Lecture Notes in Computer Science

The LNCS series reports state-of-the-art results in computer science research, development, and education, at a high level and in both printed and electronic form. Enjoying tight cooperation with the R&D community, with numerous individuals, as well as with prestigious organizations and societies, LNCS has grown into the most comprehensive computer science research forum available.

The scope of LNCS, including its subseries LNAI and LNBI, spans the whole range of computer science and information technology including interdisciplinary topics in a variety of application fields. The type of material published traditionally includes

- proceedings (published in time for the respective conference)
- post-proceedings (consisting of thoroughly revised final full papers)
- research monographs (which may be based on outstanding PhD work, research projects, technical reports, etc.)

More recently, several color-cover sublines have been added featuring, beyond a collection of papers, various added-value components; these sublines include

- tutorials (textbook-like monographs or collections of lectures given at advanced courses)
- state-of-the-art surveys (offering complete and mediated coverage of a topic)
- hot topics (introducing emergent topics to the broader community)

In parallel to the printed book, each new volume is published electronically in LNCS Online.

Detailed information on LNCS can be found at
www.springer.com/lncs

Proposals for publication should be sent to
LNCS Editorial, Tiergartenstr. 17, 69121 Heidelberg, Germany
E-mail: lncs@springer.com

ISSN 0302-9743

ISBN 978-3-540-69564-6

Lecture Notes in Computer Science
LNCS LNAI LNBI

springer.com

Jedlitschka • Salo (Eds.)

LNCS 5089

Product-Focused Software Process Improvement

PROFES

LNCS 5089

Andreas Jedlitschka
Outi Salo (Eds.)

Product-Focused Software Process Improvement

9th International Conference, PROFES 2008
Monte Porzio Catone, Italy, June 2008
Proceedings

Springer

Lecture Notes in Computer Science 5089

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

Editorial Board

Andreas Jedlitschka Outi Salo (Eds.)

Product-Focused Software Process Improvement

9th International Conference, PROFES 2008
Monte Porzio Catone, Italy, June 23-25, 2008
Proceedings

Springer

Volume Editors

Andreas Jedlitschka
Fraunhofer Institute for Experimental Software Engineering
Fraunhofer Platz 1, 67663 Kaiserslautern, Germany
E-mail: andreas.jedlitschka@iese.fraunhofer.de

Outi Salo
VTT Technical Research Centre of Finland
Kaitoväylä 1, 90570 Oulu, Finland
E-mail: Outi.Salo@vtt.fi

Library of Congress Control Number: 2008929491

CR Subject Classification (1998): D.2, K.6, K.4.2, J.1

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-69564-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-69564-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12320066 06/3180 5 4 3 2 1 0

Preface

On behalf of the PROFES Organizing Committee, we are proud to present to you the proceedings of the 9th International Conference on Product-Focused Software Process Improvement (PROFES 2008) held in Frascati - Monteporzio Catone, Rome, Italy.

Since 1999, PROFES has established itself as one of the recognized international process improvement conferences. The main theme of PROFES is professional software process improvement (SPI) motivated by product and service quality needs. Focussing on a product to be developed, PROFES 2008 addressed both quality engineering and management topics including processes, methods, techniques, tools, organizations, and enabling SPI. Both solutions found in practice and the relevant research results from academia were presented.

Domains such as the automotive and mobile applications industry are growing rapidly, resulting in a strong need for professional development and improvement. Nowadays, the majority of embedded software is developed in collaboration, and distribution of embedded software development continues to increase. Thus, PROFES 2008 addressed different development modes, roles in the value chain, stakeholders' viewpoints, collaborative development, as well as economic and quality aspects. Agile development was included again as one of the themes.

Since the beginning of the series of PROFES conferences, the purpose has been to bring to light the most recent findings and novel results in the area of process improvement, and to stimulate discussion among researchers, experienced professionals, and technology providers from around the world.

The technical program was selected by a committee of leading experts in software process improvement, software process modeling, and empirical software engineering research. This year, 61 papers from 23 nations were submitted, with each paper receiving at least three reviewers. After thorough evaluation, the Program Committee selected 31 technical full papers. The topics addressed in these papers indicate that SPI is still a vibrant research discipline, but is also of high interest for the industry; many papers report on case studies or SPI-related experience gained in industry.

The technical program consisted of the tracks quality and measurement, cost estimation, capability and maturity models, lessons learned and best practices, software process improvement, systems and software quality, and agile software development.

We were proud to have three keynote speakers, Antonia Bertolini, Kurt Schneider, and Horst Degen-Hientz. Interesting tutorials and workshops were co-located with PROFES 2008.

We are thankful for the opportunity to have served as Program Co-chairs for this conference. The Program Committee members and reviewers provided excellent support in reviewing the papers. We are also grateful to the authors, presenters, and Session Chairs for their time and effort in making PROFES 2008 a success. The General Chair, Frank Bomarius, and the Steering Committee provided excellent guidance. We wish to thank Fraunhofer IESE, the VTT Technical Research Centre of Finland, and University of Rome Tor Vergata for supporting the conference. We are also grateful to the authors for high-quality papers, the Program Committee for their hard work in

reviewing the papers, and the Organizing Committee for making the event possible. In addition, we sincerely thank Frank Bomarius for his work as a General Chair of PROFES 2008. Last, but not least, many thanks to Giovanni Cantone and his team at University of Rome Tor Vergata for the local organization of this conference and the maintenance of the PROFES 2008 website, and Sonnhild Namingha and Isabelle Schlitzer at Fraunhofer IESE for her support in copyediting this volume.

June 2008

Andreas Jedlitschka
Outi Salo

Organization

General Chair

Frank Bomarius, Fraunhofer IESE and University of Applied Sciences Kaiserslautern, Germany

Program Co-chairs

Andreas Jedlitschka, Fraunhofer IESE, Germany
Outi Salo, VTT Technical Research Centre, Finland

Tutorial and Workshop Chair

Darja Šmite, Rigas Informacijas Tecnologijas Instituts, Latvia

Organization Chair

Giovanni Cantone, Università degli Studi di Roma Tor Vergata, Italy

Local Organization Committee

Università degli Studi di Roma Tor Vergata, Italy

Anna Lomartire, Centro di Calcolo e Documentazione (CCD)
Gianfranco Pesce, Centro di Calcolo e Documentazione (CCD)
Davide Falessi, Dipartimento di Informatica, Sistemi e Produzione
Maurizio Saltali, Dipartimento di Informatica, Sistemi e Produzione
Alessandro Sarcià, Dipartimento di Informatica, Sistemi e Produzione

PR Chair

S. Alessandro Sarcià, Università degli Studi di Roma Tor Vergata, Italy

Publicity Co-chairs

Benelux	Ko Doorns, Philips
Canada	Dietmar Pfahl, University of Calgary
Central Europe	Frank Seelisch, Fraunhofer IESE

Finland	**Minna Isomursu, VTT**
Japan	**Shuji Morisaki, NAIST**
Scandinavia	**Tore Dybå, SINTEF**
South America	**Christiane Gresse von Wangenheim, Universidade do Vale do Itajaí**
USA	**Raimund L. Feldmann, FC-MD, USA**

Program Committee

Zeiad A. Abdelnabi, Garyounis University - IT College, Libya
Silvia Abrahão, Universidad Politécnica de Valencia, Spain
Muhammad Ali Babar, Lero, University of Limerick, Ireland
Bente Anda, Simula Research Laboratory, Norway
Maria Teresa Baldassarre, University of Bari, Italy
Andreas Birk, SWPM - Software.Process.Management, Germany
Danilo Caivano, University of Bari, Italy
Gerardo Canfora, University of Sannio, Italy
Jeff Carver, Mississippi State, USA
Marcus Ciolkowski, Fraunhofer IESE, Germany
Reidar Conradi, Norwegian University of Science and Technology, Norway
Beniamino Di Martino, Second University of Naples, Italy
Torgeir Dingsøyr , SINTEF, Norway
Tore Dybå, SINTEF, Norway
Davide Falessi, University of Rome "Tor Vergata", Italy
Raimund Feldmann, Fraunhofer Center Maryland, USA
Jens Heidrich, Fraunhofer Institute for Experimental Software Engineering, Germany
Martin Höst, Lund University, Sweden
Frank Houdek, Daimler AG, Germany
Hajimu Iida, NAIST, Japan
Katsuro Inoue, Osaka University, Japan
Janne Järvinen, F-Secure, Finland
Erik Johansson, Ericsson Mobile Platforms, Sweden
Natalia Juristo, Universidad Politécnica de Madrid, Spain
Kari Kansala, NOKIA, Finland
Pasi Kuvaja, University of Oulu, Finland
Marek Leszak, Alcatel-Lucent, Germany
Lech Madeyski, Wroclaw University of Technology, Poland
Annukka Mäntyniemi, VTT Technical Research Centre of Finland, Finland
Annukka Mäntyniemi, Nokia, Finland
Kenichi Matsumoto, Nara Institute of Science and Technology, Japan
Makoto Matsushita, Osaka University, Japan
Nils Brede Moe, SINTEF ICT, Norway
Maurizio Morisio, Politecnico di Torino, Italy
Mark Mueller, Robert Bosch GmbH, Germany
Jürgen Münch, Fraunhofer IESE, Germany
Haruka Nakao, Japan Manned Space Systems Corporation, Japan
Risto Nevalainen , FiSMA ry, Finland

Mahmood Niazi, Keele University, UK
Paolo Panaroni, INTECS, Italy
Dietmar Pfahl, University of Calgary, Canada
Minna Pikkarainen, VTT, Finland
Teade Punter, Embedded Systems Institute (ESI), The Netherlands
Austen Rainer, University of Hertfordshire, UK
Karl Reed, La Trobe University, Australia
Daniel Rodríguez , University of Alcalá, Spain
Kurt Schneider, Leibniz Universität Hannover, Germany
Carolyn Seaman, UMBC and Fraunhofer Center Maryland, USA
Darja Smite, University of Latvia, Latvia
Michael Stupperich, Daimler AG, Germany
Guilherme Travassos, COPPE/UFRJ, Brazil
Markku Tukiainen, University of Joensuu, Finland
Mark van den Brand, Eindhoven University of Technology, The Netherlands
Rini van Solingen, LogicaCMG and Delft University of Technology, The Netherlands
Sira Vegas, Universidad Politecnica de Madrid, Spain
Matias Vierimaa, VTT, Finland
Hironori Washizaki, National Institute of Informatics, Japan
Claes Wohlin, Blekinge Institute of Technology, Sweden
Bernard Wong, University of Technology, Sydney, Australia

External Reviewers

Ramón Garcia-Martinez, Buenos Aires Institute of Technology, Argentina
Anna Grimán Padua, Simón Bolívar University, Venezuela
Martín Solari, ORT University, Uruguay
Adam Trendowicz, Fraunhofer IESE, Germany

Table of Contents

Capability and Maturity Models

Systems and Software Quality

Quality and Measurement II

Software Process Improvement

Lessons Learned and Best Practices I

Agile Software Development

Lessons Learned and Best Practices II

Workshops

MIS-PyME Software Measurement Maturity Model-Supporting the Definition of Software Measurement Programs

María Díaz-Ley[1], Félix García[2], and Mario Piattini[2]

[1] Sistemas Técnicos de Loterías del Estado (STL)
Gaming Systems Development Department 28234 Madrid, Spain
Maria.diaz@stl.es
[2] University of Castilla-La Mancha
Alarcos Research Group – Institute of Information Technologies & Systems
Dep. of Information Technologies & Systems – Escuela Superior de Informática
13071 Ciudad Real, Spain
{Felix.Garcia, Mario.Piattini}@uclm.es

Abstract. An important reason why measurement program implementation fails is that the maturity of companies as regards measurement has not been taken into account at its definition phase. Unfortunately, the major methods and frameworks supporting measurement programs –such as Goal Question Metric (GQM), Goal-Driven Software Measurement, GQ(I)M, PSM and ISO/IEC 15939– do not explicitly address, this issue, which is especially important in small and medium settings, where low measurement maturity level is typical and there are more measurement implementation constraints. Additionally, these companies usually have poor measurement knowledge, limited resources and budget, which prevent measurement integration in the corporate culture. This restricts measurement support in these companies and increases the chances of failure. In this paper we will be looking at an adaptation of the software measurement maturity method developed by Daskalantonakis. The so-called "MIS-PyME maturity model" is focused on giving support towards measurement program definition and is integrated in MIS-PyME, a methodological framework for measurement suited to small and medium settings.

Keywords: Software measurement maturity model, measurement program definition, success factor, SMEs, MIS-PyME.

1 Introduction

A software measurement program is the result of an initiative meant to define and implement the whole process required to obtain and treat certain software information needs. A successful measurement program is such that becomes a good tool [1], i.e. it directly contributes to solving a part of the engineering problem at hand and generates value rather than data [2]. However, software measurement has proved to be a complex and difficult undertaking in the field of software, especially within the context of small and medium enterprises [3].

A. Jedlitschka and O. Salo (Eds.): PROFES 2008, LNCS 5089, pp. 19–33, 2008.

In literature one can find that many factors are involved in the successful implementation of measurement programs. As an example, Gopal et al. [4] identified and checked some success factors by analyzing their effects on the success of measurement programs. The success of a measurement program was measured using two variables: use of metrics in decision-making and improved organizational performance. The success factors selected were divided into two groups: organizational and technical factors.

Daskalantonakis also developed a good practice guide based on his experience at Motorola [5, 6]. He gives major importance to the integration of measurement programs with the rest of the software processes of an organization. In addition, he argues that the best people to analyse measurement results are the project managers and engineers involved in the measurement program, since they are experts in that particular field and understand perfectly the meaning of that data.

Fenton et Hall [7] identified fifteen success factors based on their experience, which are as follows: incremental implementation, well planned metrics framework, use of existing metrics materials, involvement of developers during implementation, measurement process transparent to developers, usefulness of metrics data, feedback to developers, ensuring that data is seen to have integrity, and that measurement data is used and seen to be used, securing commitment on the part of project managers, use of automated data collection tools, constantly improving the measurement program, internal metrics champions used to manage the program, use of external metrics gurus and provision of training from practitioners.

In [8] Pfleeger states that it is necessary to link the establishment of a measurement program to the maturity level of an organization. "Metrics are welcome only when they are clearly needed and easy to collect and understand." As an example, a measurement immature organization should not intend to implement a predictive model. This may lead to results that are unexpectedly negative, positive but spurious, difficult to interpret, or difficult to build on in subsequent studies [9]. Also, measurement cannot exceed software process: if the development process does not define the types of tests, it is not possible to evaluate the efficiency of some tests as regards others.

In this paper we look at how this last success factor is integrated in MIS-PyME, a methodological framework for defining software measurement programs focused on small and medium enterprises (SMEs) or settings. We describe an adaptation of Daskalantonakis'[6] software measurement maturity method and the interface for integrating this model into MIS-PyME in order to support it for the purpose of defining measurement programs adapted to the measurement maturity of each company.

This paper is organized as follows: Section 2 brings this work into context by summarizing existing software measurement maturity models. Section 3 introduces MIS-PyME. Section 4 describes MIS-PyME measurement maturity module. Section 5 gives an example of a real-life application for this module and underlines its advantages, and Section 6 sums up the content of this paper and outlines future research.

2 Related Work

In this section the major measurement maturity methods and models found in literature are summarized. We start with Daskalantonakis' [6] method for assessing an

organization's software measurement technology which is consistent with the SEI Software process assessment methodology [10]. This method is based on a number of assumptions which determine the focus of the Measurement Technology Assessment. From these assumptions, ten themes are derived according to which the company is characterized and evaluated:

1. Formalization of the development process
2. Formalization of the measurement process
3. Scope of measurement within the organization
4. Implementation support for formally capturing and analyzing knowledge
5. Measurement evolution within the organization
6. Measurement support for management control of software projects
7. Project improvement using measurement technology
8. Product improvement using measurement technology
9. Process improvement using measurement technology
10. Predictability of project, product, and process characteristics

For each theme, five evolutionary stages are defined that a software development organization may follow in order to reach the highest level of maturity for that particular theme. These five evolutionary stages correspond to the five levels of software process maturity as defined by SEI: initial, repeatable, defined, managed and optimized. Some questions have been classified by maturity level in order to perform the assessment.

Niessink and Vliet define a capability maturity model for measurement (M-CMM) as that which can be used to assess the measurement capability of software organizations and to identify ways to improve their measurement capability [11].The model measures the measurement capability on a five ordinal scale which matches Daskalantonakis' maturity stages. However, Niessink and Vliet define a set of pre-established processes which are different for each level and have to be in place so that an organization can reside on that level. On the other hand, following Daskalantonakis' method, each theme has its own development path.

As regards measurement treatment in software capability maturity models, we must highlight CMM [10] and its successor, CMMI [12], which both include a key process called Measurement and Analysis. This process defines good practices to implement a measurement process in an organization and reach maturity level 2.

In MIS-PyME, the measurement maturity model is used as a support module to help define measurement programs which are adapted to the measurement maturity of the organization. It will not be initially used for organization evaluation purposes. The measurement maturity module will be used as a reference to seek detailed information about a number of measurement aspects, helping the user to decide whether it is convenient or not to implement an indicator for a particular maturity measurement aspect (e.g., can the organization implement the indicator for evaluation purposes?).

Based on this assumption, we found Daskalantonakis' [6] method to be the most suitable for our needs, since the themes defined for assessing maturity mostly match the measurement aspects we want to assess, and because each measurement aspect (theme) has an evolution path organized into different levels, thus allowing the user to adjust its definition depending on what can be achieved.

CMMI [12] deals with most of the measurement aspects. However, they are distributed across most of the key process areas: software project planning at level 2, integrated software management at level 3, quantitative process management at level 4, etc. [13] but it does not deal with this information in a separate module.

Niessink and van Vliet [11] developed their own model to try and evaluate an organization's measurement maturity, and we focus on encouraging the user to define a measurement program which matches the organization's measurement maturity. The key processes defined in this model do not look in sufficient detail at some important measurement capability issues, such as what the company can measure (product, process, project, etc), to what extent (some projects, the whole organization, etc.), their analysis capability (characterizing, evaluating, etc.). This model makes a broader evaluation of measurement processes and does not go into detail as much as would be necessary for users to define their measurement program.

The major models supporting software measurement program definition include: Goal Question Metric (GQM) [14], Goal-Driven Software Measurement GQ(I)M [15], PSM [16] and ISO/IEC 15939 [17]. None of them give explicit support to users in defining measurement programs suitable for their measurement maturity.

3 MIS-PyME

MIS-PyME (Marco metodológico para la definición de Indicadores de Software orientado a PyME) is a methodological framework focused on defining measurement programs based on software indicators in small and medium settings [18].

MIS-PyME framework is classified in three main modules: the methodology and roles (MIS-PyME methodology), the workproducts which give support to the methodology (MIS-PyME Measurement Goals Table, MIS-PyME Indicator Template and MIS-PyME Database) and the third module - the measurement maturity (MIS-PyME Measurement Maturity Model).

MIS-PyME Methodology is based on GQ(I)M [15, 19], but it is designed to define basic indicators which are commonly used and required in most small and medium software development settings. Like GQ(I)M, MIS-PyME is a top-down methodology since it develops a measurement program with the ultimate goal in mind, but restricts actual changes to software process improvement, and may be conditioned by the MIS-PyME table of measurement goals and the indicator templates provided. MIS-PyME work-products are as follows:

- MIS-PyME table of measurement goals: MIS-PyME framework proposes a set of structured measurement goals usually required to implement improvement activities related to software processes.
- MIS-PyME indicator templates: An indicator template is defined for each measurement goal. The indicator template will guide users and help them define indicators and measures for a specific measurement goal. An indicator template shows, among other things, the possibility of implementing the indicator as regards the measurement maturity of the company, the conditions required to successfully implement the indicator regarding previous indicators required, conditions which must be fulfilled in order to successfully implement the indicator and how to integrate this indicator into the software process. The typical questions which the indicator tries to answer are proposed. Typical outcomes and their related analysis may also be described and show the user what the potential of an indicator is, etc.
- MIS-PyME database: Each MIS-PyME indicator template contains a set of examples of real indicators which have been defined in a successfully implemented measurement program.

One of the objectives of MIS-PyME is to define and implement measurement programs which are adapted to the measurement maturity of the setting. Companies should work in defining and implementing measurement programs which they can successfully implement, rather than trying to obtain the best measure when there are several obstacles that make a successful implementation impossible.

This paper aims to describe the third module which contains MIS-PyME measurement maturity model, and how this model is linked to the indicator templates which are intended as a guide for users.

4 MIS-PyME Measurement Maturity Model

As indicated in the second section, MIS-PyME measurement maturity (MIS-PyME-MM) model is based on Daskalantonakis' [6] method, but modified as follows:

- MIS-PyME model does not only take into account the development process, but also the quality and management processes. Additionally, it deals with the process from the point of view of capability, rather than formalization.
- The scope theme has been deeply specified by indicating what the company is able to measure at each capability level.
- Implementation support does not only take into account measurement support tools, but also the development and management tools required for the company to reach each measurement capability level.
- Some themes specified in Daskalantonakis' [6] method have been unified for the sake of simplicity: "scope", "measurement evolution" and "predictability" have been joined into one, and product, project and process improvement themes have been included in other themes.
- The theme known as "Formalization of the measurement process" has not been included in MIS-PyME measurement maturity model since it is mainly used to evaluate measurement process and not so much to support measurement program definition.
- An interface between MIS-PyME measurement maturity model and the rest of the MIS-PyME framework has been defined in order to give support to the measurement analyst.

MIS-PyME measurement maturity model, which is defined in table 2, will be mainly required during the indicator definition phase. When the measurement analyst defines the indicators, he will be supported by the corresponding MIS-PyME indicator template. This template will make recommendations for measurement maturity (amongst others) in terms of indicator implementation. These recommendations come from the interface of MIS-PyME measurement maturity module.

The interface of MIS-PyME measurement maturity module, which is shown in table 3, 4 and 5, establishes a relationship between the measurement maturity model and MIS-PyME Indicator templates. This interface helps users decide if their measurement maturity is enough for certain values of the indicator field by posing questions based on MIS-PyME measurement maturity model. Therefore, some indicator fields depend on the maturity of the company as regards measurement, especially these fields are those which determine the goal of the indicator and are the following:

Table 1. Indicator template fields which depend on measurement maturity

Indicator Field	MIS-PyME-MM theme	Description
Purpose	Software management, quality and development capability	Measurement process has to fit with the rest of the processes. Otherwise, the implementation of the measurement program will in all probability fail. For example, you cannot measure the effectiveness between test phases if test phases are not well differentiated.
	Measurement scope	There are certain kinds of measures which require a certain degree of measurement maturity and previous experience. As an example, you cannot make reliable predictions on a particular aspect when there has not been any previous, frequent and rigorous measurement of that aspect.
	Tools support	In order to implement some measurement programs, some tools are required such as databases, tools that make it possible to visualize an indicator control panel, etc.
	Measurement support for management issues	Measurement should be established in order to support process improvement goals, which also means management goals. If there is not any purpose in analyzing measurement in terms of decision making or corrective actions, the implementation of a measurement program is not recommended (for example, it is not advisable to implement a measurement program for project monitoring purposes). If the existing measurement data is not used to take simple corrective actions, it is not recommended to do so for other purposes such as optimization.
Entity	Measurement scope	If organizational information is needed based on measurement usually it is previously required to measure projects or products individually. Projects are the first entities to be measured; products comes second and processes third
Focus	Tool support	There are a number of measurements which cannot be performed if certain management or development tools are not in place.
	Processes capability	The aspect to be measured has to be established by the other development, management or quality processes.

- Purpose. This field specifies the intention of the indicator. MIS-PyME suggested values based on [20] which are as follows: characterizing, monitoring, evaluating, predicting and optimizing.
- Entity: This indicator specifies what is to be measured: the process (PROC), the project (PRJ) or the product (PROD).
- Focus: It specifies the aspect or attribute to be measured, a quality attribute (reliability, portability, usability, etc.), process performance (compliance, efficiency), user satisfaction, etc.

Table 1 shows the measurement maturity aspects on which each of the above fields depends.

5 MIS-PyME Measurement Maturity Model - Case Study

In this section we show how MIS-PyME measurement maturity model was applied in an experience which consisted in implementing a measurement program in a medium-sized setting. This experience has given us an idea about the usefulness and benefits of the proposed MIS-PyME maturity model for SMEs.

The measurement program was defined and implemented in the software development and maintenance department of Sistemas Técnicos de Loterías del Estado (STL), which is formed by 39 employees. This company was created by the Spanish Government and provides operational and IT development services for the national lottery.

In 2003, the quality department in STL encouraged an initiative to implement measurement programs in the development and maintenance department in STLbut it was not well accepted and implementation was unsuccessful. The director of the development and maintenance department was nonetheless aware of the importance of measurement and was intent on mastering this. Most especially, his objective was to improve management and quality control through these measures. In July 2006, he defined two process improvement goals:

- PIG 1: Improving project and process monitoring and control. He particularly wished to improve the monitoring of the project's progress in comparison with the plan, controlling the tests phases and improving project planning.
- PIG 2: Improving development service and product quality. This goal focused on monitoring and evaluating the development service and product quality.

These process improvement goals comprise five sub-goals, and the indicators shown in figure 1.

We now show two outstanding indicators that were modified to make them be better adapted to the maturity of the company based on MIS-PyME measurement maturity model.

IND-PRJ-FiabImpl indicator aimed to evaluate the reliability of the product developed in order to take corrective actions if necessary. This indicator was necessary for the second process improvement goal (improving development service and product quality). The intention of this indicator is to "evaluate"; the focus is "reliability" and the entity is the "product".

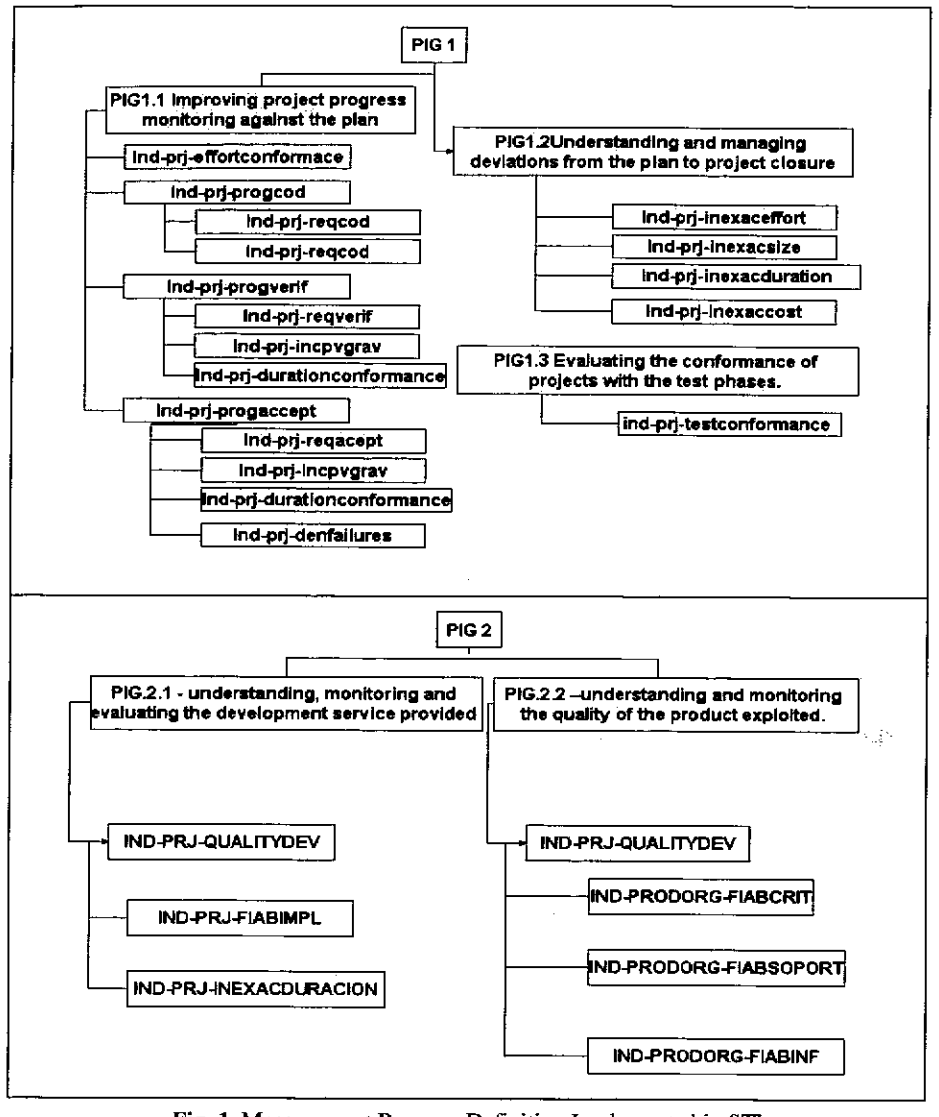

Fig. 1. Measurement Program Definition Implemented in STL

Initially, this indicator evaluated the reliability of the company based on a fix value meant as a threshold (a number of failures registered in production after the product had been installed). However, even if we had experience and we knew (more-less) the reliability of the products in production, and thanks to the suggestions included in this indicator template provided by MIS-PyME which are based on MIS-PyME measurement maturity, we realized that we were not mature enough to state what the reliability of the product would be based on the characteristics of the product developed with a fix goal. The measurement maturity model made us

reflect on this. Focusing on Table 3 and the purpose of "evaluating" the questions are: "do we rigorously, frequently and in an organized fashion measure the reliability of the product and other aspects that may have a relationship with the reliability of the product?" and "could we set reliable goals based on the available data?" Both answers were negative.

We therefore decided to evaluate indicators based on a range of values (good, normal, not too good, not acceptable). In this case we could answer affirmatively to the questions posed: we could define in a reliable way the ranges of reliability of the product developed, which would depend on the type of project: high, medium, low. As can be observed, we descend from level 4 to level 3 in terms of the measurement scope theme. Regarding the measurement support for management issues theme, the top manager was very interested in this indicator, which was included in the project close reports, and the intention was to monitor these data and take corrective actions in case of frequent negative results, therefore the answer of "¿Is measurement going to be used to take corrective actions?" is affirmative.

As regards the focus element of the indicator, the quality (see table 4), we fulfilled maturity measurement requirements. Our management process already provided a close project activity where project managers analyzed the reliability in production of the product developed in addition to other project issues. Regarding "tool support" theme, we had an incident database were failures in production were registered. Most of the people in the company used it and the process was quite well established.

The indicator Ind-PRJ-TestConformance was also modified for it to be better adapted to the maturity of the company. This indicator was defined so as to achieve the first process improvement goal: monitoring conformance with test phases. Initially, this indicator assessed conformance with test phases based on the failures detected during each test phase and compared with a threshold. We were not mature enough to define a threshold for each testing activity, but we were mature enough to define a percentage ratio threshold between test phases (e.g. more than 70% of the failures should be detected during integration test). In both cases, we went on with the fourth measurement maturity level but the second definition was easier and more reliable for us since we were experienced in analyzing the percentage of defects found. Project managers agreed to these modifications and stated that the previous definition of the indicator had not been accurate.

- The examples set out in this section illustrate how important it is to define measurement programs which are adapted to the measurement maturity of each company. Even if it seems evident, it is quite easy to make the mistake of trying to define the best measures, even if we cannot implement them. In SMEs it is still easier to make this mistake since resources, budget and measurement culture are limited, and people who define measurement programs may be from inside the company and not too experienced. MIS-PyME indicator templates advise users as to what measurement maturity requirements they should fulfill in order to define the indicator. These advise come from, MIS-PyME measurement maturity model.

Table 2. MIS-PyME. Measurement maturity model based on that developed by Daskalantonakis[6]

Themes	Level 1	Level 2	Level 3	Level 4	Level 5
Software management, quality and development capability	Immature processes. Projects depend on experienced professionals. Project management focus.	Repeat tasks which have been mastered in the past. Project depends on experienced professionals. Project management focus.	Projects characterized and reasonably understood. Project and development system management focus.	Measuring over process and process control. Focus on controlling the process.	Optimized process. Focus on process improvement. Software process improvement, benefits are quantified.
Measurement scope	Carried out occasionally with experienced people or not at all.	Carried out in big projects and with experienced people. Measurement is based on phase-by-phase project tracking, actual against estimate (size, effort, schedule, etc.). Tracking the quality of products in production. There are some estimation mechanisms and some historical data.	Organization establishes standard processes and measurement models which are followed in projects and products. Cross-projects analyses are available. Data collected and analysis are more reliable and consistent. Planning and tracking is often performed at work-package level and still involves actual vs planned performance. Defect quality measures are collected over the developed products. Threshold techniques are used.	Measurement is used in most of the projects and products. Measurement models are also process focused and there is an understanding of the process performance. The measurement process is integrated into the development process. There is a broader view of quality, not just defects but usability, maintainability, flexibility, etc. There is a set of measures that represents a quantitative model of the overall life cycle process.	Well adapted measurement models. Measuring overall process improvement, improving business results and the capability of setting quantitative improvement goals. The organization implements a control panel to keep track of achievements. Improvement can be quantitatively proved.
Tools support	There are no tools to explicitly support process measurement.	Measurement support tools focused on projects. There are some tools which support estimations. There are incident, cost and planning management tools.	Project and product focus measurement tools. There is a measurement database for storing historical data. There is a life cycle configuration management tool for each requirement, models for analysis, etc.	Measurement support tools focused on projects and products. There is an organizational database where historical data is stored. Data in the database are more reliable and there are actions to prevent recording dirty data. Development of an advanced environment is used which automatically provides product measures.	There are organizational tools which automatically collect data on the project, product and process and generate a control panel of indicators in order to provide analyses. They also generate automatic reports.
Measurement support for management issues	Management is not supported by measures	Basic project management. Milestones and commitment management. Measurement is used to take reactive decisions during project development, if there are deviations, etc.	The product developed is controlled by means of measures which are used to make decisions regarding the product. Data is used to estimate ranges and thresholds for the project and product. This allows taking corrective actions without the need of re-planning.	It is possible to predict the product, service and other attributes before the product is in production. Usual problems are controlled. It is possible to adapt processes and plans in order to achieve a certain quality degree or other kind of goal.	It is possible to predict and prevent problems. Technological needs and values are known thanks to measurement.

Table 3. MIS-PyME Measurement Maturity Interface as regards Purpose

Themes	Characterizing	Monitoring	Evaluating	Predicting	Optimizing
Software management, quality and development capability	Has the company defined the attributes which are to be measured, even informally?	(Level 3) Have the attributes which are to be measured been included in company processes? Are they correctly understood and used?	(Level 4) Is attribute evaluation included in the process?	(Level 4) Are processes stable enough to be performed rigorously and provide reliable data for the purpose of making estimations?	(Level 5) Is it possible to predict attributes in order to prevent problems and make suitable changes?

Table 3. (*continued*)

Measurement scope	(Level 1) – No maturity suggestion.	(Level 2) Is any estimating mechanism required to monitor actual vs planned? Is it available? Is data from other projects available? Is the software management process stable enough to include measurement activities? (Level 3) Has the project been monitored phase-by-phase before starting monitoring work-packages?	(Level 4) Does this attribute (and any others related to it) undergo frequent, rigorous and generalized measurement which makes it possible to define a reliable evaluation goal adapted to the characteristics of the organization? (Level 3) Is there sufficient data and business knowledge to define ranges of good, normal, bad, etc. regarding this attribute and the characteristics of the organization?	(Level 4) Has the measurement process been established organizationally? Does this attribute (and any others related to it) undergo frequent, rigorous and generalized measurement which makes it possible to predict it? Is it possible to obtain reliable estimations based on the available data?	(Level 5) Does the organization have a performance control panel of organizational processes and improvement goals? Is the organization qualitatively able to determine if a goal has been achieved? Is it possible to define a reliable improvement plan?
Tools support	(Level 1) – No maturity suggestion.	(Level 2) Are there any tools which provide the required indicators to show project progress? If any attribute product is measured based on defects or failures, is there any incident management tool in the organization?	(Level 4) Is there an organization database to store historical data?	(Level 4) Is there an organization database to store historical data? And estimation mechanisms and tools?	(Level 4) Is there a management tool to dynamically and automatically obtain indicators, reports and estimations in order to make sophisticated analyses?
Measurement support for management issues	(Level 1) What is the intended use of measurement? Is it meant to contribute to decision making and to encourage improve-	(Level 2) Is it intended to make reactive decisions based on the results of the measurement analysis regarding projects and products in production? (Level 3) – Is it intended to make decisions regarding the developed product based on	(Level 4) Is it the organization's purpose to make improvements as required to achieve a goal? (Level 3) Is measurement going to be used to take corrective actions in advance without the need of replanning?	(Level 4) Is prediction going to be used to improve project planning, avoid future problems, etc? Is prediction going to be used to adapt the planning in order to	(Level 5) What is the intended use of measurement? Is it meant to be used to make dynamic decisions in order to avoid problems and adapt the development, quality and management process? Is it intended

Table 3. (*continued*)

	ment? Otherwise, what is its purpose?	measurement results?		achieve a certain goal?	to encourage the improvement of controlled actions to achieve process and business goals?

Table 4. MIS-PyME Measurement Maturity Interface as regards Focus

Themes	Quality (maintainability, reliability, portability, usability)	Cost (effort, personnel)	Project progress, calendar	Process (compliance, effectiveness)	Client satisfaction
Software management, quality and development capability	(Level 2) Is reliability taken into account in process definition? (Level 4) Are maintainability, usability and portability taken into account in these processes? Is there any quantifiable agreement with the user as regards these aspects of the product?	(Level 2) Are workers assigned an activity or a task during the project management process? Are any estimates performed for the effort that this will involve?	(Level 2) Are projects planned? Do processes specify any monitoring activities for project progress?	(Level 4) Is the development process well defined? Are development phases well differentiated?	(Level 2) Does the process define a close collaboration for the project activity where project managers reflect about the development of the project from the point of view of the client? Do they have enough resources and experience to develop this task?
Tools support	(Level 2) Is there any incident management tool in order to obtain data based on defects and failures? Is there any configuration management tool? (Level 4) Is there any development tool where code quality attributes can be obtained such as: cyclomatic complexity, module coupling, inheritance, etc.?	(Level 2) Is there any effort/task management tool which can be used in each project?	(Level 2) Is there any tool to support project planning? (Level 3) Is there any project management tool to control project progress by work-packages?	(Level 2): Is there any incident management tool which makes it possible to obtain data based on defects and failures? And any management tool for life cycle project information?	(Level 2) Is there any incident and change request management tool which makes it possible to obtain data based on defects and failures?

Table 5. MIS-PyME Measurement Maturity Interface as regards Entity and depending on Measurement Scope theme

Project	Product	Process
(Level 3) Before monitoring work packages it is better to start with (Level 2) phase-by-phase monitoring. (Level 3) It is not possible to make cross-project analysis if a measurement model has not been established for the whole organization.	(Level 2-3) Measuring products attributes in production is usually easier, more reliable and important than measuring attributes in development, such as the reliability of products. (Level 3) Usually it is more urgent to proceed with the measurement of projects than with that of products in development, which are used for quality control. Therefore our suggestion is to start measuring projects first and then products in development. (Level 3-4) Usually it is easier and more important to measure products based on defects or failures than to measure effectiveness, reliability, and afterwards other attributes such as friendliness, complexity, maintainability, etc.	(Level 4) Usually, maturity and some experience with projects and products are required to measure the aspects of a process.

6 Conclusions and Further Research

This paper highlights a factor which must be taken into account in order to successfully implement measurement programs, which is defining measurement programs adapted to the measurement maturity of each company.

The paper gives an outline of MIS-PyME measurement maturity model, which is an adaptation of the measurement maturity method developed by Daskalantonakis [6] and the interface defined to integrate this model into MIS-PyME framework. For illustration purposes, two examples are provided and a case study (software measurement program definition in a medium setting) gives an idea of the advantages MIS-PyME measurement maturity model brings with it.

This support module, a measurement maturity framework integrated in the measurement program model for the purpose of defining measurement programs adapted to the measurement maturity of each company, is especially important for SMEs, since usually these companies have poor measurement knowledge and limited resources and budget and people from inside the company, not too experienced in the field, may be those who define the measurement program.

Our future work will revolve around testing and improving MIS-PyME measurement maturity module.

Acknowledgment. We would like to thank the staff of Sistemas Técnicos de Loterías del Estado (STL) for their collaboration. This research has been sponsored by the COMPETISOFT (CYTED, 506AC0287), ESFINGE (Dirección General de Investigación del Ministerio de Educación y Ciencia, TIN2006-15175-C05-05) and INGENIO (Junta de Comunidades de Castilla-La Mancha, PAC08-0154-9262) projects.

References

1. Hughes, R.T.: Expert Judgment as an Estimating Method. Information and Software Technology, 67–75 (1996)
2. Niessink, F., Vliet, H.v.: Measurements Should Generate Value, Rather Than Data. In: Proceedings of the Sixth International Software Metrics Symposium (METRICS 1999), Boca Raton (1999)
3. Gresse, C., Punter, T., Anacleto, A.: Software measurement for small and medium enterprises. In: 7th International Conference on Empirical Assessment in Software Engineering (EASE), Keele, UK (2003)
4. Gopal, A., et al.: Measurement Programs in Software Development: Determinants of Success. IEEE Transactions on Software Engineering 28(9), 863–875 (2002)
5. Daskalantonakis, M.K.: A Practical View of Software Measurement and Implementation Experiences Within Motorola. IEEE Transactions on Software Engineering 18(11), 998–1010 (1992)
6. Daskalantonakis, M.K., Yacobellis, R.H., Basili, V.R.: A Method for Assessing Software Measurement Technology. Quality Engineering, 27-40 (1990)
7. Hall, T., Fenton, N.: Implementing Effective Software Metrics Programs. IEEE software 14(2), 55–65 (1997)
8. Pfleeger, S.L.: Understanding and Improving Technology Transfer in Software Engineering. Systems ans Software 47 (1999)
9. Briand, L.C., Morasca, S., Basili, V.R.: An Operational Process for Goal-Driven Definition of Meassures. IEEE Transactions on Software Engineering 28, 1106–1125 (2002)
10. SEI, The Capability Maturity Model: Guidelines for Improving the Software Process, Software Engineering Institute (1995)
11. Niessink, F., Vliet, H.V.: Towards Mature Measurement Programs. Software Maintenance and Reengineering (1998)
12. CMMI Product Team: CMMI for Systems Engineering/Software Engineering, Version 1.1 - Staged Representation (CMU/SEI-2002-TR-002, ADA339224), Software Engineering Institute, Carnegie Mellon University: Pittsburgh, PA (2002)
13. Weber, C., Layman, B.: Measurement Maturity and the CMM: How measurement Practices Evolve as Processes Mature. Software Quality Parctitioner 4(3) (2002)
14. Solingen, R.v., Berghout, E.: The Goal/Quesiton/Metric Method - A practical guide for Quality Improvement of Software Development. Mc Graw Hill (1999)
15. Park, R.E., Goethert, W.B., Florac, W.A.: Goal-Driven Software Measurement-A Guidebook. Carnegie Mellon University Pittsburgh: Software Engineering Institute (1996)
16. PSM: Practical Software and Systems Measurement - A Foundation for Objective Project Management Version 4.0c. Deptartment of Defense and US Army (November 2000)
17. ISO/IEC 15939, Software Engineering-Software Measurement Process, ISO and IEC, Editors (2002)
18. Diaz-Ley, M., García, F., Piattini, M.: Software Measurement Programs in SMEs - Defining Software Indicators: A methodological framework. In: PROFES 2007 (2007)
19. Goethert, W., Siviy, J.: Applications of the Indicator Template for Measurement and Analysis. Software Engineering Measurement and Analysis Initiative (September 2004)
20. Basili, V.R., Weiss, D.: A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering 10(11), 758–773 (1984)

Printing: Mercedes-Druck, Berlin
Binding: Stein + Lehmann, Berlin

Lecture Notes in Computer Science

Sublibrary 2: Programming and Software Engineering

For information about Vols. 1– 4440
please contact your bookseller or Springer

Vol. 5095: I. Schieferdecker, A. Hartman (Eds.), Model Driven Architecture – Foundations and Applications. XIII, 446 pages. 2008.

Vol. 5089: A. Jedlitschka, O. Salo (Eds.), Product-Focused Software Process Improvement. XIV, 448 pages. 2008.

Vol. 5079: M. Alpuente, G. Vidal (Eds.), Static Analysis. X, 379 pages. 2008.

Vol. 5060: C. Rong, M.G. Jaatun, F.E. Sandnes, L.T. Yang, J. Ma (Eds.), Autonomic and Trusted Computing. XV, 666 pages. 2008.

Vol. 5055: K. Al-Begain, A. Heindl, M. Telek (Eds.), Analytical and Stochastic Modeling Techniques and Applications. XI, 323 pages. 2008.

Vol. 5052: D. Lea, G. Zavattaro (Eds.), Coordination Models and Languages. X, 347 pages. 2008.

Vol. 5051: G. Barthe, F.S. de Boer (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 259 pages. 2008.

Vol. 5048: K. Suzuki, T. Higashino, K. Yasumoto, K. El-Fakih (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2008. XII, 341 pages. 2008.

Vol. 5047: K. Suzuki, T. Higashino, T. Hasegawa, A. Ulrich (Eds.), Testing of Software and Communicating Systems. XII, 303 pages. 2008.

Vol. 5030: H. Mei (Ed.), High Confidence Software Reuse in Large Systems. XII, 388 pages. 2008.

Vol. 5026: F. Kordon, T. Vardanega (Eds.), Reliable Software Technologies – Ada-Europe 2008. XIV, 283 pages. 2008.

Vol. 5025: B. Paech, C. Rolland (Eds.), Requirements Engineering: Foundation for Software Quality. X, 205 pages. 2008.

Vol. 5020: J. Barnes, Ada 2005 Rationale. IX, 267 pages. 2008.

Vol. 5016: M. Bernardo, P. Degano, G. Zavattaro (Eds.), Formal Methods for Computational Systems Biology. X, 538 pages. 2008.

Vol. 5014: J. Cuellar, T.S.E. Maibaum (Eds.), FM 2008: Formal Methods. XIII, 436 pages. 2008.

Vol. 5007: Q. Wang, D. Pfahl, D.M. Raffo (Eds.), Making Globally Distributed Software Development a Success Story. XIV, 422 pages. 2008.

Vol. 4989: J. Garrigue, M.V. Hermenegildo (Eds.), Functional and Logic Programming. XI, 337 pages. 2008.

Vol. 4966: B. Beckert, R. Hähnle (Eds.), Tests and Proofs. X, 193 pages. 2008.

Vol. 4954: C. Pautasso, É. Tanter (Eds.), Software Composition. X, 263 pages. 2008.

Vol. 4951: M. Luck, L. Padgham (Eds.), Agent-Oriented Software Engineering VIII. XIV, 225 pages. 2008.

Vol. 4949: R.M. Hierons, J.P. Bowen, M. Harman (Eds.), Formal Methods and Testing. XIII, 367 pages. 2008.

Vol. 4937: M. Dumas, R. Heckel (Eds.), Web Services and Formal Methods. IX, 169 pages. 2008.

Vol. 4916: S. Leue, P. Merino (Eds.), Formal Methods for Industrial Critical Systems. X, 251 pages. 2008.

Vol. 4909: I. Eusgeld, F.C. Freiling, R. Reussner (Eds.), Dependability Metrics. XI, 305 pages. 2008.

Vol. 4906: M. Cebulla (Ed.), Object-Oriented Technology. VIII, 204 pages. 2008.

Vol. 4902: P. Hudak, D.S. Warren (Eds.), Practical Aspects of Declarative Languages. X, 333 pages. 2007.

Vol. 4899: K. Yorav (Ed.), Hardware and Software: Verification and Testing. XII, 267 pages. 2008.

Vol. 4888: F. Kordon, O. Sokolsky (Eds.), Composition of Embedded Systems. XII, 221 pages. 2007.

Vol. 4880: S. Overhage, C.A. Szyperski, R. Reussner, J.A. Stafford (Eds.), Software Architectures, Components, and Applications. X, 249 pages. 2008.

Vol. 4849: M. Winckler, H. Johnson, P. Palanque (Eds.), Task Models and Diagrams for User Interface Design. XIII, 299 pages. 2007.

Vol. 4839: O. Sokolsky, S. Taşıran (Eds.), Runtime Verification. VI, 215 pages. 2007.

Vol. 4834: R. Cerqueira, R.H. Campbell (Eds.), Middleware 2007. XIII, 451 pages. 2007.

Vol. 4829: M. Lumpe, W. Vanderperren (Eds.), Software Composition. VIII, 281 pages. 2007.

Vol. 4824: A. Paschke, Y. Biletskiy (Eds.), Advances in Rule Interchange and Applications. XIII, 243 pages. 2007.

Vol. 4821: J. Bennedsen, M.E. Caspersen, M. Kölling (Eds.), Reflections on the Teaching of Programming. X, 261 pages. 2008.

Vol. 4807: Z. Shao (Ed.), Programming Languages and Systems. XI, 431 pages. 2007.

Vol. 4799: A. Holzinger (Ed.), HCI and Usability for Medicine and Health Care. XVI, 458 pages. 2007.

Vol. 4789: M. Butler, M.G. Hinchey, M.M. Larrondo-Petrie (Eds.), Formal Methods and Software Engineering. VIII, 387 pages. 2007.

Vol. 4767: F. Arbab, M. Sirjani (Eds.), International Symposium on Fundamentals of Software Engineering. XIII. 450 pages. 2007.

Vol. 4765: A. Moreira, J. Grundy (Eds.), Early Aspects: Current Challenges and Future Directions. X, 199 pages. 2007.

Vol. 4764: P. Abrahamsson, N. Baddoo, T. Margaria, R. Messnarz (Eds.), Software Process Improvement. XI, 225 pages. 2007.

Vol. 4762: K.S. Namjoshi, T. Yoneda, T. Higashino, Y. Okamura (Eds.), Automated Technology for Verification and Analysis. XIV, 566 pages. 2007.

Vol. 4758: F. Oquendo (Ed.), Software Architecture. XVI, 340 pages. 2007.

Vol. 4757: F. Cappello, T. Herault, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XVI, 396 pages. 2007.

Vol. 4753: E. Duval, R. Klamma, M. Wolpers (Eds.), Creating New Learning Experiences on a Global Scale. XII, 518 pages. 2007.

Vol. 4749: B.J. Krämer, K.-J. Lin, P. Narasimhan (Eds.), Service-Oriented Computing – ICSOC 2007. XIX, 629 pages. 2007.

Vol. 4748: K. Wolter (Ed.), Formal Methods and Stochastic Models for Performance Evaluation. X, 301 pages. 2007.

Vol. 4741: C. Bessière (Ed.), Principles and Practice of Constraint Programming – CP 2007. XV, 890 pages. 2007.

Vol. 4735: G. Engels, B. Opdyke, D.C. Schmidt, F. Weil (Eds.), Model Driven Engineering Languages and Systems. XV, 698 pages. 2007.

Vol. 4716: B. Meyer, M. Joseph (Eds.), Software Engineering Approaches for Offshore and Outsourced Development. X, 201 pages. 2007.

Vol. 4709: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. VIII, 297 pages. 2007.

Vol. 4680: F. Saglietti, N. Oster (Eds.), Computer Safety, Reliability, and Security. XV, 548 pages. 2007.

Vol. 4670: V. Dahl, I. Niemelä (Eds.), Logic Programming. XII, 470 pages. 2007.

Vol. 4652: D. Georgakopoulos, N. Ritter, B. Benatallah, C. Zirpins, G. Feuerlicht, M. Schoenherr, H.R. Motahari-Nezhad (Eds.), Service-Oriented Computing ICSOC 2006. XVI, 201 pages. 2007.

Vol. 4640: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development IV. IX, 191 pages. 2007.

Vol. 4634: H. Riis Nielson, G. Filé (Eds.), Static Analysis. XI, 469 pages. 2007.

Vol. 4620: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development III. IX, 201 pages. 2007.

Vol. 4615: R. de Lemos, C. Gacek, A. Romanovsky (Eds.), Architecting Dependable Systems IV. XIV, 435 pages. 2007.

Vol. 4610: B. Xiao, L.T. Yang, J. Ma, C. Muller-Schloer, Y. Hua (Eds.), Autonomic and Trusted Computing. XVIII, 571 pages. 2007.

Vol. 4609: E. Ernst (Ed.), ECOOP 2007 – Object-Oriented Programming. XIII, 625 pages. 2007.

Vol. 4608: H.W. Schmidt, I. Crnković, G.T. Heineman, J.A. Stafford (Eds.), Component-Based Software Engineering. XII, 283 pages. 2007.

Vol. 4591: J. Davies, J. Gibbons (Eds.), Integrated Formal Methods. IX, 660 pages. 2007.

Vol. 4589: J. Münch, P. Abrahamsson (Eds.), Product-Focused Software Process Improvement. XII, 414 pages. 2007.

Vol. 4574: J. Derrick, J. Vain (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2007. XI, 375 pages. 2007.

Vol. 4556: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part III. XXII, 1020 pages. 2007.

Vol. 4555: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part II. XXII, 1066 pages. 2007.

Vol. 4554: C. Stephanidis (Ed.), Universal Acess in Human Computer Interaction, Part I. XXII, 1054 pages. 2007.

Vol. 4553: J.A. Jacko (Ed.), Human-Computer Interaction, Part IV. XXIV, 1225 pages. 2007.

Vol. 4552: J.A. Jacko (Ed.), Human-Computer Interaction, Part III. XXI, 1038 pages. 2007.

Vol. 4551: J.A. Jacko (Ed.), Human-Computer Interaction, Part II. XXIII, 1253 pages. 2007.

Vol. 4550: J.A. Jacko (Ed.), Human-Computer Interaction, Part I. XXIII, 1240 pages. 2007.

Vol. 4542: P. Sawyer, B. Paech, P. Heymans (Eds.), Requirements Engineering: Foundation for Software Quality. IX, 384 pages. 2007.

Vol. 4536: G. Concas, E. Damiani, M. Scotto, G. Succi (Eds.), Agile Processes in Software Engineering and Extreme Programming. XV, 276 pages. 2007.

Vol. 4530: D.H. Akehurst, R. Vogel, R.F. Paige (Eds.), Model Driven Architecture - Foundations and Applications. X, 219 pages. 2007.

Vol. 4523: Y.-H. Lee, H.-N. Kim, J. Kim, Y.W. Park, L.T. Yang, S.W. Kim (Eds.), Embedded Software and Systems. XIX, 829 pages. 2007.

Vol. 4498: N. Abdennahder, F. Kordon (Eds.), Reliable Software Technologies - Ada-Europe 2007. XII, 247 pages. 2007.

Vol. 4486: M. Bernardo, J. Hillston (Eds.), Formal Methods for Performance Evaluation. VII, 469 pages. 2007.

Vol. 4470: Q. Wang, D. Pfahl, D.M. Raffo (Eds.), Software Process Dynamics and Agility. XI, 346 pages. 2007.

Vol. 4468: M.M. Bonsangue, E.B. Johnsen (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 317 pages. 2007.

Vol. 4467: A.L. Murphy, J. Vitek (Eds.), Coordination Models and Languages. X, 325 pages. 2007.

Vol. 4454: Y. Gurevich, B. Meyer (Eds.), Tests and Proofs. IX, 217 pages. 2007.

Vol. 4444: T. Reps, M. Sagiv, J. Bauer (Eds.), Program Analysis and Compilation, Theory and Practice. X, 361 pages. 2007.