Available online at www.sciencedirect.com

ScienceDirect

Journal of Systems Architecture 52 (2006) 627–639

JOURNAL OF SYSTEMS ARCHITECTURE

www.elsevier.com/locate/sysarc

FMESP: Framework for the modeling and evaluation of software processes

Félix García [a,*], Mario Piattini [a], Francisco Ruiz [a], Gerardo Canfora [b], Corrado A. Visaggio [b]

[a] Alarcos Research Group, Computer Science Department, University of Castilla-La Mancha, 13071 Ciudad Real, Spain
[b] RCOST – Research Centre on Software Technology, University of Sannio, Benevento, Italy

Available online 7 August 2006

Abstract

Nowadays, organizations face with a very high competitiveness and for this reason they have to continuously improve their processes. Two key aspects to be considered in the software processes management in order to promote their improvement are their effective modeling and evaluation. The integrated management of these key aspects is not a trivial task, the huge number and diversity of elements to take into account makes it complex the management of software processes. To ease and effectively support this management, in this paper we propose FMESP: a framework for the integrated management of the modeling and measurement of software processes. FMESP incorporates the conceptual and technological elements necessary to ease the integrated management of the definition and evaluation of software processes. From the measurement perspective of the framework and in order to provide the support for the software process measurement at model level a set of representative measures have been defined and validated.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Software process modeling; Software measurement; Conceptual framework; Software engineering environment

1. Introduction

The successful management of software processes is one of the main goals for software organizations in order to improve products quality, given the existent correlation between process and product quality [10]. To satisfy the quality requirements entails that: software processes must produce the expected results, be correctly defined, and any improvements made should be in accordance with the objectives of the enterprise, which may change very often in highly competitive companies. "Software Process Management" [9] involves four key responsibilities: to define, measure, control and improve the process. In order to take these responsibilities into account, it is very important to consider the integrated management of two key aspects:

- *Process modeling*. Given the particular complexity of software processes, deriving from the high diversity of elements which have to be considered when managing them, it is necessary to effectively carry out the definition of the software processes. The process models constitute the starting point

* Corresponding author. Tel.: +34 926 295300; fax: +34 926 295354.
E-mail address: Felix.Garcia@uclm.es (F. García).

1383-7621/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2006.06.007

to analyze, enact and improve the processes. Against the diversity of existing process modeling proposals, a reference process metamodel becomes necessary. With this goal, the Object Management Group (OMG) has proposed the SPEM (Software Process Engineering Metamodel Specification) metamodel [20], which defines a reference language for process modeling.

- *Process evaluation*. In order to promote software process improvement, it is very important to previously establish a framework for analysis (with the aim of determining the strong and weak points of the software processes). The previous step of the software processes improvement is their evaluation and this goal requires the definition of measures related to the different elements involved in software processes. Due to the great number of different entities involved in the evaluation of software processes, the establishment of a common terminology for the definition, calculation and exploitation of measures is fundamental for the integrated and effective management of the measurement process. The integration of the modeling and evaluation of software processes can be beneficial for a Software Organization for different reasons:
  – it is a critical factor to reach a high degree of maturity in its processes with accordance to the maturity models;
  – by using the same notations and reference models for modeling and evaluation, misunderstanding could be more easily avoided;
  – the integration could let the adoption of one software tool for both modeling and evaluating: this reduces the costs, in terms of licenses, time for training the people who will have to use it.

In this article we propose FMESP: a framework for the integrated management of the modeling and measurement of software processes in order to promote their improvement. FMESP includes the conceptual and technological elements necessary to ease the integrated management of the definition and evaluation of software processes. From the measurement perspective, FMESP provides a systematic and integrated way to effectively measure the basic kinds of entity candidate for the measurement in the context of the software processes: process models, projects and products. The existing research efforts on software process measurement have been mainly focused on the measurement of projects – in terms of cost and schedule –, products and concrete aspects of models as accuracy [7], intended as the degree to which the model reflects the actual process. To complement the process measurement support, we have proposed a set of measures to evaluate the maintainability of SPMs as part of FMESP. They may provide the quantitative basis for easing the changes and evolution of the models in the context of software process improvement.

The rest of the paper proceeds as follows. Firstly, we present the characteristics of the FMESP framework by describing its main components: a conceptual framework and a software engineering environment (SEE). In Section 3 the measures for the evaluation of SPMs are described. In Section 3.1 the definition of the measures is presented. Then, the results obtained with the empirical validation of the measures are shown. Finally, some conclusions and further works are outlined.

2. FMESP overview

The main components of FMESP are a *conceptual framework* and a *software engineering environment*.

2.1. Conceptual framework

The FMESP Conceptual Framework provides the support necessary for the representation and management of the knowledge related with the software processes from the modeling and measurement perspectives. FMESP represents the necessary mean to define and measure the software processes in an integrated way, by tackling their intrinsic complexity to support these key aspects in the process management. In Fig. 1 the structure and main elements of the conceptual framework are shown.

As we can observe in Fig. 1, the conceptual framework of FMESP is composed of

(a) A *conceptual metadata architecture* of four Abstraction layers. The aim of this architecture is to provide the integration management of the modeling and measurement by representing the elements related in different abstraction layers. This integration is carried out by including the modeling languages (metamodels) necessary to define the processes, the measurement models of the process entities and the required metamodels (domain metamodels) to represent any software process-related

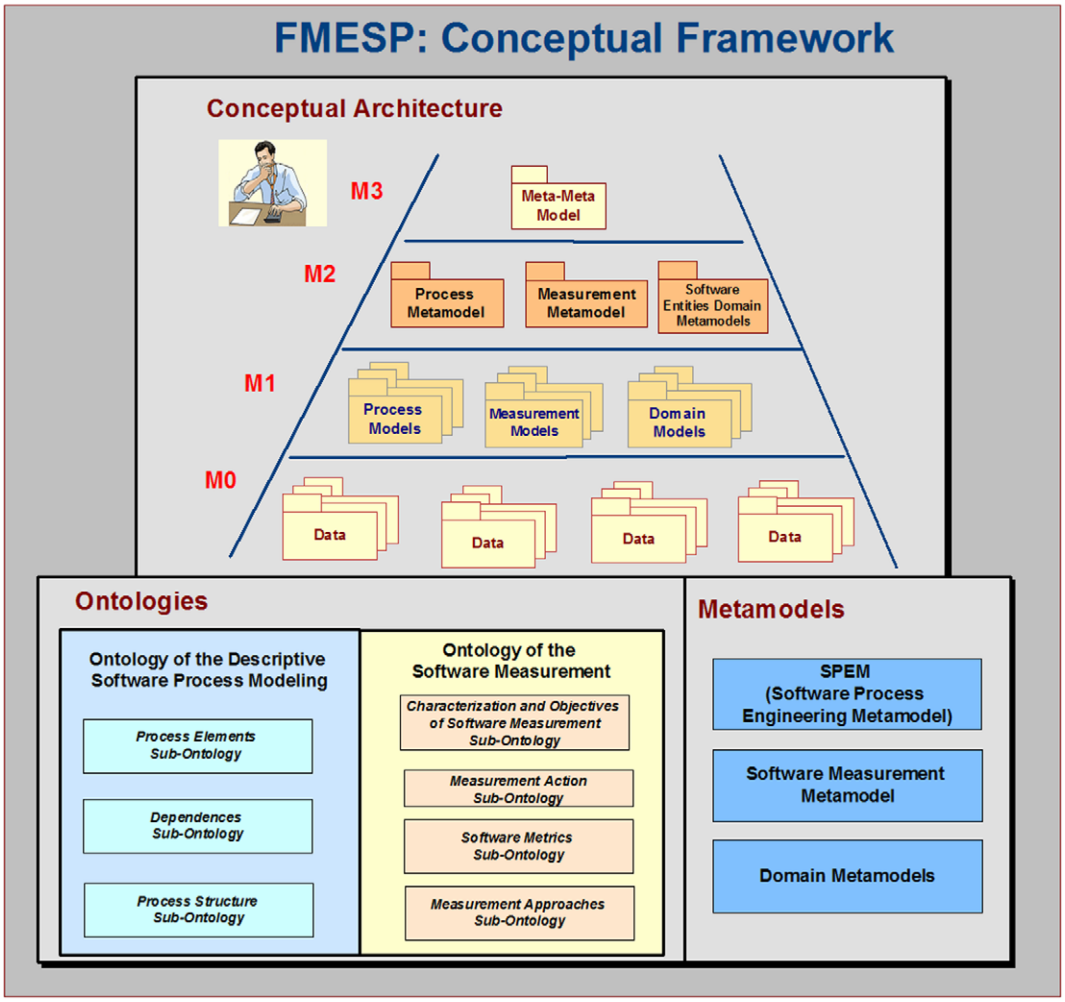

FMESP: Conceptual Framework
Conceptual Architecture
M3
Meta-Meta Model
M2
Process Metamodel
Measurement Metamodel
Software Entities Domain Metamodels
M1
Process Models
Measurement Models
Domain Models
M0
Data
Data
Data
Data
Ontologies
Ontology of the Descriptive Software Process Modeling
Process Elements Sub-Ontology
Dependences Sub-Ontology
Process Structure Sub-Ontology
Ontology of the Software Measurement
Characterization and Objectives of Software Measurement Sub-Ontology
Measurement Action Sub-Ontology
Software Metrics Sub-Ontology
Measurement Approaches Sub-Ontology
Metamodels
SPEM (Software Process Engineering Metamodel)
Software Measurement Metamodel
Domain Metamodels

Fig. 1. FMESP: conceptual framework.

entity (process models, projects, products) which can be candidate for measurement. In this architecture also the concrete models for the definition and measurement of software processes (level M1) are included.

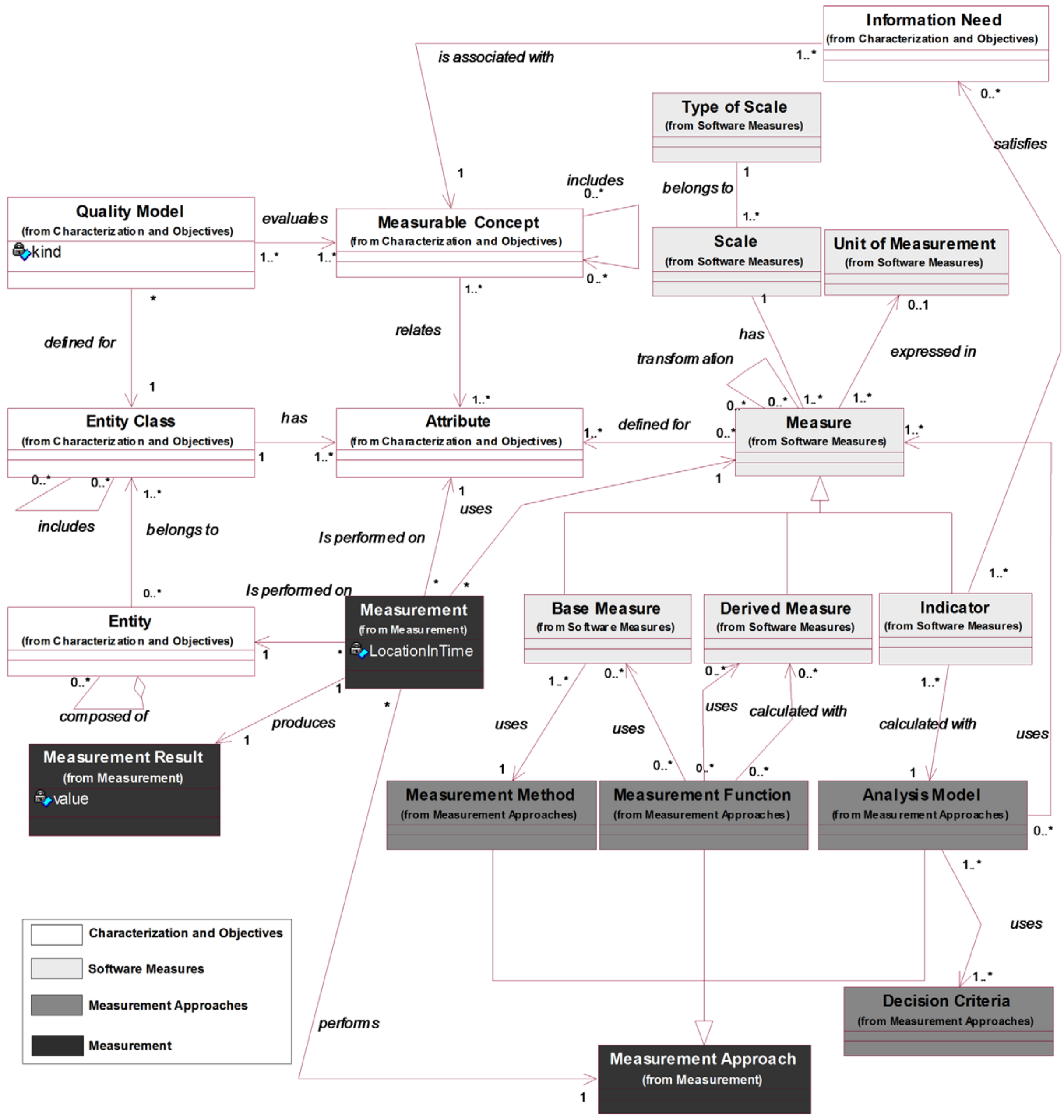
(b) A collection of *ontologies*. A fundamental aspect to consider in order to successfully integrate the modeling and measurement of software processes is that all the models and metamodels necessary must be based on the same conceptualization (set of objects, concepts, entities and their relationships which are supposed to exist in these domains of interest) and such conceptualization must be specified by building the suitable ontologies [15]. With this aim in mind and as a previous step to the metamodel definition we developed the following ontologies by applying the REFSENO methodology [21]:

- *Descriptive software process modeling ontology*. This ontology has been defined to clarify the domain of descriptive modeling of software processes and the SPEM specification [20] has been used as reference.
- *Software measurement ontology* (SMO) [13]. At the best knowledge of the authors, there is a need to establish and clarify the terminology; the relevant concepts and relations in the field of the software measurement. By this ontology the software measurement domain is clarified and it has eased the later definition of the measurement metamodel in order to carry out a systematic an effective measurement process given the inherent complexity of the software process measurement. Also, by this ontology a common measurement terminology can be provided in companies to ease the communication and understanding among the staff responsible of the

measurement process and provide the possibility to register the results of this process in a consistent an integrated way. The current version of this ontology is shown in Fig. 2 with an UML class diagram. The SMO ontology is subdivided in four sub-ontologies:

(i) *Software measurement characterization and objectives*, which includes the concepts required to establish the scope and objectives of the software measurement process. The main goal of a software measurement process is to satisfy certain information needs by identifying the entities (which belong to entity classes) and the attributes of these entities (which are the focus of the measurement process). Attributes and information needs are related through measurable concepts (which belong to a quality model).

(ii) *Software measures*, which aims at establishing and clarifying the key elements in the definition of a



Information Need (from Characterization and Objectives)
Type of Scale (from Software Measures)
Quality Model (from Characterization and Objectives)
kind
Measurable Concept (from Characterization and Objectives)
Scale (from Software Measures)
Unit of Measurement (from Software Measures)
Entity Class (from Characterization and Objectives)
Attribute (from Characterization and Objectives)
Measure (from Software Measures)
Entity (from Characterization and Objectives)
Measurement (from Measurement)
LocationInTime
Base Measure (from Software Measures)
Derived Measure (from Software Measures)
Indicator (from Software Measures)
Measurement Result (from Measurement)
value
Measurement Method (from Measurement Approaches)
Measurement Function (from Measurement Approaches)
Analysis Model (from Measurement Approaches)
Decision Criteria (from Measurement Approaches)
Measurement Approach (from Measurement)
is associated with
satisfies
includes
belongs to
evaluates
relates
defined for
has
transformation
expressed in
includes
belongs to
Is performed on
uses
composed of
produces
calculated with
performs
Characterization and Objectives
Software Measures
Measurement Approaches
Measurement

Fig. 2. FMESP: software measurement ontology.

software measure. A measure relates a defined measurement approach and a measurement scale (which belongs to a type of scale). A measure is expressed in a unit of measurement, and can be defined for more than one attribute. Three kinds of measures are distinguished: base measures, derived measures, and indicators.

(iii) *Measurement approaches*, which introduces the concept of measurement approach to generalize the different "approaches" used by the three kinds of measures for obtaining their respective measurement results. A base measure applies a measurement method. A derived measure uses a measurement function (which rests upon other base and/or derived measures). Finally, an indicator uses an analysis model (based on a decision criteria) to obtain a measurement result that satisfies an information need.

(iv) *Measurement*. It establishes the terminology related with the act of measuring software. A measurement (which is an action) is a set of operations having the object of determining the value of a measurement result, for a given attribute of an entity, using a measurement approach. Measurement results are obtained as the result of performing measurements (actions).

(c) A collection of *Metamodels*. The knowledge related with the software process modeling and measurement domains were represented with the ontologies, the aim to automate the conceptual framework. It was enriched with the following metamodels:

• A *process modeling language* (PML). The language selected has been SPEM (Software Process Engineering Metamodel Specification). This is the process modeling metamodel more suitable for the FMESP framework which requires a generic metamodel with the constructors adequate for the descriptive definition of the software processes. SPEM has a wide industrial acceptance and since it is based on UML (Unified Modeling Language) it is expected to be used largely as well as the UML in the product modeling.

• *Software measurement metamodel*. In order to provide the quantitative support necessary for the process improvement, the measurement models must be represented by using a consistent metamodel. For this reason we have defined a measurement metamodel based on the software measurement ontology. With this metamodel the measurement process is effectively supported by providing a homogeneous and consistent representation of the huge diversity of entities which can be measured in the context of the software process evaluation (project measurement, quality of the work-products, quality of the process models, etc.).

• *Metamodels* for the definition of the *software-process related entities*. In order to provide the support necessary for the measurement of the software process it is necessary to represent and measure the relevant entities related with the process of interest. In order to achieve this, the metamodels which represent these entities must be defined and included in the repository. Hence, the measurement models can be defined and this definition is performed relying on the metamodels constructors by defining a set of base measures (for example, the number of entities in an entity-relationship model) and then a set of derived measures and indicators (maintainability rate of an entity relationship model). These metamodels are called "Domain Metamodels".

2.2. Software engineering environment

The FMESP SEE is composed of two integrated tools which provide the technological support to the conceptual framework: METAMOD for the definition and management of the metadata conceptual architecture; and GenMETRIC to support the development of measurement models. The SEE can be extended with new tools for the support of the evaluation and improvement of the processes. All the metadata managed by the tools are stored in a repository which permits the sharing of all the overall software process information and knowledge. In Fig. 3 a general overview of the SEE is shown.

As we can observe in Fig. 3, the three main elements which constitute the SEE are

(a) A *metadata repository*. The metadata representation at different abstraction levels is the most suitable means to handle the complexity and diversity of the information needed to support the definition and evaluation of the software processes. To support this representation and storage one metadata repository is provided. The repository constitutes the key component of the SEE and it provides the tools with the metadata they need to support the concrete aspects related to the definition and measurement of software processes. Also, these tools storage in the repository new metadata which makes its growth possible. The models and metamodels are stored in the repository as XMI documents [23].

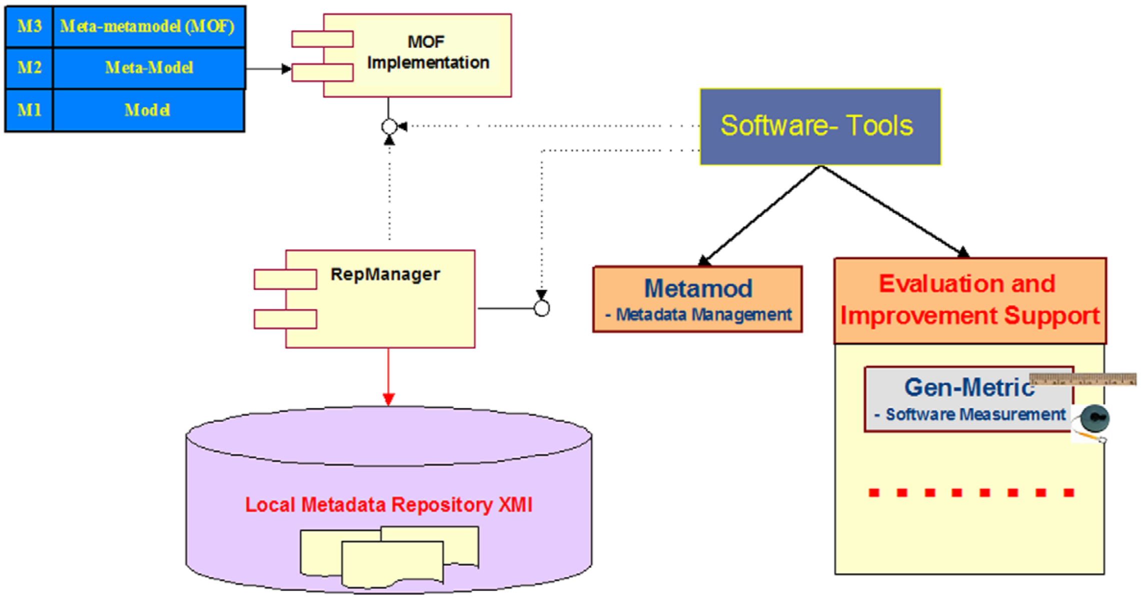
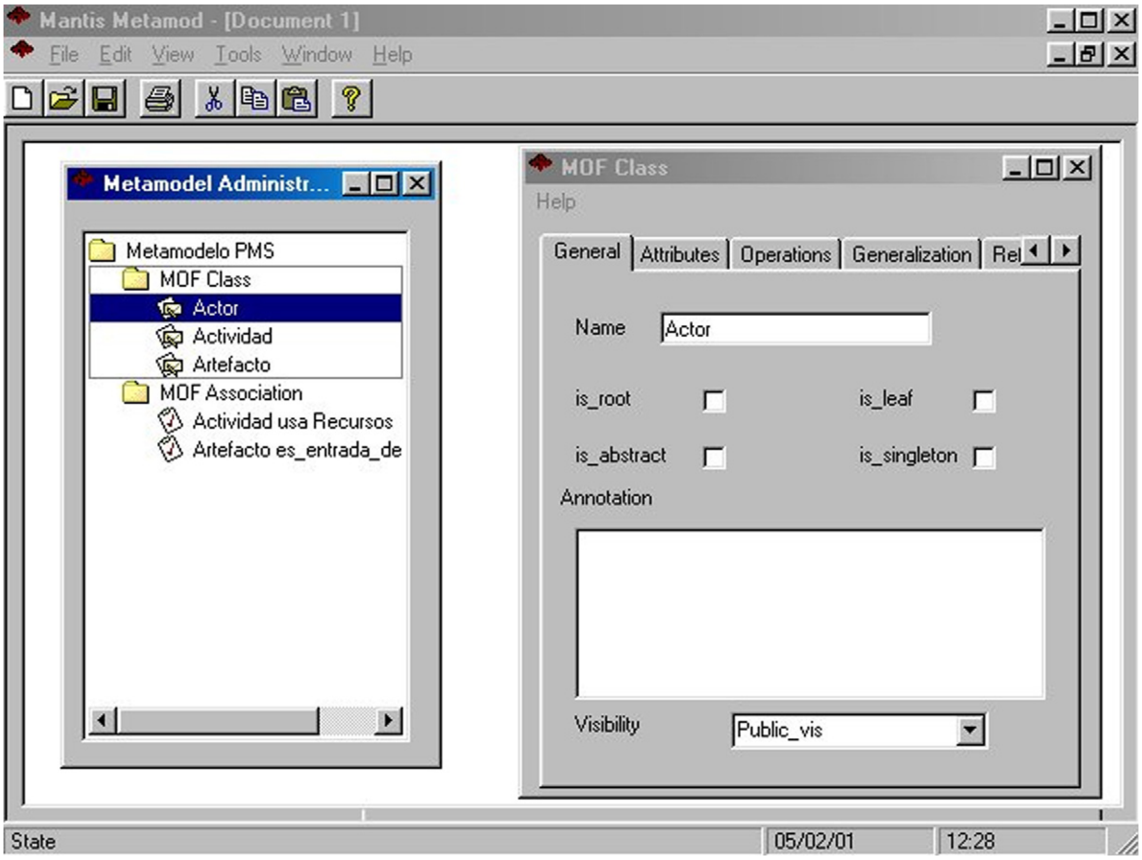


Fig. 3. FMESP SEE.



Metamodelo PMS
MOF Class
Actor
Actividad
Artefacto
MOF Association
Actividad usa Recursos
Artefacto es_entrada_de

Name: Actor
is_root ☐ is_leaf ☐
is_abstract ☐ is_singleton ☐
Annotation
Visibility: Public_vis

State 05/02/01 12:28

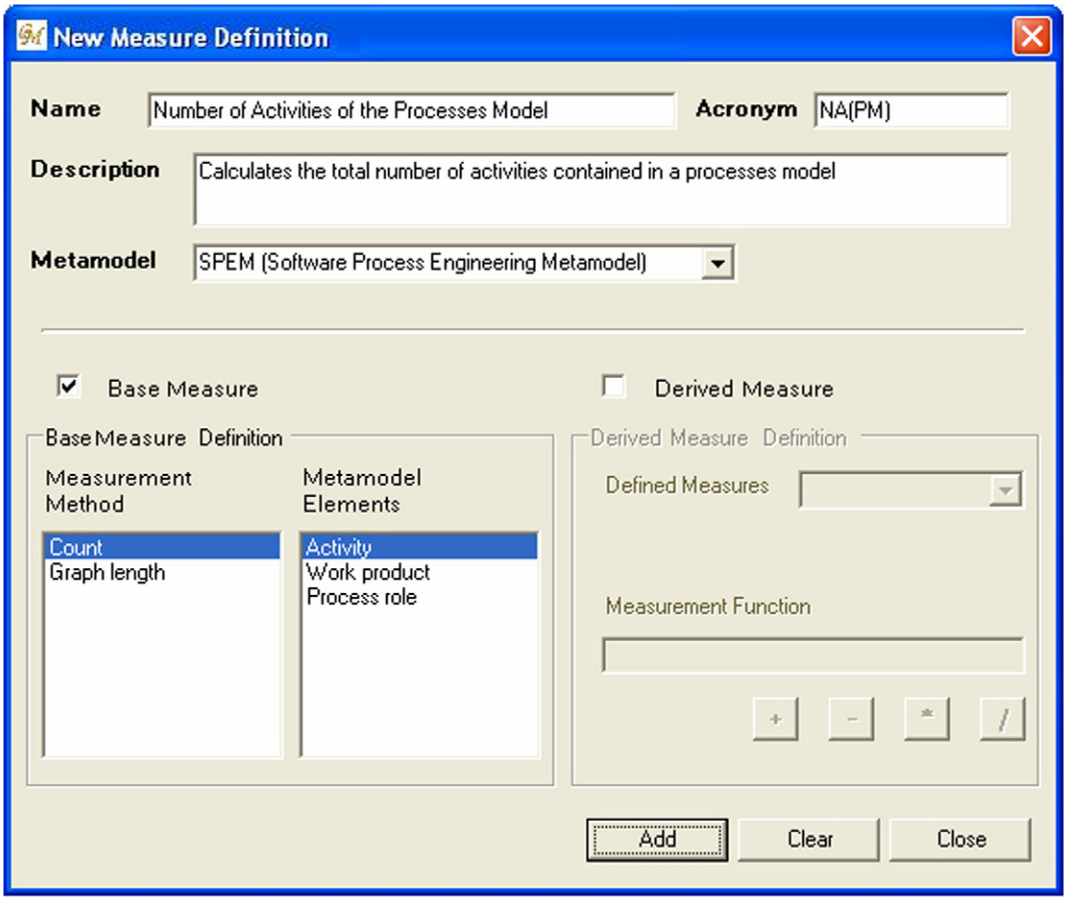
Fig. 4. METAMOD tool.

(b) *Software components* for the *management* and *storage* of *metadata*. To support the metadata definition and to store them in the repository according to the MOF [16] and XMI standards, two software components have been developed: *MOFImplementation*, which manages the representation of metamodels and models according to the MOF standard; and *RepManager* [19], to manage the XMI representation and storage of the metadata defined. The services of these components are used by the SEE software tools to save and retrieve metadata of the repository.

(c) *Software tools*. The aim of the SEE software tools is to provide the support for the definition of the elements related to the software process – from the proper software process model to any entity of interest related with it – and the support for the measurement of such elements. With this goal two software prototypes have been developed:

• *METAMOD* [11] which supports the definition, query and storage of the models and metamodels of the repository. To achieve this it uses the services of the components *MOFImplementation* and *RepManager*. With METAMOD, therefore, it is possible to manage the software process knowledge base stored in the repository. Fig. 4 shows the interface of METAMOD for the definition of a metamodel by using the MOF language.

• *GenMETRIC* [12], for the measurement of the software entities. With GenMETRIC it is possible the measurement of any software entity, for which it is required to have stored in the repository the domain metamodel which represents such entities. Also it is possible the definition of any software measure. The base measures are defined on the domain metamodel elements (classes and associations) by using standard measurement methods as "count" or "graph length". For the definition of derived measures and indicators the tool includes an evaluator of arithmetical and logical expressions. In Fig. 5 the frame for the definition of base and derived measures is shown. Therefore,


New Measure Definition

Name: Number of Activities of the Processes Model

Acronym: NA(PM)

Description: Calculates the total number of activities contained in a processes model

Metamodel: SPEM (Software Process Engineering Metamodel)

[✓] Base Measure

[ ] Derived Measure

Base Measure Definition

Measurement Method	Metamodel Elements
Count	Activity
Graph length	Work product
	Process role

Fig. 5. Definition of software measures with GenMETRIC.

GenMETRIC is an open and extensible measurement tool thanks to its usage of the metadata stored in the repository (using the software components services). The measurement models defined with GenMETRIC are stored in the repository according to the measurement metamodel of FMESP.

With the framework proposed the management of the modeling and measurement of software processes is supported by including a metadata repository in which a knowledge base of the process management is established orientated to its evaluation and improvement.

3. Measures for software process models

The improvement of software processes involves the need to effectively maintain them and the maintenance of the software process deserves the same attention as well as any other kind of software [8]. Considering that "software processes are software too" [17], the evaluation of the maintainability of the processes in the early stages of their development and specially, in the modeling stage, is very important. SPMs constitute the starting point to carry out the later enactment, evaluation and improvement. Therefore, as the software processes change, process models may change accordingly and it is necessary to maintain effectively the process models with the aim to facilitate: the communication of process modifications, the understanding of new responsibilities and procedures and the automation of guidance in performing new activities. With this goal, in order to integrate (within the context of the FMESP framework) the existing proposals in literature related with process measurement at project and product levels with the process measurement at conceptual level, a set of measures for software process models were defined and empirically validated to find useful SPMs maintainability indicators.

A representative set of measures for software process models has been defined in order to evaluate the influence of the structural complexity of the models on their maintainability (see Table 1).

The measures evaluate the software process model structural complexity, and they could be useful maintainability indicators, taking into account that a software process model with high degree of complexity will be much more difficult to maintain by considering the relationship between structural complexity and maintainability of software artifacts [3] in the domain of software processes.

The measures have been defined following the SPEM terminology [20] by examining its key software process constructors, but they can be directly applied to other PMLs. The defined measures are

Table 1
Measures of SPMs

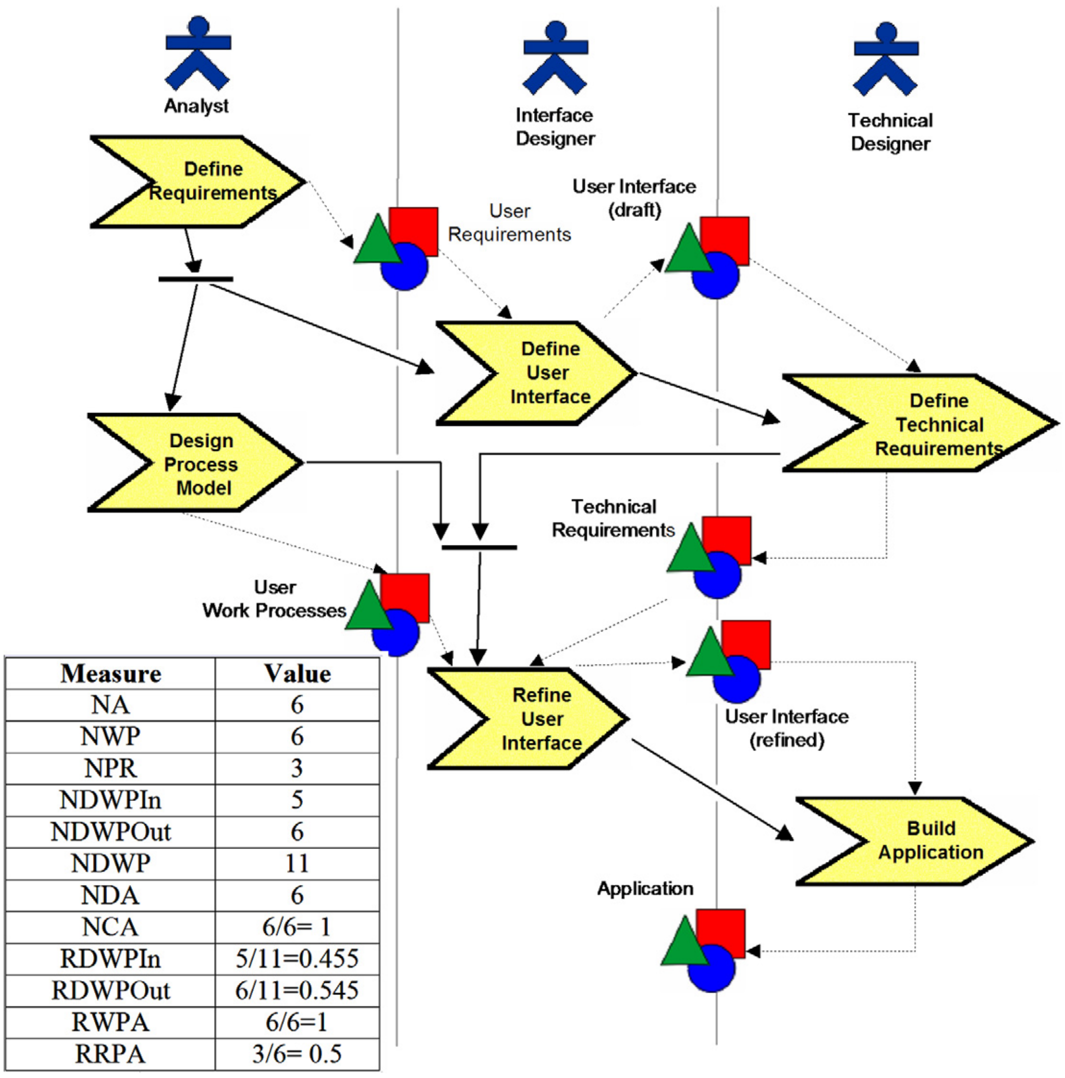
Measure	Definition
NA	Number of **Activities** of the software process model
NWP	Number of **Work Products** of the software process model
NPR	Number of **Roles** which participate in the process
NDWPIn	Number of input dependences of the **Work Products** with the **Activities** in the process
NDWPOut	Number of output dependences of the **Work Products** with the **Activities** in the process
NDWP	Number of dependences between **Work Products** and **Activities** $NDWP(PM) = NDWPIn(MP) + NDWPOut(MP)$
NDA	Number of precedence dependences between **Activities**
NCA	Activity Coupling in the process model $NCA(PM) = \frac{NA(PM)}{NDA(PM)}$
RDWPIn	Ratio between **input dependences** of Work Products with Activities and **total number of dependences** of Work Products with Activities $RDWPIn(PM) = \frac{NDWPIn(PM)}{NDWP(PM)}$
RDWPOut	Ratio between **output dependences** of Work Products with Activities and **total number of dependences** of Work Products with Activities $RDWPOut(PM) = \frac{NDWPOut(PM)}{NDWP(PM)}$
RWPA	Ratio of **Work Products** and **Activities**. Average of the work products and the activities of the process model $RWPA(PM) = \frac{NWP(PM)}{NA(PM)}$
RRPA	Ratio of **Process Roles and Activities** $RRPA(PM) = \frac{NPR(PM)}{NA(PM)}$

Model Scope (see Table 1), because they measure the structural complexity of the overall software process model. The measures have been theoretically validated by using the DISTANCE framework [18] and they belong to the ratio scale.

In Fig. 6 an exemplar software process model represented with SPEM is shown, for which the Activity Diagram UML notation and the stereotypes which represent the SPEM constructors can be used. The measures values are also shown.

3.1. Empirical validation

In order to empirically validate the proposed measures and generalise the results in the best way possible, a family of experiments [4] has been carried out. As Basili et al. [2] remark, by a family of experiments it is possible to accumulate the knowledge necessary to extract significant conclusions that can be applied in practice. By carrying out an empirical validation in the context of a family of experiments it is possible to obtain better results than if we carry out the empirical studies in an individual way. With this objective we have planned and performed a family of experiments according to the methodology of Ciolkowski et al. [6]. The general goal of the experiments is to demonstrate the suitability of the selected measures of software process models as maintainability indicators. By using the Goal Question Metric (GQM) template



Measure	Value
NA	6
NWP	6
NPR	3
NDWPIn	5
NDWPOut	6
NDWP	11
NDA	6
NCA	6/6= 1
RDWPIn	5/11=0.455
RDWPOut	6/11=0.545
RWPA	6/6=1
RRPA	3/6= 0.5

Fig. 6. Software process example and measure values.

[1] the general goal can be defined in the following way:

- *analyze* measures of SPMs;
- *with the purpose of* evaluating;
- *with respect to* the capability of being used as maintainability indicators;
- *from the point of view* of the researchers;
- *in the context of* computer science undergraduate students and professionals of information systems.

According to the general plan of the family we have carried out five individual experiments (Fig. 7) in order to satisfy the general goal.

As can be observed in Fig. 7, the individual experiments were grouped under two main categories depending on the kind of tasks to perform by the subjects:

- *Subjective rating*. In this group, the maintainability sub-characteristics were rated in a subjective way according to the opinion of the subjects.
- *Objective rating*. In the objective experiments the subjects had to perform a set of tasks on the models related with their maintainability (understandability and modifiability). In these experiments the dependent variables were measured in an objective way by calculating the time spent by the subjects in performing these tasks.

The context of this group of experiments has been composed of students and professionals. In Table 2 are summarized the obtained results.

As a result of the empirical study, the measures NA, NWP, NDWPIn, NDWPOut, NDWP and NDA were demonstrated to be empirically valid and hence, they can be used like SPMs maintainability indicators. This significant group of measures has been correlated in all the experiments with the dependent variables studied.

4. Conclusions and further works

In this paper the FMESP framework has been presented. FMESP integrates the modeling and measurement of software processes by providing the conceptual and technological support necessary in order to successfully manage these two key responsibilities of the process management. The framework was successfully applied in a software company and some significant benefits were obtained [5].

From the measurement perspective of the framework and in order to provide the support for the software process measurement at conceptual level, a set of representative SPM measures have been

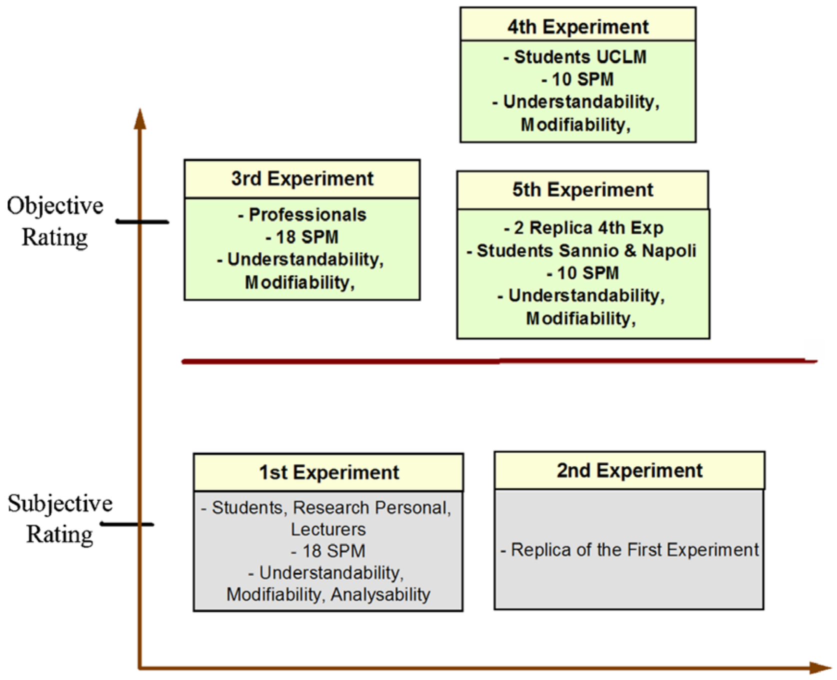

4th Experiment
- Students UCLM
- 10 SPM
- Understandability, Modifiability,
Objective Rating
3rd Experiment
- Professionals
- 18 SPM
- Understandability, Modifiability,
5th Experiment
- 2 Replica 4th Exp
- Students Sannio & Napoli
- 10 SPM
- Understandability, Modifiability,
Subjective Rating
1st Experiment
- Students, Research Personal, Lecturers
- 18 SPM
- Understandability, Modifiability, Analysability
2nd Experiment
- Replica of the First Experiment

Fig. 7. Overview of the experiments family.

Table 2
Summary of the results of the experiments family

Experiments	Subjects	No. of subjects	No. of mod	Dependent variables	Measurement of dependent variables	Empirically validated measures
1st	Professors, researchers, students	20	18	Understandability (U) Analysability (A) Modifiability (M)	Subjective rating of subjects (U, A, M) Understandability time (UT)	**U, A, M**: NA, NWP, NDWPIn, NDWPOut, NDPT, NDA **UT**: NA, NWP, NDWPIn, NDWPOut, NDPT, NDA
2nd (replica of the 1st)	Professors, researchers, students	25	18	Understandability (U) Analysability (A) Modifiability (M)	Subjective rating of subjects (U, A, M) Understandability time (UT)	**E, A, M**: NA, NWP, NDWPIn, NDWPOut, NDPT, NDA, RRPA (only A) **UT**: NA, NWP, NDWPIn, NDWPOut, NDPT
3rd	Professionals	29	18	Understandability (U) Modifiability (M)	Understandability time (UT) Modifiability time (MT)	**UT:** NA, NWP, NDWPIn, NDWPOut, NDPT, NDA **MT**: –
4th	Students	86	10	Understandability (U) Modifiability (M)	Understandability time (UT) Modifiability Time (MT)	**UT**: NA, NWP, NDWPIn, NDWPOut, NDPT, NDA, NCA **MT:** NWP, NDWPIn, NDWPOut, NDPT
5th (replica of the 4th)	R1 Students	26	10	Understandability (U) Modifiability (M)	Understandability time (UT) Modifiability time (MT)	**UT**: NWP, NDWPIn, NDWPOut, NDPT **MT**: NA, NWP, NDWPIn, NDWPOut, NDPT, NDA
	R2 Students	38	10	Understandability (U) Modifiability (M)	Understandability Time (UT) Modifiability time (MT)	**UT**: NA, NWP, NDWPIn, NDWPOut, NDPT, NDA, NCA **MT**: NWP, NDWPIn, NDWPOut, NDPT

defined and validated. These measures can be very useful to select the models with the most easiness of maintenance among various alternatives in companies: changing their SPMs helps improve their software processes. It facilitates the software processes evolution by assessing the process improvement at conceptual level.

The application of the FMESP framework provides companies with a suitable support in order to fulfil the requirements of higher maturity levels in their processes, namely

- *Process definition support*, by means of which the company has institutionalized their software processes and can manage their improvement and consequent evolution. It provides the means necessary to satisfy the general goal of a CMMI level 3: "Institutionalize a defined process".
- *Process measurement support*. The FMESP framework includes a systematic approach to measure all the necessary relevant artefacts related with software processes in a consistent and integrated way. It supports the measurement of projects, products and processes by providing a suitable measurement metamodel and a flexible method to measure these kinds of artefacts at metamodel scope. Therefore, it provides companies with the initial support necessary to go advance in the management of the measurement process and it could enable them to satisfy the CMMI measurement-related key process areas (levels 3–5) [14].

Some important ideas and improvements to tackle in future works have been detected according to the following categories:

- *Measurement metamodel.* It would be convenient to develop a graphic notation for the representation of measurement models according to the measurement metamodel proposed. As a result, a software tool which extends the functionality of GenMETRIC should be incorporated to the SEE of FMESP.

- *SEE*. The tools developed must be refined in order to develop a robust set of tools which can be used in an industrial environment.
- *Improvement based on knowledge management*. One important future line identified is the knowledge management application to promote the software process improvement. The FMESP framework is suitable for this application because it provides an open repository in which the meta-data related with the definition and measurement of software processes are included. This information can be used by a system of intelligent agents in order to derive improvement actions. We are building a prototype in this sense [22].
- *SPM measures:*
  – Carry out new families of experiments focused on the evaluation of concrete measures we consider relevant (NPR, NCA) and that according the results obtained in the empirical study does not seem to be clearly correlated with the maintainability of software process models.
  – Carry out study cases using real software process models.
  – Develop new empirical studies to find out if the SPMs complexity has influence in the project execution results.

Acknowledgments

This research has been partially supported by the projects: ESFINGE (Dirección General de Investigación of the Ministerio de Educación y Ciencia, TIN2006-15175-C05-05; MECENAS (Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, PBI06-0024); and ENIGMAS (Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, PBI-05-058).

References

[1] V. Basili, H. Rombach, The TAME project: towards improvement-oriented software environments, IEEE Transactions on Software Engineering 14 (6) (1988) 728–738.
[2] V. Basili, F. Shull, F. Lanubile, Building knowledge through families of experiments, IEEE Transactions on Software Engineering 25 (4) (1999) 435–437.
[3] L. Briand, J. Wüstn, H.A. Lounis, Comprehensive investigation of quality factors in object-oriented designs: an industrial case study, Technical Report ISERN-98-29, International Software Engineering Research Network, 1998.
[4] G. Canfora, F. Garcia, M. Piattini, F. Ruiz, C.A. Visaggio, A family of experiments to validate metrics for software process models, Journal Systems and Software 77 (2005) 113–129.
[5] G. Canfora, F. Garcia, M. Piattini, F. Ruiz, C.A. Visaggio, Applying a framework for the improvement of the software process maturity in a software company, Journal Software Practice and Experience 36 (3) (2006) 283–304.
[6] M. Ciolkowski, F. Shull, S. Biffl, A family of experiments to investigate the influence of context on the effect of inspection techniques, in: Proceedings of the 6th International Conference on Empirical Assessment in Software Engineering (EASE), Keele (UK), 2002, pp. 48–60.
[7] J. Cook, A. Wolf, Software process validation: quantitatively measuring the correspondence of a process to a model, ACM Transactions on Software Engineering and Methodology 8 (2) (1999) 147–176.
[8] B. Curtis, Maintaining the software process, in: Proceedings of the International Conference on Software Maintenance (ICSM), IEE Computer Society Press, Orlando, Florida, 1992, pp. 2–8.
[9] W.A. Florac, A.D. Carleton, Measuring the software processStatistical Process Control for Software Process Improvement, Addison Wesley, Boston, 1999.
[10] A. Fuggetta, Software process: a roadmap, in: Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, 2000, pp. 25–34.
[11] F. García, F. Ruiz, M. Piattini, M. Polo, Conceptual architecture for the assessment and improvement of software maintenance, in: Enterprise Information Systems IV, Kluwer Academic Publishers, Hingham, 2003, pp. 219–226.
[12] F. García, F. Ruiz, J. Cruz, M. Piattini, Integrated measurement for the evaluation and improvement of software processes, in: Proceedings of the 9th European Workshop on Software Process Technology (EWSPT'9), Lecture Notes in Computer Science 2786 (2003) 94–111.
[13] F. García, M.F. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, M. Genero, Towards a consistent terminology for software measurement, Information and Software Technology 48 (8) (2006) 631–644.
[14] D. Goldenson, J. Jarzombek, T. Rout, Measurement and analysis in capability maturity model integration models and software process improvement, CROSSTALK The Journal of Defense Software Engineering 6 (7) (2003) 20–24.
[15] T. Gruber, Towards principles for the design of ontologies used for knowledge sharing, International Journal of Human-Computer Studies 43 (5/6) (1995) 907–928.
[16] Meta Object Facility (MOF) Specification, version 1.4, Object Management Group (OMG), 2002.
[17] L.J. Osterweil, Software process are software too, revisited, in: Proceeding of the 19th International Conference on Software Engineering (ICSE), ACM Press, 1997, pp. 540–548.
[18] G. Poels, G. Dedene, Distance-based software measurement: necessary and sufficient properties for software measures, Information and Software Technology 42 (1) (2000) 35–46.
[19] F. Ruiz, M. Piattini, F. García, M. Polo, An XMI-based repository for software process metamodeling, in: Proceedings of Product Focused Software Process Improvement Conference (PROFES'2002), Lecture Notes in Computer Science 2559 (2002) 546–558.
[20] Software Process Engineering Metamodel Specification (SPEM); adopted specification, version 1.0, Object Management Group, November 2002.

[21] C. Tautz, C.G. Von Wangenheim, REFSENO: a representation formalism for software engineering ontologies, version 1.1. 015.98/E. Fraunhofer IESE, 1998.

[22] A. Vizcaino, J. Favela, M. Piattini, F. García, Supporting software maintenance in web repositories through a multiagent system, in: Proceedings of Atlantic Web Intelligence Conference (AWIC'03), Lecture Notes in Computer Science 2663 (2003) 307–317.

[23] XML Metadata Interchange (XMI) Specification, version 1.2, Object Management Group (OMG), 2002.

Félix García holds M.Sc. and Ph.D. degrees in Computer Science from the University of Castilla–La Mancha (UCLM). He is assistant Professor at the Department of Computer Science at UCLM and member of Alarcos Research Group. He is the author of several papers and book chapters on software processes management, from the point of view of their modeling, measurement and technology. His research interests are business process management, software processes, and software measurement.

Mario Piattini holds M.Sc. and Ph.D. degrees in Computer Science from the Polytechnical University of Madrid, and an M.Sc. in Psychology from the UNED. He is a Certified Information System Auditor and Certified Information Security Manager from ISACA (Information System Audit and Control Association). He is Full Professor at the Department of Computer Science at the University of Castilla–LaMancha in Ciudad Real, Spain. He is the author of several books and papers on databases, software engineering and information systems. He leads the Alarcos research group specialized in information system quality. His research interests are software quality, advanced database design, metrics, software maintenance, information system audit, and security.

Francisco Ruiz is Ph.D. in Computer Science for the University of Castilla-La Mancha (UCLM) and M.Sc. in Chemistry–Physics for the Complutense University of Madrid (Spain). He is full time associate professor of the Department of Computer Science at UCLM in Ciudad Real (Spain). He has been Dean of the Faculty of Computer Science between 1993 and 2000. Previously, he was Computer Services Director's in the mentioned university (1985–1989) and he has also worked in private companies like analyst-programmer and project manager. He is sub-director of Alarcos research group (http://alarcos.inf-cr.uclm.es/english/). His current research interests include business process modeling, management and measurement, software process technology and modeling, software maintenance, and methodologies to software projects planning and managing. He has been member of nine program committees and seven organizing committees. He belongs to several scientific and professional associations: ACM, IEEE-CS, ISO JTC1/SC7, EASST.

Gerardo Canfora is a full professor of computer science at the Faculty of Engineering and the Director of the Research Centre on Software Technology (RCOST) of the University of Sannio in Benevento, Italy. He serves on the program committees of a number of international conferences. He was a program co-chair of the 1997 International Workshop on Program Comprehension, of the 2001 International Conference on Software Maintenance, and of the 2004 European Conference on Software Maintenance and Reengineering; he was the General chair of the 2003 European Conference on Software Maintenance and Reengineering. His research interests include software maintenance, program comprehension, reverse engineering, workflow management, metrics, and experimental software engineering. He serves on the Editorial Board of the IEEE Transactions on Software Engineering and The Journal of Software Maintenance and Evolution. He is a member of the IEEE and the IEEE Computer Society.

Aaron Corrado Visaggio obtained his Ph.D. in Software Engineering at the University of Sannio, Italy in the 2005. He had his degree in Electronic Engineering at the Politecnico of Bari, Italy, in 2001. He developed his master thesis at the Fraunhofer IESE, Kaiserslautern, Italy, in the field of Software Process Modeling. He currently works as a researcher at the Research Centre on Software Technology (RCOST), University of Sannio, Benevento, Italy. His main research interests are empirical software engineering, agile methods, software process modeling and management, knowledge management applied to software engineering.