification. Note that it is important to reduce the effort of documenting experiences in order to improve the chances of a developer actually doing it. [11, 12]

The documentation of experiences is only one of the challenges of learning organizations. The other big challenge is to apply relevant experiences to the development activities. In the example above we introduced overviews of existing experiences into relevant Wiki pages that relate to the activities of planning vehicle prototype construction and its execution.

6 Conclusion and Outlook

In this paper we showed that Information Flow Analysis is a fast and lightweight method to reveal and document informally organized processes like pre-development processes in auto industry. While traditional approaches proved to be unsuitable, we were able to capture the important aspects of pre-development with Information Flow Analysis. This is because the Information Flow concepts enable to distinguish between different states of information. The specialized Information Flow Notation makes it possible to depict non-document-based communication that is particularly important and common in pre-development.

The results of Information Flow Analysis give a good starting point to build supporting information systems. In our study we were able to enhance a Wiki system to better support the important information flows. While this Wiki is a good prototype for this area, further work in IT-support for pre-development is needed. On the one hand, effort for documenting project relevant information has to be further reduced. On the other hand tool support has to ensure that enough documentation is created to support the start of production. In our opinion, a good strategy of lightweight tools is to support developers doing their daily tasks while systematically storing additional documentation as a by-product.

Generally, our experiences show that an Information Flow Map of a pre-development project is a valuable artifact. It serves as an important foundation for discussion. This way such a map can lead to faster education of new project members or to process improvements.

References

1. Weber, M., Weisbrod, J.: Requirements Engineering in Automotive Development: Experiences and Challenges. IEEE Software 20(1), 16–24 (2003)
2. Allmann, C.: Automotive Pre-Development, requirements management based on the green field? Softwaretechnik Trends 27(1) (2007)

3. Stapel, K., et al.: Improving an Industrial Reference Process by Information Flow Analysis: A Case Study. In: Münch, J., Abrahamsson, P. (eds.) PROFES 2007. LNCS, vol. 4589. Springer, Heidelberg (2007)
4. Schneider, K.: Software Process Improvement from a FLOW Perspective. In: Learning Software Organizations Workshop. Springer, Kaiserslautern (2005)
5. Allmann, C., Oppermann, N., Kovacevic, S.: Simulation-driven functional safety evaluation of embedded automotive systems. In: 8th International Stuttgart Symposium "Automotive and Engine Technology", Stuttgart, Germany (2008)
6. Allmann, C., Winkler, L., Kölzow, T.: The Requirements Engineering Gap in the OEM-Supplier Relationship. Journal of Universal Knowledge Management 1(2), 103–111 (2006)
7. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley Professional, Reading (2003)
8. VMXT, V-Modell XT (Version 1.2), Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (2006), http://v-modell.iabg.de/v-modell-xt-html-english/index.html
9. Wikimedia-Foundation, MediaWiki, http://www.mediawiki.org
10. Wikimedia-Foundation, Semantic MediaWiki, http://semantic-mediawiki.org
11. Basili, V., Caldiera, G., Rombach, D.H.: The Experience Factory. Encyclopedia of Software Engineering. John Wiley and Sons, Chichester (1994)
12. Schneider, K.: LIDs: A Light-Weight Approach to Experience Elicitation and Reuse. In: Bomarius, F., Oivo, M. (eds.) PROFES 2000. LNCS, vol. 1840. Springer, Heidelberg (2000)
13. Allmann, C., Oppermann, N.: Lightweight requirements management in the automotive pre-development. In: Software Engineering 2008, Munich, Germany (2008)
14. Jacobson, I.: Object Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Reading (1992)
15. Benner, K., et al.: Utilizing Scenarios in the Software Development Process. In: Proceedings of the IFIP WG8.1 Working Conference on Information System Development Process, Amsterdam, Netherland (1993)
16. Sutcliffe, A.: Scenario-based requirements analysis. Requirements Engineering Journal 3(1), 48–65 (1998)

Employees' Motivation for SPI: Case Study in a Small Finnish Software Company

Anu Valtanen and Hanna-Miina Sihvonen

University of Kuopio, Department of Computer Science, P.O.B 1627,
FI-70211 Kuopio, Finland
{anu.valtanen,hanna-miina.sihvonen}@uku.fi
http://www.cs.uku.fi

Abstract. In small software companies the resources available for SPI are often limited. With limited resources, the motivation of the employees becomes one of the key factors for SPI. In this article, the motivational factors affecting a small company's SPI efforts are discussed. In the research, we carried out interviews and a survey in a small Finnish software company considering the motivation towards SPI. The results are presented here and compared with earlier motivation research. There were differences revealed while comparing the motivating factors of smaller companies to those of larger ones. In large companies the focus seems to be on the business related motivators and in small ones the motivators related to comfortability of work are emphasized. Motivation survey and the interviews proved to be useful tools in planning the future SPI strategy. A lot of valuable information was discovered for planning and implementing the next steps of SPI.

1 Introduction

Small and very small software companies are fundamental to the growth of many national economies and it is crucial to note that small companies should not be seen less important and influential than large ones, while the term small may imply this. Majority of software companies are small [1], for example in Finland vast majority of companies operating in both data processing and software engineering fields employ less than 50 employees[1]. Small companies need to maintain and enhance their competitiveness and for that they need to improve their processes. However, small companies do not necessarily share the same characteristics and goals as large ones, which affect SPI efforts. There are certain unique features of small companies that need to be understood [1,2]. Their resources, both financial and human, are often limited, and management, work, and organizational culture may differ greatly from the ones in large organizations. For example, in small software companies employees often work in several roles and practically every employee is involved directly or indirectly in software development process, whether one wants or does not want to be involved, whether one has software engineering background or not.

[1] http://www.stat.fi (2006)

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 152–163, 2008.

Considering the limited resources – especially the employees, the human resources - who represent a crucial part of companies' process infrastructure [3], their motivation for SPI has to be taken into account as one important factor for successful improvement initiatives and implementation. Their motivation to adapt new practices to daily work is significant for the SPI success. Practitioners' SPI motivators and de-motivators have been studied earlier in different cultural contexts for example by Baddoo et al. in UK [4,5,6] and Niazi et al. in Vietnam [7]. However, previous studies of SPI motivation have concentrated on exploring motivation mainly in large and SME sized software companies, and not emphasized the small and very small software companies. In this paper, we present a case study of SPI motivators in a small Finnish software company. We have studied SPI motivation applying methods and motivators from studies of Baddoo et al. [4] and Niazi et al. [7]. The focus in this research is on considering motivation from employees' and management's perspectives. The results are compared with the previous studies [4,7].

This paper aims at providing a more comprehensive perspective of small software companies' employees' motivation and point of views about SPI. Additionally, the effect of the management's motivation towards SPI in a small company is discussed. The paper is organized as follows: In Section 2, the case context and research questions are presented. In Section 3, the research strategy is described. In section 4, we present the findings from the research. In Section 5, we present comparison of the results to earlier studies. Finally, in Section 6 we conclude the results and point out the most important findings.

2 Case Context and Research Questions

The motivation of the employees is an important and much researched subject in context of SPI [4,5,6,7]. However, the research usually neglects small and very small software companies. In this research, SPI motivational factors of a small company's employees are contemplated and answers to following questions are searched:

- What motivates the employees of a small company to SPI?
- What motivates the management of a small company to SPI?
- What is the role of motivation for SPI success in a small company?

The research was carried out in form of a case study in a small Finnish company that has 17 employees. A case study is said to have "a distinct advantage when a 'how' or 'why' question is being asked about a contemporary set of events over which the investigator has little or no control." [12] and as a method, suits this research very well.

The company where the research was performed in is a traditional software house. The company's working environment is quite free and the company has a low hierarchy. The employees of the company are loosely divided into departments where their main responsibility areas are. Division of the workforce that participated in this research is presented in Table 1. The CEO of the company

participates practically in every activity and because of this has a good knowledge of what is going on in the company. The employees feel free to talk about their work related problems and other issues.

Table 1. The Profiles of the Employees

	MainResponsibility Area	*Sales*	*Adv.*	*Con.*	*Def.*	*Des.*	*Prog.*	*Doc.*	*Adm.*	*Test.*	*Man.*	*SPI%*
1	Management	-	-	-	x	x	-	-	-	x	x	85
2	Sales and Marketing	x	-	-	-	-	-	-	-	-	-	6
3	Sales and Marketing	x	-	x	-	-	-	-	x	x	-	3
4	Development	-	-	-	x	x	x	x	x	-	-	66
5	Development	-	-	-	-	-	x	-	-	-	-	6
6	Development	-	x	-	x	x	x	x	x	x	-	12
7	Development	-	-	-	x	x	x	x	x	-	-	15
8	Development	-	-	-	x	x	x	x	x	-	-	51
9	Development	-	-	-	x	x	x	x	-	-	-	45
10	Customer Support	x	x	-	x	x	x	x	x	x	-	90
11	Customer Support	x	x	-	x	x	x	x	x	x	-	90
12	Customer Support	x	x	x	-	-	-	-	-	x	-	3
13	Customer Support	x	x	-	x	x	-	x	x	x	-	85

Sales = Salesperson
Adv. = Advisor
Con. = Consigner
Def. = Definer
Des. = Designer
Prog. = Programmer
Doc. = Documenter
Adm. = Administrator
Test. = Tester
Man. = Manager

SPI % = The participation in the SPI efforts. (Percentage value count from the documented process improvement sessions in the target company *n=33*)

The target company has an ongoing SPI project. The company started its SPI efforts from a situation where the processes were at initial state and the situation was quite chaotic. Like many small companies, also this company has a lack of SPI resources and skills. The improvement project has been so far realized using two outside SPI researchers, from whom another is the main author of this article, and the target company's personnel. Selected processes have been modeled and streamlined using a light-weight modeling technique, that does not require special SPI personnel or other special resources from the target company. The techinique is described in [13]. After 18 months of improvement work the situation has improved quite a bit. A lot of this seems to be thanks to the enthusiasm and involvement of the target company's employees.

The second phase of improvement project is about to start. At this point part of the company's processes are modeled and streamlined to the state from which it is possible to start aiming at mature processes. The motivation research described here was conducted to assure the next phases success. To measure the motivation, there was a series of interviews made in the company considering the

Table 2. Definition of SPI Motivators (Baddoo and Hall, 2002)

	Motivator	*Definition*
1	Automation	Tools to eliminate paper work
2	Autonomy	Enables practitioners to perform different roles
3	Bottom-up initiatives	Low and middle management have input into the design and planning of SPI
4	Career prospects	Improves career prospects
5	Communication	Improved communication about SPI
6	Compulsory	All SPI practices are made mandatory
7	Cost beneficial	Improved cost/revenue ratio through SPI
8	Critical mass	The presence of sufficient staff members who want to see SPI happen
9	Eliminates bureaucracy	Eliminates spending time on bureaucratic processes
10	Empowerment	Empower staff to take decisions on SPI
11	External audits	Provision of some external body to maintain SPI practices
12	Feedback	Feedback from stakeholders
13	Job satisfaction	Practitioners get job satisfaction from producing good quality process
14	Justifiable benefits	The ability to justify the long-term benefits of SPI
15	Knowledgeable team leaders	Having team leaders who know about SPI
16	Maintainable/easy processes	Processes that are easy to understand, follow and maintain
17	Meeting targets	SPI practice help company to meet company's goals
18	Phased introduction	SPI is introduced through small and incremental implementation
19	Process ownership	Stakeholders own and therefore are able to change processes
20	Resources	Sufficient time and resources allocated to SPI
21	Reward schemes	Stakeholders are rewarded for SPI work
22	Saleability	The perception that SPI will lead to more saleable job market skills
23	Shared best practices	Best practice is shared in companies
24	SPI forum	Creating a forum where SPI ideas can be discussed
25	Standardisation	SPI makes practitioners work in a standardised way
26	Taller hierarchy	Taller company hierarchies for more opportunity and promotion
27	Task forces	Using task forces for SPI
28	Top-down commitment	Senior management support for SPI
29	Training	Training provided to practitioners in SPI practices
30	Reduced admin	SPI leads to reduced administration
31	Visible success	Evidence of the benefits of SPI

motivation towards improvement efforts. Additionally, the interviewees assessed the factors influencing their motivation using a survey based on [4].

3 Research Strategy

In order to be able to carry out a research of this type it requires firstly, a confidential relationship between the researchers, employees and company's management and secondly a company that carries out SPI efforts. These requirements also present a limitation for sample size. For these reasons, our sample in this case is only one company.

However, we have quite an extensive sample collected within one company, taking into account that we collected data from 13 out of 17 employees. Those who were left out were doing either remote work or were subcontractors working on sales and marketing in different geographical locations. In addition to the small sample size, other limitations of this research are the same ones as described in [4,7].

3.1 Data Collection Method

Data collection consisted of two basic instruments: individual interviews and motivation survey. The confidentiality and anonymity of the collected data was explained to the employees. The interviews were semi-structured interviews, carried out in the form of discussion. Individual interviews were chosen in order to avoid influence of the others on the interviewee's answers and to explore the personal dimension of motivation. The interview questions and format were constructed by two researchers. The motivation survey is based on software practitioner motivators researched earlier by Baddoo et al. [4], used also by Niazi et al. in [7]. Survey consists of 31 items and uses a 5-point response scale. The survey examines what motivators the employees evaluate motivating but also whether they may consider some of those de-motivating. The interviews and survey were examined for consistency of information across the two instruments. Comments from interviews were compared to the motivators pointed out in the survey. The interviews and surveys were combined with common identity number in order to be able to remain traceability between each interview and survey.

We interviewed individually the company's CEO and twelve employees, who are directly or indirectly involved in software development process, see Table 1. Two researchers were present in each interview. One researcher asked the interview questions and guided through the discussions. Interviewing researcher has not participated in the company's SPI project. Another researcher, who was making notes, has been involved with the SPI project of the company. Interviews varied in length from 20 to 60 minutes. Interviews were recorded and transcribed afterwards. Interview questions were as follows:

1. How did you choose a career in software engineering?
2. Did you have prior knowledge of SPI before the ongoing process improvement project? (e.g. SPI models and standards)
3. How were the processes defined in your company prior to SPI project?
4. How were new tools, methods, and ways of work introduced and implemented earlier?
   - Were there defined common ways of work?
   - Were your roles and responsibilities defined?
5. What motivated you to take part in process improvement efforts?
6. What results and impacts has the process improvement work had?
7. How were you informed about the upcoming SPI project?
8. Are the resources available for the improvement work?
9. What affect the outside process improvement consultants have had?
10. What has motivated you keeping up the improvement work?

The interviewed employees were given the motivator survey to fill in after the interviews in order to avoid preconceived conceptions of motivators. The employees were explained the meaning of each SPI motivator. The employees were advised to consider the motivators from their personal perspective towards SPI. Motivators with explanation are listed in Table 2. Each employee was asked to evaluate the motivators with a scale from one to five. The scale was as follows: 1 = strong negative impact on motivation, 2 = negative impact on motivation, 3 = no impact on motivation, 4 = positive impact on motivation, 5 = strong positive impact on motivation. Employees were also asked to prioritize three most important motivators respectively.

3.2 Data Analysis

Each interview and the related motivation survey were studied as one entity and they were combined with an identity number. The interview answers were coded and traceability from the smallest instances to the original answers was maintained. Answers were collected and grouped by each interview question. From grouped answers, all company and product specific references were removed in order to preserve the anonymity of the company. From grouped answers, keywords and sentences were marked and listed into tables in respect of interview question. From provided lists, the common, unique, and contradictory motivators were identified and compared with results from motivation survey. The motivation survey data was combined in tables, describing with scale from one to five the motivators across the employees, including CEO's motivators. CEO's motivators were marked with a star in Table 3, in order to be able to perceive his motivators, presenting the managerial point of view.

4 Findings

As it can be seen from Table 3, the employees of the company are motivated to SPI. They see most of the motivational factors presented in Baddoo et al's [4] work as motivating. The 4's and 5's dominate the table, meaning that the employees see the factor mentioned having a positive or very positive impact on motivation.

The factors having the strongest positive influence on motivation were autonomy, bottom-up initiatives, communication, critical mass, job satisfaction, justifiable benefits, process ownership, shared best practices, standardization, top-down commitment and training. Most of these motivating factors also occured during the interviews. For example, majority of the interviewees stressed the importance of communication saying that it is essential to disseminate the improvement project's results and plans to the whole company. There were only three factors that were considered having a negative impact on motivation. One of the motivators having a negative impact was interestingly cost beneficiality. Other de-motivating factors were reward schemes and reduced admin. In the next sections the motivational factors of the target company's CEO and the

Table 3. Results of the motivation query

		Occurrence in answers *(n=12)*												
		Frequency[3]					%[4]							
	Motivator[2]	1	2	3	4	5	1	2	3	4	5	*P1*[5]	*P2*[5]	*P3*[5]
1	Automation	0	1	2	6	3*	0	8	17	50	25	0	0	0
2	Autonomy	0	0	0	9	3*	0	0	0	75	25	3	0	0
3	Bottom-up initiatives	0	0	0	6*	6	0	0	0	50	50	0	1	1
4	Career prospects	0	0	4	6*	2	0	0	33	50	17	0	2	0
5	Communication	0	0	0	7*	5	0	0	0	58	42	0	1	1
6	Compulsory	0	1*	5	6	0	0	8	42	50	0	0	0	0
7	Cost beneficial	0	2	3	5	2*	0	17	25	42	17	0	1*	0
8	Critical mass	0	0	0	7	5*	0	0	0	58	42	0	1	0
9	Eliminates bureaucracy	0	0	5	5*	2	0	0	42	42	17	0	0	0
10	Empowerment	0	0	1*	6	5	0	0	8	50	42	2	0	0
11	External audits	0	1	3	4*	4	0	8	25	33	33	0	0	2
12	Feedback	0	0	1	9*	2	0	0	8	75	17	0	0	0
13	Job satisfaction	0	0	0	7	5*	0	0	0	58	42	3	2	0
14	Justifiable benefits	0	0	0	10*	2	0	0	0	83	17	0	0	0
15	Knowledgeable team leaders	0	0	3	7*	2	0	0	25	58	17	0	0	0
16	Maintainable/easy processes	0	0	2*	6	4	0	0	17	50	33	1	0	1
17	Meeting targets	0	0	3	6	3*	0	0	25	50	25	1*	0	1
18	Phased introduction	0	0	3	8*	1	0	0	25	67	08	0	0	0
19	Process ownership	0	0	0	7*	5	0	0	0	58	42	1	0	0
20	Resources	0	0	1	5*	6	0	0	8	42	50	0	1	0
21	Reward schemes	0	0	7*	3	2	0	0	58	25	17	0	0	1
22	Saleability	0	0	4	3*	5	0	0	33	25	42	0	0	0
23	Shared best practices	0	0	0	6	6*	0	0	0	50	50	0	0	0
24	SPI forum	0	0	1	8*	3	0	0	8	67	25	0	0	0
25	Standardisation	0	0	0	8	4*	0	0	0	67	33	1	2	1
26	Taller hierarchy	0	0	3	7*	2	0	0	25	58	17	0	0	0
27	Task forces	0	0	3	8	1*	0	0	25	67	8	0	0	0
28	Top-down commitment	0	0	0	5	7*	0	0	0	42	58	0	1	1
29	Training	0	0	0	7	5*	0	0	0	58	42	0	0	1
30	Reduced admin	0	0	5	4	3*	0	0	42	33	25	0	0	0
31	Visible success	0	0	1	7	4*	0	0	8	58	33	0	0	2*

Table 4. The three most motivating factors

	1	2	3
CEO	Meeting targets	Cost beneficial	Visible Success
Employees	Job satisfaction	Autonomy	Standardization

employees are compared and analyzed. The analysis presented is based on both, the interviews and the motivation survey.

4.1 Motivation of the CEO

In the interviews, the CEO expressed cost beneficiality and autonomy as motivators in several answers. While describing his earlier job assignments and career

[2] The motivational factors researched by Baddoo et al.
[3] The number of the occurrences in total.
[4] Percentage values of occurrences.
[5] P1, P2, P3=The three most important motivators.

in the interview, it became obvious that these motivators have been strong from the early stages of his career. Furthermore, he mentioned shared best practices and standardization as motivators, which he also sees as main goals of the SPI project. Although maintainable/easy processes was pointed out by the CEO to be one of the most important goals too, in the survey he had marked that to be irrelevant. He considered external audits useful and motivating saying that the external auditing compels to carry out improvements. While seeing the external compulsion as motivating trigger, he sees internal compulsion having a negative impact. He evaluates using imposing or compulsion for employees and himself being a de-motivating factor, which contradicts with the previous statement. He sees top-down commitment as one of the most important motivators and insists on being committed to SPI. However, the top-down commitment is not visible for the employees. He emphasized also training, knowledgeable team leaders, SPI forum, and visible success as motivators. In the interviews, he pointed out the defined roles and easy processes as motivating factors, which allow him to reduce administration and to redirect some of his tasks to other personnel. From the CEO's point of view the possibilities to work in a cost effective and reduced administrative way were important. In the CEO's interview we presented additional question, what he thinks motivates his employees to SPI. He predicted that job satisfaction, visible success, autonomy and communication would motivate the employees.

In the survey he listed following as the most important motivators: automation, autonomy, cost beneficiality, critical mass, job satisfaction, meeting targets, shared best practices, standardization, task forces, top-down commitment, training, reduced admin and visible success. As the three most important motivators he listed the following respectively: meeting targets, cost beneficiality and visible success, see Table 4. The results of the CEO's motivation survey are marked with * in Table 3.

Furthermore, he added five other motivators he evaluated highly motivating: improved quality, improved scheduling, improved resource usage, and impression of professionalism for customers.

4.2 Motivation of the Employees

None of the target company's employees had previous experience in SPI. The initiative to the SPI project came from the company's CEO and the participation in the project was compulsory. While discussing in the interviews about the communication issues related to SPI project in the interviews, it became obvious that the employees who had not participated in the SPI work so far had little knowledge on how the improvement work was going, and what were the real goals. However, neither the compulsory nor the lack of previous knowledge had not influenced the motivation of the employees negatively. Quite the contrary, compulsion was seen as a fairly motivating factor by 6 of the employees. Considering the communication problems, the situation was different. In the motivation query communication was seen as an important motivation factor by all of the

employees and many of them stated in the interviews that it was frustrating not to have information about the progress of the SPI project.

Despite the communication problems, compulsion and lack of experience and knowledge, as a whole, the employees of the target company are very motivated to SPI. In the motivation survey there were 13 out of 31 factors that all of the employees regarded as motivating or strongly motivating. Summary of the answers can be seen in Table 3. When asking them to name the three most important motivational factors respectively, job satisfaction, autonomy and standardization rose above others, see Table 4.

In the interviews, in addition to communication, most of the employees expressed that they saw top-down commitment and standardization as very important motivators. Interestingly, only standardization was seen as the most important factor in the motivation survey.

Even though the CEO of the company insists on being committed to SPI some of the employees have a different opinion. Especially the employees in the marketing department and part of the software engineers experience that there are no resources available for them to improve their processes. This conception is a result and a problem caused by the poor communication and resource allocation, which is not justified to the employees. The resources are allocated without further explanations. The SPI project has not yet reached the marketing department but improving their processes is discussed and included in the future plans. This problem describes the motivation of the target company's employees well. They see the fact that their own work processes have not yet been improved, acknowledge the unsatisfactory state of processes, and look forward on improvement actions influencing those processes.

Comparing the motivators the CEO predicted to motivate the employees, job satisfaction, visible success, autonomy and communication, it appears that he knows his employees quite well. All the motivational factors the CEO predicted were high on the motivator list of the employees.

5 Discussion

With a small company's limited resources, motivation of the employees is essential to make process improvement happen. In the research presented here, the results of the motivation survey and the interviews proved the employees to be highly motivated in process improvement efforts. Furthermore, the employees who have not yet had the chance to participate actively in improvements, showed also great interest and motivation to be involved in the SPI project.

Perhaps not surprisingly, the most important motivational factors between the employees and the CEO differed quite a lot. The CEO named productivity related issues to have the most positive impact on his motivation. Meanwhile, among the employees the comfortability issues were seen as the most significant. The factor "cost beneficial" distributes opinions strongly. Part of the employees evaluate that it has a negative impact on motivation, however there are others who consider it having positive or very positive impact. Based on the

interviews, cost beneficiality might be seen as intrusive managerial procedure. Reward schemes, that have been referred to in [4,7] as a motivating factor, was not considered having an impact on motivation in the research presented here. The same employees who had valued reward schemes strongly motivating in the survey expressed this also in the interviews.

When comparing the results of the research presented here with the results of other motivation surveys, these results support the earlier ones. In Baddoo et al's survey [4] among the most important motivators, excluding the management, were visible success, bottom-up initiatives, resources, training, empowerment, and process ownership. In Niazi and Babar's research [7] six highly motivating factors were cost beneficiality, job satisfaction, knowledgeable team leaders, maintainable/easy processes, shared best practices and top-down commitment. In the research presented here the most important motivators among the employees were autonomy, bottom-up initiatives, communication, critical mass, job satisfaction, justifiable benefits, process ownership, shared best practices, standardization, top-down commitment and training. In addition to these, the CEO named cost beneficiality and autonomy as important factors.

Critical SPI success factors are higher management support, training, awareness, allocation of resources, staff involvement, experienced staff and defined SPI implementation methodology [15]. All these were also listed as motivators in the motivation survey and the answers presented that these factors have a positive impact on the target company's employees. In the light of these results, the factors motivating the target company should facilitate the success of the SPI efforts.

Traditionally the management/top-down commitment has been seen as an vital success factor for SPI [15,14,10] and the results from this research strengthen this conception. However, what became evident, especially from the interview answers, was the fact that it is not enough if the manager insists on being committed. The actual practical actions has to be visible for employees. By taking into account bottom-up initiatives, empowerment, allocating resources fairly and communicating the SPI related issues to the employees, the motivation of the management reflects to the employees and increases their motivation. This can also be done through compulsion, which was not considered de-motivating by any of the employees. On the contrary, all of the employees regarded this as motivating. When SPI is made compulsory by the management, it can be perceived as a sign of SPI being a part of the company's essential business processes and there by, beneficial for everyone. In the interviews some of the employees were clearly pointing out the need for following up SPI, seeing this motivating, because this provides visible success and justifiable benefits, which were pointed out motivating by majority of the employees.

6 Conclusion

In the research presented here, the motivation of a small company's employees for SPI was researched and analyzed. According to the motivation survey, the most

important factors having strong positive impact on motivation were top-down commitment, shared best practices, resources and bottom-up initiatives. The most important motivators having a positive impact were autonomy, feedback and justifiable benefits. The factors the employees themselves chose to be the most important, were job satisfaction, standardization, and autonomy.

Comparing the results to [7] of small company practitioner motivators, the important ones have been cost beneficiality, job satisfaction, knowledgeable team leaders, and maintainable/easy processes. Our results show that job satisfaction is important, however, more emphasis is on top-down commitment, bottom-up initiatives, resources, and shared best practices. Comparing both of these to the motivators in large companies [7,4], career prospects, eliminating bureaucracy, and reward schemes were not seen as important as in large companies. In large companies the focus is on the business related motivators and in small ones the motivators related to comfortability of work are emphasized.

During the research presented, motivation survey and the interviews have proven to be useful tools in planning the future SPI strategy. A lot of valuable information was discovered for planning and implementing the next steps of SPI. When beginning SPI or planning future SPI cycles in the company, it could be useful to carry out a motivation survey before SPI actions or latest, in the early stages of the SPI project. This provides the SPI planners with a conception of how to involve and interact with the employees or management, for instance how often SPI results should be reported and to whom, who should be involved at what stage of the project, justification scheme etc.

Despite the fact that the results presented in this article are from one small company, the researchers suggest that the main results can be generalized when keeping some limitations in mind. The most significant motivational factors presented here represent opinions of one small company, but seem to support previous research. In addition to small sample of the research, other limitations are the language and human issues. The interviews and the survey were conducted in Finnish. Original motivational factors were translated from English to Finnish. In addition to language issues, when interviewing people there is always the risk that people offer answers that they expect the interviewer wants, and also, they might understand questions or words differently regardless of explanations. To address these limitations in future work, it would be valuable to increase the sample size and expand research in different cultural contexts in small companies. However, this requires a good groundwork.

References

1. Richardson, I., von Wangenheim, C.G.: Guest Editors' Introduction: Why are Small Software Organizations Different? IEEE Software 24, 18–22 (2007)
2. Horvat, R.B., Rozman, I., Gyrks, J.: Managing the complexity of SPI in small companies. Software Process: Improvement and Practice 5, 45–54 (2000)
3. Kaltio, T., Kinnula, A.: Deploying the defined SW process. Software Process: Improvement and Practice 5, 65–83 (2000)

4. Baddoo, N., Hall, T.: Motivators of Software Process Improvement: an Analysis of Practitioners' Views. Journal of Systems and Software 62, 85–96 (2002)
5. Baddoo, N., Hall, T.: De-motivators for Software Process Improvement: an Analysis of Practitioners' Views. Journal of Systems and Software 66, 23–33 (2003)
6. Baddoo, N., Hall, T.: Software Process Improvement Motivators: An Analysis using Multidimensional Scaling. Empirical Software Engineering 7, 93–114 (2004)
7. Niazi, M., Ali Babar, M.: Motivators of Software Process Improvement: An Analysis of Vietnamese Practitioners' Views. In: Proceedings of EASE 11th International Conference on Evaluation and Assessment in Software Engineering (2007)
8. Demirörs, O., Demirörs, E.: Software Process Improvement in a Small Organization: Difficulties and Suggestions Software Process Technology. In: Proceedings of the 6th European Workshop on Software Process Technology EWSPT. Springer, Heidelberg (2006)
9. Curtis, B., Hefley, W.E., Miller, S.A.: The People Capability Maturity Model: Guidelines for Improving the Workforce. Addison-Wesley, Reading (2002)
10. Zahran, S.: Software process improvement: practical guidelines for business success. Addison-Wesley Longman Ltd., Essex (1998)
11. Abrahamsson, P.: Is Management Commitment a Necessity After All in Software Process Improvement. In: Proc. 26th Euromicro. Conf., vol. 2, pp. 246–253 (2000)
12. Yin, R.K.: Case Study Research: Design and Methods. Sage Publications Inc., Thousand Oaks (2003)
13. Ahonen, J.J., Forsell, M., Taskinen, S.K.: A modest but practical software process modeling technique for software process improvement. Software Process Improvement and Practice 7, 33–44 (2002)
14. Niazi, M., Wilson, D., Zowghi, D.: Critical Success Factors for Software Process Improvement Implementation: An Empirical Study. Software Process Improvement and Practice 11, 193–211 (2006)
15. Dybå, T.: Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context. In: Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, pp. 148–157. ACM Press, New York (2003)

A Knowledge Management Approach to Support Software Process Improvement Implementation Initiatives

Mariano Angel Montoni, Cristina Cerdeiral, David Zanetti, and Ana Regina Cavalcanti da Rocha

COPPE/UFRJ – Universidade Federal do Rio de Janeiro
Caixa Postal 68511 – CEP 21945-970 – Rio de Janeiro – RJ – Brasil
{mmontoni,cerdeiral,zanetti,darocha}@cos.ufrj.br

Abstract. The success of software process improvement (SPI) implementation initiatives depends fundamentally of the strategies adopted to support the execution of such initiatives. Therefore, it is essential to define adequate SPI implementation strategies aiming to facilitate the achievement of organizational business goals and to increase the benefits of process improvements. The objective of this work is to present an approach to support the execution of SPI implementation initiatives. We also describe a methodology applied to capture knowledge related to critical success factors that influence SPI initiatives. This knowledge was used to define effective SPI strategies aiming to increase the success of SPI initiatives coordinated by a specific SPI consultancy organization. This work also presents the functionalities of a set of tools integrated in a process-centered knowledge management environment, named CORE-KM, customized to support the presented approach.

Keywords: Software process improvement implementation, process-centered knowledge management environment, SPI implementation strategy.

1 Introduction

The increase of organizations competitive advantages is crucial to guarantee market survival. In order to increase organizations capability to develop software, several approaches can be adopted, being Software Process Improvement (SPI) implementation the most recognized one.

Software development is a complex activity and software processes depend strongly in human commitment for its implementation [1]. Individual and organizational behavioral aspects also have great influence in the success of SPI initiatives [2, 3]. Therefore, an important issue to be considered in the execution of SPI initiatives is that those initiatives are conducted by people in a highly collaborative process. Motivated and satisfied teams are likely to implement process improvements more efficiently and the benefits achieved from the SPI implementations are rapidly observed. Nevertheless, when the opposite occurs, resistance to changes can be a critical barrier to effective SPI implementation [3].

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 164–175, 2008.

SPI practitioners must have a great amount of knowledge about software engineering and they must also be capable to use these knowledge to guide SPI implementation in software organizations aiming to achieve the expected results [4]. Moreover, mostly process improvement actions require a significant amount of financial resources and some benefits achieved with implemented improvements are not easily measurable, increasing the difficulty to observe return of SPI investments [5].

According to Zaharan [6], the lack of adequacy of approaches to support SPI implementation is one of the most common reasons to SPI initiatives failure. The majority of SPI approaches only supports the identification of "what" activities must be implemented, and not the "how" to implement those activities [7, 8].

Considering that SPI consultancy organizations have the SPI implementation as their core business, and that the success of SPI initiatives coordinated by those consultancy organizations implies directly in the success of the organizations that acquire their services, it is fundamental to provide adequate mechanisms to support the management of SPI implementation initiatives by SPI consultancy organizations.

This work presents a knowledge management approach to support SPI implementation initiatives conducted by SPI consultancy organizations. As part of the approach, a set of tools were developed and integrated in a process-centered knowledge management environment aiming: (i) to support SPI consultancy organizations to establish SPI strategies; (ii) to support the execution and follow-up of such strategies; and (iii) to support the management of knowledge necessary to conduct SPI initiatives effectively and efficiently. In order to facilitate the definition of SPI strategies, we developed and applied a methodology to identify critical success factors that influence SPI initiatives. Both the methodology and the results achieved by its application are also presented in this work.

In the next section, we discuss some issues related to management of SPI initiatives. Section 3 presents the methodology developed to support the identification of critical success factors that influence SPI initiatives. Section 4 presents the approach to support SPI initiatives conducted by SPI consultancy organizations, and also the tools developed to support the approach. Section 5 presents the conclusions and points out future directions.

2 Management of SPI Implementation Initiatives

An important aspect to be considered in the management of SPI implementation initiatives is the relevant issues that have influence on SPI success. Most of the time, these issues are penetrated deeply into the organization so that it is difficult to recognize its existence and importance [6]. For instance, SPI implementation involves the introduction of innovative practices in the organizations that require knowledge about new technologies. But it is not always easy to manage barriers for a specific SPI implementation, such as lack of technical knowledge required to implement process improvements.

Issues that have influence on SPI implementation initiatives are object of research studies in the last decades. These issues are commonly known as Critical Success Factors (CSF). Although there is considerable number of studies focusing this theme, there is neither consensus in the area about what are those factors, nor understanding about how they interact or influence SPI implementation initiatives.

According to Niazi *et al.* [7], most of the software organizations do not treat their SPI initiatives as real projects. Therefore, it is difficult to apply methods and techniques well established in the project management area to support SPI initiatives execution. Therefore, it is important to apply efficient mechanisms to monitor and control SPI actions and to manage critical success factors that may affect a specific SPI initiative, for instance, resistance to processes changes. Another important aspect to be considered in SPI implementation initiatives is that most of the activities related to SPI involve transference of large amounts of knowledge (for instance, knowledge about patterns, methods, techniques, supporting tools and about the software process implemented, and also knowledge about the software organization and its projects).

Baddoo and Hall [9] suggest that the improvement of knowledge about the relationship among motivators to SPI for different software engineering groups can increase the efficiency of SPI implementation. El-Emam et al. [10] also point that the evaluation of the effects of critical success factors interaction that influence SPI initiatives amplify substantially the organizational pre-requirements for the success or failure of those initiatives and, in this way, increases the viability to define more generic models about critical success factors that may affect SPI. El-Emam et al. [10] also indicates that cultural differences affect directly the success of SPI.

Niazi et al. [7] suggest that the actual problem in SPI is not the lack of standards or models, but the lack of an effective strategy to successfully implement such standards and models. Some approaches were developed to support SPI implementation management. Niazi et al. [7] developed a SPI implementation framework, named SPI-IF, from empirical studies about critical success factors (CSF) that influence SPI. This approach aimed to identify "what" is critical in SPI implementation and "how" to implement the improvements in the organizations. Wilson et al. [11] developed a framework to evaluate the success of SPI initiatives and validated this approach with group interviews in seven organizations of the UK. Dybå [12] developed an instrument to measure critical success factors in SPI based in data collected from 120 software development organizations.

3 A Methodology for Identifying Critical Success Factors That Influence Software Process Improvement Initiatives

In order to cope with the lack of knowledge about factors that influence SPI initiatives, we planned and executed an empirical study to develop a knowledge-body of factors that influence SPI initiatives in the context of Brazilian software industry. The complete methodology developed to support the study is presented in [13]. Next, we briefly describe the relevant characteristics and results of such study that were used as a base for elaborating the approach to support SPI implementation initiatives that is the main focus of this work.

We applied in the study a multi-strategy approach that combined review of empirical studies in the SPI field, a survey and application of qualitative and quantitative analysis techniques. In the study, we applied the Grounded Theory (GT) method [14]. The first step for applying this method was to collect data by applying questionnaires

aiming to identify factors that have influence in SPI implementation. The questionnaires were applied to a selected group of SPI practitioners that participated on SPI initiatives based on recognized models and standards, such as CMMI [15]. In total we analyzed 25 questionnaires of SPI practitioners. The next step in the study was to associate codes to declarations on each questionnaire representing types of findings of critical success factors that influence SPI initiatives. The coded data (findings) were then analyzed and categorized. These categories were denominated critical success factors properties. The association between the findings and a critical success factor property was classified as a finding representing the presence or absence of a critical success factor in a specific SPI implementation initiative. As a result of this analysis we elaborated a theoretical framework composed of knowledge related to critical success factors, its properties and relationships.

In order to derive and aggregate the main critical success factors, we applied two widely recognized statistical analysis techniques: Multi-Dimensional Scaling (MDS) [16] and Principal Components Analysis (PCA) [17]. The MDS and PCA techniques were used in conjunction to identify the critical success factors with statistical significant relationship. More specifically, the PCA technique provided us a systematic way for identifying a reduced set of CSF components relative to the original set of variables. 5 major critical success factors components with statistical reliability were identified.

The first factor was labeled "Environment" since all variables measure the organizational environment capability to establish and maintain SPI initiatives. These variables measure if there are favorable conditions for initiating and sustaining an SPI initiative with two points of view: the individual and the organization. The individual measures are related to members' satisfaction and relationship among members and the SPI team. The organization measures are related to conciliation of strategic goals and SPI interests and to organization internal stability.

The second factor is labeled "Strategy" and indicates that an efficient SPI strategy is concerned on guaranteeing that organization members are aware of the potential benefits that can be achieved by implementing SPI.

The third factor component is "Institutionalization" since the variables of this factor measure the institutionalization of SPI implementation initiatives across the organization by characterizing the degree of resistance to processes institutionalization and to organizational changes, for instance, people turnover, and to inherent difficulties of implementing SPI in different organizational levels.

Since all variables of the fourth factor are considered indicators of commitment to SPI, we labeled this factor as "Commitment". A higher management committed to SPI provides adequate financial resources since the conception of an SPI program and throughout the SPI projects. Moreover, a committed senior management guarantees that organization members have adequate competences and available time to efficiently execute process changes.

The fifth factor is termed "Motivation and acceptance" and indicates that the SPI team is a facilitator of organization members' acceptance to institutionalization of process changes promoted by SPI initiatives.

4 A Knowledge Management Approach to Support Software Process Improvement Implementation Initiatives

An approach supported by a computational infra-structure was defined to support the management of SPI implementation initiatives by consultancy organizations. This approach is based on a customized process-centered knowledge management environment. Next, we describe some important definitions, the approach architecture, and the functionalities of supporting tools to the presented approach.

4.1 Standard SPI Implementation Process and SPI Implementation Strategy Concepts

Two concepts used in this work need more discussion. The first concept is *standard SPI implementation process*. A SPI consultancy organization may have one or more different processes to guide SPI implementation. Nevertheless, these processes do not differ much when put in practice, i.e., they share common characteristics. Therefore, we can say that SPI consultancy organizations may have one or more standard processes which are composed of explicit phases constituted of subprocesses defined in a high level. These standard processes can be defined in accordance to existence SPI implementation approaches. One example would be a standard process defined based on the phases and subprocesses of the IDEAL model [18].

The second concept is *SPI implementation strategy*. A SPI consultancy organization may have one or more SPI implementation strategies applicable to one or more of the subprocesses that constitute each one of the organizations' standard SPI implementation process. These strategies specialize and adapt a subprocess of a SPI standard process by defining in a lower level the activities to be executed in the context of each subprocess. The defined process for a specific SPI implementation project is constituted of the standard SPI process adapted by the selection and incorporation of specific SPI implementation strategies. A SPI implementation strategy applied to guide a specific subprocess of a standard SPI process defines: (i) the activities executed in the context of the subprocess and its descriptions, like the products required and produced by the activities and the competences requirements that activities executants must satisfy in order to be allocated in the activity; (ii) the organizational contexts which define scenarios for applying the strategies; (iii) the required knowledge to execute the activities defined in the strategy; (iv) the resources necessary to execute the activities defined in the strategy; (v) the risks associated to a specific strategy and that occurred in previous SPI implementation projects; (vi) the mitigation and contingency actions related to risks associated to a specific strategy and the results of the execution of such actions in previous SPI implementation projects; (vii) the time and effort usually required to execute each one of the activities defined in a specific strategy based on data collected from previous SPI implementation projects; and (viii) the communication procedures necessary to guarantee the success of the activities defined in a specific strategy.

As an example of an strategy, we can consider the subprocess "Establish the infra-structure" defined in the "Initiating" phase of the IDEAL approach [18]. Depending in the specific characteristics of the software organization focus of the SPI

implementation, SPI consultancy organizations can chose to provide or indicate different infra-structures. For instance, if the software organization does not have adequate tools to support processes, an SPI consultancy organization may suggest specific tools according to organizational context. One example is a process-centered software engineering environment, named TABA Workstation [19-21], that COPPE/UFRJ provides to software organizations that acquire its SPI services. COPPE/UFRJ is a research and development institute that has been providing SPI consultancy services to Brazilian organizations for over two decades.

4.2 Architecture of the Approach

The approach presented in this work relies on an architecture constituted of different components with well defined objectives and responsibilities aiming to address critical issues related to SPI implementation initiatives. The conceptual understanding of critical success factors, standard SPI implementation process and SPI implementation strategy were essential for designing such architecture.

Figure 1 presents the main components of this architecture and their relationships. The first group of components on the left part of Figure 1 has the objective to manage knowledge related to (i) critical success factors that have influence on SPI implementation initiatives, and (ii) SPI strategies to guide SPI implementation initiatives coordinated by a SPI consultancy organization. The components of this group are the following:

- *Identification of Critical Success Factors:* the objective of this component is to support the acquisition of knowledge related to critical success factors that have influence in SPI implementation initiatives. The component must support knowledge capture from different sources, for instance, technical papers and reports, and empirical studies, and also to support the application of the methodology for identifying critical success factors that influence SPI initiatives presented in section 3 of this work.
- *Definition of SPI implementation Strategies:* the objective of this component is to support the acquisition of knowledge related to SPI implementation strategies in specific contexts, considering organizational characteristics that may have positive or negative impact on SPI. The component must provide the means for different SPI consultancy organizations to preserve their knowledge related to (i) their SPI implementation strategies defined based on software process standards and models, (ii) the context which these strategies are most applicable, and (iii) the directives for adapting the SPI strategies to support the conduction of specific SPI implementation initiatives.

The second group of components on the upper right part of Figure 1 treats the application of benchmarking techniques in SPI implementation projects. The main component of this group is:

- *Benchmarking of SPI Implementation Projects:* the objective of this component is to support the identification of best practices of a SPI consultancy organization and the application of these practices in new SPI

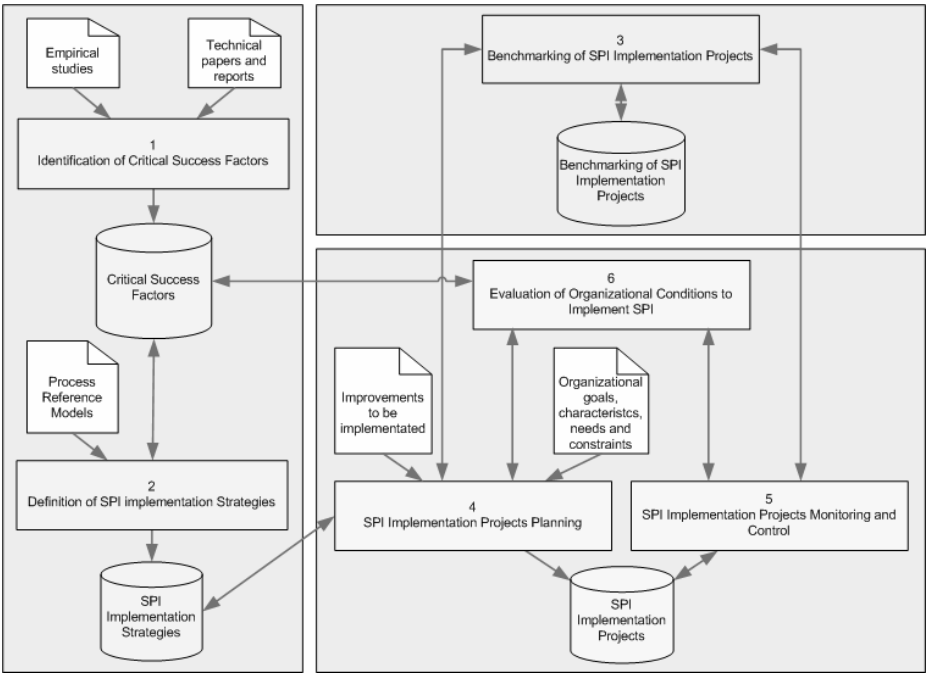


Fig. 1. Components of the approach to support SPI implementation initiatives

initiatives. The component must support the maintenance of knowledge related to the performance of SPI implementation strategies extracted from execution data of SPI implementation projects. This knowledge provides the means to SPI consultancy organizations predict both current and future SPI implementation projects performance.

The third group of components on the lower right part of Figure 1 focuses SPI implementation projects management and assessment. The components of this group are the following:

- *SPI Implementation Projects Planning:* the objective of this component is to support planning of SPI implementation projects aiming to provide the means for selecting and adapting SPI strategies based on organizational characteristics that have positive or negative impact in SPI initiatives.
- *SPI Implementation Projects Monitoring and Control:* the objective of this component is to support SPI activities monitoring by analyses of both qualitative and quantitative data of projects execution. The component must also support the effective control of factors that may significantly influence the SPI implementation projects results.
- *Evaluation of Organizational Conditions to Implement SPI:* the objective of this component is to support the identification of factors that may influence the success of SPI implementation initiatives since the begging of the SPI project and throughout its execution.

4.3 Supporting Tools

A set of tools were developed to support the application of the presented approach. These tools are integrated in a customizable process-centered knowledge management environment, named CORE-KM (Customizable Organizational Resources Environment with Knowledge Management) [22].

A knowledge management environment was customized to the COPPE/UFRJ SPI consultancy organization based on the CORE-KM. This environment is being used to support the application of COPPE/UFRJ's SPI implementation strategy, named SPI-KM, in software development organizations. Although this strategy was used to coordinate several SPI implementation initiatives in different Brazilian software organizations [23, 24], the SPI consultants had to deal with innumerous difficulties due to lack of supporting tools, for instance, difficulty to transfer knowledge about the strategy to new SPI consultants.

The set of tools developed to support the presented approach is the following:

- *Tool to support Management of SPI Implementation Strategies:* the objective of this tool is to support (i) the management of critical success factors information that have influence in SPI initiatives, and (ii) the management of SPI implementation strategy information of a specific SPI consultancy organization.
- *Tool to support Benchmarking of SPI Implementation Projects:* the objective of this tool is to support (i) the definition of a benchmarking base of SPI implementation projects coordinated by a specific SPI consultancy organization, (ii) the management of information of SPI implementation projects coordinated by a specific SPI consultancy organization, and (iii) the access of information of similar SPI implementation projects coordinated by a specific SPI consultancy organization. Figure 2 presents the screen of this tool that consolidates information about different SPI implementation strategies performances.
- *Tool to support Management of SPI Implementation Projects:* the objective of this tool is to support (i) the planning of SPI implementation projects coordinated by a SPI consultancy organization, (ii) the monitoring and control of SPI implementation projects coordinated by a specific SPI consultancy organization, and (iii) the assessment of organization conditions to implement SPI. Figure 3 presents the screen of this tool that supports the identification of critical success factors that influence SPI implementation projects. From this screen, the SPI management can identify the critical success factors present in a specific SPI implementation project and define a risk management plan constituted of mitigation and contingency actions to manage the factors assessed negatively as potential threats to the success of the project.

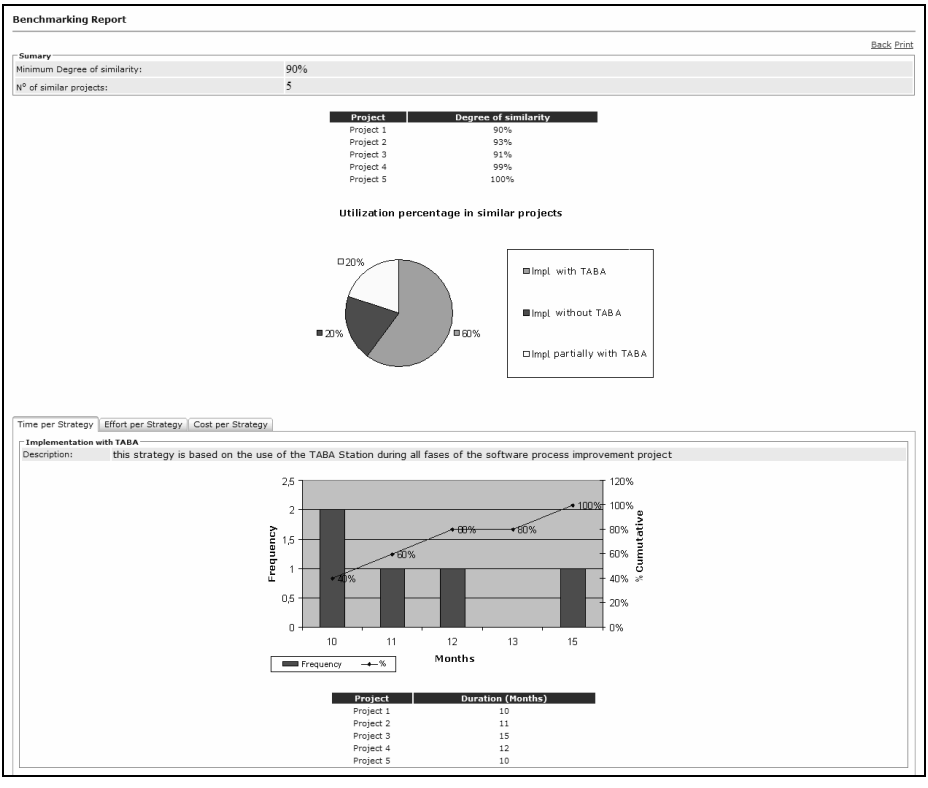


Fig. 2. Screen of the Benchmarking report

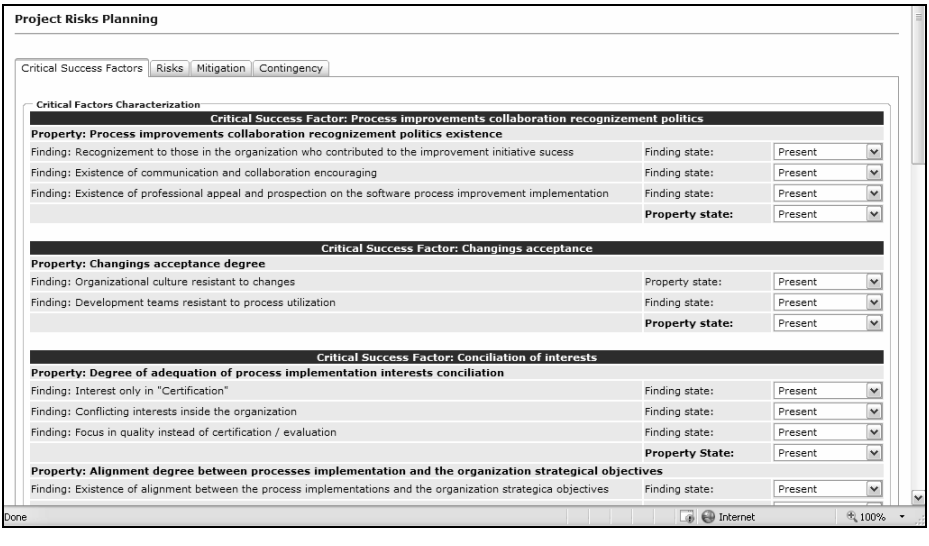


Fig. 3. Screen to support the identification of critical success factors

5 Conclusions

This work presented an approach to support SPI implementation initiatives. We also presented a methodology for identifying critical success factors that influence SPI initiatives. The main components of the approach and the supporting tools were also discussed in this work.

A case study was designed aiming to answer the following research question: *How has the use of the developed approach facilitated the SPI implementation initiatives conducted by a SPI consultancy organization?* In order to answer this research question, we assumed that SPI implementation initiatives are facilitated if some expected benefits are achieved by an SPI consultancy organization while conducting a specific SPI implementation initiative.

Therefore, we stated the following proposition to guide our study effort: The main benefits expected to be achieved by applying the presented approach are: (i) to accumulate knowledge about critical success factors that have influence on SPI implementation initiatives; (ii) to accumulate knowledge about SPI implementation strategies; (iii) to facilitate the selection and adaptation of SPI implementation strategies in specific contexts; (iv) to facilitate the evaluation of organization conditions to implement SPI; (v) to facilitate the management of SPI initiatives based on effective and efficient SPI implementation strategies; (vi) to increase the visibility of SPI initiatives results; (vii) to diminish costs, time and effort to conduct SPI initiatives; (viii) to preserve knowledge related to SPI implementation execution; and (ix) to facilitate the identification of best practices in SPI implementation.

The unit of analysis for this study, i.e., the case itself, is a specific SPI implementation initiative coordinated by the COPPE/UFRJ SPI consultancy organization. The case study will show how the presented approach increases the success of this specific SPI implementation initiative.

The data from which the study derive its conclusions will be collected in the form of defined metrics that provides indications whether or not the expected benefits of the application of the presented approach are being observed in the case of the study. The SPI initiative object of this study has already began and we expect to conclude this study by April 2009.

Statistical analysis will be applied aiming to conclude statistically significant results. These results will be validated by testing the study findings in another SPI implementation initiative planned to start in early 2009. We are also planning to have an external auditor to evaluate the procedures executed in the context of this research aiming to guarantee the study reliability.

References

1. Coleman, G., O'Connor, R.: Software process in practice: A Grounded Theory of the irish software industry. In: Richardson, I., Runeson, P., Messnarz, R. (eds.) EuroSPI 2006. LNCS, vol. 4257, pp. 28–39. Springer, Heidelberg (2006)
2. Baddoo, N., Hall, T.: Motivators of Software Process Improvement: An analysis of practitioners' views. Journal of Systems and Software 62, 85–96 (2002)

3. Baddoo, N., Hall, T.: De-motivators for software process improvement: An analysis of practitioners' views. Journal of Systems and Software 66, 23–33 (2003)
4. Niazi, M., Wilson, D., Zowghi, D.: Critical success factors for software process improvement implementation: An empirical study. Software Process Improvement and Practice 11, 193–211 (2006)
5. Goldenson, D.R., Herbsleb, J.D.: After the Appraisal: A Systematic Survey of Process Improvement, its Benefits and Factors that Influence Success. Software Engineering Institute (1995)
6. Zaharan, S.: Software Process Improvement – Practical Guidelines for Business Success. Addison-Wesley, Reading (1998)
7. Niazi, M., Wilson, D., Zowghi, D.: A framework for assisting the design of effective software process improvement implementation strategies. Journal of Systems and Software 78, 204–222 (2005)
8. Wu, M., Ying, J., Yu, C.: A methodology and its support environment for benchmark-based adaptable software process improvement, vol. 6, pp. 5183–5188. Institute of Electrical and Electronics Engineers Inc., New York (2004)
9. Baddoo, N., Hall, T.: Software process improvement motivators: An analysis using multidimensional scaling. Empirical Software Engineering 7, 93–114 (2002)
10. El-Emam, K., Goldenson, D., McCurley, J., Herbsleb, J.: Modelling the likelihood of software process improvement: An exploratory study. Empirical Software Engineering 6, 207–229 (2001)
11. Wilson, D.N., Hall, T., Baddoo, N.: A framework for evaluation and prediction of software process improvement success. Journal of Systems and Software 59, 135–142 (2001)
12. Dyba, T.: An Instrument for measuring the key factors of success in software process improvement. Empirical Software Engineering 5, 357–390 (2000)
13. Montoni, M., Rocha, A.R.: A Methodology for Identifying Critical Success Factors that Influence Software Process Improvement Initiatives: An Application in the Brazilian Software Industry. In: Abrahamsson, P., Baddoo, N., Margaria, T., Messnarz, R. (eds.) EuroSPI 2007. LNCS, vol. 4764, pp. 175–186. Springer, Heidelberg (2007)
14. Strauss, A., Corbin, J.M.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Thousand Oaks (1998)
15. SEI: CMMI® for Development (CMMI-DEV), V1.2. Software Engineering Institute (2006)
16. StatSoft: STATISTICA Electronic Manual. StatSoft Inc. (2004)
17. Kim, J., Mueller, C.: Factor Analysis: Statistical Methods and Practical Issues. Sage Publications, Thousand Oaks (1978)
18. Gremba, J., Myers, C.: The IDEAL Model: A Practical Guide for Improvement (1997)
19. Montoni, M., Santos, G., Rocha, A.R., Weber, K.C., Araujo, E.E.R.d.: MPS Model and TABA Workstation: Implementing Software Process Improvement Initiatives in Small Settings. In: Santos, G. (ed.) Fifth International Workshop on Software Quality. WoSQ 2007: ICSE Workshops 2007, p. 4 (2007)
20. Montoni, M., Santos, G., Rocha, A.R., Figueiredo, S., Cabrai, R., Barcellos, R., Barreto, A., Scares, A., Cerdeiral, C., Lupo, P.: Taba workstation: Supporting software process deployment based on CMMI and MR-MPS. In: Münch, J., Vierimaa, M. (eds.) PROFES 2006. LNCS, vol. 4034, pp. 249–262. Springer, Heidelberg (2006)
21. Ferreira, A.I.F., Santos, G., Cerqueira, R., Montoni, M., Barreto, A., Rocha, A.R., Figueiredo, S., Barreto, A., Filho, R.C.S., Lupo, P., Cerdeiral, C.: Taba workstation: Supporting software process improvement initiatives based on software standards and maturity

models. In: Richardson, I., Runeson, P., Messnarz, R. (eds.) EuroSPI 2006. LNCS, vol. 4257, pp. 207–218. Springer, Heidelberg (2006)
22. Galotta, C., Zanetti, D., Rocha, A.R., Oliveira, K.M.: Organizational Learning Based on a Customizable Environment for Knowledge Management Using Intranet. In: E-LEARN 2004 – World Conference on e-Learning in Corporate, Government, Healthcare & Higher Education, Washington, EUA, vol. 2, pp. 2626–2633 (2004)
23. Santos, G., Montoni, M., Figueiredo, S., Rocha, A.R.: SPI-KM Lessons Learned from Appling a Software Process Improvement Strategy Supported by Knowledge Management. In: Münch, J., Abrahamsson, P. (eds.) PROFES 2007. LNCS, vol. 4589, pp. 81–95. Springer, Heidelberg (2007)
24. Santos, G., Montoni, M., Vasconcellos, J., Figueiredo, S., Cabral, R., Cerdeiral, C., Katsurayama, A.E., Lupo, P., Zanetti, D., Rocha, A.R.: Implementing Software Process Improvement Initiatives in Small and Medium-Size Enterprises in Brazil. In: 6th QUATIC (International Conference on the Quality of Information and Communications Technology), Lisboa, Portugal (2007)

An Operational Approach for Selecting Open Source Components in a Software Development Project*

Annick Majchrowski and Jean-Christophe Deprez

CETIC, Charleroi, Belgium
{am,jcd}@cetic.be
http://www.cetic.be

Abstract. Many organizations have started to integrate Free/Open Source Software (F/OSS) components in their applications. It is therefore crucial for these companies to select the most appropriate F/OSS components in terms of functional and non-functional needs. Although F/OSS selection methods have appeared in the last few years, they lack an operational description. In turn, this has slowed their use in software development project. This work presents an operational approach for selecting F/OSS components where the client, the development team and their respective quality assurance teams are involved in the selection process. Although the case study applying the F/OSS selection approach is left to future work, this article already describes an industrial case where the approach presented in this paper has been approved for use by the various partners, i.e., the client, the development firm and their respective quality teams.

1 Introduction

Many public and private organizations have started to integrate Free (*libre*) Open-Source Software (F/OSS[1]) in their products and systems. Furthermore, software solutions that rely on F/OSS components become more frequently available to customers or external users. In turn, organizations want assurance regarding the quality of F/OSS projects before integrating them in their solutions.

Based on this need, several methodologies to help select appropriate F/OSS projects were created in the past couple of years. Two prominent methodologies are the Qualification and Selection Open Source (QSOS) backed by Atos Origin and Open Business Readiness Rating (OpenBRR) created by Carnegie Mellon West and Intel [1, 2]. Unfortunately, these two methods are not presented in an operational way in turn they cannot readily be applied in software development project that involve several partners such as the client, the development team and their respective quality assurance teams.

* This work is partly funded by QUALOSS (#33547), a research project funded under the FP6 programme of the European Commission.

[1] F/OSS stands for Free *libre* Open Source Software.

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 176–188, 2008.

The contribution of this paper is to propose an operational approach for selecting F/OSS components to be integrated in a software application. In particular, the roles of the various teams involved are clearly identified and their tasks described.

Furthermore, where QSOS and OpenBRR are lightweight methodologies that do not require in-depth evaluation of F/OSS development endeavors, our approach suggests that a much more thorough evaluation is often needed to respond to customer's demand. We are currently working on creating this in-depth evaluation methodology as part of the QUALOSS project (www.qualoss.eu). Consequently, once this in-depth evaluation method is finalized, our selection approach will propose more objective results and more accurate risk estimation regarding the integration of the F/OSS components considered.

The challenge in creating our operational approach is to remain applicable to all development projects interested in integrating F/OSS components. Furthermore, the approach assigns the appropriate responsibilities to obtain an appropriate balance of power between the client and the development team.

The rest of this paper is organized as follows. Section 2 presents the context where our approach applies. Section 3 describes our F/OSS selection approach in details. Section 4 explains how we plan to apply our approach on an industrial case. We then compare our effort to other related works in Section 5 and present our conclusion and planned future work in Section 6.

2 Context of Methodology

The methodology for selecting F/OSS components presented in this work is applicable to the software development context illustrated in Figure 1.

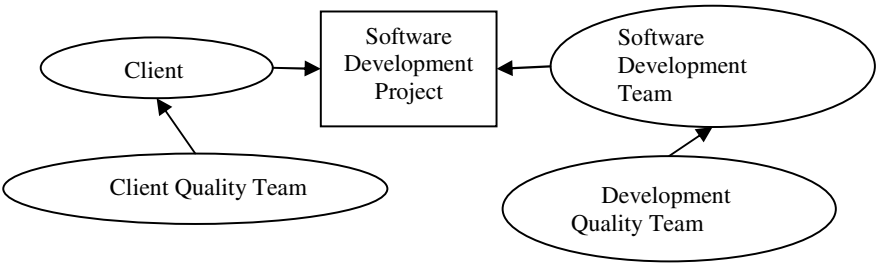


Fig. 1. Software Development Context where our F/OSS Selection approach is applicable

The context of Figure 1 handles the following development scenarios:

Scenario 1: Internal Software Development Project.
In this scenario, the client and the software development team work for the same company and the quality assurance may be handled by a single quality team. In other words, the Client Quality team and the Development Quality team may be merged, i.e., handled by a single team or person. Indeed, in the case of very small projects, the client or the development team may also be responsible for performing the task of

quality assurance. Incidentally, our selection approach is independent of whether the resulting software application will be for use internal to the developing firm only or whether it will be distributed externally (free or for a fee).

Scenario 2: Consulting Software Development Project.
In this scenario, the client and the software development are different entities. In most cases, the consulting firm will not only assign a development team to the project but also a quality team in charge of monitoring the quality from the consulting firm's viewpoint. Depending on the client's expertise and resource availability, the quality assurance tasks may be performed by the client's resources. However, there are also cases where the client is not an IT expert or prefers to contract out the quality assurance task to yet another third party different from the consulting firm in charge of development.

Due to large portions of software project failing and running over budget [3], clients have become increasingly concerned about monitoring the quality of the software product delivered as well as other work products related to the project such as project planning and monitoring documents. Furthermore, clients are interested in monitoring the quality of the code developed and also insist on software reuse. Where reuse was synonymous to the use of component off-the-shelf (COTS) in the past, the advent of Free *libre* Open Source Software (F*l*OSS) is shifting this trend. Many customers now request that consulting firms reuse existing F*l*OSS components. Even in the case of internal development projects many companies start considering integrating F*l*OSS components to avoid reinventing the wheel.

We observe two trends when using F*l*OSS components:

- A client does not wish to become an integrating part of the FlOSS community and thus, does not want to feedback its modifications to the F*l*OSS development endeavor.
- A client wishes to become an integrating part of a F*l*OSS community and desires that its modification be included in the new versions of the F*l*OSS component.

The optimal use of F*l*OSS is achieved when following the second model. However, the F*l*OSS selection approach presented in this paper equally applies to both cases. In both cases, the role of the two quality teams is to ensure that the F*l*OSS component selected respect the quality criteria selected. In other words, both quality teams must work in concert and take the context of the client and of the software development team into account. For example, the client may be in a context where certain partnership imposes certain F*l*OSS component. On the other side, the software development team may have an internal expert on a particular F*l*OSS component in turn favoring that component may lower risk of project failure, at least in terms of cost and time schedule estimation. Our F*l*OSS selection approach includes steps where each party can express its strengths and weaknesses. In turn, the F*l*OSS selection decision become much more transparent and removes most of the subjectivity often hidden in this kind of activities.

3 F*l*OSS Selection Approach

This section describes our approach for selecting F*l*OSS components to later integrate in a custom-made software application. The main activities to conduct are summarized in the workflow shown in Figure 2. Besides stating the responsibilities of the

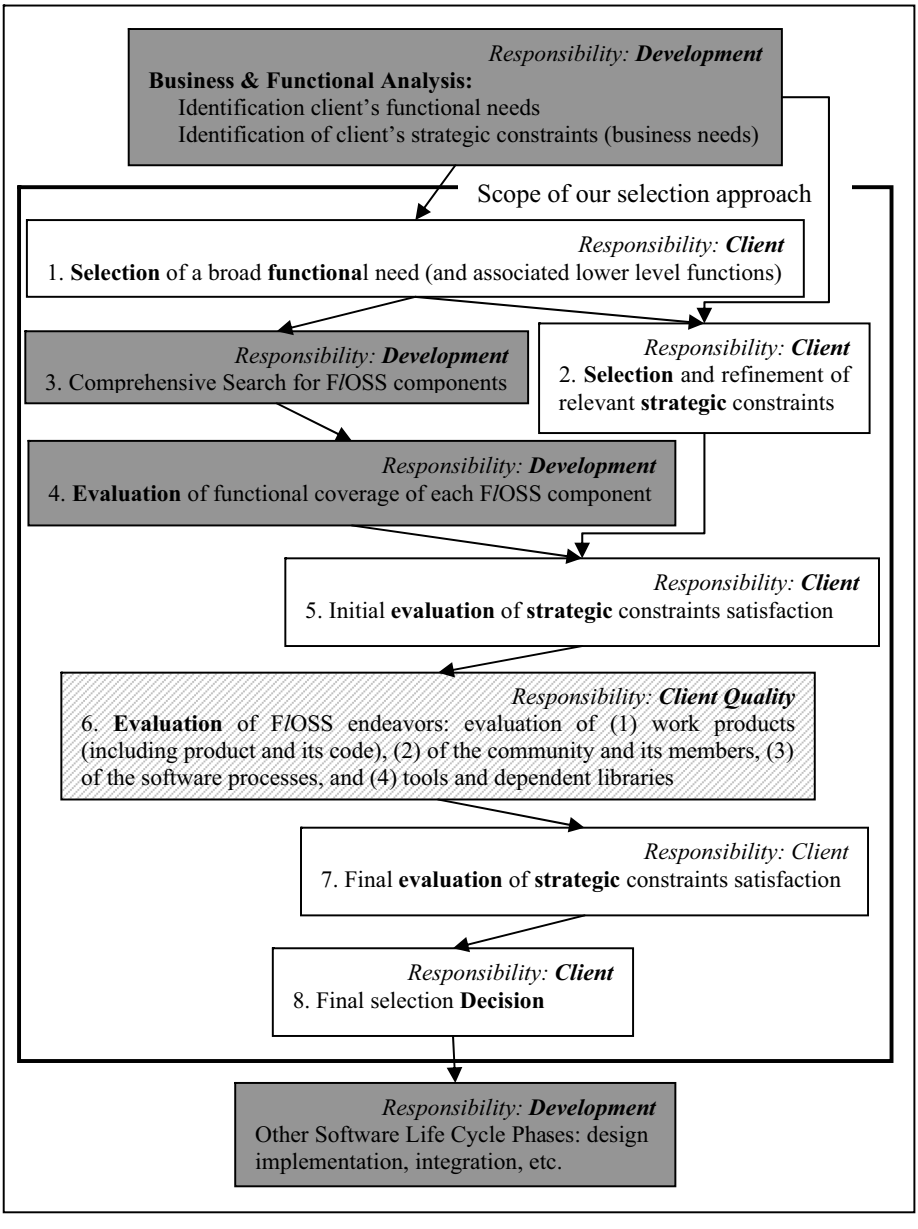


Fig. 2. Workflow used by our approach for Selecting FlOSS components

different parties involved, Figure 2 also emphasizes it using various grey shed boxes. The rationale for the leadership assignment of each activity is explained later.

It is worth noting that the top and bottom unnumbered boxes are not part of our approach. They indicate the point of the life cycle when our approach takes place, in particular, between the analysis and the design phases.

Below, we present a detailed description of the activities within the scope of our approach. In addition to explaining the job performed by each actor taking part in an activity, our description also specifies the usual amount of time needed to perform an activity. This duration information can be used to figure the resource needed to perform the selection process within a specified time frame.

It is understandable that selecting a F/OSS component should not slow down the development process excessively. Nonetheless, it would also be useless to perform light analysis that would barely increase our confidence in the selected F/OSS component. In turn, our approach was created with the idea that the selection process could take place in about 1 month, assuming adequate staffing and prompt availability of each partner.

1) Selection of a broad functional need (and its associated lower level functions)

Short description: The first activity consists of selecting a broad functional need of the application to develop. The broad functional need is a high level description of a function needed in the application such as an ERP task manager module. Associated lower level functions are functions that further details how the high level function is to really behave. For example, registered users must be able to share tasks.

Rationale: It is important to have an initial step to communicate to every party involved the beginning of the FlOSS selection process for a given functional need.

Input of the activity: A list of business functional analysis expressed by the client. This list is usually specified in an analysis document.

Output of the activity:

- A one-paragraph description of the broad functional need that may be fulfilled by a F/OSS component. This short description.
- A check list that enumerates all lower level functions associated to the broad functional description also needed.
- A communication sent by the client to the contact person of each team involved, that is, (1) the client, (2) the client quality team, (3) the consulting development team, and (4) the consulting quality team. This communication informs the four teams that the F/OSS component selection process has been initiated for the specified broad functional need.

Hypothesis: Our premise for initiating this activity assumes that the result of the business and functional analysis is far enough advanced that all broad functional needs have been identified (in the context of the given project). This is required so that our selection process is not impacted by the apparition of new broad functional need that could affect the F/OSS selection context.

Actors and their roles: The client is the leader of this activity. However, that actual bulk of the work is really performed by the consulting development team who will actually select the appropriate need and explain to the client why that need was

selected. For example, if the implementation of need A is a prerequisite to the implementation of need B, it then makes sense to treat B before A. Furthermore, the development team must also create the check list of lower level functions required by the client.

The reason for assigning the leadership to the client is to guarantee that the client and its quality team are aware that the selection process for one broad functional need has been initiated. This leadership is express by requiring that the client sends a communication to the four teams involved.

Estimated time: combined effort of 1 to 2 person-days depending on how easy it is to transform the input functional analysis document into the check list of lower level functions.

2) Selection and refinement of relevant strategic constraints

Short description: This activity consists of selecting the relevant strategic constraints from the analysis document. Most constraints such as the cost of adapting a F*l*OSS component equality applies to the selection exercises of all F*l*OSS components. However, certain strategic constraint may need to be refined to be better adapted to the functionality selected in activity 1. For example, a strategic constraint may require verifying the reputability of a F*l*OSS component, the notion of reputation may actually be adapted in accordance to the functional need. In some cases, it may be sufficient to show a few reference uses of a F*l*OSS component while in other cases, it may be required to show the publication of several books and the use of a F*l*OSS component in business critical applications.

Strategic constraints identified in this step will be evaluated in a later activity. However, this activity may already select the methods or procedure to use later for quantifying the degree to which a strategic constraint is satisfied. For example, it may be necessary to back estimation cost of adapting a F*l*OSS component with a method such as COCOMO or similar but lighter approach. It is then during this step that such methods are also selected (and eventually specified in details).

Besides cost estimation, in the context of selecting F*l*OSS components, strategic constraints usually also include F*l*OSS license issues, F*l*OSS components already used by the client's partners. Furthermore, if the client desires to feedback its contribution to the F*l*OSS communities, she will then be concerned with the permeability of F*l*OSS communities, that is, how open are they to new contributors?

Rationale: It is important that early in the F*l*OSS selection process, the client communicates precisely her strategic constraints to the other teams involved. This will avoid wasting time analyzing F*l*OSS components that do not meet certain strategic constraints.

Input of the activity:

- A list of strategic constraints identified in the business and functional analysis document.
- The functional need selected in the previous step of our approach

Output of the activity:

- A list of specific strategic constraints (to verify in a later activity). This list should be presented in the form of a check list to ease constraints verification.

Furthermore, if the verification of some strategic constraints requires specific methods or procedures, such methods and procedures must also be listed.

- A communication sent by the client to the contact person of each team involved, that is, (1) the client, (2) the client quality team, (3) the consulting development team, and (4) the consulting quality team. This communication informs the four teams that the strategic constraints have been identified for this execution of the F*l*OSS selection process.

Hypothesis: Our premise for initiating this activity is that the result of the business and functional analysis is far enough advanced that all generic strategic constraints have been identified. This is required so that the F*l*OSS selection process is not impacted by the appearance of new business strategic constraints.

Actors and their roles: The client reviews the list of business constraints in the analysis document to identify the strategic business constraints relevant to the particular functional context. This activity may require refining generic strategic constraints into specific ones adapted to the functional context.

Most of the work is performed by the client, the client quality team may need to get involved in order to refine the generic strategic constrains into more specific ones.

Minimal Estimated Effort: combined effort of 1 to 2 person-days.

3) Comprehensive Search for F*l*OSS components

Short description: The activity consists of selecting a fairly exhaustive list of F*l*OSS components that match with the broad functional need selected in activity 1. The usual way performed to search and found such components consists on using the broad functional need as the keyword search in a search engine on websites. Another possible way is to post a question related to the searched free component under the site of some free communities.

Rationale: To avoid missing F*l*OSS components that may be of interest, it is important to perform a fairly exhaustive search.

Input of the activity: The one-paragraph description of the broad functional need.

Output of the activity: a list of F*l*OSS components that implement the broad functional need. We note that the coverage of the lower level functions is not verified at this time.

Hypothesis: Our premise for this activity is that the general architecture specifying the technical needs and environment in which the application being developed is known. It is often already described in the call for tender.

Actors and their roles: Each of the four teams is free to suggest a list of F*l*OSS components related to the broad functional need. However, the F*l*OSS components that do not meet the architectural and technical needs are not even included. There is no reason to include a Java component if the final solution is to be developed in C++ or to include a MS Windows-only component if the final solution must solely run on Unix.

Minimal Estimated Effort: About 2 person-days. Each team is allocated half a person-day for searching for relevant F*l*OSS components.

4) Evaluation of functional coverage of each F/OSS component

Short description: The activity consists of evaluating, that is, assigning scores to each F/OSS components listed as output of activity 3 above. The scores will be obtained by first filling for each F/OSS component, the checklist of the lower level functions provided by activity 1. The process of transforming a filled check list in a score must be decided at the beginning of this activity. Moreover, to insure consistent treatment, the same scoring rules should be applied to all F/OSS components. Efficient scoring procedures usually work in two steps. First, they specify how to assign a raw score to each criteria and second, they allow to weigh each criteria according to its importance, in our context, the weight would represent the importance of each lower level function in the checklist.

Rationale: Functional requirements are universally very important. It is therefore a priority for our F/OSS selection approach to measure functional coverage early in the process.

Input of the activity:

- The list of FlOSS components identified in activity 3
- The check list of the lower level functional needs created in activity 1

Output of the activity: Each F/OSS component is attributed a functional coverage score.

Hypothesis: Our premise for this activity is that all F/OSS components meet the architectural and technological needs of the client. Furthermore, the checklist of lower level functional needs should be complete.

Actors and their roles: The development team fills the check list for all F/OSS components provided as input. It is important that the location of data used to fill the checklist and compute the scores be recorded in order to explain the reason for the proposed F/OSS component ranking to the client and its quality team.

Beside the objective scores obtained using the selected scoring procedure, the client and the development team should be allowed to introduce subjective criteria, for example, the client and the development teams may create a list of strengths and weaknesses for each component. Furthermore, the client may also indicate the F/OSS component that seems to best fit her trade.

Clearly, it is not the objective of this activity to be extremely thorough. In other words, each F/OSS component being scored needn't be installed and tested to evaluate its functional coverage. The mature F/OSS components frequently provide list of features and functionalities. Furthermore, they also propose tutorials and reference manuals. The requirement of not installing F/OSS component to evaluate their functional coverage is specified for the whole activity 4 to take place within a two week time frame. Exceptionally, particular functional needs will not be satisfied by mature F/OSS components and only very few, young F/OSS components will match the need. Young F/OSS endeavors may not always propose functional documentation. In turn, a quick installation of the few F/OSS components may be installed in order to assess their functional coverage. Only a limited number of F/OSS components should be analysis in this way so as to remain under the two weeks timeframe for activity 4.

Although the ultimate goal of this activity is not to eliminate F*l*OSS components from the initial list, this can however happen if functional coverage is below an expected threshold. It is crucial for the four teams of actors to agree on the threshold.

Minimal Estimated Effort: maximum 1 person-day per F*l*OSS component

5) Initial evaluation of strategic constraints satisfaction

Short description: The activity consists of evaluating all FlOSS components in the list regarding its fulfillment of strategic constraints. The scores will be obtained by filling the checklist provided as outcome of activity 2. The logical way to proceed could be to follow the ranking provided by activity 4. The process of transforming a filled checklist in a score must be decided at the beginning of this activity. Moreover, to insure consistent treatment, the same scoring rules should be applied for all F*l*OSS components.

Rationale: In addition to functional coverage, it is important to evaluate whether F*l*OSS components fulfill the strategic constraints of the client and also some of the development team.

Input of the activity:

- List of F*l*OSS components ranked according to their functional coverage

Output of the activity:

- List of F*l*OSS components ranked according to both (1) functional coverage and (2) fulfillment of strategic constraints.
- An expert opinion on which few F*l*OSS components should be analyzed thoroughly as part of activity 6.

Hypothesis: Even F*l*OSS components with lower functional coverage deserve being evaluated as their ranking may improve thanks to better fulfillment of strategic constraints. It is however possible to agree on a threshold of functional coverage under which F*l*OSS components mustn't be analyzed for the strategic constraints.

Actors and their roles: The work in this activity is shared between the development quality team and the client quality team. However, the client is responsible for this activity because she will finally decide the few F*l*OSS components to analyze in more details as part of the next activity.

Minimal Estimated Effort: 1 person-week.

6) Evaluation of F*l*OSS endeavors

Short description: The activity consists of evaluating a few F*l*OSS endeavors thoroughly. This evaluation presents objective, reproducible results. It is worth noting that this activity evaluates F*l*OSS endeavors and not solely F*l*OSS components. A F*l*OSS endeavor is composed of a set of community members, a set of the work products (including the product and its code) produced by the F*l*OSS community, a set of development processes executed by the community members, and a set of software tools used to produce, support or run the F*l*OSS component. Where the evaluations of activities 4 and 5 respectively, focused on the F*l*OSS components, the evaluation in this activity analyzes the F*l*OSS endeavor that produces a F*l*OSS component. In analogy, it is like evaluating the enterprise that produces a software application vs. merely analyzing a software application.

The method to help evaluate F*l*OSS endeavor is currently being developed as part of the QUALOSS project (www.qualoss.eu). In the meantime, a combination of QSOS, OpenBRR and various source code analyses including historical code evolution may be performed.

Rationale: To insure that the client's concerns are well taken into account in the final selection, the client quality team must be in charge of this evaluation. Furthermore, although late in this selection process already, it is still better to backtrack at this moment due the bad quality of the F*l*OSS endeavors under current consideration.

Input of the activity: F*l*OSS endeavors corresponding to the F*l*OSS components selected in activity 5.

Output of the activity:

- The objective results of the evaluation of selected F*l*OSS endeavors.
- The client quality team communicates its results during a meeting with the four teams involved.

Hypothesis: The client and development teams have accepted the ranking and selection performed in activities 4 and 5.

Actors: The client quality team leads and performs most of the work. If required, the development quality team may also intervene and help in this activity. However, it will always be under the control of the client quality team.

Minimal Estimated Effort: 1 person-week per F*l*OSS endeavor

7) Final evaluation of strategic constraints satisfaction

Short description: The activity 6 consists of re-evaluating and measuring some strategic constraints of the project in regards of the results of activity 6. For example, the cost estimation method employed in activity 5 may not have taken into account that a F*l*OSS component was poorly documented and contained little code comments. Likewise, the architecture of the F*l*OSS components may show a high level of coupling and low cohesion in its modules hence impacting further potential maintenance effort. Besides, code assessment, the evaluation of activity 6 may show that the community heavily relies on the contribution of just a very few developers who do not accept input from new contributors.

Rationale: In light of the new findings of activity 6, it is worth reevaluating certain strategic constraints, particularly those related to cost and risk estimates.

Input to the activity: the strategic constraints identified in activity 2 and their current evaluation from activity 5.

Output of the activity: A new, more accurate evaluation of the strategic constraints based on the outcome of activity 6.

Actors: The leadership is given to the client along with the client quality team, they decide on the strategic constraints that must be re-evaluated. It is then the development team and its quality team who re-evaluate the strategic constraints based on the outcome of activity 6.

Minimal Estimated Effort: 1 to 2 days per F*l*OSS endeavor

8) Final selection Decision

Short description: This last activity exploits the results of activities 4 , 6 and 7 to perform the final selection decision of which F*l*OSS component will be integrated in the application.

Input to the activity:

- Results of the evaluation of functional coverage of F*l*OSS components identified in activity 3
- Results of the evaluation of a few F*l*OSS endeavors from activity 6
- Results of the re-evaluation of the strategic constraints from activity 7

Output of the activity: the selected F*l*OSS Component to integrate in the software application.

Actors: All actors meet but it is finally the client who takes the final decision

Estimated time: 0.5 person-day per team

4 Our FlOSS Selection Approach in the Tabellio Project

The purpose of the Tabellio project is to develop a parliamentary software application for the two francophone parliaments of Belgium (PCF and PFB). The application must include an information system for drafting, managing and publishing legislative and parliamentary documents. It must also include other generic tools commonly found in public local administration to enable the sharing of information between regional and local administrative authorities.

Furthermore, the two parliaments also desire to distribute the software application developed during the Tabellio project under a GPL-compatible license. In turn, they insist that the application reuses existing F*l*OSS components when it is possible. The long-term objective is to initiate a F*l*OSS community around parliamentary software components so that different level of government throughout the world can exchange generic software components.

In one of the Tabellio subprojects, the four teams depicted in our F*l*OSS selection approach are the following: The *clients* are the two Belgian parliaments, the *client quality team* role is handled by CETIC, an independent research center where both authors of this paper are working, the *development team* and the *development quality team* are both handled by the software consulting firm Software AG.

Currently, Software AG is conducting the functional analysis, that is, the phase in Figure 2 before our F*l*OSS selection approach. Besides functional needs, CETIC is also gathering the generic strategic constraints already expressed by the parliaments in the call for tender as well as other work products of the Tabellio project. Two important generic strategic constraints are:

- A cost estimate must be computed for the effort needed to adapt every F*l*OSS component considered.
- The ability to collaborate with a F*l*OSS community and contribute to its F*l*OSS endeavor must be estimated.

An initial iteration of our F*l*OSS selection approach is in preparation. We plan to present this application of our approach in our future work.

5 Related Works

The Qualification and Selection Open Source (QSOS) backed by Atos Origin and Open Business Readiness Rating (OpenBRR) created by Carnegie Mellon West and Intel are two methodologies to help selecting among F/OSS components [1, 2]. A comparison of both methodologies is presented in [4]. Both methodologies are light-weight and their descriptions are not operational in nature. Consequently, their application in software development project seems not to gain in popularity.

Our approach has certain similarity to the main step of OpenBRR, which decomposes its evaluation in two steps. First, a viability check is performed, which is similar to our strategic constraints, and then an evaluation, a bit more in-depth, of a few viable F/OSS projects is undertaken. However, OpenBRR neither explains clearly the scenario in which it is useful nor does it specify which persons (roles) are to perform the various activities prescribed by OpenBRR.

Orthogonal to our research, we note that prior to being concerned with selecting and evaluating F/OSS, the software engineering research addressed the evaluation and selection of COTS (component-off-the-shelf) [5, 6]. Although similarities exist in principle, the actual methods are quite different due to the unavailability of most data in the case of COTS. Hence, the COTS selection approaches are quite different from ours in practice. Furthermore, none of the COTS selection approach survey were operationally described.

6 Conclusions and Future Works

Our F/OSS selection approach applies to most software development project interested in integrating (or reusing) F/OSS component in their custom-developed application. We now plan to study the applicability of our approach during real world development projects. In particular, we are already involved in an industrial case where public organizations (two Belgian parliaments) have hired a consulting firm for assembling F/OSS components into a new application. The parliaments and the software development consulting firm have approved our F/OSS selection approach. Since our selection approach will be applied for selection several modules of the application, we will later report on all selection occurrences of that software project.

References

1. Method for Qualification and Selection of Open Source software (QSOS) version 1.6 © Atos Origin (April 2006), http://qsos.org/
2. Business Readiness Rating for Open Source © OpenBRR.org, BRR 2005 – Request for Comment 1 (2005), http://www.openbrr.org
3. Whittaker, J.A., Jorgensen, A.: Why software fails. SIGSOFT Softw. Eng. Notes 24(4), 81–83 (1999)
4. Deprez, J.-C., Alexandre, S.: Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR & QSOS. In: Proc. of the 9th International Conference on Product Focused Software Process Improvement (PROFES 2008), Rome, Italy, June 23-25 (to appear, 2008)

5. Briand, L.C.: COTS evaluation and selection. In: Proc. International Conference on Software Maintenance, Bethesda, November 1998, pp. 222–223. IEEE Computer Society, Los Alamitos (1998)
6. Henderson-Sellers, B., Gonzalez-Perez, C., Serour, M.K., Firesmith, D.G.: Method engineering and COTS evaluation. SIGSOFT Softw. Eng. Notes 30(4), 1–4 (2005)

A Case Study of Coordination in Distributed Agile Software Development

Steinar Hole[1] and Nils Brede Moe[2]

[1] NTNU Department of Computer and Information Science
NO-7491 Trondheim, Norway
steinaho@stud.ntnu.no
[2] SINTEF Information and Communication Technology
NO-7465 Trondheim, Norway
Nils.B.Moe@sintef.no

Abstract. Global Software Development (GSD) has gained significant popularity as an emerging paradigm. Companies also show interest in applying agile approaches in distributed development to combine the advantages of both approaches. However, in their most radical forms, agile and GSD can be placed in each end of a plan-based/agile spectrum because of how work is coordinated. We describe how three GSD projects applying agile methods coordinate their work. We found that trust is needed to reduce the need of standardization and direct supervision when coordinating work in a GSD project, and that electronic chatting supports mutual adjustment. Further, co-location and modularization mitigates communication problems, enables agility in at least part of a GSD project, and renders the implementation of Scrum of Scrums possible.

Keywords: Agile development, Scrum, case study, coordinating work, mutual adjustment, direct supervision, standardization, global software development.

1 Introduction

Many organizations turn toward global software development (GSD) in their quest for cheap, higher-quality software with a short development cycle. GSD is becoming the norm by promising potential advantages like global resources, attractive cost structures, round-the-clock development and closeness to local markets [1].

To unleash the potential, methods and tools for distributed software development are designed to enable dispersed team members to share programming tasks and development practices [2]. Methods and tools are needed to mitigate GSD problems related to coordination, communication, control [3], and increased complexity [4].

Recently, there has been a growing interest in applying agile approaches in GSD to solve some of the coordination and communication challenges [3]. Several reports on the successful implementation of agile values and principles in different GSD projects conclude that there are significant differences between the fundamental principles of agile and distributed approaches, while there is a growing interest in assessing the viability of using agile practices for distributed teams [5-7].

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 189–200, 2008.

Agile development approaches and GSD approaches differ significantly in their key tenets, e.g. regarding coordination mechanisms [6]. Traditional GSD focuses on command-and-control, formal communication, and is usually implemented using a mechanistic (bureaucratic with high formalization) organizational structure. Agile development focuses on leadership-and-collaboration, informal communication and the desire for an organic organizational form [8]. Therefore, applying agile principles to GSD marks an intersection of two seemingly incompatible approaches.

Ramesh et al. [6] demonstrate how the balancing between agile and distributed approaches can help when introducing agility in GSD. They suggest that project leaders and champions should participate in coordinating the activities of the local and remote teams to help achieve project goals. Motivated by the work of Ramesh et al. [6], we investigate how work is coordinated when introducing agile methods in a GSD environment. Our research question is:

"How are tasks coordinated in GSD teams applying agile methods?"

The remainder of the paper is organized as follows. Section 2 describes GSD and agile development, and the challenges associated with merging these two approaches. Section 3 describes our research method. In Section 4, we present results from a multiple case study on agile methods and practices applied to three GSD projects. Findings are discussed in Section 5. Section 6 concludes and suggests future research.

2 Background

In this section we present background information on agile development and GSD. We use literature to describe challenges with coordination in an agile GSD context.

2.1 Agile Methods and Scrum

Agile software development comprises a number of practices and methods [9-11]. Among the most known and adopted agile methods are Extreme Programming (XP) [12] and Scrum [13]. XP focuses primarily on the implementation of software, while Scrum focuses on agile project management [14]. In this study the focus is on Scrum since Scrum is an agile approach to the management of software development projects [9-11], and thus focuses on the coordination of work.

Scrum and agile development favor a leadership-and-collaboration style of management where the traditional project manager's role is replaced with the Scrum master's role of a facilitator or coordinator [9-11]. The Scrum master is in charge of solving problems that prevents the Scrum team (5-9 people) from working effectively. He or she is often described as a coach or facilitator and does not organize the team (designers and developers); the team organizes itself and makes decisions concerning what to do. The Scrum master works to remove the impediments of the process, makes decisions in the daily meetings and validates them with management [13].

Software is developed by the self-organizing team in increments called "sprints", starting with planning and ending with a review. The team coordinates on a daily basis. Features to be implemented are registered in a backlog, and a product owner decides which backlog items should be developed in the following sprint. These items are specified in a sprint backlog.

The product backlog comprises a prioritized and constantly updated list of business and technical requirements for the system being built or enhanced. Backlog items can include features, functions, bug fixes, requested enhancements and technology updates. Multiple stakeholders can participate in generating product backlog items, such as customer, project team, marketing and sales, management and support [11].

2.2 Coordinating Mechanisms in Agile Development and GSD

The issue of agile approaches in distributed development has caught the attention of several researchers. There have been many studies reporting on the successful implementation of agile practices in GSD [5-7, 15-17], but a number of implementation barriers are also mentioned by these authors. The combination of both agile and GSD is poorly understood although it is expected to be beneficial [3]. Exploring theories on coordination of work is one way of understanding this combination.

Coordination of work is an important aspect of teamwork and team leadership [18]. Coordination together with communication and collaboration are recognized as the key enablers of software development processes [19]. There are three basic coordinating mechanisms that seem to describe the fundamental ways in which organizations can coordinate their work [20]:

1. Mutual adjustment - based on the simple process of informal communication
2. Direct supervision - one person takes responsibility for the work of others by issuing instructions and monitoring their actions
3. Standardization - of which there are four types: work processes, output, skills (as well as knowledge) and norms

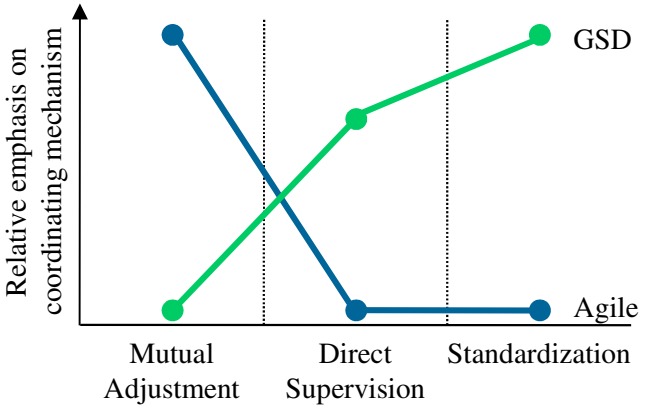


Fig. 1. Relative emphasis on coordinating mechanisms: Agile development relies purely on mutual adjustment, while GSD emphasizes standardization and some direct supervision

GSD usually relies mainly on formal mechanisms (coordination by standardization), which exploit detailed architectural design and plans to address impediments to team communication induced by geographical separation [3, 6]. Agile development relies on people and their creativity rather than on processes [21], and emphasizes informal communication (mutual adjustment) as the primary coordinating mechanism [8].

The major challenge of applying agile methods or practices in a GSD context is to balance the coordinating mechanisms (Fig. 1). However there are obvious conflicts when trying to balance mutual adjustment, direct supervision and standardization.

3 Research Design and Method

The goal of this research is to understand how the introduction of agility affects coordination of tasks in global software development teams. It is therefore important to study software development teams in practice. We have collected data from three teams using Scrum and participating in globally distributed software projects.

We report on a multiple case holistic study [22], in which we studied one phenomenon in several projects in one company. In a multiple case study, each case must be selected carefully so that it either a) predicts similar results or b) predicts contrasting results but for predictable reasons [22]. We chose option a).

3.1 Study Context

This study was done in the context of a larger action research program, where several companies have introduced elements from agile development in response to identified problems. The software company is medium-sized with approximately 150 employees in four major departments. The second author of this paper participated in the introduction and training of Scrum, and observed the company while using Scrum. The first author conducted the interviews, which we use as the primary source of data for this study. The projects participating in the study were all using Scrum for the first time; however this company was experienced with using GSD.

3.2 Data Sources and Analysis

To address the research questions, we conducted semi-structured interviews with the persons most responsible for coordination of work in the three projects, i.e. a Scrum master, a project manager and a product owner. One person was selected from each project. The interviews lasted from 30 to 40 minutes, and aimed at understanding how Scrum was applied in a GSD context. The interview guide was based on the three coordinating mechanism as proposed by Mintzberg [20] in addition to questions related to Scrum. We focused on understanding coordination of work, communication within and between the teams, feedback-sessions, planning and estimation, use of documentation, roles and specializations, and how decisions were made. All the interviews were transcribed.

4 Agility in GSD Projects

We now present the three GSD projects under study, how Scrum was implemented in these projects, and how work was coordinated in the projects.

4.1 Project India I

The goal of the project is to develop a system for integrity management of pipelines both offshore and onshore. Today several customers are interested in buying the

product, and so far three contracts have been signed. One of the biggest challenges in this project is to align requirements from potential customers from all over the world. Scrum was introduced one year after the project had started.

The project consists of six developers working full time (one is a Scrum master), two GUI designers, one product owner, and one project manager working 50% on this project. Four of the developers are situated in India together with one tester. To improve communication one of them was in periods moved to Norway.

The sprints usually lasted three weeks, ending on a Friday with a retrospective- and review-meting. The next sprint was planned the following Monday. The team organized a 15 minutes stand-up every morning discussing project related issues. The product owner usually joined all the different Scrum meetings.

Coordinating GSD Work in the India I Project. Before using Scrum the team relied on standardization and direct supervision when coordinating work with their Indian team. In the beginning, the remote team was given some easy tasks specified by the Norwegian team. The Scrum master said: "In the beginning the quality was varying, and then we thought they should only concentrate on the testing. Then they said 'No, this is not fun, please give us something more exiting to work on', and then they got different tasks, and this worked pretty well."

After using Scrum for 6 months the project had implemented all the Scrum practices, and felt they were succeeding with continuously improving their Scrum process. The team tried to work as if they were all collocated, ignoring the geographical and time differences. The Scrum master said: *"It is a big barrier being distributed. We used a lot of time on discussions between people in the two sub-teams. It did not work. The solution was to appoint one of the remote developers the role of a local Scrum master. And then we mostly communicated with her."*

To improve the communication it was decided to let the Indian Scrum master stay in Norway for a period. The Scrum master said: *"This improved the situation a lot. The productivity increased while she was here. The important issue is to communicate with only one person."* She was participating in all the Scrum meetings while situated in Norway. At the same time it was also decided to let the remote team work on its own module.

Even though they started applying Scrum, and assigning a member of the remote team as a local Scrum master, the coordination between the two teams was still described as a traditional way of developing software. During the planning meetings in Norway, the local team would plan and suggest initial estimates for all the tasks in the project, and then assign tasks to their remote partner. Later the remote team would turn these tasks into sub-tasks, and provide new estimates. In the end, the Norwegian team would check the results.

The Norwegian Scrum master, the Scrum master from India and one of the Norwegian developers had frequent meetings (2-3 times a week) with the remote team. This was a kind of distributed stand-up. In the meetings between the two sub-teams they relied on chat and e-mail. The Scrum master said: *"We tried to use telephone-conferences, but it did not work well, because of language problems. It is also easier to understand each other when relying on written communication. Also extensive use of chatting makes it possible to ask a question right away. It takes time to organize a telephone-conference."* He continued: *"It was also difficult to only use 15 minutes on the telephone. Often we used an hour. Chat is better."*

Coordination of work with the remote team was mostly based on direct-supervision. The Scrum master from India was involved in the meetings but she then decided who should do what.

4.2 Project India II

The goal of the project was to develop a system for quality audits in organizations. This project represents the second release of the system and will provide multi user support. Two departments of the studied company are involved, each acting as an internal customer responsible for contracts with their own international customer.

The project consists of a product owner, who is also a project manager, and an architect from Norway, while development is outsourced to India. Four remote developers are working 100% on the project, one of them as a team leader. In addition a few remote developers contribute part time on the project. The Indian team members are given specialized responsibilities, like GUI.

Scrum was applied from the inception of this project because, according to the product owner, "*our customer didn't understand the creation of an old-fashioned functional specification, so we thought: Okay, let's try an agile approach*." They agreed on a contract that allowed the use of a backlog with a constantly updated list of business and technical requirements, and continuous deployment of short deliveries. The backlog was maintained by the product owner. In addition to the described Scrum practices, they used continuous integration and semi-automatic deployment, and code reviews.

Coordinating GSD Work in the India II Project. The project started after the first initial backlog was created by the product owner. After the initial design was created, the work was then planned and divided into sprints in cooperation with the Indian team. This failed. The product owner said: *"I quickly gave up these sprints, that is, to define them together with the remote team."* She continued: *"It was very difficult because of problems with the communication. […] We didn't understand each other, and then there were cultural differences, too."*

The product owner explained how they changed their way of coordinating work, after finding it too time consuming to do the sprint planning in cooperation with the remote team: *"We then started sending them work-packages specified in detail, but we realized it would be a too big job to do this for each work package."* The solution was then to create a principal work plan and then further specify and document backlog items with use-cases described in documents.

The product owner and the remote team leader communicate daily, often several times a day. She said: *"There has been a team leader down there who assigned the tasks to the team, so I've only been dealing with him."*

The assignment of tasks to the Indian team became less detail oriented and instead there was an increased focus on continuous communication. It seems like the product owner tried to act more as described in the Scrum literature. She was maintaining the backlog and specifications, while letting the Indian team work out the specifications: *"I do not know everything, therefore I try to communicate: 'This is the use case, you need to solve this. Work it out.' And it works, and then they ask: 'Can we discuss', and of course, we do."*

Coordination of work with the remote team was mainly based on direct supervision and standardization in the form of written specifications and reporting of status, but the team was also relying on frequent informal communication. However, the biggest challenge was getting feedback from the remote team. The product owner said: *"What I miss, though, is that they should detect problems and show initiative."*

4.3 Project Eastern Europe

The goal of the project is to develop a system for collection and visualization of data from ship-inspections. When ships are inspected, the results are stored in the system, and the collected data are visualized through 3D models. The 3D engine was first developed as a prototype 5 years ago, before it was integrated into the core system and then released. Each time the product is sold to a new customer it requires adaptation and modification of the system. Several contracts with different customers from all over the world have been signed.

Four to five developers are situated in the remote team in an East European country, while two developers are situated in Norway, together with two persons from the support department, one from sales and a project manager acting as a product owner. The Norwegian team implements the daily Scrum. These meetings are also used for discussion of future solutions. They tried to implement sprints for the whole project, but failed. Tasks are mostly assigned to the Norwegian team's members by the project manager, who said, while pointing at the backlog: *"There, I've been putting some signatures on who is going to do what."*

Coordinating GSD Work in the Eastern Europe Project. The project was originnally applying a traditional, waterfall inspired model. This changed a year ago when a new project manager was assigned. The two distributed teams tried to use a common Scrum process. They were conducting several joint stand-ups each week, and implemented shared responsibilities. Originally, the remote team was only responsible for the creation of 3D models, but when it was decided to integrate them in the total development process, they faced new challenges. The project manager said: *"We thought that we should try Scrum, but because we wanted the remote team to take part in development and bug fixing, stand-up became a challenge. […] We didn't manage to interact and cooperate, it became too time consuming."*

According to the project manager, the remote team was unfamiliar with the system. This unfamiliarity made communication time consuming. The project manager said: *"We felt that the Norwegian team members used too much time communicating with the remote team."* The project manager also felt that the remote team did not deliver as expected. She said: *"Often, the software seemed inadequately tested."* This dissatisfaction was communicated to the remote team.

The project manager considered the problem to be difficulties gaining a thorough understanding of the complex source code, and commented on how tasks were divided: *"If we had managed to identify bigger chunks of new functionality to be developed by the remote team, it might have been easier for them."* To improve the situation it was decided to divide responsibility between the teams and to give the remote team tasks that required less cross-site coordination. The Norwegian team is now responsible for the core system, bug fixing, new functionality and customer relations, while the remote team is mainly responsible for system configuration and the

creation of 3D models for each customer. The project manager said: *"Because of their 3D competency, it works, because then they don't have to communicate with us all the time. […] It's only if they lack a specification or domain knowledge, for instance when they miss an overview of what to put on the ship, then they come back and ask."*

Coordination of work between the teams was mainly based on standardization, and to some degree direct supervision. The level of mutual adjustment was low.

5 Discussion

In this section we present our key observations in light of our research question: *How are tasks coordinated in GSD teams applying agile methods?* To answer this question we need to evaluate the degree to which the projects conformed to the generally accepted elements of the Scrum methodology. None of the projects succeeded in implementing a shared Scrum process for both the local and remote team, and only the local team in the India I project was using Scrum as intended. One reason for the reported problems was the failed attempt to implement mutual adjustment in the distributed process. Agile development relies on mutual adjustment. All the projects ended up using the traditional approach relying on direct supervision and standardization when coordinating remote work. Figure 2 summarizes the coordinating mechanisms used between the teams. The graphs are drawn from the bases of the interview data and are also discussed with the interviewees.

5.1 Challenges Implementing Mutual Adjustment in GSD

All three projects tried to implement daily stand-ups as they are the most important instrument for mutual adjustment. However, they all experienced these meetings as time consuming, because of the flow of questions from the remote site. Language and cultural differences were also a reason for the problems with these meetings. Communication problems, often reported in GSD projects [6, 23], led to the replacement of daily meetings with direct supervision and detailed specifications. This probably made it difficult to solve the communication problems [24], discuss the backlog, and to self-organize; one of the key tenets of agile development [25].

Ramesh et al. [6] suggest four practices to improve communication; synchronize work hours, provide for informal communication through formal channels, balanced coordination and constant communication. India II was only partly synchronized, but managed to communicate frequently and relied on formal channels, i.e. communication through people with dedicated roles. India I reduced the need for synchronization and coordination through modularization and communicated frequently with the remote Scrum master. The team from Eastern Europe used synchronized work hours, enabling constant communication, but the amount of communication and the lack of formalized channels negated the positive effect.

All three projects were using Scrum for the first time, and it is possible that more mature Scrum teams would communicate more efficiently because they may be more knowledgeable about and have a better understanding of issues related to applying an agile approach in a GSD project. Furthermore, none of the remote teams were trained in Scrum and this probably resulted in a lack of process understanding.

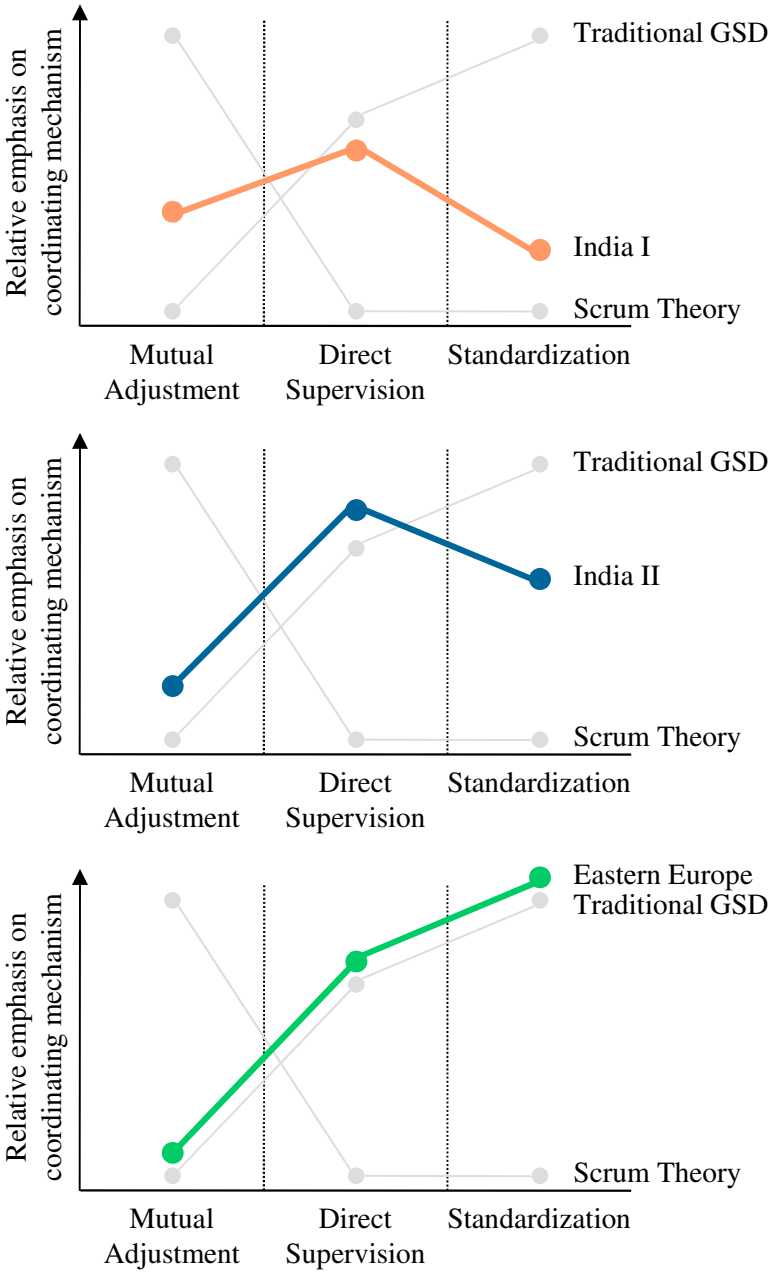


Fig. 2. Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on direct supervision and standardization than on mutual adjustment

5.2 Implementing Scrum Practices

There was no joint Scrum process between the teams; however India I succeeded in implementing Scrum in Norway by dividing the project into modules, appointing a remote Scrum master, and by moving her to Norway for periods. The other projects used a similar approach, making the remote team responsible for specific modules. This reduced the need for everyone to communicate with everyone, and made communication less critical. The Eastern Europe project chose to assign standardized tasks to the remote team, as less complex tasks reduce the need for mutual adjustment [20]. Fowler [26] argues that this kind of modularization is important to succeed with distributed Scrum, because a remote team that is responsible for an entire module from planning to testing gets a deeper understanding of the tasks it is working on. He also suggests continuous integration to avoid surprises when integrating the modules.

Modularization also makes it possible to implement a Scrum of Scrums approach [27], where several teams follow their own Scrum process. The total process will then be coordinated through meetings between the Scrum masters. India I was in an early phase of implementing Scrum of Scrums.

Two of the projects improved their level of mutual adjustment after first substituting this coordinating mechanism with standardization and direct supervision. Electronic chatting was the best remedy to support mutual adjustment, since it is instant, written text is less hampered by noise than speech, and it was perceived as timesaving compared to using a telephone conference.

All projects focused on direct supervision after failing to use Scrum, but after some months, they all felt they could reduce their level of direct supervision because of an increased level of trust. Among the reasons for increased trust are frequent and reliable communication [24] and frequent visits by distributed partners [6]. Trust is a prerequisite for effective mutual adjustment [24].

6 Conclusion and Future Work

This paper presented data from a multiple case study. None of the projects succeeded in implementing mutual adjustment, and Scrum was only implemented in one local team. In the end the projects applied a subset of Scrum practices. We found that:

- A high level of trust is important for reducing direct supervision and standardization which is important to enable mutual adjustment.
- Co-locating the remote Scrum master with the local team and making the remote team responsible for dedicated modules, makes it possible to implement Scrum in part of a GSD project, and to implement Scrum of Scrums. This also reduces the need for everyone to communicate with everyone in the GSD project.
- The communication problems caused by distribution are a threat to mutual adjustment, however electronic chatting enables mutual adjustment.
- In addition, there is a need for more research utilizing formal analytical methods on how work is coordinated in mature agile GSD teams, e.g. teams using Scrum of Scrums, and when there is a common Scrum process.

Acknowledgement

We appreciate the input received from the project participants of the investigated company and from the review by Hamish Barney and Odd Nordland. This research is supported by the Research Council of Norway under Grant 174390/I40.

References

1. Damian, D., Moitra, D.: Global software development: How far have we come? IEEE Software 23, 17–19 (2006)
2. Canfora, G., Cimitile, A., Di Lucca, G.A., Visaggio, C.A.: How distribution affects the success of pair programming. International Journal of Software Engineering and Knowledge Engineering 16, 293–313 (2006)
3. Agerfalk, P.J., Fitzgerald, B.: Flexible and distributed software processes: Old petunias in new bowls? Communications of the ACM 49, 26–34 (2006)
4. Carmel, E., Agarwal, R.: Tactical approaches for alleviating distance in global software development. IEEE Software 18, 22–29 (2001)
5. Holmstrom, H., Fitzgerald, B., Agerfalk, P.J., Conchuir, E.O.: Agile practices reduce distance in global software development. Information Systems Management 23, 7–18 (2006)
6. Ramesh, B., Cao, L., Mohan, K., Xu, P.: Can distributed software development be agile? Communications of the ACM 49, 41–46 (2006)
7. Paasivaara, M., Lassenius, C.: Could Global Software Development Benefit from Agile Methods? In: Casper, L. (ed.) ICGSE, International Conference on Global Software Engineering, pp. 109–113 (2006)
8. Nerur, S., Mahapatra, R., Mangalaraj, G.: Challenges of migrating to agile methodologies. Communications of the ACM 48, 72–78 (2005)
9. Erickson, J., Lyytinen, K., Siau, K.: Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research. Journal of Database Management 16, 88–100 (2005)
10. Cohen, D., Lindvall, M., Costa, P.: An Introduction to Agile Methods. In: Zelkowitz, M.V. (ed.) Advances in Computers, Advances in Software Engineering, vol. 62. Elsevier, Amsterdam (2004)
11. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods - Review and analysis, vol. 478. VTT Publications (2002)
12. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading (2004)
13. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall PTR, Upper Saddle River (2001)
14. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J.: New directions on agile methods A comparative analysis, pp. 244–254 (2003)
15. Farmer, M.: DecisionSpace infrastructure: agile development in a large, distributed team. Agile Development Conference, pp. 95–99 (2004)
16. Nisar, M.F., Hameed, T.: Agile methods handling offshore software development issues. In: Hameed, T. (ed.) International Multitopic Conference 2004, pp. 417–422 (2004)
17. Sulfaro, M.: Agile Practices in a Large Organization: The Experience of Poste Italiane. In: Concas, G., Damiani, E., Scotto, M., Succi, G. (eds.) XP 2007. LNCS, vol. 4536. Springer, Heidelberg (2007)

18. Salas, E., Sims, D.E., Burke, C.S.: Is there a "big five" in teamwork? Small Group Research 36, 555–599 (2005)
19. Layman, L., Williams, L., Damian, D., Bures, H.: Essential communication practices for Extreme Programming in a global software development team. Information and Software Technology 48, 781–794 (2006)
20. Mintzberg, H.: Mintzberg on Management: Inside Our Strange World of Organizations (1989)
21. Cockburn, A., Highsmith, J.: Agile software development: The people factor. Computer 34, 131–133 (2001)
22. Yin, R.K.: Case Study Research: Design and Methods. Sage Publications Inc., Thousand Oaks (2003)
23. Herbsleb, J.D., Mockus, A.: An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering 29, 481–494 (2003)
24. Moe, N.B., Smite, D.: Understanding Lacking Trust in Global Software Teams: A Multi-Case Study. In: Münch, J., Abrahamsson, P. (eds.) PROFES 2007. LNCS, vol. 4589, pp. 20–32. Springer, Heidelberg (2007)
25. Dyba, T., Dingsoyr, T.: Empirical Studies of Agile Software Development: A Systematic Review. Information and Software Technology (2008)
26. Fowler, M.: Using an Agile Software Process with Offshore Development (2003), http://www.martinfowler.com
27. Sutherland, J., Viktorov, A., Blount, J., Puntikov, N.: Distributed Scrum: Agile Project Management with Outsourced Development Teams. In: HICSS, p. 274 (2007)

ProPAMet: A Metric for Process and Project Alignment

Paula Ventura Martins[1] and Alberto Rodrigues da Silva [2]

[1] INESC-ID, FCT/Universidade do Algarve
Campus de Gambelas, Faro, Portugal
pventura@ualg.pt
[2] INESC-ID /Instituto Superior Técnico
Rua Alves Redol, nº 9 –1000-029 Lisboa, Portugal
alberto.silva@acm.org

Abstract. Software Process Improvement (SPI) is one of the main software development challenges. Unfortunately, process descriptions generally do not correspond to the processes actually performed during software development projects. Process and project alignment is essential to really find out how process improvement is important to achieve an organization's strategic objectives. Considering this approach, this paper presents a new software SPI methodology designated by Process and Project Alignment Methodology (ProPAM). As a complement to be aware about project changes and facilitate the migration to an improved process, we propose a metric called ProPAMet to analyze the alignment between process and projects. To conclude, a case study contributed to validate the effectiveness of ProPAM and ProPAMet.

1 Introduction

Software process improvement (SPI) is a challenge to organizations trying to continually improve the quality and productivity of software and to keep up their competitiveness [1]. However, there has been limited success for many SPI efforts. Recent reports concluded that 70% of organizations attempting to adopt the CMM (Capability Maturity Model) failed in achieving the intended goals [2].

There is a vast literature about process improvement approaches, such as: CMM [3], CMMI [4], ISO/IEC 15504 [5-7], BOOTSTRAP[8]. However, they don't tell though how to improve and which are the specific means to get into a particular maturity level. These approaches don't provide methods for process elicitation and modelling in order that projects follow specific development processes. They don't show how project practices and knowledge is gathered to contribute for process improvement. They don't explain the mechanisms of team members' collaboration to cope with changing contexts or react to existing problems. These are the main reasons for limited success in many SPI programs. Also important is the fact that some studies recognize the need of further research on implementing SPI [9].

Evaluation and, as a consequent, improvement of software processes would be impossible without software measurement [10]. Software process assessment is a mean for organizations to identify their strengths, weakness, existing improvement activities and key disciplines for improvement. Measurement-based SPI enables organizations

R.V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 201–212, 2008.

to determine the current state of their software process and to evaluate results of developed SPI programs. It also allows: (1) determining the effectiveness of applied processes [11]; (2) studying the effects of new practices introduced through improvement programs [12]; and (3) finally specifying process models that correspond to the processes actually performed [11].

All these factors allowed identifying several problems associated to existing SPI approaches, such as: (1) improvement actions focused on SPI models and ignoring organizational culture; (2) existing SPI models require several investments, such as: budget, time and human resources; (3) absence of key practitioners involvement result in resistance to change; (4) process descriptions generally do not correspond to the processes actually performed during software development projects; (5) existing SPI models don't provide methods for process and project representation; (6) existing SPI models identify *what* to improve but don't give any information about *how* to do it; and (7) no indicator about how project practices are diverging from the base process.

One of the contributions of this paper is to present a new SPI approach, **Pro**cess and **P**roject **A**lignment **M**ethodology (ProPAM) is a SPI approach based on process and project data in order to detect misalignments between projects and supporting processes. The development of a metric for evaluating the accuracy of process and project alignment and the need to improve the process is another contribution that we intend to introduce in the domain of SPI. **Pro**cess and **P**roject **A**lignment **Met**ric (ProPAMet) allows determining the alignment between processes and projects considered as an indicator to perform changes in base processes.

This paper is organized in the following sections. Section 2 describes briefly the proposed ProPAM methodology. Section 3 presents the proposed metric to analyse process and project alignment. Section 4 presents a case study performed in a Portuguese organization. Finally, Section 5 concludes and introduces future trends.

2 ProPAM Methodology

As mentioned in previous section, existing SPI models are insufficient to guide change in a constantly changing, constrained and increasingly unpredictable environment. Process and Project Alignment Methodology (ProPAM) directs attention to organization's needs for communication, coordination and collaboration within and between project teams. The methodology is about how the process and project are represented and how project teams acquire and use knowledge to improve work. ProPAM methodology proposes solving the problems faced in software development projects carried within the organizations. A critical feature in ProPAM is the integration of SPI activities with software development activities. This way, we considered project teams and projects as the baseline for improvement. A detailed specification of ProPAM can be found in [13].

As Figure 1 illustrates, ProPAM methodology includes SPI activities that intends to develop and implement the software process of an organization (process level). Nevertheless, SPI activities also include monitoring and tracking of software projects (project level). At project level, the methodology proposes to assist organizations in its efforts to assess and manage problematic situations of specific projects, and develop and implement solutions that help manage these problems. The project level

covers project(s) information needed to systematically support or reject many of decisions about the process. At process level, project's feedbacks conduct to process reviews and iterative process improvement. The dynamic interplay between these two levels (project level and process level) show the synergy between the activities performed by project roles (project manager and team member) and the activities performed by the process roles (process manager) involved in SPI.

Figure 1 overviews the ProPAM methodology that includes the alignment between the process and project(s) illustrated through process and projects levels. The scope of the levels is well defined in order for process and projects actors collaborate on SPI programs. However, to manage the inherent complexity of these levels, namely ProPAM represented at process level, it is current practice to divide such models into views. In general, a view is defined as a projection of a process model that focuses on selected features of the process [14]. ProPAM is organized in two correlated and complementary views, the static view and the dynamic view that represent the behaviour at that particular level. Whereas the static view describes aspects of the methodology as core and supporting disciplines in terms of activities, work products and roles, the dynamic view shows the lifecycle aspects of ProPAM expressed in terms of stages and milestones.

ProPAM static view integrates project management, process management, SPI and knowledge management (KM) disciplines, as illustrated in diagram of Figure 1. These disciplines assure alignment of projects with organization vision and goals, and the adopted and improved software process.

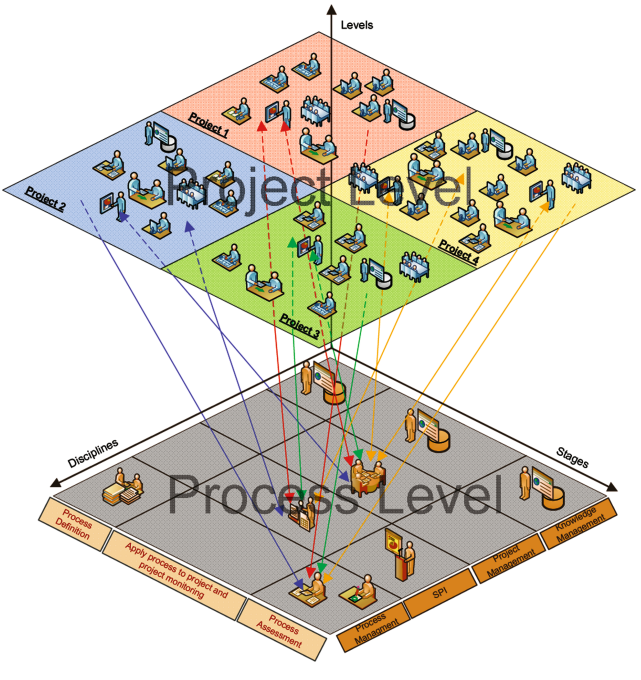


Fig. 1. Process and Project Alignment Methodology (ProPAM)

ProPAM dynamic view covers iterative process improvement through a SPI life cycle with three stages: (1) process definition; (2) apply process to project(s) and monitoring; and (3) process assessment and refinement. The **process definition** stage main goal is dedicated to an initial process specification through the application of the PIT-ProcessM metamodel specified in Figure 2. **Apply process to project(s) and monitoring** stage involves planning and executing the project within the base process best practices. It also provides assurance that the project is progressing according to the base process or reveals the need to take SPI actions because the activities performed by team members are different from those specified in the process. Project problems may occur and a new set of practices must be imposed or the process manager detects that new practices are needed. In the **process assessment and refinement** stage, initially, the project manager and process manager analyze project data and produce assessments focused on project issues and process issues, respectively. After that, the results gathered during assessments enable improvements and consistent refinements of the base process creating a new process version.

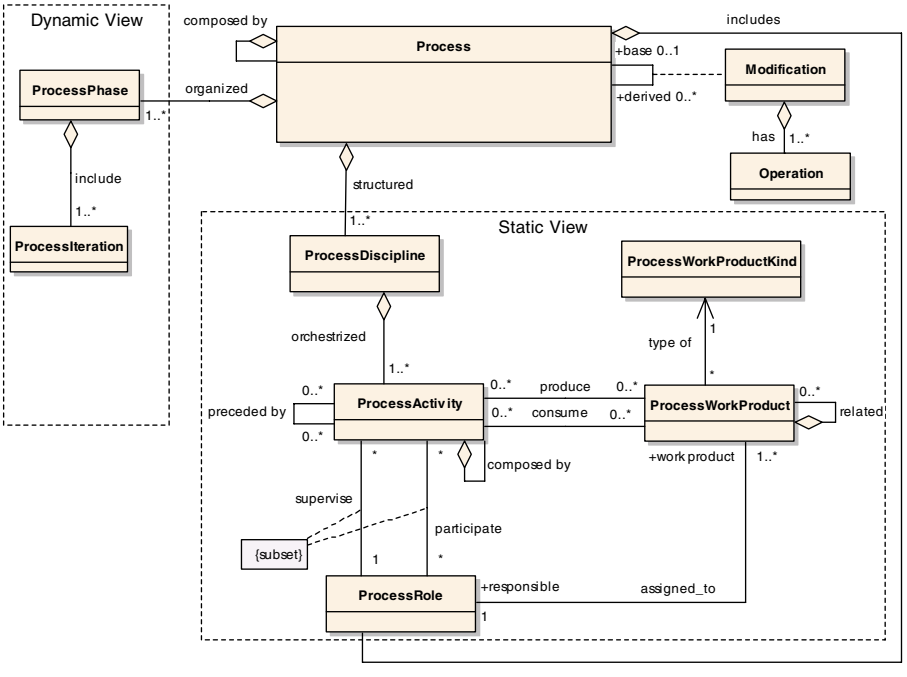


Fig. 2. ProjectIT Process Meta-model

3 Process and Project Alignment Metric (ProPAMet)

Process descriptions generally do not correspond to the processes actually supporting software development projects. They just represent high-level plans and so, do not contain the concrete information necessary for a software project. This lack of alignment between the process and project(s) results from processes unrelated to project

activities and failure in detecting project changes to improve the process. Process and project alignment is essential to really find out how process management is important to achieve an organization's strategic objectives.

However, progressive modifications in projects can cause misalignments with the original process. These modifications can be management innovations or changes in the way the activities are executed. Furthermore, a modification may regard not only the considered activity, product or actor but it can also affect other elements having a dependence relation with the modified one. ProPAM methodology provides mechanisms to detect misalignments between processes and projects through detection of changes and innovations in project's activities. This mechanism is based on a metric that allows defining the alignment degree between process and project.

Process and project alignment is defined as the degree to which the project activities support and are supported by the process practices. Moreover, it involves a real match between process practices and projects activities, products and actors. We propose a metric where the process is considered the reference, and the measure provides a balanced assessment of the fidelity of matches and gaps.

Project metrics goals are important to improve project-by-project performance, divisional/sector performance, or organizational performance. Process metrics are also important to quantify attributes of the development process and the development environment. However, none of these approaches allows identifying the similarities between the features considered in both domains (process and project). The alignment metric that we propose intends to characterize how closely the projects are related to their base process.

The **Pro**cess and **P**roject **A**lignment **Met**ric (ProPAMet) evaluates the mapping features of one project to features of the process. The alignment measure intends to evaluate the correctly match between process and project features (aligned features) divided by the total number of features identified in a project. The measurement process contains the following phases:

- identification and classification of project features by categories, considering: (1) unaligned features in process entities and (2) aligned features in process entities;
- calculating an aligned features value;
- calculating general project features value;
- calculating final process and project alignment value.

For measurement purposes, project features are organized according to process entities in the following five categories: phase, discipline, role, work product and activity. The formula to derive process and project alignment can be written:

$$\text{ProPAMet}(AF,TPF)=AF/TPF\ . \tag{1}$$

Where:

AF = number of project's entities that have a correspondence in process' entities (**A**ligned **F**eatures).

TPF = number of project's features (aligned and unaligned features) in all five categories (**T**otal **P**roject **F**eatures).

In (1) each term **AF** and **TPF** is calculated by formulas (2) and (3). However, features from different categories have a different impact in the process. To consider the

Table 1. ProPAM weights

Category	Weight factor
Phase	0,2
Discipline	0,25
Role	0,1
Work Product	0,05
Activity	0,05

relevance of each category, project features must be multiplied by a constant weight w_i in formulas (2) and (3). Each category has its own constant weight w_i (i=1,2,3,4,5) as presented on Table 1.

To derive the **AF** function, first identify all project entities that have correspondence in process entities. Then weight each aligned feature based on one of the five categories. The sum of these weights is called the aligned features function (AF):

$$AF(i,a,w)= \Sigma (a_i.w_i) . \quad (2)$$

In (2) each term **i**, $\mathbf{a_i}$ and $\mathbf{w_i}$ represent:

i = features are classified through five categories (phase, discipline, role, work product and activity)

$\mathbf{a_i}$ = number of project aligned features classified in category i.

$\mathbf{w_i}$ =weight assigned to category i.

However, some features could have no correspondence in process entities for respective categories. Then, consider project features as the project activities with and without correspondence in process entities. Then, weight each project feature based on one of the five categories (Table 1). The sum of these weights is called the total project features (TPF):

$$TPF(i,p,w) = \Sigma (p_i.w_i) . \quad (3)$$

In (3) each term **i**, $\mathbf{p_i}$ and $\mathbf{w_i}$ represent:

i = features are classified through five categories (phase, discipline, role, work product and activity)

$\mathbf{p_i}$ = number of project features (aligned and unaligned) classified in category i.

$\mathbf{w_i}$ = weight assigned to category i.

Over time, process and project misalignment happens when projects practices gradually changes to a point where differences to the base process are considered relevant. ProPAMet, a metric to evaluate process and project alignment enables to compare performed practices in current project with practices of the base process. The metric compares the base process at time t_0 with practices in projects at time t_i, allowing notifying practitioners about differences between process and project practices. When the ProPAMet threshold is crossed, differences are significant and it is a recommended to start a new SPI program which probably will conduct to an improved process version.

Figure 3 illustrates the second stage (apply process to projects and monitoring stage) of a hypothetic SPI program. Three iterations were executed until achieve organization goals. Iteration ends when the metric threshold is crossed and a new process version is delivered. This cycle concludes when the SPI program goals are fulfilled.

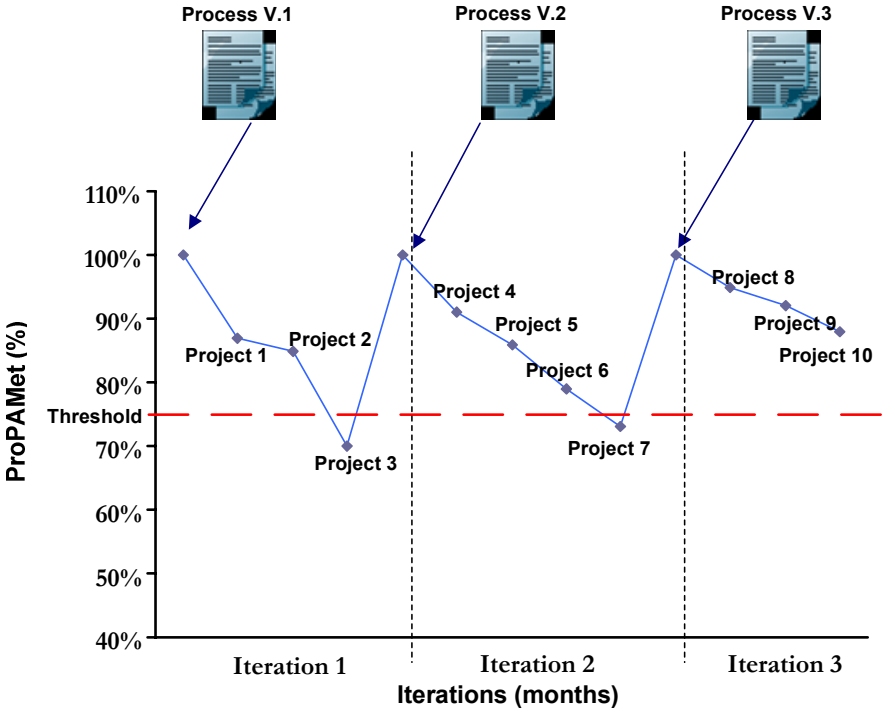


Fig. 3. Example of a SPI program with three iterations

4 Case Study

The purpose of the case study was to evaluate the effectiveness of ProPAM as a new methodology for SPI in small and medium organizations. We collaborate with a Portuguese software house that had demonstrated interest to define and improve their software development process. The case study includes the observation of three different projects and the application of the proposed methodology – ProPAM – to define and improve their software development process.

A SPI program was conducted in order to control and analyse projects developed by this organization. The SPI program was organized in three stages. The first stage was dedicated to an initial process specification at process level. Wile in the second stage several activities had been realized at process and project level. At project level, three projects had been under inspection to detect, introduce and validate new software development practices. Then, these practices had been analysed at process level as candidates for future improvements in the base process. Final stage main is dedicated to specify the improved process and includes a final feedback meeting to discuss introduced practices.

SPI roles planned and performed improvement activities over a period of ten months, which resulted in the definition of the process (a process model, process documentation guideline) and a knowledge base (documents, guidelines, projects data, template library). At the end, the changed process had been presented to senior manager and project teams and further modified and improved based on their feedback.

Critical work of a SPI program was developed during the second stage of the SI program. At project level, the project PTF had been monitored during 12 iterations that lasted one, two or three week's time each. Project NGRID and PIS were organized in fewer iterations, respectively 5 and 4 as showed in Table 2. This table also provides a profile of attributes for the three projects.

Table 2. Main features of the three inspected projects

Characteristic	Project NGRID	Project PIS	Project PTF
Project name	NGRID	PIS	PTF
Application	Web-based application	Web-based application	Portal (front-end and back-office)
Duration	7 weeks (planned) 10 weeks (actual)	6 weeks (planned) 9 weeks (actual)	18 weeks (planned) 25 weeks (actual)
Number of Iterations	5 iterations	4 iterations	12 iterations
Iteration length	5 x 2 weeks	1 x 2 weeks 1 x 1 week 2 x 3 weeks	3 x 2 weeks 1 x 1 week 6 x 3 weeks
Project Team Size	5	4	4

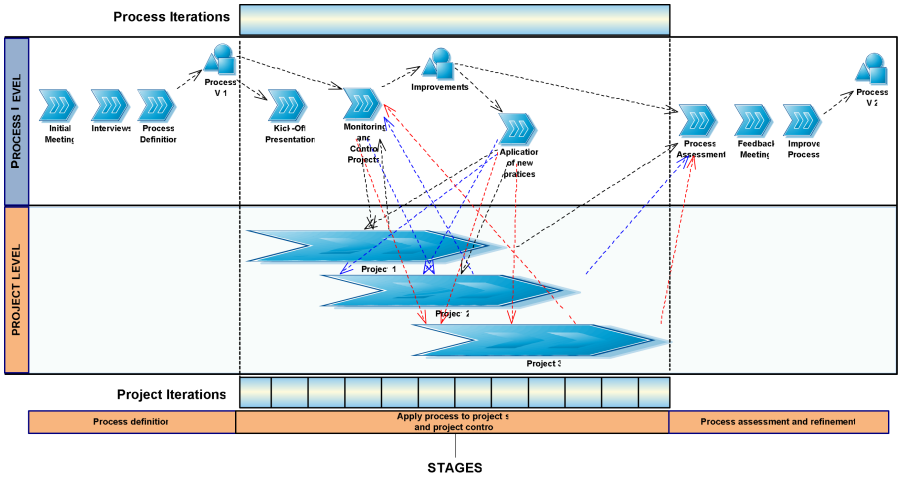


Fig. 4. SPI program performed at the Portuguese software house

At process level, only an iteration took place during the second stage. As we can see, at process level, iterations act in a different time scale expressed in months. In this case study, this iteration lasted six months. The nature of the project and process level iterations won't necessarily change much, so we recommend at least one SPI program each year. Figure 4 illustrates the difference between the time scale of the iterations at process and project level. It also identifies main activities and demonstrates the interaction between these two levels.

Table 3. Existing and new practices organized by disciplines

Analyse and design			
Interviews	+	+	+
Prototyping	+	-	-
Use cases	-	+	+
Requirements specification	+	+	+
Issue and change request management	+	+	+
Modelling	±	±	+
Design information	-	-	-
Requirements management	±	±	±
Requirements traceability through design	-	±	±
Development			
Write code	+	+	+
Test-Driven Development (TDD) techniques	-	-	-
Pair programming	-	-	-
Pair programming training	-	-	-
TDD training	-	-	-
Tests			
Unitary tests	±	±	±
System tests	±	±	±
Final client test	±	±	+
Pre-production debug	+	+	+
Test cases	±	±	±
Client participation on test cases	-	-	±
Independent tester	-	-	-
Peer review	-	-	-
Cross-reference between requirements and test cases	-	-	-
Defects management	±	±	±
Deployment			
Prepare client installation	+	+	+
Client installation	+	+	+
Prepare project presentation	+	+	+
Present project to client	+	+	+
Project Management			
Kick-off meeting	+	+	+
Prepare project meetings	-	-	-
Iteration meetings	+	+	+
Elaborate commercial proposal	±	±	±
Historical data	-	-	-
Estimation	-	-	-
Planning/replanning	±	±	±
Tracking project	-	-	±
Periodic reports	±	±	±
Risk management	-	-	±
Quality management	±	±	±
Software configuration management	-	-	-
	NGRID	**PIS**	**PTF**

During the period of the pilot case study from September 2006 to July 2007 we collect data from these three projects. All the data presented here was obtained through analysis of several projects' work products and SPI documents. The data collected were analysed statistically, and proposals were developed for improving the software development process based on the results of an analysis of the qualitative data collected in the assessment and other quality improvement findings from developed projects.

Finally, after analysing data collected through the three projects and comment final metrics results, it is the moment to verify the impact of the proposed practices in the base process. In accordance with ProPAM methodology, ProPAMet is a metric which allow identifying the degree of alignment between projects and the correspondent process. The main objective is to determine the degree of alignment in order to advice improvements in the base process. A high degree of alignment indicates that projects practices are highly synchronized with correspondent process.

Here the challenge occurs under changing project practices, such as the ones introduced in these three projects (case study) that could result in a shift to a new process version. When such changes take place, ProPAMet is used to evaluate the degree of alignment or consensus between projects and the correspondent process. ProPAMet calculus involved identification and classification of the features from projects and from the base process according to specific categories (see table 1). To determine the match between process and projects, two functions were applied (equation 2 and 3) to determine aligned features (AF) and total project features (TPF) (see section 3). In total and relative to the base process, project NGRID included the following unaligned features: 1 discipline, 10 work products and 6 activities. Whereas the corresponding values were 1, 13 and 7 in project PIS and 1, 18 and 10 in project PTF (table 3). Projects PIS and PTF included one more role in the unaligned features, the Web Designer.

Figure 5 shows the variance of the ProPAM metric (ProPAMet) within the three projects. The graphic illustrates a decrease in the value of ProPAMet from project NGRID to project PTF and, consequently, shows how changes proposals influenced project practices and conduced to process improvement.

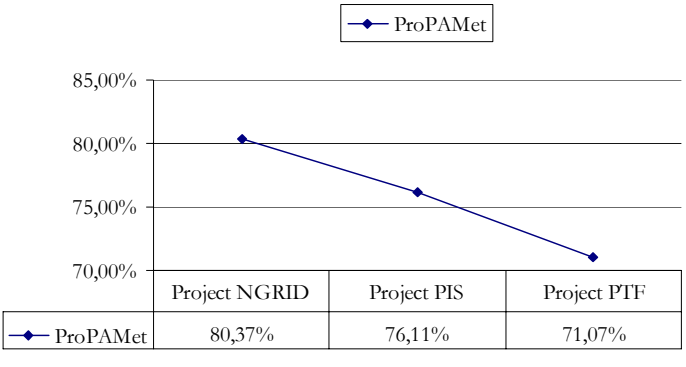


Fig. 5. ProPAM metric (ProPAMet) for each project

5 Conclusions

Currently, existing process metamodels are not suitable for SPI, since their main goal is on process specification without any consideration regarding project changes to improve the process. These problems redirected our efforts to define two metamodels (PIT-ProcessM and PIT-ProjectM) that are applied in process and project specifications and respective alignment. On the other hand, within ProPAM, PIT-metamodels contributed to the solution of previously identified SPI problems. So, these metamodels had been proposed to fulfil the following requirements: (1) provide support for process definition and improvement; (2) specify projects based on a previous process description; (3) track project issues back to the process. Round trip was also an important feature, since we used reverse engineering to improve the process model based on the changes introduced in the project description (process and project alignment).

The contribution of this thesis was not just a modelling approach to align process and project specifications, within ProPAM, we also proposed a mechanism to process evolution based on the changing needs of the software development organization. The case study, and consequent results evaluation, demonstrated the effectiveness of ProPAM to improve an organization software process.

Concerning ProPAMet metric, this paper introduced a metric that graphically helps to decide about process and project misalignments that will conduct to SPI actions to improve an organization process. When an organization applies ProPAMet metric during successive software projects, it obtains a feedback about how project practices are diverging from proposed process practices. Therefore, we highly recommend using ProPAMet during all developed software projects for two reasons: (1) evaluate if projects changes proposed by SPI programs are relevant to justify a new and improved version of the base process and (2) determine if current project practices are not aligned with base project, justifying a new SPI program to improve the process.

References

[1] Salo, O.: Improving Software Development Practices in an Agile Fashion. In: Agile Newsletter 2, Agile-ITEA edn., p. 8 (2005)
[2] Krasner, H.: Accumulating the body of evidence for the payoff of software process improvement 1997. Krasner Consulting (1997)
[3] SEI: Capability Maturity Model for Software (CMM), Version 1.1. Carnegie Mellon University (1993)
[4] SEI: Capability Maturity Model Integration (CMMI), Version 1.2. Software Engineering Institute, USA (2002)
[5] ISO/IEC: 15504-2 Information technology - Software process assessment – Part 2: A reference model for processes and process capability, July ISO/IEC TR 15504-2 (1998)
[6] ISO/IEC: 15504-5 Information technology – Software process assessment – Part 5: An assessment model and indicator guidance, ISO/IEC JTC1 / SC7 (1998)
[7] ISO/IEC: 15504-7 Information technology – Software process assessment – Part 7: Guide for use in process improvement, International Organization for Standardization ISO/IEC TR 15504-7 (1998)

[8] Kuvaja, P., Simila, J., Krzanik, L., Bicego, A., Koch, G., Saukkonen, S.: Software Process Assessment and Improvement: The BOOTSTRAP Approach. Blackwell Publishers, Malden (1994)
[9] El Emam, K., Fusaro, P., Smith, B.: Success Factors and Barriers for Software Process Improvement. Better Software Practice For Business Benefit: Principles and Experience, 355–371 (1999)
[10] Arthur, L.: Improving Software Quality: An Insider's Guide to TQM, New York (1993)
[11] Pfleeger, S., Rombach, H.: Measurement Based Process Improvement. IEEE Software, 8–11 (1994)
[12] Van Solingen, R., Berghout, E.: The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development. McGraw-Hill, Cambridge (1999)
[13] Martins, P.V., Silva, A.R.: ProPAM: SPI based on Process and Project Alignment. In: 2007 IRMA Internacional Conference. IGI Publishing, Vancouver (2007)
[14] Verlage, M.: Multi–view modeling of software processes. In: Third European Workshop on Software Process Technology. Springer, Grenoble (1994)

Author Index