ACS/IEEE International Conference on

Computer Systems and Applications

Tunis, Tunisia
14-18 July, 2003

IEEE Catalog Number: 03EX722
ISBN: 0-7803-7983-7
Library of Congress: 2003106612

Search This CD-ROM
CD-ROM Help

EXIT

Metamodeling and Measurement for the Software Process Improvement

Félix García, Francisco Ruiz, Mario Piattini
Alarcos Research Group. Department of Computer Science
University of Castilla-La Mancha
Paseo de la Universidad, 4
13071- Ciudad Real (Spain)
{Felix.Garcia, Francisco.RuizG, Mario.Piattini}@uclm.es

Abstract

Software development and maintenance organizations are more and more concerned about software process assessment and improvement when they are promoting the improvement in the quality of their final products. Software process assessment and improvement are very complex activities due to the great number of different aspects to be considered. In order to manage this complexity, it is useful to establish a conceptual architecture which includes all the aspects necessary. This conceptual architecture must involve the definition of the metamodels and models necessary in order to carry out an assessment and improvement process in an effective and integrated way. In this paper we present a conceptual architecture of four abstraction levels to represent and manage the assessment and improvement of the software process. The management of the concepts involved in the conceptual architecture is provided by MANTIS-Metamod, which is a component of MANTIS, an integral environment for the management of the Software Maintenance Process (SMP).

Keywords

Assessment, Improvement, Conceptual Architecture, Software Process, Measurement

1. INTRODUCTION

Research into software processes has undergone a boom over the past few years mainly due to the growing complexity of software systems. Software applications are very complex products and, therefore, difficult to develop and maintain. Any organization devoted to producing and maintaining software follows a definite process. Therefore, the software process is a process, but with particular characteristics stemming from the special complexity of the software products obtained. In fact, the final quality of a software product is directly related to the way in which it is developed and maintained. This is encouraging software development and maintenance organizations to concern about improving processes when they are promoting improvement in the quality of their products. To this end, a few different initiatives to establish a reference framework for organizations to improve their products have come about. The most noteworthy are CMM [18], CMMi [19], the ISO 15504 [5;6;7] standard and, given its importance, improvement has been included in the new family of ISO 9000:2000 standards [9;10]. These proposals promote the adoption of a focus based on processes when a quality management system is developed, implemented and improved.

In order to treat with the complexity of software processes, first, it's necessary to know all of the elements involved. The software process may be defined as: the conjunction of coherent policies, organizational structures, technologies, procedures and artifacts which are necessary in order to conceive, develop, package and maintain a software product [2].

Given the diversity of elements involved in a software process, the definition and management of software processes in a organization isn't a trivial task. One of the basic elements for successful management of software processes is to define them correctly, based on the notion of software process model. Numerous initiatives have sprung up on this notion. The modeling of software processes has become a very acceptable solution for dealing with the inherent complexity of the whole software process. In related literature, one can find diverse languages and modeling formalities, known as "Process Modeling Languages" (PMLs), which are aimed at precisely and clearly representing the different elements related to a software process. In general, the following elements, (which are general concepts with different notations and terms) can be identified in a software process in the different MPLs [2]: Activity, Product, Resource and Organizations and Roles.

Faced with the diversity of existing process modeling proposals, the need for a standard software process metamodel becomes apparent. This metamodel can serve as a common reference and should include all of the aspects needed to define, as semantically as possible, the way in which the software is developed and maintained. With this aim in mind, the Object Management Group has recently proposed the SPEM metamodel (Software Process Engineering Metamodel Specification) [14], which constitutes a language for the creation of models for concrete processes in an organization. This language is within the conceptual framework of the OMG four level architecture based on the

0-7803-7983-7/03/$17.00 ©2003 IEEE

MOF standard [13]. It allows the effective management of the different concepts related to software processes on different levels of abstraction.

On the other hand, to support the integrated measurement in an organization, it's necessary to establish a measurement framework in order to provide a common reference for the definition and management of the different measures used for the assessment process. To achieve this, it's necessary the definition of a measurement metamodel, from which it could be possible to derive concrete measurement models.

In this article we describe a conceptual framework based on the four conceptual level architecture of OMG for the assessment and improvement of the software process. First, a general view of the architecture is given. In Section 3 the SPEM metamodel for the definition of process models is described. In the next section, the importance of incorporating the measurement aspects with the definition of a metamodel which constitutes the common reference in the organizations that want to improve their processes is analyzed. A generic metamodel for process and product measurement must be established to carry this out. In Section 5 a concrete model for the assessment and improvement of the software process is described as a instance of the SPEM metamodel. In the next section a tool for automatic management of this conceptual architecture is described. Finally, the conclusions and related works are presented.

2. CONCEPTUAL FRAMEWORK

The proposed conceptual framework is within the framework of the MANTIS project [16], whose aim is the definition and construction of an environment for the integral management of the software maintenance process (SMP). MANTIS defines the way in which SMP is organized, managed, measured and supported. This framework is extensible to any software process.

For the management of SMP, MANTIS integrates, among other aspects, the people (with certain skills who carry out certain roles in the project) the techniques (or methodologies) used by the people, the tools and activities (in which the teams participate and that help them to be familiar with significant targets).

Aiming to integrate all of these concepts (inherent in all software processes) in MANTIS, a four level conceptual architecture is defined, based on the MOF standard (Meta Object Facility) for metamodeling based on object technology [13] proposed by the Object Management Group (OMG). The aim of MOF is to specify and manage metadata at different levels of abstraction. MOF describes an abstract modeling language (based on the core of UML). In Table 1 the conceptual architecture of the MOF standard and its application in the MANTIS environment for improving the SMP, and which is extensible to any software process, is shown:

Table 1. Conceptual Levels in MOF & MANTIS

Level	MOF	MANTIS
M3	MOF-model (Meta-Metamodel)	*MOF-model*
M2	Meta-model	Software Process Engineering Metamodel (SPEM)
M1	Model	Model of Assessment and Improvement of SMP based on ISO 15504
M0	Data	Instances of Software Process Assessment (real-world concrete Software assessment projects)

In the lower level of the architecture, M0, the results of the application of an assessment and improvement process to a concrete software maintenance project are found. Therefore, the data that represent the results of applying an assessment process (and subsequent improvement) to maintenance projects are found on this level. These results make it possible to deduce the strong and weak points of the maintenance process of an organization, fundamental for applying improvement plans. The data managed at this level are instances of the data represented in the next level up, M1. The process model used in level M1 is based on the assessment and improvement model proposed in the ISO 15504 standard. Therefore, this assessment and improvement model makes up a process model and is represented using an abstract language for the definition of any concrete software process. In MANTIS, SPEM is used as a process metamodel. This metamodel contains the constructors needed to define any concrete software process model, and therefore, it includes the abstract concepts needed to define any element of the assessment model of ISO 15504. For example, the generic concept of "activity" of SPEM is enacted in the concepts "Present the assessment results" or "Derive a plan of action" pertaining to the concrete model from level M1 (see Section 5).

In the last conceptual level of MANTIS, M3, all of the metamodel process concepts are represented using the abstract language MOF, which is basically made up of two constructors: Class-MOF and Association-MOF (these are the principle elements from our point of view, although others do exist: package, for reuse, types of data, etc.). In this way, all of the concepts from level M2 are instances of Class-MOF or Association-MOF. For example, the concepts of M2 Activity, Work Product are instances of Class-MOF, and the relations "Activity precedes Activity", or "Work Product precedes Work Product" are instances of Association-MOF.

The other fundamental element of this conceptual architecture is the possibility of representing these metadata, structures at different levels of abstraction, in accordance with a

format which facilitates exchange. With this goal and to promote the portability of MANTIS, XMI (XML Meta-data Interchange) [15], a language based on XML (eXtensible Markup Language) [20], is used to openly represent the defined metamodels and models according to the conceptual architecture of MOF.

3. SOFTWARE PROCESS ENGINEERING METAMODEL (SPEM)

SPEM (Software Process Engineering Metamodel) is a OMG specification in the final phase in which a generic metamodel is described for the description of concrete software processes. This metamodel is based on UML [12], and therefore, on the principles of object orientation and it serves as a template for creating concrete process models, such as: Rational Development Unified Process (RUP), the ISO 15504 assessment and improvement model, etc.. In the specification SPEM is defined like a software process metamodel and like a UML profile. Like a metamodel SPEM specifies the minimal set of elements needed to describe any concrete process of software development, without including constructors for specific areas or disciplines, making SPEM a generic metamodel. The main objective of this specification is to attempt to make the already existing diverse terminology in the languages of software process modeling more homogeneous, since the same concepts are sometimes referred to by different names.

The conceptual model of SPEM is based on the idea that a software development process consists of the collaboration between abstract and active entities, referred to as process roles, that carry out operations, called activities, on tangible entities, called work products.

The SPEM specification is composed of a set of packages in which each of their elements is described. All of these packages are constructed through the SPEM_Foundation package, a sub-set of UML 1.4, and the SPEM_Extensions_Package, which adds on the constructors and the semantics needed for the software process engineering. SPEM is basically structured in 5 packages that are:

- **Basic Elements**. In this package the basic elements needed to describe processes are included They are: *External Description*, which contains a description of the elements of the model and *Guidance* which are associated with each model element and provide more detailed information for the process performers.
- **Dependences**. This package contains the following dependences defined in SPEM: *Categorizes*, which is a relation directed from a determined package to a process element corresponding to another package. It provides a means for associating multiple categories to the process elements; *Impacts* that relates work products, and indicates that one work product could invalidate another; *Import*, which has the same semantics as the Import element of ULM, with the difference being that all of the elements have public visibility in SPEM; *Precedes*, which is a relationship between activities or between work definitions and it indicates "beginning-beginning", "end-beginning" or "end-end" dependences between these elements depending on the value of their *kind* attribute; *Refers to* that relates process elements and indicates whether or not they are included in the same process component; and *Trace* which is a relationship between Work Definitions and it has the same semantics as the Trace relation in UML.
- **Process Structure**. The main structural elements through which a process description is constructed are described in this package. In the Figure 1 the components of this package are represented. The main elements of this package are: *WorkProduct* or artifact, which is anything that is produced, consumed or modified by a process. It could be a document, a model, source code, etc..; *Work Definition* which is an not abstract operation that describes the work carried out in a process. They have explicit inputs and outputs referred to via *Activity Parameter* constructor; *Activity* which is the main subclass of Work Definition and it describes a part of work carried out by a process role such as tasks, operations and actions that a determined role carries out or helps. An activity can be made up of a set of atomic elements called *Steps*; *Process Reformer*, which describes the one responsible for carrying out a set of Work Definitions that make up a process and Process Role that is a subclass of Process Performer and it describes the responsibilities associated to Work Products and defines the roles that are carried out and help in specific activities.

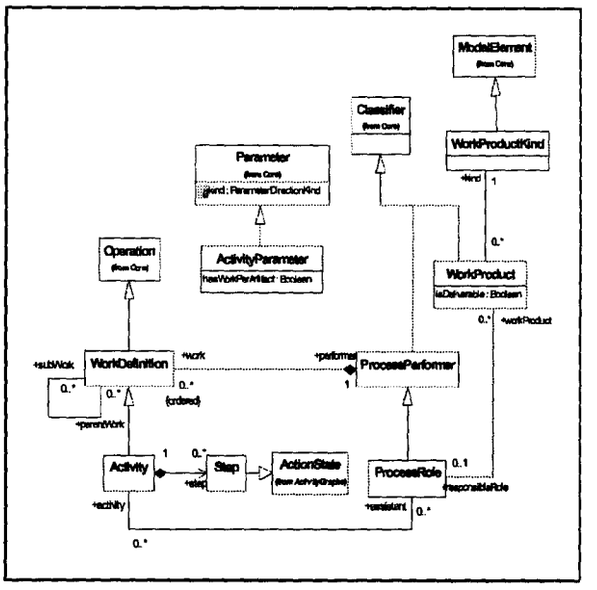

Figure 1. SPEM Structure Process Package

- **Process Components.** This package contains the elements needed to divide one or more process descriptions into self-contained parts upon which configuration management processes or version controls can be applied.
- **Process Life Cycle.** This package includes the process definition elements that help to define how the processes will be executed. They describe or restrict the behavior of the process to be carried out and they are used to help plan, execute and monitor the process. With these elements the order of the execution process is established and it is possible to define the iterations and phases of the process.

In short, SPEM makes the software process integrated management within the proposed conceptual architecture easier, since the concepts of the different models are grouped under a common terminology.

4. Incorporation of measurement in the conceptual architecture

A fundamental element to be taken into consideration when modeling processes is the possibility of defining objective indicators that allow a software organization to assess and improve its processes in an efficient way. To do so, it is vital to establish a framework of process measurement. A good base for developing a measurement process is the one provided by the standard ISO 15504. In Figure 2 the framework of process measurement of the ISO 15504 norm is presented.

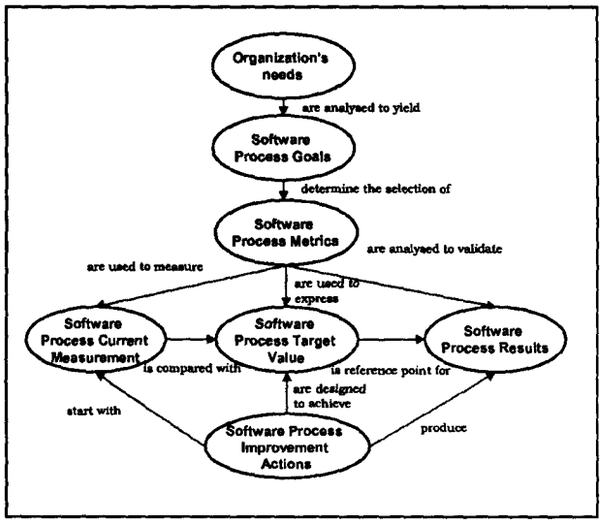

Figure 2 Framework of Software Process Measurement according to ISO 15504

As can be observed in the Figure 2, it is necessary to define process metrics to satisfy certain business objectives and they are used to measure the needs of an organization. These metrics make up the objective base to be able to quantitatively indicate what can be expected from a certain process. In this manner, it is possible to know in function of the value of these associated indicators whether or not a process improvement will be needed.

Besides, it is very important for an organization wishing to introduce an effective measuring process to be able to precisely define concrete measuring models. These, when supported by concrete tools, allow for the appropriate and necessary automation for process assessment. The majority of the problems in collecting data on a measurement process are mainly due to a poor definition of the software measures being applied. Therefore, it is important not only to gather the values pertaining to the measurement process, but also to appropriately represent the metadata associated to this data. In Kitchenham *et al.* [11] a method for the specification of measurement models is defined with the aim of capturing the definitions and relationships between software measurements. The proposed framework is made up of three levels of abstraction for measurement, starting form a generic measurement model and moving up to automation of the gathering of metric values on a project level. This idea of abstraction is fundamental to be able to integrate the measurement process effectively into the organization.

Therefore, it is very convenient to introduce a generic meta model for measurement, making it possible to derive concrete measurement models that make up the base for assessment and improvement processes in an organization. In Figure 3 our proposal for a measurement metamodel based on the ISO 15939 [8] standard is represented in UML.

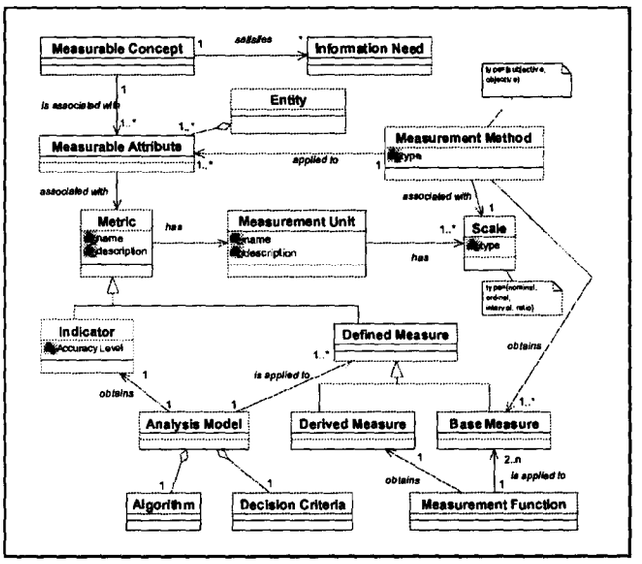

Figure 3. Generic Metamodel for the Measurement Process

As can be observed in Figure 3, from the point of view of measurement, the elements on which properties can be measured are "Entities". An entity is an object (for example, a process, product, project or resource) that can be characterized through the measurement of its "Measurable Attributes" which describe properties or characteristics of entities, which can be distinguished quantitatively or qualitatively by human or automatic means. The aim of attributes is to satisfy specific information needs such as, "the need to compare software development productivity with respect to a determined value". This abstract relation between attributes and information needs is represented by the element called "Measurable Concept", that, in this case, would be "productivity ratio of software development". As measurable attributes, attributes of the developed product size or of development effort could be used.

All measurable attributes are associated to a metric, which is an abstraction of the different types of measurements used to quantify and to make decisions concerning the entities. All metrics are associated to a unit of measure (for example, code lines), which at the same time pertain to a determined scale. In accordance with the standard, the 4 scales distinguished are: nominal, ordinal, interval and ratio, although other classifications can be established like in Kitchenham *et al.* [11]. The three types of metrics are:

- **Base Measurement,** defined in function of an attribute and the method needed to quantify it (a measurement is a variable to which a value is assigned).
- **Derived Measurement,** a defined measurement in function of two or more values of base measurements.
- **Indicator,** a measurement that provides an estimate or assessment of specific attributes derived from a model with respect to information needs. The indicators are the base for analysis and decision making. These measurements are the ones that are presented to the users in charge of the measurement process.

The procedures for calculating each of the metric types are:

- The values of the base measurements are reached with ***"Measurement Methods"*** that consist of a logical sequence of operations, generically described, used to quantify an attribute with respect to a specific scale. These operations can imply activities such as, counting occurrences or observing the passing of time. The same measurement method can be applied to multiple attributes.
- The derived measurements are obtained by applying a ***"Measurement Function"***, which is an algorithm or calculation carried out to combine two or more base measurements. The scale and unit of the derived measurement depend on the scales and units of the base measurements.
- The indicators are obtained with an ***"Analysis Model"***. An analysis model produces estimates and assessments relevant to the defined information needs. It consists of am algorithm or calculation that combines one or more base measurements and/or derivates with determined decision-making criteria. All decision-making criterion is composed of a series of limit values or used objects for determining the need to research, or to describe the confidence level with regard to a determined result. These criteria help to interpret the measurement results.

Using this reference metamodel it is possible to measure any element of a process or data model. For example, metamodel elements are not explicitly identified in SPEM (only one metric can be included as a kind of *Guidance*) for measurement. Thus, with the incorporation of this measurement extension to the SPEM metamodel, the possibility of specifying which metrics will be associated to the process characteristics (activities or work products or some other measurable entity) arises for defining concrete process models. If we apply this metamodel to the elements of the SPEM model, we could measure important elements like the class *Activity,* and the classes *Work Product and Process Performer.* These elements of the model have a set of measurable attributes, such as for an activity: *"the number of activities with which there is a precede type dependence"*. This attribute would be calculated with a metric to satisfy an information need like *"Evaluate the software process coupling"* and, in this case, the unit of measure would be of *"Ratio"* type. In this way, the work of an assessment and improvement process is eased, since the fulfillment of the software processes carried out in a determined organization are quantitatively registered.

5. Model for the Software Process Assessment and Improvement

Once a process metamodel has been defined, it is possible, based on this metamodel, to describe concrete process models in an organization. In this work a concrete model for assessment and improvement based on the ISO 15504 [5;6;7] standard is proposed. This norm defines a bi-dimensional reference model for describing processes through the assessment of their attributes, which are structured at different capacity levels. The dimensions considered in the norms are: the process dimension, in which the measurable objectives that should reach each process and their functionality are defined, and the capability dimension, based on assessing a set of attributes associated to each process to determine its capacity, necessary for its management and improvement. ISO 15504 provides the guide for carrying out a software assessment process and a method for continuously carrying out software improvements.

So that an organization can improve its processes, their strong and weak points must be previously identified, making necessary an assessment process. The ISO 15504 norm provides the guide needed for an organization to carry out assessment and improvement processes. The use of this guide along with the definition of effective measurement models greatly facilitates the execution of an assessment and improvement process. In the Figure 4 the relationship between these processes according to ISO 15504 shown:

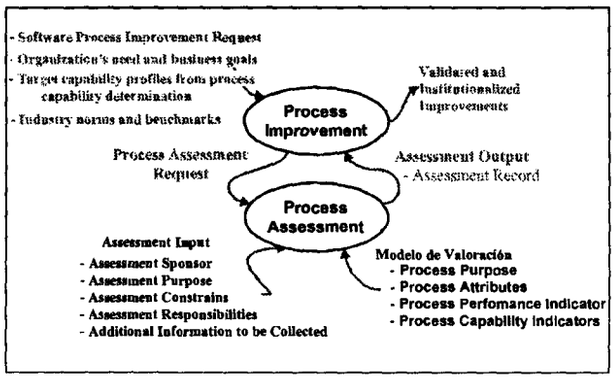

Figure 4. Software Process Assessment and Improvement according to ISO 15504

There is a strong relation between the assessment and improvement processes. For an organization to apply an improvement plan, it must first develop an assessment model that allows for identification of the aspects of the processes that need to be improved. Next, the elements of the assessment and improvement model proposed in this paper in accordance with the SPEM metamodel will be briefly represented in the Table 2:

Table 2. Mapping between SPEM and Concrete Assessment and Improvement Model based on ISO 15504

SPEM (M2)	M2 Instances (M1 Elements)
Lyfecycle	Assesment and Improvement ISO 15504
Phase	*Process Assessment* *Process Improvement*
Activity	Reviewing the assessment input Selecting the Process Instances Preparing for a team-based assessment Collecting and verifying information Determining the Actual Ratings for Process Instances Determining derived ratings Validating the ratings Presenting the assessment output Examine the organization's needs and business goals Initiate process improvement Prepare for and conduct a process assessment Analyse assessment output and derive action plan Implement improvements Confirm improvements
	Sustain improvement gains Monitor performance
WorkProduct	Organization's needs and business goals Assessment Request Assessment Input Process Profile Action Plan
Process Role	Owner of the Assessment Qualified Assessor
Process Role assists Activity	Owner of the Assessment **assists** Prepare for and conduct a process assessment Qualified Assessor **assists** Reviewing the assessment input
Process Role responsible of WorkProduct	*Qualified Assessor* ***is responsible of*** *Assessment Input*

As can be seen in the Table 2, the concrete elements that make up the proposed assessment and improvement model are defined with SPEM constructors.

This model provides the formal basis to carry out an effective assessment and improvement process and, with the integration of this model in the four level conceptual architecture, it's possible to adapt it easily and interchange it with other companies within the same domain.

6. Tool for the automatic management of the conceptual architecture

As support to this conceptual architecture the tool MANTIS-METAMOD [4] has been developed. The aim of this tool is effectively managing the metadata at different levels of abstraction, using an intuitive and simple way of working with the correspondences between different levels and being able to openly represent these metadata for exchange. The structure of the tool is the following:

For the entrance of user data, the main components of the application are composed by and model and metamodel administrator, and by window systems that allow for a visual description of the different projections of such models and metamodels. For example, the main elements complying with the projection between levels M3-M2 are a metamodel administrator and a windows system that permits a visual description of the classes which are the core of the MOF model (Package, Class, Type of Data, Attribute, Operation, Reference, Association, End of Association and Restriction). The metamodel administrator, as well as the associated windows system that allows us to describe the classes of the MOF metamodel can be seen in the Figure 5.

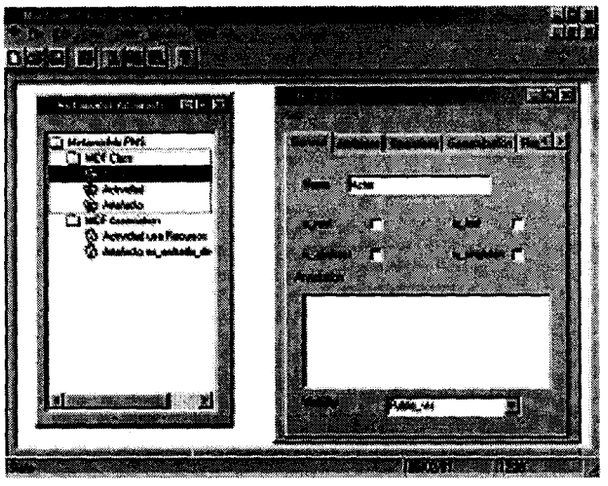

Figure 5. MANTIS-Metamod interface

The models defined by the user through the interface are validated and internally represented according to the hierarchy of classes of the MOF model.

To store the metadata the tool uses a metadata storage manager [17] which store them according to the XMI standard [15]. Using XMI, it is possible for the information on models and metamodels to be exchanged between tools that use this open format. This greatly increases the capacity for integration with other tools and environments. The storage manager is designed to:

- Store the MOF models defined in the tool in a metadata local repository in XMI
- Import metamodels

The repository manager provides the application with two basic services: storage and importation of metamodels that are stored in a metadata repository represented in XMI. With the use of XMI it's possible the interchange of the models (like the model of assessment and improvement proposed) and metamodels (like SPEM) with other organizations, which increases the capacity of the tool.

7. Conclusions and Further Works

In this paper a conceptual architecture based on the MOF standard for the effective management of the software process assessment and improvement has been presented. The assessment and improvement model based on ISO 15504 has been integrated into this architecture, defined with the SPEM metamodel constructors. Moreover, a generic measurement metamodel is proposed as an extension to the above-mentioned metamodel to incorporate the measurement process with the aim of providing an integrated framework for the assessment process. This architecture is being used in the MANTIS environment to apply the concepts of the software maintenance process assessment and improvement.

Given the diversity of the existing proposals on metamodeling of software processes [1], the need for a conceptual architecture on 4 levels that allows for effective and flexible incorporation of new elements to the defined metamodels or models is apparent.

Our main objective with the framework proposed is to provide integration support for two of the four key responsibilities of software process management [3] which are fundamental to improve the software process:

- **Define the Process**, for which a generic metamodel is used. With SPEM, companies can define their software processes in a standard way.
- **Measure the Process.** To provide an integrated framework for the measurement process a measurement metamodel based on ISO 15939 has been incorporated to the conceptual architecture. With this metamodel, a company could define all its concrete measurement models in an integrated way by using the same constructors.

Among future lines of research, it is worthwhile to mention:

- Refinement and improvement of a software process assessment model through definition, verification and validation of a set of software process models metrics.
- Description of concrete measurement models, through the generic metamodel of the measurement proposed, to effectively support software process assessment.
- Development of a tool for the management of the measurement process in an integrated way using the measurement metamodel based on ISO 15939 proposed.

Acknowledgements

This work has been partially funded by the TAMANSI project financed by "Consejería de Ciencia y Tecnología, Junta de Comunidades de Castilla-La Mancha" of Spain (project reference PBC-02-001) and by the DOLMEN project (Languages, Methods and Environments), which is partially supported by FEDER with number TIC2000-1676-C06-06.

REFERENCES

[1] Breton, E. y Bezivin, J. Process-Centered Model Engineering. 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2001). September 4-7, 2001. Seattle.

[2] Derniame, J.C., Kaba, B. A., and Wastell, D. Software Process: Principles, methodology and technology. *Lecture Notes un Computer Science 1500* . Springer, 1999.

[3] Florac, W.A. y Carleton, A.D. Measuring the Software Process. Statistical Process Control for Software Process Improvement. SEI Series in Software Engineering. Addison Wesley. 1999.

[4] Garcia, F., Ruiz, F., Piattini, M. and Polo, M. Conceptual Architecture for the Assessment and Improvement of Software Maintenance. 4th International Conference on Enterprise Information Systems (ICEIS'02). Ciudad Real, Spain, 610-617. 2002.

[5] ISO IEC 15504 TR2:1998, part 2: A reference model for processes and process capability, ISO/IEC JTC1/SC7, 1998.

[6] ISO IEC 15504 TR2:1998, Software Process Assessment – part 4: Guide to conducting assessment, ISO/IEC JTC1/SC7, 1998.

[7] ISO IEC 15504 TR2:1998, Software Process Assessment – Part 7 : Guide for use in process improvement, ISO/IEC JTC1/SC7, 1998.

[8] ISO IEC 15939, Information Technology – Software Measurement Process, Committee Draft, December 2000.

[9] International Organization for Standardization (ISO). 2000. Quality management systems - Fundamentals and vocabulary. ISO 9000:2000. See http://www.iso.ch/iso/en/iso9000-14000/iso9000/selection_use/iso9000family.html

[10] International Organization for Standardization (ISO). 2000. Quality management systems - Requirements ISO 9001:2000

[11] Kitchenham, B.A., Hughes, R.T. y Linkman, S.G. Modeling Software Measurement Data. IEEE Transactions on Software Engineering. 27(9), pp. 788-804. 2001.

[12] OMG Unified Modeling Language Specification; versión 1.4, septiembre 2001. Object Management Group. Available in http://www.omg.org/technology/documents/ formal/uml.htm.

[13] Meta Object Facility (MOF) Specification; versión 1.4, abril 2002. Object Management Group. In http://www.omg.org/technology/ documents/formal/mof.htm.

[14] Software Process Engineering Metamodel Specification; adopted specification, mayo-2002. Object Management Group. Available in http://cgi.omg.org/cgi-bin/doc?ptc/02-05-03.

[15] OMG XML Metadata Interchange (XMI) Specification; versión 1.2, enero 2002. Object Management Group. In http://www.omg.org/technology/documents/formal/xmi.htm

[16] Ruiz, F., Piattini, M. and Polo, M. (2001). An Conceptual Architecture Proposal for Software Maintenance. International Symposium on Systems Integration (ISSI, Intersymp'2001). Baden-Baden, Germany (2001), VIII:1-8

[17] Ruiz, F., Piattini, M., García, F. y Polo, M. (2002). A XMI-based Repository for Software Process Metamodeling. *Proceedings of Fourth International Conference on Product Focused Software Process Improvement* (PROFES'2002). Lecture Notes in computer Science. Rovaniemi (Finland). December 2002.

[18] Software Engineering Institute (SEI): The Capability Maturity Model: Guidelines for Improving the Software Process,1995. In http://www.sei.cmu.edu/cmm/cmm.html

[19] Software Engineering Institute (SEI): Capability Madurity Model Integration (CMMI^{SM}), version 1.1. March 2002. Software Engineering Institute (SEI). In http://www.sei.cmu/cmmi/cmmi.html

[20] W3C: Extensible Markup Language (XML) 1.0 (second edition), October, 2000. In http://www.w3.org.