ISSN 0302-9743

Lecture Notes in Computer Science
LNCS LNAI LNBI

❯ springer.com

Münch
Abrahamsson (Eds.)

LNCS 4589

LNCS
4589

Product-Focused
Software Process Improvement

PROFES
2007

Jürgen Münch
Pekka Abrahamsson (Eds.)

Product-Focused Software Process Improvement

8th International Conference, PROFES 2007
Riga, Latvia, July 2007
Proceedings

Springer

[illegible]
[illegible]

Jürgen Münch   Pekka Abrahamsson (Eds.)

Product-Focused Software Process Improvement

8th International Conference, PROFES 2007
Riga, Latvia, July 2-4, 2007
Proceedings

Springer

Volume Editors

Jürgen Münch
Fraunhofer Institute for Experimental Software Engineering
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
E-mail: juergen.muench@iese.fraunhofer.de

Pekka Abrahamsson
VTT Electronics
Kaitovayla 1, 90570 Oulu, Finland
E-mail: pekka.abrahamsson@vtt.fi

Library of Congress Control Number: 2007929634

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	3-540-73459-7 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-73459-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12086863 06/3180 5 4 3 2 1 0

Preface

The Eight International Conference on Product-Focused Software Process Improvement (PROFES 2007) brought together researchers and industrial practitioners to report new research results and exchange experiences and findings in the area of process and product improvement. The focus of the conference is on understanding, learning, evaluating, and improving the relationships between process improvement activities (such as the deployment of innovative defect detection processes) and their effects in products (such as improved product reliability and safety). Consequently, major topics of the conference include the evaluation of existing software process improvement (SPI) approaches in different contexts, the presentation of new or modified SPI approaches, and the relation between SPI and new development techniques or emerging application domains.

This year's conference theme focused on global software development. More and more products are being developed in distributed, global development environments with many customer–supplier relations in the value chain. Outsourcing, off-shoring, near-shoring, and even in-sourcing aggravate this trend further. Supporting such distributed development requires well-understood and accurately implemented development process interfaces, process synchronization, and an efficient process evolution mechanisms. Overcoming cultural barriers and implementing efficient communication channels are some of the key challenges. It is clear that process improvement approaches also need to consider these new development contexts.

A second key focus of PROFES 2007 was on agile software development. Market dynamics require organizations to adapt to changes of the development environment and to enforce innovations better and faster. This often results in process changes that impose risk challenges for SPI approaches. Advanced SPI is required to support the assessment of the impact of process changes such as the introduction of agile methods. Due to the fact that software development processes are human-based and depend heavily on the development context, process changes and their resulting effects should be considered carefully. We consider the development context to include at least the domain-specific characteristics, the workforce capabilities, and the level of work distribution.

The technical program was selected by a committee of leading experts in software process modeling and SPI research. This year, 56 papers from 21 nations were submitted, with each paper receiving at least three reviews. The Program Committee met in Riga for one full day in February 2007. The Program Committee finally selected 30 technical full papers. The topics indicate that SPI remains a vibrant research discipline of high interest for industry. Emerging technologies and application domains, a paradigm shift to global software and system engineering in many domains, and the need for better decision support for SPI are reflected in these papers. The technical program consisted of the tracks global software development, software process improvement, software process modeling and evolution, industrial experiences, agile software development, software measurement, simulation and decision support, and processes and methods. We were proud to have four distinguished keynote speakers,

External Reviewers

Table of Contents

Keynote Addresses

Software Development and Globalization (Abstract) 1
H. Dieter Rombach

Software Development Globalization from the Baltic Perspective (Abstract) 2
Guntis Urtāns

Experiences in Applying Agile Software Development in F-Secure (Abstract) 3
Jari Still

People Side of IT Globalization (Abstract) 4
Carol Dekkers

Global Software Development

An Industrial Survey of Software Outsourcing in China 5
Jianqiang Ma, Jingyue Li, Weibing Chen, Reidar Conradi, Junzhong Ji, and Chunnian Liu

Understanding Lacking Trust in Global Software Teams: A Multi-Case Study 20
Nils Brede Moe and Darja Šmite

Utilization of a Set of Software Engineering Roles for a Multinational Organization 35
Claude Y. Laporte, Mikel Doucet, Pierre Bourque, and Youssef Belkébir

Software Process Improvement

Software Verification Process Improvement Proposal Using Six Sigma ... 51
Tihana Galinac and Željka Car

Software Development Improvement with SFIM 65
René Krikhaar and Martin Mermans

SPI-KM - Lessons Learned from Applying a Software Process Improvement Strategy Supported by Knowledge Management 81
Gleison Santos, Mariano Montoni, Sávio Figueiredo, and Ana Regina Rocha

Organisational Readiness and Software Process Improvement 96
Mahmood Niazi, David Wilson, and Didar Zowghi

Software Process Improvement Through Teamwork Management 108
Esperança Amengual and Antònia Mas

De-motivators of Software Process Improvement: An Analysis of Vietnamese Practitioners' Views 118
Mahmood Niazi and Muhammad Ali Babar

Software Process Modeling and Evolution

Defining Software Processes Through Process Workshops: A Multicase Study 132
Finn Olav Bjørnson, Tor Stålhane, Nils Brede Moe, and Torgeir Dingsøyr

Improving an Industrial Reference Process by Information Flow Analysis: A Case Study 147
Kai Stapel, Kurt Schneider, Daniel Lübke, and Thomas Flohr

Connecting the Rationale for Changes to the Evolution of a Process 160
Alexis Ocampo and Martin Soto

Industrial Experiences

Use of Non-IT Testers in Software Development...................... 175
Vineta Arnicane

Requirements Management Practices as Patterns for Distributed Product Management... 188
Antti Välimäki and Jukka Kääriäinen

SPI Consulting in a Level 1 Company: An Experience Report 201
Tomas Schweigert and Michael Philipp

Agile Software Development

On the Effects of Pair Programming on Thoroughness and Fault-Finding Effectiveness of Unit Tests 207
Lech Madeyski

An Agile Toolkit to Support Agent-Oriented and Service-Oriented Computing Mechanisms 222
Asif Qumer and Brian Henderson-Sellers

Achieving Success in Supply Chain Management Software by Agility ... 237
Deepti Mishra and Alok Mishra

Software Measurement

Software Measurement Programs in SMEs – Defining Software Indicators: A Methodological Framework 247
María Díaz-Ley, Félix García, and Mario Piattini

Smart Technologies in Software Life Cycle.......................... 262
Zane Bičevska and Jānis Bičevskis

Convertibility Between IFPUG and COSMIC Functional Size Measurements 273
Juan Jose Cuadrado-Gallego, Daniel Rodríguez, Fernando Machado, and Alain Abran

A Framework for Measuring and Evaluating Program Source Code Quality 284
Hironori Washizaki, Rieko Namiki, Tomoyuki Fukuoka, Yoko Harada, and Hiroyuki Watanabe

Software Fault Prediction with Object-Oriented Metrics Based Artificial Immune Recognition System 300
Cagatay Catal and Banu Diri

Simulation and Decision Support

Operational Planning, Re-planning and Risk Analysis for Software Releases 315
Ahmed Al-Emran and Dietmar Pfahl

Project Cost Overrun Simulation in Software Product Line Development 330
Makoto Nonaka, Liming Zhu, Muhammad Ali Babar, and Mark Staples

E-Service Architecture Selection Based on Multi-criteria Optimization.. 345
Edzus Zeiris and Maris Ziema

Processes and Methods

A Component-Based Process for Developing Automotive ECU Software 358
Jin Sun Her, Si Won Choi, Du Wan Cheun, Jeong Seop Bae, and Soo Dong Kim

A Systematic Approach to Service-Oriented Analysis and Design....... 374
Soo Ho Chang and Soo Dong Kim

Improving the Problem Management Process from Knowledge Management Perspective ... 389
Marko Jäntti, Aki Miettinen, Niko Pylkkänen, and Tommi Kainulainen

Workshop

Experience on Applying Quantitative and Qualitative Empiricism to Software Engineering (Workshop Description) ... 402
Marcus Ciolkowski and Andreas Jedlitschka

Tutorials

Using Metrics to Improve Software Testing (Tutorial Description) ... 405
Alfred Sorkowitz

Increase ICT Project Success with Concrete Scope Management (Tutorial Description) ... 407
Carol Dekkers and Pekka Forselius

Agile Software Development: Theoretical and Practical Outlook (Tutorial Description) ... 410
Pekka Abrahamsson and Jari Still

Author Index ... 413

Software Development and Globalization

H. Dieter Rombach

Chairman, ICT Group, Fraunhofer Gesellschaft e.V.,
Executive Director, Fraunhofer IESE, Kaiserslautern,
Software Engineering Chair, CS Dept., University of Kaiserslautern

Developing software across borders has become an emerging area of software engineering. It is one of the important competitive advantages in today's industry. However, the increased globalization of software development creates many challenges brought by distribution of software life cycle activities among teams separated by various boundaries, such as contextual, organizational, cultural, temporal, geographical, and political.

and incremental development as well as adaptive development. As our Supply chain management software project was an innovative project, key practices of agile methods such as scheduling according to feature priorities, incremental delivery of software, feedback from expert users, emphasis on face-to-face communication, pair development, minimalist design combined with refactoring, test-driven development, daily integration, self organizing teams, and periodic tuning of methods helped significantly to achieve its successful implementation. As agile methods provide flexibility, it encourages the development teams and individuals towards creativity which is essential for successful implementation of innovative projects.

As it was innovative, large scale, high risk project, we formally did the architectural design along with documentation. This design documentation played an important role in the successful implementation of this project and it will be helpful in the maintenance phase also. The most important characteristic of development methods used in this project is that they were adapted to circumstances in each phase of the development. Agile development methods were combined so that new approaches are resulted from this self-adaptivity approach. It was not possible to complete and fix all the requirements because of the business domain and product characteristics. The software development team handling such a large project was small. Communication between team members was strong, as they were working in a small office and a business expert already aware of the business domain was in the same office so that they could interact whenever needed. The development approaches used in the supply chain management software project involved less documentation than the process-oriented approaches, usually emphasizing a smaller amount of documentation for a given task or only the critical parts were documented.

References

1. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods- Review and analysis. VTT Publications 478, 1–112 (2002)
2. Cockburn, A.: Agile Software Development. Addison-Wesley, London (2000)
3. Cockburn, A.: Agile Software Development. Addison-Wesley, London (2002)
4. Fowler, M.: The New Methodology, (April 2003), http://www.martinefowler.com/articles/
5. Jeffries, R., et al.: Extreme Programming Installed, vol. 172. Addison Wesley Longman, Redwood City (2001)
6. Larman, C.: Agile and Iterative Development: A Manager's Guide. Addison-Wesley, London (2003)
7. Leffingwell, D., Muirhead, D.: Tactical Management of Agile Development: Achieving Competitive Advantage, Rally Software Development Corporation, p. 1–23 (2004)
8. Shekaran, C., Garlan, D., Jackson, M., Mead, N.R., Potts, C., Reubenstein, H.B.: The role of software architecture in requirements engineering. In: Proceeding of the First International Conference on Requirements Engineering, pp. 239–245 (April 18-22, 1994)
9. Tichy, W.F.: Agile Development: Evaluation and Experience, University of Karlsruhe, tichy@ipd.uka.de
10. www.agilemanifesto.org

Software Measurement Programs in SMEs – Defining Software Indicators: A Methodological Framework

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] ALARCOS Research Group
Information Systems and Technologies Department
UCLM-Soluziona Research and Development Institute
University of Castilla-La Mancha,
13071 Ciudad Real, Spain
{Felix.Garcia,Mario.Piattini}@uclm.es

Abstract. Implementing a measurement program is not an easy task. It requires effort, resources, budget, experts in the field, etc. The challenges to successfully implement a measurement program in small settings are considerable and greater than in large companies. Small and medium-sized enterprises (SMEs) have an additional handicap: the existing methods and frameworks that support measurement programs such as Goal Question Metric (GQM), Goal-Driven Software Measurement, GQ(I)M, PSM and ISO/IEC 15939 do not fully satisfy the needs of such companies. We propose MIS-PyME, a methodological framework which supports small and medium enterprises (SMEs) in establishing software measurement programs, especially as regards the definition of software indicators. MIS-PyME is based on GQ(I)M and aims at supporting SMEs measurement activities related to software process improvement tasks. This framework has been applied in STL, where the main benefit derived from the use of MIS-PyME has been an effortless, more accurate program definition integrated into software process improvement practices.

Keywords: Methodological framework, measurement programs definition, GQ(I)M, MIS-PyME.

1 Introduction

Measurement is a key technology for supporting the basic tasks of an improvement program. Collecting and interpreting well-defined data provides organizations with the necessary information to make well-founded decisions about process improvement[1]. Process improvement results in increased productivity, better quality, and reduced cycle time, all of which make a company competitive in the software business[2].

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 adoption, and must have the wisdom of taking new business opportunities. Therefore, implementing serious measurement programs is even more important in small organizational settings [3].

Although measurement is 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 SMEs [4]. The existing obstacles include limited resources and a limited budget, a need for training, tight schedules, poor software measurement knowledge, a low cash flow, a restricted mentality as regards software measurement, etc. [5]. [6]. [7].

Table 1. Requirements for measurement program models suited to SMEs

SMEs Restrictions	Measurement program model for SMEs Requirements	Benefits
Limited resources, schedule and budget	- Easy to Understand and Manage (EUM) - Effortless (EFL): It should guide users and help them define measurement programs in an effortless way - Complete. (COM): It should cover their process improvement needs	- prevents users from spending too much time defining measurement programs. - does away with the need to contract measurement experts to develop measurement programs.
Limited training , poor software measurement knowledge	- Informative (INF): It should contain full information about the measurement possibilities, the benefits of using measurement and its potential use in technical and organizational management.	- makes users learn about measurement. - makes users learn about the benefits deriving from its use. - makes the measurement program more accurate.
	- Integration into the Processes (INTP): It should contain full information about its practical integration into the organization's software processes	- help measurements activities implemented for different aims be coherent. - measurement usefulness can be better understood since its potential use is clearly shown when the measurement goal is derived from the software process practices. - measurement activities are better integrated into software processes.
	Measurement Maturity Model (MMM): It integrates a measurement maturity model	- make the organization progress across software measurement. - advise the user to implement those measurement goals which suit its measurement maturity and prevent the user from defining measurement goals which cannot be successfully reached.

Our research aims at overcoming some of the obstacles mentioned in order to favor the implementation of measurement programs in small settings. MIS-PyME (Marco metodológico para la definición de Indicadores de Software orientado a PyME) is based on GQ(I)M [8, 9] and its main focus is on defining software indicators, which are the basic instruments for the analysis and interpretation of measurement objectives, and therefore for decision making.

With MIS-PyME we intended to provide a methodological framework that would help implement measurement practices which would support software process improvement activities suited to SMEs.

After listing the restrictions SMEs are confronted with in terms of software measurement, we conclude by showing the requirements which software measurement models should fulfill in order to be suited to SMEs regarding the definition of measurement (Table 1). We will use this table to look at the benefits and the contributions of our work.

This paper is organized as follows: Section 2 brings the MIS-PyME into context. Section 3 provides a more detailed insight into the framework, and briefly lists its specifications. Section 4 provides a case study examining the benefits derived from the implementation of this framework in a medium-sized company and Section 5 outlines the benefits resulting from present and intended research.

2 Related Work

In this section we show the most outstanding models and standards which support the definition of measurement programs, and the deficiencies of such standards as regards their implementation in SMEs. Special attention should be paid to the measurement models for the definition of measurement for SMEs which will be described below. An analysis of software products, process re-use and the integration of these practices into the measurement model will also be provided.

- Goal Question Metric GQM [10]. According to this model, there should exist an independent team which leads measurement program initiatives. This team must possess deep knowledge of measurement issues and have unrestricted access to the leaders of each project. Implementing this model in a company made up of 10 people which desires to perform measurement initiatives by itself will pose a challenge. Besides, there are authors who think that GQM encourages practitioners to define measures that are difficult to analyze and collect [10-12]. This is a drawback, especially for SMEs.
- In 1996, the SEI (Carnegie Mellon Software Engineering Institute) published the Goal-Driven Software Measurement guidebook [8]. This extension to GQM is called Goal Question Indicator Metric, GQ(I)M. Even if the definition of a measurement program is easier with this model than with GQM [10] (since it provides an intermediate layer - the indicator layer - as well as analyses, measurement examples, and supporting templates [9]), the project team still needs to have great knowledge and high insight on the field of measurement, and must make big efforts to define the measurement program.

Moreover, neither GQM [10] or GQ(I)M [8] give any guideline as to how to integrate measurement into software processes (INTP). GQ(I)M[8] contains some information on how to learn about measurement possibilities but GQM [10] does not, and none of them contains a measurement maturity model.

- PSM (Practical Software and Systems Measurement) [13] is a framework created by the Department of Defense in 1994 and its goal is to provide project and technical managers with the Best Practices and guidelines in software measurement. PSM focuses on issues in software projects which typically require management and control. It does not however clearly show the usefulness of measurement programs in supporting process improvement, and it does not provide any guideline on how to help the company improve through measurement maturity.
- ISO/IEC 15939 [14] lists the activities and tasks required in order to successfully identify, define, select, apply, and improve software measurement or the measurement structure in the organization under a generic project. However, it does not give any detailed methodology for defining the measurement program, it is not easy to implement effortlessly, and it does not give any information regarding measurement benefits, software process integration or measurement maturity.

Table 2 sums up how the above measurement models fulfill the requirements needed for a measurement definition model to be adapted to SMEs. The resulting values may be: "No", "Yes", "Some" or "-", which occurs when the concept does not fit into the model.

Table 2. Fulfillment of the measurement models requirements

Requirements	GQM	GQ(I)M	PSM	ISO/IEC
EUM	YES	YES	YES	SOME
EFL	NO	NO	YES	NO
COM	-	-	SOME	-
INF	NO	SOME	YES	NO
INTP	NO	NO	NO	NO
MMM	NO	NO	NO	NO

There are some studies that tailor some of the most widely known software measurement models and standards mentioned to the needs of SMEs. One of these is the work by Gresse et al. [4], who suggest the GQM Lightweight method. This approach consists in integrating the re-use of context-specific quality and resource models into the GQM model. This makes it unnecessary to start the model from scratch. This approach saves some effort and by so doing matches one of the aims of our approach, although the means to achieve this differ.

As far as process and product re-use are concerned, the latter having to do with the re-use of indicators and measures in our methodology, Medonça et al. [15] approach is to be taken into account. It is understood as a measurement model which integrates the previous experience of a company into its methodology. Finally, some studies believe in the benefits of re-using software products, processes, and experiences to achieve higher quality systems at a lower cost [16]. Some enterprises have developed this issue as observed in [17], [18].

3 MIS-PyME Specification

Firstly, we present an overview of MIS-PyME; we go on to look at the main work products of the framework, and provide a more detailed analysis of MIS-PyME. We finish by summing up what the contributions of this measurement framework have been.

3.1 MIS-PyME Framework Overview

MIS-PyME Framework deals with the definition of software indicators for SMEs. It is based on GQ(I)M [8, 9]. However, the steps of the methodology have been modified and tailored to the needs of SMEs. The main adaptations are:

- MIS-PyME supports this definition by providing measurement goals and indicator templates which function as a guide to the definition of measurement programs in the frame of software process improvement.
- MIS-PyME includes a useful database of indicators and measures taken from successfully implemented measurement programs.
- MIS-PyME provides a measurement maturity model which helps the company improve its software measurement in an orderly fashion.

It must be borne in mind that the scope of MIS-PyME in the measurement process only covers the "measurement planning process".

3.2 MIS-PyME Specifications

This section gives an overview of MIS-PYME main work products and characteristics.

MIS-PyME Work Products. The main work products identified in MIS-PyME are the following:

- MIS-PyME guide: A document which is intended as a guide for MIS-PyME model. It is focused on two working methods. The first one is used when a precise measurement goal is required to be defined for process improvement. The second one guides the organization and helps it progress through measurement maturity in an orderly fashion.
- MIS-PyME measurement goals table: MIS-PyME framework proposes a set of structured measurement goals usually required to implement improvement activities related to software processes. The goals are organized in a structure based on the measurement maturity required to implement each goal defined.
- 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 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. These are typical questions which the indicator tries to answer. 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. The measures used as input for these MIS-PyME indicator templates are also included in the database.

MIS-PyME Roles. MIS-PyME defines only three roles which have to be performed by different people. The first one is the measurement analyst, who should be familiar with the activities and processes carried out by the software development and maintenance department. It is preferable if his or her usual work relates to the definition of requirements, testing, configuration management or security, rather than design or development tasks. The second one should be a top manager who supports the measurement program initiative and has in-depth knowledge of the working method, software processes and process improvement needs. The third one is the reviewer, who will act at the "Verify the measurement program" status of the methodology. This role will be played by at least two people; one of them will be a project manager and the other one, a developer. The other steps in the methodology are performed by the measurement analyst, who should ask the top manager for all necessary information.

MIS-PyME Methodology Steps. We shall now briefly describe the MIS-PyME methodology and the most important changes made in MIS-PYME methodology as compared with GQ(I)M [8, 9] basic model. Figure 1 succinctly outlines this methodology.

1. Identifying your process improvement goals: MIS-PYME first step refers to GQ(I)M [8] step three, "Identifying your sub-goals", but our initial goals will derive from the software process model and they will not be business goals, but management process improvement goals.
- Description: Defining the process improvement goals that you want to carry out aided by software measurement. Identifying the related entities that will help achieve this goal.
- Input: Needs of the organization in order to establish and improve software processes. Output: List of process improvement goals and related entities.
2. Formalizing measurement goals
- Description: Measurement goals are specified. After that, the object of study, the purpose, the point of view, the environment and the measurement constraints are defined. The template for the definition of measurement goals has been changed with respect to that in GQ(I)M[8] so that it is easier to use. This has been achieved inasmuch as the purpose of the measurement is restricted to a set of precise purposes, as Briand et al. [5] measurement goal template does.
- Input: List of process improvement goals and related entities. Output: MIS-PyME measurement goal templates filled out.
3. Identifying if measurement goals have been defined:
- Description: Once measurement goals have been defined, verification of whether the MIS-PYME measurement goals table already defines the required measurement goals may follow.

- Input: MIS-PyME measurement goal table. Output: Register in MIS-PYME measurement goal table and related MIS-PyME indicator template.
4. Defining Indicators: This is a three step status.
- Description: Indicators required to implement measurement goals are defined.
- Input: MIS-PYME indicator templates related to each measurement goal. Output: MIS-PYME indicator templates filled out.

4.1. Specifying the indicators
- Description: If the measurement goals were in the MIS-PYME measurement goals table, the measurement analyst might take a look at the recommendations, restrictions, preliminary actions, information needs, etc. according to the MIS-PYME indicator template established for that goal. He/she should otherwise be guided by general recommendations provided by the generic MIS-PYME indicator template.
- Input: MIS-PYME indicator templates related to each measurement goal. Output: MIS-PyME indicator templates filled out.

4.2. Searching in MIS-PyME database:
- Description: 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.
- Input: MIS-PYME indicator templates related to each measurement goal. Output: MIS-PYME indicator examples.

4.3. Identifying sub-goals derived:
- Description: Any of the questions posed or the prerequisites 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 3.1 and 3.2 may then be repeated until all measurement-derived goals and their relevant indicators have been defined.
- Input: MIS-PyME indicator template filled out. Output: list of derived measurement goals.

5. Defining your measures and identifying the actions needed to implement them: Step 7 and 8 of GQ(I)M[8] are joined at one point.
- Description: 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
- Input: MIS-PyME indicator templates filled out. Output: measure definition checklists and data collection specifications.

6. Integrating measurement.
- Description: Integrating the measurement activities into previous measurement processes and into other software processes is the aim of this step. MIS-PYME provides guidance as to the structure of the (recommended) html document where all the indicators, measures, and measurement sub-processes of the organization are defined.
- Input: MIS-PyME indicator templates, measure definition checklists and collection specifications. Output: (updated) measurement process specification and (updated) software process specification.

7. Verifying the measurement process
- Description: The measurement process resulting from the process is verified by reviewers and modified if required.
- Input: Measurement process specification (updated) and software processes (updated). Output: verified measurement process specification and verified software processes.

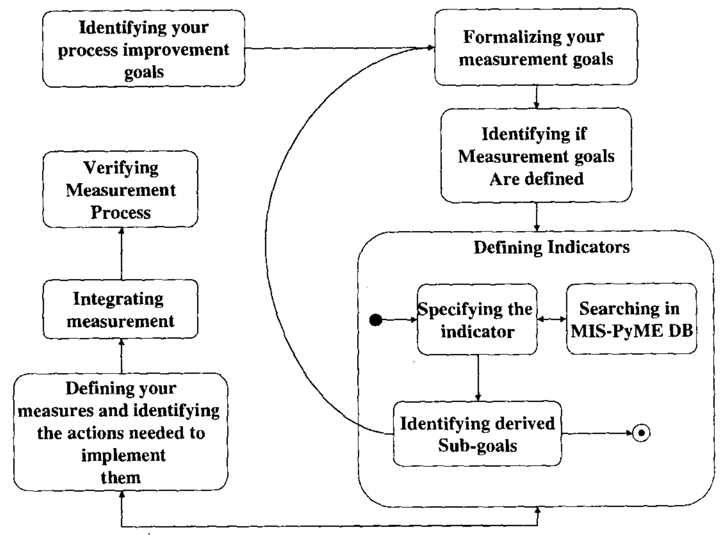
Identifying your process improvement goals
Formalizing your measurement goals
Identifying if Measurement goals Are defined
Verifying Measurement Process
Defining Indicators
Integrating measurement
Specifying the indicator
Searching in MIS-PyME DB
Defining your measures and identifying the actions needed to implement them
Identifying derived Sub-goals

Fig. 1. MIS-PyME methodology steps

3.3 MIS-PyME Contribution

Next table (see Table 3) shows how MIS-PyME fulfills the requirements mentioned in section one for the definition of software measurement programs suited to SMEs. After examining the table we can conclude that none of the measurement models shown in section 2 equals our contribution in terms of requirements met.

Table 3. Contribution of MIS-PyME

REQ	How it is fulfilled	Description
EUM	MIS-PyME Guide provides two working methods	MIS-PyMe supports the definition of measurement through any process improvement goal or progressive measurement maturity.
	Linkage between MIS-PyME measurement goals table – indicator templates –examples in DB	Users can easily obtain an overview of the whole definition process of a typical measurement goal by defining the need (improvement process practice) to define measures.
EFL	MIS-PyME measurement goals table	Typical process improvement measurement goals have been written down in the structured table, which should help users to shape their idea of measurement goals.
	MIS-PYME indicator templates	Questions asked to project managers are better focused thanks to the guidelines provided in the section "questions" in the indicator templates.
		Section "analysis and interpretation" in the indicator templates gives guidelines regarding the type of analysis that can be performed on the indicator, as well as its possible outcomes and interpretations.
		Guidelines regarding typical indicator inputs or how the indicator could be graphically displayed make the definition easier, especially if the measurement analyst is not an expert in the measurement area.
	MIS-PYME database	These indicator and measure examples related to a measurement goal make it easier to define the measurement program and let the user improve and check that definition.
COM	MIS-PyME measurement goals table	The measurement goals proposed. The related indicator templates and examples cover the basic needs for typical software process models. This deals with the measurement of product, projects and software processes.
INF	MIS-PYME indicator templates	Section "analysis and interpretation" gives guidelines regarding the type of analysis that can be performed on the indicator, as well as its possible outcomes and interpretations, which also show the potential of the use of measurement.
	MIS-PyME measurement goals table, indicator templates, DataBase	All the practical and theoretical information contained in these products helps users learn and understand the potential of the use of measurement and facilitates its exploitation.
INTP.	MIS-PyME measurement goals table	The link between process improvement and the measurement program is clearly established.
	indicator templates	Fields regarding "post definition" and "indicator integration" give guidelines about how to integrate the indicator into software processes.
IMM	MIS-PYME indicator templates	"Restrictions of purpose" and "evolution" fields meet the purpose of adjusting the indicator definition to the current measurement maturity.
	MIS-PYME guide	This guides users and helps them define and implement measurement goals according to their maturity level.

4 Applying MIS-PyME Framework in the Context of STL

The first application of MIS-PyME in real small settings is described in this section. The goal of this application was not to formally validate the framework but to obtain some hands-on experience with the framework and see if it was fit to solve the problems of a real organization. This gave us the opportunity to detect the deficiencies existing in its practical application and obtain the first feed-back regarding the acceptance of the framework by the organization and the preliminary benefits brought about by its use.

4.1 Introduction

Software measurement initiatives have been encouraged for many years by the software development and maintenance department in this company, which is formed by 39 people. However, the measurement process defined was not accurate enough and had not been properly established throughout the department. Some deficiencies had been detected, especially regarding those measures dealing with reliability. One of them was for example that the purpose of some of the indicators implemented had not been accurately established. An effort was made to try and evaluate the reliability of the project at a time when no collective and stable model existed for the definition of a fair evaluation threshold. Also, the input data was not enough for the analyses and interpretations that had been performed on projects.

The need of accurately measuring products, projects and processes in the organization increased since the number of projects and their scope was gradually increasing. A commitment to establish a well defined and accepted measurement program started to take shape.

The goals of the measurement program were as follows:

- To improve the definition of the indicators which had been previously established but had not been much accepted. These are related to the following areas: reliability of products under production, reliability of products under development within the scope of a specific project, and precision of the estimates of the duration of projects.
- To define other indicators required for project, product and process management related to estimation, development, service quality, and software process effectiveness. These were divided into two periods:
    - In the first period, only the measurement objectives which required data to be collected just once during the project or those objectives related to the monitoring of a process were defined and implemented
    - In the second phase, those measurement objectives which required data to be collected quite frequently, such as those related to project monitoring, would be defined later once phase I had been implemented. We will not deal with this phase in the present paper.
- To establish an easy and well documented methodological framework for the definition of measurement programs.
- To develop easy measurement management tools.

4.2 Development and Implementation of the Measurement Program

The measurement program was carried out by a person from the development department whose measurement knowledge was not bad, but he was not an expert in the area. The director of the department, who supported the initiative and had a good knowledge of the existing measurement needs, was the supervisor.

Initially, the measurement program was developed using GQ(I)M[8] measurement definition model. It took one month to define the first phase of the measurement program. The results were reviewed by some research members and by the director of the development department. Among the weaknesses and potential improvements that were detected are the following :

- In spite of the time the measurement analyst had spent trying to understand the limitations of the purpose of the indicators, he failed to define some of the same.
- The measurement program was not meant to be applied to the development processes existing at the time, and there was not a clear idea as to what these indicators were intended to be used for.
- The measurement process was not well structured. It was not well documented either, and turned out to be hard to follow.

These and other problems detected were not solely caused by the fact that GQ(I)M was used. If a measurement expert had defined the measurement program, he or she might have succeeded, but it became evident that it would be quite easy to fail if GQ(I)M was used even if the scope of the measurement program was restricted and known, and even if the responsible for defining the measurement program was already somehow familiar with software measurement.

Some aspects of GQ(I)M which could have lead to an unsuccessful measurement program were indicated by the measurement analyst:

- Since the measurement analyst did not want to bother project managers, he posed some of the questions to specify the goal himself. He failed in some of them.
- It was difficult to specify the indicators which would fulfill the measurement goal.
- It was difficult to know the purposes of the indicator which it would be possible to implement.
- It was difficult to define how to analyze the indicator.
- It was difficult to implement the measurement goal since many measures could not be collected.
- It was difficult to document the measurement process and integrate it into other measurement processes already established in an easy and understandable fashion.

Due to these problems, the measurement program had to be interrupted. It would be resumed two months later, since at that moment the responsible for the measurement program had to devote his time to some other urgent project.

In the second attempt at using MIS-PyME framework, the participants in the measurement program were the same as in the first attempt. The definition of the measurement program took one month and a half.

As may be noticed, the measurement program took more time using MIS-PyME than GQ(I)M. The reason is that, starting from the first version, the measurement program was more accurately defined and it was reviewed several times, as the development director trusted the resulting measurement program definition, but did not trust the one defined using GQ(I)M.

Part 1 of the measurement program defined 5 measurement goals and 20 software indicators. The data collected for the other indicators that had already been implemented stayed the same, but the way of analyzing the indicators was modified in most of the cases.

The implementation of part 1 of the measurement program required some developments to be made in order to create tools that automated the measurement activities as much as possible and tailor others. These developments were the following:

- Some new reports were implemented in the request for change & incident management system (Remedy).
- Some excel sheets were created in order to manage the measurement program. One excel sheet was developed to manage development processes; the other, for product management issues. The plan was to start using these easy sheets, and a more complex and powerful tool in the future if necessary.

In figure 2 we show one of the implemented goals which consists in evaluating the quality of project development services. This goal was defined by indicator IND-PROC-CALIDADSRV which uses two input indicators. One indicator characterizes the deviation between the first formal duration estimation of the projects and the real duration of the same (IND-PROC-INEXACDURACION), and the second indicator measures the reliability of the software developed by measuring the incidents occurred in the course of production for each project one month after the last installation of software (IND-PROC-FIABIMPL) related to the project.

The first indicator had already been defined and used before this initiative, but this was not the case of the second. The common goal had been well defined by means of IND-PROC- CALIDADSRV indicator definition.

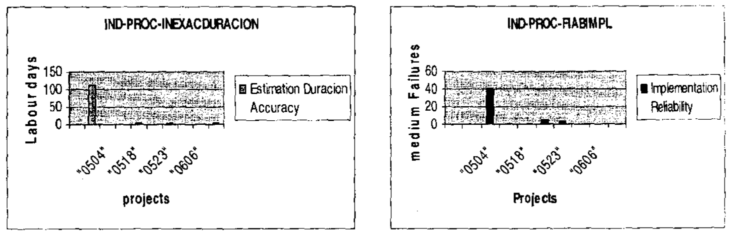

Fig. 2. Service quality indicator (IND-PROC-CALIDADSRV) which contains two input indicators: IND-PROC-INEXACDURACION and IND-PROC-FIABIMPL

Once the measurement program defined was implemented we analyzed this indicator at the time as defined and required. IND-PROC-FIABIMPL had to be retrospectively collected so as to be used with the other indicator IND-PROC-INEXACDURACION. This indicator allows us to understand and evaluate the general quality of a software

project provided to our clients, and to ascertain if the on-time release of the software product had a negative impact on software reliability, etc.

4.3 Lessons Learnt

The lessons learnt were analyzed after the first period of analysis was performed by the responsible for the definition of the measurement program and the director of the development department. We have divided their conclusions in two: The measurement program that resulted from the experience and the methodology (MIS-PyME) used.

A far as the measurement program is concerned, they reported that they trusted this measurement program but did not trust other previous measurement programs. The reasons were:

- This measurement program covers most of their basic needs.
- Its definition is quite complete as regards the questions which should be answered for each indicator and the analysis and interpretation that can be done, which also prevents users from making a free analysis or interpretation since they have to adjust to what is defined.
- The measurement program was well integrated with the software processes.
- The measurement program was documented based on the Web, which made it easier to read, access and use since it contained links to other reports on the software processes, sheets, etc that might be required.
- However, there was still quite a lot of data that had to be manually collected and included in the excel sheet, which they found to be quite bothersome.

Regarding the use of MIS-PyME framework as compared with our previous experience:

- Questions asked to project managers are better focused thanks to the guidelines provided by the MIS-PYME indicator templates.
- Fitness to the purpose is more easily achieved thanks to the guidelines regarding the types of indicators and the restrictions in the implementation of each type of indicator.
- MIS-PYME indicator templates give guidelines regarding the type of analysis that can be performed on the indicator, as well as its possible outcomes and interpretation. This information allows the analyst to learn some information and pass it on to project managers as far as the potential of the results obtained from the indicators analysis is concerned.
- Guidelines regarding typical indicator inputs (how the indicator could be graphically displayed), and indicator examples make the definition easier, especially if the measurement analyst is not an expert in the measurement area.
- The integration of the measurement process into software processes is also easier using MIS-PyME, since it informs about how to do this.
- MIS-PyME documentation is Web-based, which facilitates its use (easy access to the required templates, etc.).
- The guide regarding how to document the measurement process helps to increase the reliability of the resulting measurement process, making it easier to use and more integrated as well.

However, there are still some deficiencies in MIS-PyME model. Among several mistakes found, the most outstanding problem was that MIS-PyME database, containing the indicators and measure examples, is still quite small and does not cover most of the needs.

5 Conclusions and Further Research

In this paper we have proposed a methodological framework which makes it easier to define measurement programs. The framework, which is called in Spanish MIS-PyME (Marco metodológico de definición de Indicadores de Software para PyMEs) is based on GQ(I)M [8, 9] and is designed to be used with software process improvement practices. It provides a full and detailed guide that helps define common required measurement goals based on indicator templates and a database that includes examples of indicators and measures that have been implemented in successful measurement programs. It also integrates a model to make the organization progress through measurement.

We started by showing the characteristics required for a measurement definition model suited to SMEs. We have shown how the most outstanding measurement models do not fulfill the requirements and, after reviewing MIS-PyME framework, we have given a number of reasons why MIS-PyME framework does match these requirements, thus proving the extent of our contribution.

The end of the paper presents the use of MIS-PyME for developing a measurement program in the development department of Sistemas Técnicos de Loterías del Estado (STL) where the results were positive.

In the future, we shall continue monitoring and refining this measurement framework in order to validate the MIS-PyME framework for different SMEs. We have to increase the MIS-PyME database and include other MIS-PyME measurement goals and related indicators to consider other needs. The maturity model integrated in MIS-PyME is the least developed area and we may focus on it in the years to come.

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) and ESFINGE (Dirección General de Investigación del Ministerio de Educación y Ciencia, TIN2006-15175-C05-05) projects.

References

1. Brijckers, A., Differding, C.: The Role of Software Process Modeling in Planning Industrial Measurement Programs. In: Proceedings of theThird International Sysmposium on Software Metrics (METRICS'96) pp. 31–40 (1996)
2. Daskalantonakis, M.K.: A Practical View of Software Measurement and Implementation Experiences Within Motorola. IEEE Transactions on Software Engineering 18(11), 998–1010 (1992)

3. 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, pp. 41–44 (2005)
4. 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)
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, pp. 23–29 (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. Park, R.E., Goethert, W.B., Florac, W.A.: Goal-Driven Software Measurement-A Guidebook: Carnegie Mellon University Pittsburgh: Software Engineering Institute (1996)
9. Goethert, W., Siviy, J.: Applications of the Indicator Template for Measurement and Analysis, in Software Engineering Measurement and Analysis Initiative (September 2004)
10. Solingen, R.v., Berghout, E.: The Goal/Quesiton/Metric Method - A practical guide for Quality Improvement of Software Development. Mc Graw Hill, New York (1999)
11. Solingen, R.v., Berghout, E.: Integrating Goal-Oriented Measurement in Industrial Software Engineering: Industrial Experiences with and Additions to the Goal/Question/Metric Method (GQM). In: Seventh International Software Metrics Symposium. London, England, pp. 246–258 (2001)
12. Shepperd, M.: Foundations of Software Measurement. Prentice Hall, Hemel Hempstead, England (1995)
13. PSM: Practical Software and Systems Measurement - A Foundation for Objective Project Management Version 4.0c: Deptartment of Defense and US Army (November 2000)
14. ISO/IEC 15939, in Software Engineering - Software Measurement Process (2002)
15. Mendonça, M.G., et al.: An approach to improving existing measurement frameworks. IBM Systems Journal 37(4) (1998)
16. Basili, V.R., Caldiera, G., Rombach, H.D.: The experience factory, in Encylopedia of Software Engineering, J.J.M. (ed.) John Wiley & Sons, pp. 469–476
17. Druffel, E., Redwine, S.T., Riddle, W.E.: The STARS Program: Overview and Rationale, pp. 21–29. IEEE Computer, Los Alamitos (1983)
18. Guerrieri.: Searching for Reusable Software Components with the RAPID Center Library System. In: Proceedings of the Sixth National Conference on Ada Technology, pp. 395–406 (1988)