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 978-3-540-85552-1

Lecture Notes in Computer Science

LNCS LNAI LNBI

Cuadrado-Gallego et al. (Eds.)

LNCS 4895

LNCS 4895

Software Process and Product Measurement

Juan J. Cuadrado-Gallego
René Braungarten
Reiner R. Dumke
Alain Abran (Eds.)

Software Process and Product Measurement

**International Conference, IWSM-Mensura 2007
Palma de Mallorca, Spain, November 2007
Revised Papers**

Springer

Lecture Notes in Computer Science 4895

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

Juan J. Cuadrado-Gallego
René Braungarten Reiner R. Dumke
Alain Abran (Eds.)

Software Process and Product Measurement

International Conference, IWSM-Mensura 2007
Palma de Mallorca, Spain, November 5-8, 2007
Revised Papers

Springer

Volume Editors

Juan J. Cuadrado-Gallego
University of Alcalá
Department of Computer Science
Edificio Politécnico, Autovía A2, Km. 31,7, 28805 Alcalá de Henares, Spain
E-mail: jjcg@uah.es

René Braungarten
Reiner R. Dumke
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Institut für Verteilte Systeme
AG Softwaretechnik
Universitätsplatz 2, 39106 Magdeburg, Germany
E-mail: {braungar, dumke}@ivs.cs.uni-magdeburg.de

Alain Abran
Université du Québec
École de technologie supérieure
Département de génie logiciel et des TI
1100, rue Notre-Dame Ouest, Montréal, Québec H3C 1K3, Canada
E-mail: Alain.Abran@etsmtl.ca

Library of Congress Control Number: 2008933374

CR Subject Classification (1998): D.2.8, D.2

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-85552-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-85552-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12320127 06/3180 5 4 3 2 1 0

Preface

Since 1990 the International Workshop on Software Measurement (IWSM) has been celebrated annually in Montréal (Québec), Canada, and different places all over Germany by turns. The Montréal editions were organized by the Software Engineering Research Laboratory (GELOG)[1] of the École de technologie supérieure (ÉTS) at the University of Québec at Montréal (UQAM), which is directed by Professor Alain Abran. The German editions were organized jointly by the Software Measurement Laboratory (SMLAB)[2] of the Otto-von-Guericke-University Magdeburg, Germany, which is directed by Professor Reiner R. Dumke; and the German-speaking user association for software metrics and effort estimation (DASMA e. V.)[3]. Partially, the editions of IWSM were held jointly with the DASMA Software Metrik Kongress (MetriKon).

Organized by an initiative of José Javier Dolado[4] from the University of the Basque Country at San Sebastian and Juan J. Cuadrado-Gallego[5] from the University of Alcalá in Madrid the first edition of the International Conference on Software Measurement (*Mensura*) could be convened in Cádiz, Spain in 2006. Motivated by this success and with the first edition of Mensura finding special approval, the organizers of IWSM and Mensura decided to complement each other and, thus, to organize the next conference edition together. In November 2007, the typical convention month for both conferences, that joint conference was held in Palma de Mallorca, Spain.

This volume is the post-proceedings of the IWSM-Mensura 2007 conference and consists of a set of 16 final papers selected from the conference. Each one of these papers has been thoroughly revised and extended in order to be accepted for this edition. The IWSM-Mensura Steering Committee is very proud to have obtained the approval of Springer to publish the first edition of the joint conference post-proceedings in the prestigious *Lecture Notes in Computer Sciences* (LNCS) series and hope to maintain this collaboration for the future editions of the conference.

February 2007 Juan J. Cuadrado-Gallego

[1] http://www.lrgl.uqam.ca/
[2] http://ivs.cs.uni-magdeburg.de/sw-eng/us/
[3] http://www.dasma.org/
[4] http://www.sc.ehu.es/dolado/
[5] http://www.cc.uah.es/jjcg/

Organization

General Chairs

Reiner R. Dumke	Otto-von-Guericke-University, Magdeburg, Germany
José Miguel Toro	University of Seville, Spain

Program Committee Chairs

Alain Abran	University of Québec, Montréal, Canada
Antonia Mas	University of the Balearic Islands, Palma de Mallorca, Spain

Program Committee

Moteoi Azuma	University of Waseda, Japan
Manfred Bundschuh	DASMA e.V., Germany
David Card	DNV IT Global Services, USA
François Coallier	ÉTS, Montréal, Canada
Juan J. Cuadrado-Gallego	University of Alcalá, Spain
Jean-Marc Desharnais	ÉTS, Montréal, Canada
Ton Dekkers	Shell Information Technology, The Netherlands
José Javier Dolado	University of the Basque Country, Spain
Christof Ebert	vector Consulting, Stuttgart, Germany
Khaled El-Emam	University of Ottawa, Ottawa, Canada
Felix García	University of Castilla-La Mancha, Spain
Elena García-Barriocanal	University of Alcalá, Spain
George Ghinea	Brunel University, UK
Naji Habra	FUNDP, Namur, Belgium
Nadine Hanebutte	University of Idaho, USA
Mark Harman	Kings College, London, UK
Rachel Harrison	Stratton Edge Consulting Ltd., UK
Ali Idri	INSIAS, Morocco
Natalia Juristo	Politechnic University of Madrid, Spain
Adel Khelifi	ALHOSN University, Abu Dhabi, UAE
Pedro Lara	University Europea de Madrid, Spain
Miguel Lopez	University of Namur, Belgium
Mathias Lother	Robert Bosch GmbH, Germany
Hakim Lounis	University of Québec, Montréal, Canada

Fernando Machado	Catholic University of Uruguay, Montevideo, Uruguay
John McGarry	U.S. Army, USA
Roberto Meli	DPO, Italy
Pam Morris	Total Metrics, Australia
John C. Munson	University of Idaho, USA
Olga Ormandjieva	Concordia University, Montréal, Canada
Oscar Pastor	Technical University of Valence, Spain
Mario Piattini	University of Castilla-La Mancha, Spain
Tony Rollo	Software Measurement Services Ltd., UK
Salvador Sánchez	University of Alcalá, Spain
Daniel Rodríguez	University of Alcalá, Spain
Miguel-Ángel Sicilia	University of Alcalá, Spain
Maya Daneva	University of Twente, The Netherlands
Manoranjan Satpathy	General Motors, India
Andreas Schmietendorf	Berlin School of Economics, Germany
Manuel Serrano	University of Castilla-La Mancha, Spain
Marcio Silveira	EDS, Brazil
Harry Sneed	SES, Munich/Budapest, Hungary
Esperança Amengual	Univ. of the Balearic Islands, Palma de Mallorca, Spain
Stanimir Stojanov	University of Plovdiv, Bulgaria
Hannu Toivonen	Nokia, Finland
Claes Wohlin	Blekinge Institute of Technology, Sweden
David Zubrow	Software Engineering Institute, USA
Horst Zuse	Technical University Berlin, Germany
Marcus Ciolkowski	Fraunhofer IESE, Kaiserslautern, Germany
Javier Aroba	University of Huelva, Spain
Dietmar Pfahl	University of Calgary, Canada
Andreas Jedlitschka	Fraunhofer IESE, Kaiserslautern, Germany

Special Sessions/Workshop Chairs

Mercedes Ruiz	University of Cádiz, Spain
Luis Fernández	University Europea de Madrid, Spain
Luigi Buglione	Engineering.it S.p.A., Italy
Luca Santillo	Independent consultant, Italy
Sylvie Trudel	CRIM, Canada

Table of Contents

Implementing Software Measurement Programs in Non Mature Small Settings

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
{Felix.Garcia,Mario.Piattini}@uclm.es

Abstract. Although measurement has been successfully applied in various areas, it has proved to be a complex and difficult undertaking in the field of software and especially in the context of small and medium enterprises (SMEs). Measurement programs in SMEs with a low maturity level should be tailored to their limitations (limited resources, experts, etc.) in order to carry out the measurement initiatives efficiently. In this paper we report the method, principles and practices followed in our experience for defining and implementing a measurement program in the development department of Sistemas Técnicos de Loterías del Estado (STL). We also show the characteristics of this company which guided our approach towards tackling the problem, and the resulting software measurement program. As a result of the application of certain practices in the company, some significant benefits were obtained, which could be replicated in similar environments.

Keywords: Software measurement program, small and medium settings, case study.

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 the one that becomes a good tool [1], this means that it directly contributes to solving a part of the engineering problem at hand and generates value rather than data [2].

Although measurement is applied in various areas, it has proved to be a complex and difficult undertaking in the field of software, especially within the context of small and medium enterprises [3].

Small organizational units are just as likely to be confronted with demands for credible evidence about their ability to deliver quality products on time and on budget as large, multinational organizations. Similarly, managers in small settings are equally or even more likely than their counterparts in larger organizational units to have to make well-founded business decisions about process improvement and technology

J.J. Cuadrado-Gallego et al. (Eds.): IWSM-Mensura 2007, LNCS 4895, pp. 154–167, 2008.

adoption, and must have the wisdom of taking new business opportunities. Therefore, implementing serious measurement programs is even more important in small organizational settings [4].

Unfortunately, small and medium enterprises have some common characteristics which eventually become obstacles in establishing software measurement initiatives Some of these are: limited resources and training, poor software measurement knowledge, restricted cash flow, a restricted mentality as regards software measurement, etc. [5-7].

This paper aims to report our experience in setting up a measurement program in the software development department of a medium-sized company with a low software measurement maturity level. We show the methodological approach and implementation strategy chosen which fits the characteristics of the company, and we briefly present the resulting measurement program. Finally it is exposed the benefits of using these practices in similar settings.

The paper is organized as follows: Section 2 sets this work in context by showing some measurement program implementation experiences. Section 3 specifies the characteristics of the company, the measurement program definition framework chosen, the organizational approach, the goals of the measurement program, the resulting measurement program and some practices for implementing the measurement program are also expounded. Section 4 shows the conclusions and lessons learned from the experience and Section 5 shows further research.

2 Related Work

In this section we provide an overview of the implementation of measurement programs in software technological companies and we address some studies which identifies good practices for implementing software measurement programs successfully.

Daskalantonakis presented a company-wide software measurement initiative which was implemented to fulfill organizational quality policy [8]. In this work some cultural issues were identified and an organizational approach through which to carry out measurement programs was defined. Some benefits were an improvement in terms of defect density and customer satisfaction. In [9] a measurement program at Daniro J-Technologies in the context of software development projects is described where a measurement program definition primary focussed on understanding productivity and defects for the organization was proposed. However the measurement program shown is only in its initial (i.e. planning and definition) stage and the implementation and validation phase is yet to be done.. In [10] Kilpi shows how Nokia organizationally carries out its measurement activities. It explains the differences as regards GQM method which basically are: the use of quality metrics library instead of defining a new set of metrics for each project, the use of a Quality Plan and Metrics Guidelines instead of a GQM plan, the automation of data collection and reporting, and part-time instead of full-time people involved.

With regard to the implementation of measurement programs in small and medium settings, in [11] MacDonell et al. present a measurement program definition whose purpose was to define a metrics framework which serve to determine the development effort in organizations that develops multimedia systems. Lavazza et al. presented a measurement program experience which took place in the Banca Conboto where

GQM was used and adapted due to the operational, budget and time constrains. The resulting measurement program gave valuable information about the quality of the maintenance process [12].

These experiences gave us a valuable insight into the matter but there is very little information about experiences related to small and medium settings in defining and implementing measurement programs.

1. **Incremental implementation**
2. **Well-planned metrics framework**
3. **Use of existing metrics materials**
4. **Involvement of developers during implementation**
5. **Measurement process transparent to developers**
6. **Usefulness of metrics data**
7. **feedback to developers**
8. **Ensure that data is seen to have integrity**
9. **Measurement data is used and seen to be used**
10. **Commitment from project managers secured**
11. **Use automated data collection tools**
12. **Constantly improving the measurement program.**
13. **Internal metrics champions used to manage the program**
14. **Use of external metrics gurus**
15. **Provision of training from practitioners**

Fig. 1. Hall and Fenton measurement success factors [13]

In the frame of measurement programs good practices, Gopal et al. [14] identified and proved some success factors by analyzing its effects on the measurement programs success. 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 in two sets: organizational and technical factors. Daskalantonakis also stated some good practices form its experience in Motorola [8, 15], and Fenton and Hall [13] identified from the experience fifteen success factors for implementing software measurement programs as shown in figure 1. However none of these studies show good practices to follow especially when the measurement program is implemented in small and medium software companies and its characteristics are typical of these environments: low measurement maturity level, poor measurement knowledge, measurement not integrated in the culture, limited resources and budget, etc. In figure 1 the success factors detected by Fenton and Hall that do not easily fit for SMEs are underlined.

3 Development of the Measurement Programs in STL

In this section we describe the company in which the case study was conducted, the methodological and organizational approach set out, the principles followed; the process improvement goals supported by the measurement program and a summary of the resulting measurement program.

3.1 Description of the Company: Measurement Programs Constraints

Sistemas Técnicos de Loterías del Estado (STL) is a company which was created by the Spanish government and which provides the operations and IT development services for the national lottery. Software measurement initiatives have been encouraged for many years by the software development and maintenance department in this company, which is formed of 39 people. Unfortunately the measurement process which was defined was not accurate enough and had not been properly established throughout the department. Some of the outstanding problems were the scarce amount of resources available for this task. However the need to measure continued and the need of defining accurate measurement programs re-emerged

The development and maintenance department is in charge of developing and maintaining the bet on-line systems including the software of the related terminals and payment channels; and the invoicing and informative systems. They support 33 products of which 21 are critical and 18 of them are on-line systems. Most of the code is written in FORTRAN, C, C++, some Java and PI/SQL. The core critical product amounts to 2000 KLOC.

The development projects where normally carried out by less than 12 people. It usually takes five or six months long and rarely above 15 months.

Some other characteristics of the company related to measurement are as follows:

- C1: The resources were limited and therefore we could not spend too much effort on defining and implementing the measurement program.
- C2: Some project managers were reluctant to use the measurement initiative
- C3: Measurement was not established in the company culture.
- C4: Measurement knowledge was quite limited throughout the company.
- C5: The software measures collected were few, there was no established measurement process and therefore the measurement maturity was quite low.

3.2 Measurement Program Definition Framework

MIS-PyME (Marco metodológico para la definición de Indicadores de Software orientado a PyME) is a methodological framework focussed on defining measurement programs based on software indicators in small and medium settings.The main characteristic of MIS-PyME, among the outstanding software measurement models, is that it fully covers the requirements of a measurement program model suited to small and medium settings with a low measurement maturity level. 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 measurement maturity model (MIS-PyME Measurement Maturity Model). MIS-PyME Methodology is based on GQ(I)M [16, 17], but it is designed to define basic indicators which are commonly used and required in most small and medium software development settings. These indicators are adapted to the measurement maturity of the setting. Like GQ(I)M, it is a top-down methodology since it develops the measurement program with the goal in mind but restricts the domain of those goals solely to software process improvement and may be conditioned by the MIS-PyME measurement goals table and the indicator templates provided. This methodology also makes use of a database of measurement program definitions related to software process improvement. MIS-PyME methodology also stresses and helps the integration of the measurement program in the software processes. The principles supported by MIS-PyME are the following:

- The "Reuse and project-specific tailoring" principle, stated by Basili and Rombach [18, 19]. This principle indicates that measurement planning should reuse models and metrics which have been defined for the whole organization. However these models might be tailored to the project or product specific characteristics. MIS-PyME encourages companies in defining measurement programs which will be valid and useful in almost all products or projects, by doing so it makes measurement programs establishment easier in the development unit and make possible the cross-projects and products control.
- Related to the above principle, we base the measurement program on the indicators since they give high level information and can be applied to most of projects, products, processes depending on the nature of the indicator. The indicators allow making cross-analysis and they are an easy reusable unit. Measures and derived measures used in the indicator may be specific for certain project or product, but when then measurement program is to be applied to a new project (or product) these measures or derived measurement of the measurement program will be adapted to the new project.
- To define and implement measurement programs which are adapted to the measurement maturity of the setting. Companies may work in defining and implementing measurement programs that they are able to successfully implement and not to try achieving the best measure when there are several obstacles that make impossible a successful implementation. These obstacles may be related to development, management or quality capability, support tools, etc.
- To define measurement programs with the sole purpose of giving support to the software process improvement initiatives as are also followed in [8]. If the measurement process is still not well integrated in the organization culture and it is not mature enough, it is costly to establish measurement programs. Small and medium settings could not achieve to successfully implement measurement programs which come from just any business goal; at least it will be costly to achieve it. In addition, as stated in the previous point, measurement maturity can not overcome processes maturity. Both areas are extremely connected and need to improve together. Since measurement is not an aim but a way, MIS-PyME leaves the responsibility for seeking a goal to process improvement initiatives.

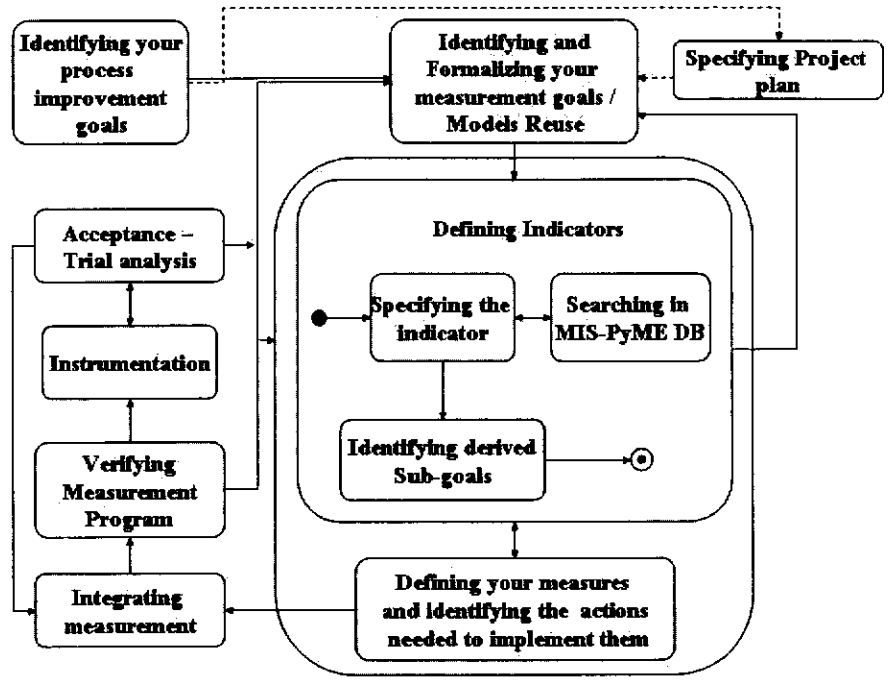


Fig. 2. MIS-PyME Methodology

Figure 2 succinctly outlines this methodology, which is composed of the following activities.

1. **Identifying your Process Improvement Goals:** The process improvement goals that you want to carry out aided by software measurement are defined and the related entities that will help achieve this goal are identified.
2. **Formalizing Measurement Goals and Check if a Measurement Model is Reused:** Measurement goals are specified. After that, the object of study, the purpose, the environment and the measurement constraints are defined. MIS-PyME measurement goal table helps the user to identify the measurement goals that will support the process improvement goals. In this step whether there is any measurement goal in the already measurement process of the organization with the same description is also verified. If affirmative, its corresponding indicator template is checked to understand if it fulfils the needs to support the process improvement goal.

⇒ **Specifying Project Plan:** A small project plan should be defined. It should only contain: A description of the measurement program, the people involved and their roles, the calendar and a specification of the acceptance phase (trial analysis and pilot project). It may be included in a process improvement project if exists.

3. **Defining Indicators:** Indicators required to implement measurement goals are defined. This is a three step status.

3.1. *Specifying the indicators:* If the measurement goals were in the MIS-PYME measurement goals table, the measurement analyst might take a look at the recommendations, restrictions, questions, etc. according to the MIS-PYME indicator template established for that goal.
3.2. *Searching in MIS-PyME database*: When defining an indicator, measurement analysts may check for any examples in the database related to the MIS-PYME indicator template required for the desired indicator. If a suitable one is found, they can directly and effortlessly adapt the indicator proposed to the measurement program being defined.
3.3. *Identifying sub-goals derived*: Any of the questions posed or the inputs recommended in the MIS-PYME indicator template table may lead to another measurement goal. We call these measurement-derived goals, which may also have their corresponding measurement goal in the table for MIS-PYME measurement goals and their corresponding MIS-PYME indicator templates. Step 2 and 3 may then be repeated until all measurement-derived goals and their relevant indicators have been defined.

4. **Defining your Measures and Identifying the Actions Needed to Implement them:** The measures that have to be collected are identified in detail and defined in the checklists. It is defined which data is to be included/excluded from the measured values, as well as how the data will be collected. The ability of the organization to obtain the measures is analyzed, and the way in which they could be collected is established. If it is not possible to collect the desired data, the indicator specification may be modified based on this information. Maybe it should be defined several ways to get certain measure since this measures can be found in different context (different program languages, different procedures involved, different systems, etc).
5. **Integrating the Measurement Program:** Integrating the measurement activities into previous measurement processes and into other software processes is the aim of this step. MIS-PYME provides guidance in order to integrate the indicators and measurement sub-processes in the development, quality and management sub-processes of the company.
6. **Verifying the Measurement Program:** The measurement process resulting from the process is verified by reviewers and modified if required.
7. **Instrumentation**: Tools which support measurement process are developed or tailored.
8. **Acceptance of the Measurement Program:** The measurement program is used in a trial analysis or in a pilot project.

3.3 Settings

The roles necessary to carry out the measurement program and suggested by MIS-PyME model were as follows: the measurement analyst who had some knowledge about software measurement but was not too experienced. The usual work of this role is consulting, coordinating projects, defining requirements and testing. The second role was played by the manager (the director of the development department) who supports the measurement program initiative and has an in-depth knowledge of the software processes and process improvement needs. The third role was the reviewer.

The group of people who played this role was formed of the five project managers (most of them are also the sub-department managers) and some key developers.

This approach was designed in order not to disturb project managers in their usual tasks. The top manager knows the main measurement needs and he is capable of determining them. The measurement analyst will work only with the top manager during the entire definition process phase. At the "verifying measurement process" phase the measurement analyst works with the reviewers.

3.4 Software Measurement Program in STL

The highest priority software Process Improvement Goals (PIG) were the following:

- PIG 1: Improving project monitoring and control. We particularly wished to improve the monitoring of the project's progress in comparison to the plan and to understand and manage any deviations from the plan at the project's closure. Besides regarding the process, we wanted to monitor the conformance with the test phases.
- PIG 2: Improving the development service and product quality. This goal focused on understanding, monitoring and evaluating the development service and product quality exploited.

Measurement Program Resulted. For the first goal (PIG1) three sub-goals through which to improve the software process are identified: PIG 1.1- Improving project progress monitoring in comparison to the plan; PIG 1.2 - Understanding and managing any deviations from the plan at the project's closure. PIG 1.3: Evaluating the conformance of projects with the test phases.

For the first sub-goal we emphasized the indicators as follows: the first indicator which gives the information about the effort spent against what it was estimated (IND-PRJ-EffortConformance). The next indicator shows the progress of the coding phase by showing the number of requirements coded against the total, and the planned schedule (IND-PRJ-ProgCod). Another indicator shows the progress of the verification phase by showing the number of open incidences and its severity and the planned schedule (IND-PRJ-ProgVerif). For the acceptance phase progress, an indicator shows the number of requirements verified with regard to the total of requirements, the number of open incidences and its severity, the planned schedule, and the defect density (IND-PRJ-ProgAccept).

For the second sub-goal four indicators were defined in order to measure the deviation at the protect closure: deviation regarding the effort (Ind-prj-InexacEffort), the size of the software (Ind-prj-InexacSize), the duration of the project (Ind-prj-InexacDuration,) and the total developing cost (ind-prj-InexacCost).

Besides, for each of the indicators related to the PIG 1.2 sub-goal, an indicator was defined in order to perform global and cross-project analyses. All the projects were thus globally analyzed during a six-month period.

For the third sub-goal PIG 1.3 only two indicators were defined, one called IND-PRJ-TestConformance that measured the total failures detected during the integration and acceptance test and the relation of failures detected in both phases and the same for the sever failures type. For this goal an indicator was defined called IND-PRJORG-TestConformance that made a cross-project balance taking into account the type of projects.

From PIG 2, two sub-goals were identified: PIG.2.1 - understanding, monitoring and evaluating the development service provided and PIG.2.2 –understanding and monitoring the quality of the product exploited.

For PIG.2.1 a first level indicator called Ind-prj-QualityDev was defined. This indicator was simply formed from two other indicators, Ind-prj-FiabImpl and Ind-prj-InexacDuration. Ind-prj-FiabImpl aimed to evaluate the reliability of the software developed and released under a project. Ind-prj-InexacDuration aimed to evaluate the deviation in time as regards what was planned. There is also an indicator which globally measures the development process as regards the development service quality. This indicator is called Ind-prjorg-QualityDev.

PIG.2.2 was represented by three indicators which monitor the quality of the products provided by our clients: Ind-prodorg-FiabCrit, Ind-prodorg-FiabSopot, Ind-prodorg-FiabInf. These indicators show the density of failures in production for each product during the period analyzed. Each indicator includes the products related to its product classification: critical, support, informative.

In summary, the resulting measurement program defined 31 indicators, 29 measures, 6 estimations and 9 concept criteria. This measurement program was designed to be used by the whole department. The measurement program was created in two phases which lasted almost three months. PIG.1.1 was tackled in a second phase after PIG.1.2, PIG.2.1 and PIG.2.2 were already implemented and in use.

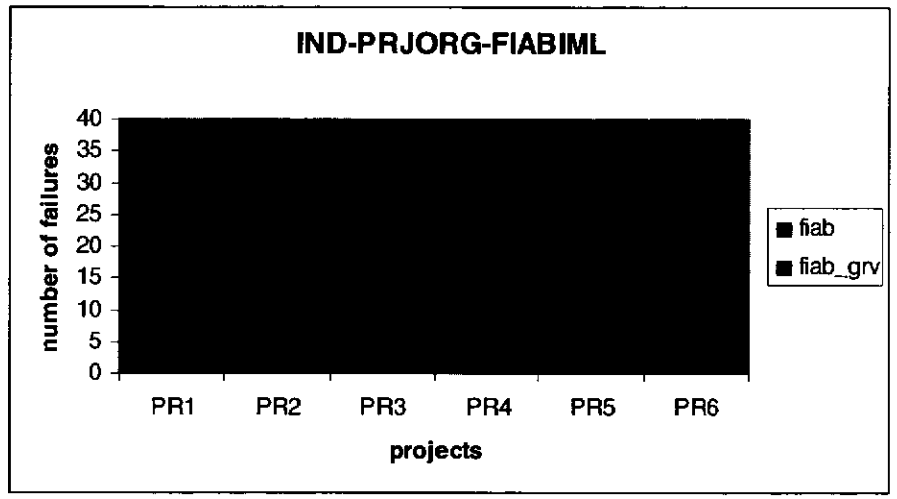
Software Measurement Process. The measurement program mentioned above formed the initial measurement process defined for the development department in STL. Three different sub-processes were identified: the project management measurement sub-process, the process management measurement sub-process and the product management measurement sub-process. As far as the integration of the measurement process is concerned, the analysis activities related to the indicators were included in the development process and its related templates (e.g. close of project report, project progress monitoring report, etc.).

3.5 Software Measurement Program Implementation

In this section we briefly explain some important issues related to the implementation of the measurement program and we finally show an example.

Instrumentation. It was intended to avoid data that could not be obtained from the tools already in use. The software measurement program collects the information from five development and configuration management tools: the Microsoft Project Manager, ActiTime (a free tool with which to register the effort dedicated to each task), Remedy (an incident management tool) and IRQA (a requirement management tool). Only Remedy and IRQA needed to be tailored in order to automatically launch the queries to the database and to create reports containing the required information and to enable requirement trace. We briefly studied some software measurement tools such as MetricFlame, MetricCenter, ProjectConsole, etc., but we preferred to use simpler tools than complicated ones since the latter may make the understanding of the basic measurement process steps more difficult; and, after weighing up the difference between the effort needed to study a complex tool, adapt it and train people, and the benefits provided, we chose to develop three simple Spread Excel Sheets to give support to each of the sub-processes.

Verification. The measurement program was first reviewed in two sessions. In the first session the measurement analyst gave an overview of the measurement program defined and suggestions were only received in the second session after the measurement program was analyzed. Afterwards the measurement program was tested in the department. The results obtained from the measurement programs during the verification phase did not become known outside the reviewers group. For the indicators related to the products, an analysis was done based on the six-month product activity. For the indicators related to PIG 1.2, PIG 1.3 and 2.1, an analysis was done using the projects which were finished between the previous six moths. The mentioned



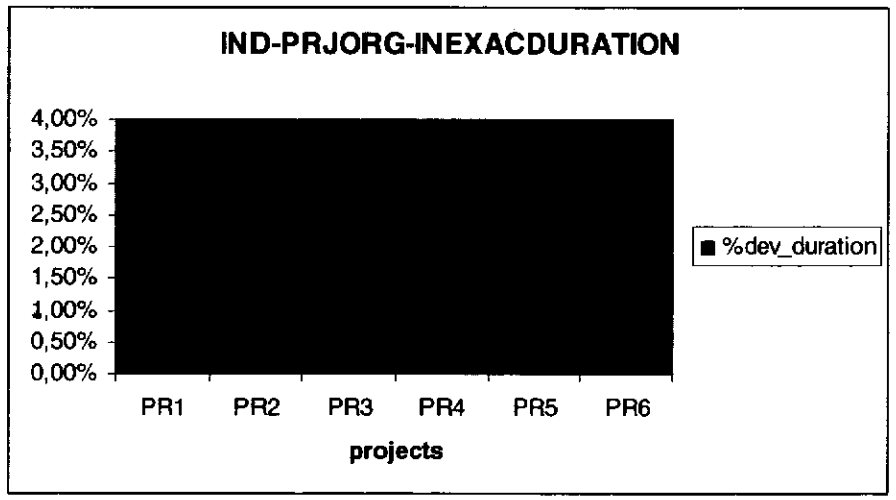


Fig. 2. Development service quality indicator (Ind-prjorg-QualityDev) which contains two input indicators: Ind-prjorg-InexacDuration and Ind- prjorg -FiabImpl

indicators were implemented quite fast but PIG1.1 related to the monitoring of the project and therefore its verification took the duration of it. This part of the measurement program is still at the verification phase.

Final Results and Example. The measurement program defined in STL finally overcame the goals stated in section 4 (pending the verification phase of PIG 1.1 sub goal). In Figure 2 an example of an indicator (Ind-prjorg-QualityDev) after its first analysis is shown. This indicator not only evaluates the development service quality focused on reliability and project delay for each type of projects, but additionally this indicator made us to ascertain whether the on-time release of the software product had a negative impact on software reliability.

4 Conclusions and Lessons Learned

In this paper we have reported an experience in defining and implementing a measurement program in the development department of Sistemas Técnicos de Loterías del Estado (STL) whose aim was to consolidate a useful and simple software measurement process.

The interest of this paper is to show the strategy followed to develop and implement the measurement program, taking into account the characteristics of the company and the good practices identified which could be applied in similar environments.

The characteristics of the company were as follows:

- People involved in the measurement program, including the measurement analyst are from inside the company and not too expertise in the field.
- Poor measurement culture in the company, poor knowledge and therefore poor measurement maturity.
- Some project managers and developers reluctant to use measurement.
- The measurement program should be established in a small or medium software development company or unit with less than 50 people approximately.
- People involved in project, in which the measurement program is established, may be normally less than 12 people.
- The duration of the project should not exceed 15 months and should normally take 5 or 6 months.

Regarding the above characteristics we suggest to follow the practices as follows:

1. The definition of Measurement Programs should not focus on defining measurement programs for certain projects or products, since it will be costly, difficult to handle and of little worth for future developments. The organization should focus on reusing its existing measurement models.
2. Measurement programs should focus on supporting software process improvement goals instead of on business goals. Low measurement maturity level settings cannot afford measurement programs from just any business goal if the aim is to define successful measurement programs effortlessly, accurately and consistently.
3. Measurement Programs definition should be adapted to the measurement maturity of the company. As an example a software measurement low maturity organization cannot expect empirical prediction results. We used MIS-PyME indicator templates to guide us in this issue.
4. Using common models for software project measurement related may be of use in order to detect the measurement goals which may support software process improvement goals. We made use of the measurement goal table provided by MIS-PyME and we found them useful.
5. With regard to the organizational approach (supported by MIS-PyME) and the roles involved, the benefits detected by our experience were as follows:

- The extra work that the measurement program definition implies for the project managers is reduced by using this approach. The aim is to seek common useful measurement goals that support software process improvement. The top manager is able to support the measurement analyst in defining the first approach of the measurement program.
- Project managers make more objective suggestions and effective modifications about the measurement program definition if they review the first approach. As some project managers are reluctant to use these initiatives, the first approach of the measurement program can show them its usefulness and motivate them to review it.

6. We discovered that the way in which the revisions were made by means of two preliminary verification sessions, and the pilot test, was quite useful: By doing so people give better suggestions, analyze better the problem and the usefulness of the measurement program, get more involved in the measurement program, and they easier agree with it.
7. The means of documenting software measurement programs and integrating them in the rest of the software process is essential for the acceptance and use of the measurement process. Our measurement process is documented in a way which clearly answers the questions: "what aspects of the projects, products, resources and process are measured?", "what for?" and "what do I have to measure at each moment of the software development and maintenance process?" Moreover, measurement activities are included in the software development and maintenance model, and in the output report templates involved. Our measurement process was documented in a ".pdf" format but we will, however, modify it to make it accessible via WEB as this is the format of the software development and maintenance model.
8. Excel Spread Sheets or familiar databases are recommended for this type of settings. First because before having powerful tools it is better to understand the process and to control the essential activities. Furthermore, the benefits that the tool provides may not make up the cost of evaluating the tool and training people. Once the company is mature enough, other more powerful tools can be acquired.
9. We also recommend taking advantage of the data provided by already existing development tools and attempting not to collect ambiguous or difficult (as regards data collection) data.

Some of the practices suggested above have been already identified as success factors for implementing measurement programs by some authors: the first recommendation was indicated by Basili and Rombach [19], the second and seventh were identified by Daskalantonakis [8, 15] and the ninth was specified by Fenton and Hall

[13]. We agree with these practices too and we propose others that made the definition and implementation of our measurement program easier.

Since the information related to measurement programs definition and implementation in small and medium settings with similar characteristics is scarce and it covers a wide industry sector, our work is worthy of consideration.

However most of the practices suggested in this study may not be valid when the measurement program is not aimed to be implemented in a software development small and medium company or single unit, when the measurement maturity level is high and the software measurement is integrated in its culture or when the budget and resources assigned are rather good.

5 Further Research

Our future research will focus upon guiding small and medium settings towards increasing their maturity with regard to software measurement and also upon observing the benefits of measurement in the development process and its consequences in businesses in this kind of environments. By doing so, we shall continue with the refinement of MIS-PyME methodology framework.

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 (2003)
4. Goldenson, D., Rout, T., Tuffley, A.: Measuring Performance Results in Small Settings: How do you do it and what matters most? In: Proceedings of the First International Research Workshop for Process Improvement in Small Settings (2005)
5. Briand, L.C., Differding, C.M., Rombach, H.D.: Practical Guidelines for Measurement-Based Process Improvement. Software Process - Improvement and Practice 2(4), 253–280 (1996)
6. Mondragon, O.A.: Addressing Infrastructure Issues in Very Small Settings. In: Proceedings of the First International Research Workshop for Process Improvement in Small Settings (2005)
7. Emam, K.E.: A Multi-Method Evaluation of the Practices of Small Software Projects. In: Proceedings of the First International Research Workshop for Process Improvement in Small Settings (2005)

8. Daskalantonakis, M.K.: A Practical View of Software Measurement and Implementation Experiences Within Motorola. IEEE Transactions on Software Engineering 18(11), 998–1010 (1992)
9. Kettelerij, R.: Designing A Measurement Programme For Software Development Projects, in Daniro System Integration and Development B.V, May 2006. University of Amsterdam, Amsterdam (2006)
10. Kilpi, T.: Implementing Software Metrics Program at Nokia. IEEE software 18(6), 72–76 (2001)
11. MacDonell, S.G., Fletcher, T.: Metric Selection for Effort Assessment in Multimedia Systems Development. In: proceedings of the Fifth International Software Metrics Symposium, Bethesda, MD, USA (1998)
12. Lavazza, L., Mauri, M.: Software Process Measurement in Real World: Dealing with Operating Constraints. In: Workshop on Software Process Simulation and Modeling (2006)
13. Hall, T., Fenton, N.: Implementing Effective Software Metrics Programs. IEEE software 14(2), 55–65 (1997)
14. Gopal, A., et al.: Measurement Programs in Software Development: Determinants of Success. IEEE Transactions on Software Engineering 28(9), 863–875 (2002)
15. Daskalantonakis, M.K., Yacobellis, R.H., Basili, V.R.: A Method for Assessing Software Measurement Technology. Quality Engineering, 27–40 (1990)
16. Park, R.E., Goethert, W.B., Florac, W.A.: Goal-Driven Software Measurement-A Guidebook. Carnegie Mellon University Pittsburgh, Software Engineering Institute (1996)
17. Goethert, W., Siviy, J.: Applications of the Indicator Template for Measurement and Analysis. In: Software Engineering Measurement and Analysis Initiative (September 2004)
18. Basili, V.R., Rombach, D.H.: The Tame Project: Towards Improvement-Oriented Software Environments. IEEE Transactions of Software Engineering 14(6), 758–773 (1988)
19. Basili, V.R., Rombach, H.D.: Support for comprehensive reuse. IEEE Software Engineering Journal, 758–773 (1988)

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

Lecture Notes in Computer Science

Sublibrary 2: Programming and Software Engineering

For information about Vols. 1– 4554
please contact your bookseller or Springer

Vol. 5153: A. Rausch, R. Reussner, R. Mirandola, F. Plášil (Eds.), The Common Component Modeling Example. VIII, 460 pages. 2008.

Vol. 5142: J. Vitek (Ed.), ECOOP 2008 – Object-Oriented Programming. XIII, 694 pages. 2008.

Vol. 5140: J. Meseguer, G. Roşu (Eds.), Algebraic Methodology and Software Technology. XIII, 432 pages. 2008.

Vol. 5136: T.C.N. Graham, P. Palanque (Eds.), Interactive Systems
Design, Specification, and Verification. IX, 311 pages. 2008.

Vol. 5135: R. de Lemos, F. Di Giandomenico, C. Gacek, H. Muccini, M. Vieira (Eds.), Architecting Dependable Systems V. XIV, 343 pages. 2008.

Vol. 5119: S. Kounev, I. Gorton, K. Sachs (Eds.), Performance Evaluation: Metrics, Models and Benchmarks. X, 323 pages. 2008.

Vol. 5111: Q. Chen, C. Zhang, S. Zhang, Secure Transaction Protocol Analysis. XI, 234 pages. 2008.

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

Vol. 5091: B.P. Woolf, E. Aïmeur, R. Nkambou, S. Lajoie (Eds.), Intelligent Tutoring Systems. XXI, 832 pages. 2008.

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

Vol. 5082: B. Meyer, J.R. Nawrocki, B. Meyer (Eds.), Balancing Agility and Formalism in Software Engineering. XI, 305 pages. 2008.

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

Vol. 5063: A. Vallecillo, J. Gray, A. Pierantonio (Eds.), Theory and Practice of Model Transformations. XII, 261 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, A. Ulrich, T. Hasegawa (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. Maibaum, K. Sere (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. 5002: H. Giese (Ed.), Models in Software Engineering. X, 322 pages. 2008.

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

Vol. 4970: M. Nagl, W. Marquardt (Eds.), Collaborative and Distributed Chemical Engineering. XII, 851 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. 4922: M. Broy, I.H. Krüger, M. Meisinger (Eds.), Model-Driven Development of Reliable Automotive Services. XVIII, 183 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. 4895: J.J. Cuadrado-Gallego, R. Braungarten, R.R. Dumke, A. Abran (Eds.), Software Process and Product Measurement. X, 203 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.