ASAI
Simposio Argentino de
Inteligencia Artificial

ASSE
Simposio Argentino de
Ingeniería de Software

AST
Simposio Argentino
de Tecnología

EST
EST 2003
Trabajos Estudiantiles

JSL
Jornadas Sobre
Software Libre

SID
Simposio Argentino de
Informática y Derecho

SIO
Simposio Argentino en
Investigación Operativa

SIS
Simposio de Informática
y Salud

WAIT
Workshop Argentino de
Infomática Teórica

Acrobat Reader

32 JAIIO - 2003

Jornadas Argentinas de Informática e Investigación Operativa
Argentine Conference on Computer Sciencie and Operational Research

Organizado por

Sociedad Argentina de InvestigaciónOperativa

Sede

Universidad Argentina de la Empresa

1 al 5 de Septiembre de 2003

SEDE
Universidad Argentina de la Empresa
Lima 717 - C1073AAO
Buenos Aires, Argentina

Auspicios

COMITÉ ORGANIZADOR

Coordinador General
Alfredo Olivero

Comité Ejecutivo
Germán Guido Lavalle

Colaboradores Locales
Aliscioni, Pablo Danilo
Dellepiane, Paola
Diaz, Cristian Andres
Gonzalez, Ximena
Lifschitz, Martín S.
Martinez, Sandra
Pugliese, Irene
Zuñiga, Javier

Coordinación Ejecutiva
Pilar Suter
Alejandra Villa

Secretaría
Yanina Viegas
Florencia Weber

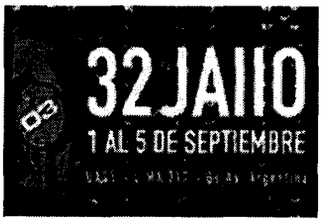
SADIO
CONSEJO DIRECTIVO

Presidente **Gabriel Baum**	

Vicepresidente **Horacio Leone** Secretario **Pablo Martinez Lopez** Tesorero **Alan March** Protesorero **Juan Carlos Fränkel**	Vocales Titulares **Alejandro Bianchi** **Pedro D'Argenio** **Marcelo Frías** **Federico Heinz** Vocales suplentes **Arnoldo Palma** **Nelson Sprejer**

Secretaría de SADIO

Uruguay 252 2º "D" (C1015ABF)
Ciudad Autónoma de Buenos Aires
Horario de atención: Lunes a Viernes de 12 a 18 hs.
Teléfono: 4371-5755 Tel/Fax: 4372-3950
E-mail: sadio@speedy.com.ar
Web: www.sadio.org.ar



ASAI
Simposio Argentino de
Inteligencia Artificial

ASSE
Simposio Argentino de
Ingeniería de Software

AST
Simposio Argentino
de Tecnología

EST
EST 2003
Trabajos Estudiantiles

JSL
Jornadas Sobre
Software Libre

SID
Simposio Argentino de
Informática y Derecho

SIO
Simposio Argentino en
Investigación Operativa

SIS
Simposio de Informática
y Salud

WAIT
Workshop Argentino de
Infomática Teórica

Acrobat Reader

ASSE 2003

Simposio Argentino de Ingeniería de Software

TITULO DEL TRABAJO	AUTOR/ES
A Fault Seeding Experience	Fabio Grigorjev, Natacha Lascano and Julio L. Staude
A proposal of a Software Measurement Ontology	Francisco Ruiz, Marcela Genero, Félix García, Mario Piattini a Calero
A systematic Approach to Generate Test Cases based on Faults	Marisa A. S´anchez and Miguel A. Felder
DesignBots: Towards a Planning-based Approach for the Exploration of Architectural Design Alternative	J. Andrés Díaz Pace, Marcelo R. Campo
Derivation of Zeus Concepts from the GAIA. Methodology to Develop Multi-Agent Systems	Josefina Povolo, Lorena Stuchi, Omar Chiotti
Dynamic Adaptive Marshalling A Dynamic Solution to Middleware Heterogeneity	Richard Slamkovic, George Fernandez, Jim McGovern
Improving the Quality of the Software through Aspects	Silvia de Castro Bertagnolli and Maria Lúcia Blanck Lisbôa
Information Integration Problem in B2B RelationShips - A state of the art	Ma. Laura Caliusco, Ma. Rosa Galli, Omar Chiotti
Managing Consistency and Conflicts in Real-time Collaborative Software Engineering	Felix Wong, Jim McGovern, and George Fernandez
Modeling of Mobile-Agent Applications with UML	Edgardo A. Belloni and Claudia A. Marcos
Solving Conflicts in Aspect-Oriented Applications	Jane L. Pryor and Claudia Marcos
Subjetive Consistency	Ing. Pedro E. Colla

A proposal of a Software Measurement Ontology

Francisco Ruiz, Marcela Genero, Félix García, Mario Piattini and Coral Calero

ALARCOS Research Group
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF CASTILLA - LA MANCHA
Paseo de la Universidad, 4
13071 - CIUDAD REAL – SPAIN
{Marcela.Genero, Francisco.RuizG,
Mario.Piattini,Felix.Garcia,Coral.Calero}@uclm.es

Abstract. Software measurement, as a young discipline, suffers from the symptoms of any unconsolidated discipline. Unfortunately, there does not yet exist a consensus about the terminology used in this field. Therefore, the first step towards software measurement become a well established engineering discipline is to get a common agreement among software measurement researchers and practitioners about the concepts used in software measurement and the relationships between them. To reach this goal, it is well-known the usage of ontologies, due to an ontology represents a certain view of an application domain in which the concepts that live in this domain are defined in an unambiguous and explicit way. So that, the aim of this is to present a preliminary, practical and semi-formal ontology for software measurement.

1. Introduction

Software measurement suffers from several symptoms of any relatively young discipline [4]. Software measurement is currently in the phase in which terminology, principles, and methods are still being defined and consolidated. However, nobody can dispute the relevant role that software measurement plays in software engineering. Software measures could be used:

- To build prediction systems for database projects [21].
- To understand and improve software development and maintenance projects [3].
- To maintain system quality, highlighting problematic areas [6].
- To determine the best ways to help practitioners and researchers in their work [26], etc.

Moreover software measures help to assess and institutionalise Software Process Improvement (SPI) in software-intensive organisations. In fact, in the most popular initiatives such as CMM, SPICE and CMMi measurement plays a fundamental role. The ISO 9000 standard emphasises the importance of measures for quality assurance and management.

Unfortunately, there does not yet exist a consensus about the terminology used in software measurement. Due to the sake of brevity we cannot enter in detail in all the discrepancies in software measurement terminology. As an example of this fact, we show the different nomenclatures used in the literature for the concepts "metric" and "observation".

Table 1. Different nomenclatures for the concepts "metric" and "observation"

Source	Metric	Observation
Our proposal	Metric or measure	Observation or measurement
[18]	Measure	Observation (only for base metrics)
[20]	Element Measure Type	Recorded Value
[2]	Not included	Value
[24]	Metric	Measure
[19]	Measure	Measurement point

Therefore, we believe that creating a common agreement about the concepts and terminology in this field is a good starting point for contributing to software management becoming a mature discipline. For this purpose it is necessary that software measurement researchers and practitioners reach a consensus about what we are talking about when we refer to "software measurement" or "software measurement process", "measure", "metric", "measurable attribute", "measurable object", etc.

In order to obtain this goal we decided to construct a common conceptualisation of software measurement domain, where objects, concepts, entities and their relationships can be explicitly represented. For this purpose, we designed a software measurement ontology, which enables explicit specification of a conceptualisation [13]. An ontology represents a certain view of an application domain in which the concepts that live in this domain are defined in an unambiguous and explicit way [7].

Our background in software measurement [5, 8, 11, 27] helped us to detect the main concepts we need to consider, including their relationships and their importance within software measurement.

The software measurement ontology we propose in this paper is part of a "super-ontology" for software maintenance, built within the MANTIS environment, a software maintenance engineering environment [31][1].

Moreover, ontology facilitates knowledge sharing, knowledge integration and knowledge reuse. All of these aspects are critical for MANTIS environment, since its goal is to promote the sharing and reusing of information and knowledge.

What we have said above led us to design a complete and semi-formal ontology following an adaptation of the REFSENO methodology (Tautz and Von Wangenheim, 1998), where both static and dynamic aspects [22] were clearly defined and specified.

---

[1] Ruíz et al. [31] provide an overview of each of the "sub-ontologies" that compose the "super-ontology" used in the MANTIS Environment, but does not provide details of the software measurementt ontology.

The ontology of software measurement we propose has been developed based on three main documents:

- The software measurement process of the ISO 15939 18].
- The conceptual model for representing collections of software data provided by Kitchenham et al. [20].
- The software measurement conceptual model proposed by Becker-Kornstaedt and Webby [2].

Other authors have demonstrated the importance of suitably defining the measurement process and the data and correspondent metrics. For example Olsina et al. [24] have recently presented a conceptual framework for the definition and exploitation of quality metrics. In this line is the measurement ontology proposed by Kim [19]. Even though Kim´s ontology is defined within a model for the quality of traditional industry based on the ISO 9000, some of its contributions can be transferred to the field of software measurement in software engineering.

Summarising the introduction, the main focus of this work is to present a practical and semi-formal ontology for software measurement which was designed using UML [23] as an ontology representation language and an adaptation of REFSENO [33], an ontology definition methodology.

The rest of this paper is organized as follows: Section 2 points out the importance of the definition of ontologies using methodologies that have been created for this purpose and provides a brief description of REFSENO. Our proposal of a software comes in Section 3. A software measurement metamodel, which allows the implementation of the proposed ontology, is presented in Section 4. Finally, the last section presents some concluding remarks.

2. Ontology Design

Many authors have pointed out the importance of formalising the definition and design of information systems with ontologies [14]. However, to design ontology it is advisable to follow a methodology suitable for this goal. Different methodologies and representations have been proposed. For instance, Hikita and Matsumoto [15] used a representation based on the first order predicate logic. Other authors prefer frame-based approaches, such as those that are used in Ontolingua [9], one of the most frequently used ontology languages.

We have used and adaptation of the REFSENO methodology, proposed by Tautz and Von Wangenheim [33]. Although other sets of techniques exist for developing ontologies [12, 32, 34, 35] they have not been applied to software engineering as REFSENO has. Moreover, we choose REFSENO for the following reasons:

- It allows the modelling of software engineering knowledge in a precise and complete manner, by using alternate representations. The ontologies specified using REFSENO are precise, since the semantic relationships are defined and are complete, in the sense that all conceptual knowledge necessary to instantiate an experience base are provided.
- It has a clear terminology, differentiating between conceptual and context-specific knowledge, thus enabling the management of knowledge from different contexts.

- It guarantees a consistent ontology since consistency criteria must be fulfilled.

Basically, the adaptation consists of the usage of ontology diagrams modelled by UML and the simplification of the REFSENO tables (glossary of concepts, attributes, relationships, etc.) eliminating the aspects related to computational inference.

2.1 A Brief Description of REFSENO

REFSENO provides constructs to describe concepts where each concept represents a class of experience items. Besides concepts, their properties (called terminal attributes) and relationships (non-terminal attributes) are represented. One relevant feature of REFSENO is that it enables us to describe similarity functions, which are used for similarity-based retrieval. In this way the methodology facilitates the implementation of the retrieval component. On the other hand, REFSENO also incorporates integrity rules such as: cardinalities and range of values for attributes, assertions and preconditions.

REFSENO extends the formalism of Ostertag, et al. [25] by additional integrity rules and by clearly separating the schema definition and characterisation. Althoff et al. [1] discovered from their previous experience that the storage of experiences in the form of documents (in contrast to codified knowledge) results in an important reduction of learning effort, typically associated with knowledge-based systems.

REFSENO is an improved adaptation of Methontology (Fernández et al., 1997; Gómez-Pérez, 1998), which imitates the software life-cycle proposed by the IEEE 1074 standard (IEEE, 1995). The main steps of REFSENO are:

1. Planning.
2. Specification of the ontology requirements.
3. Conceptualisation. (This step is similar to the phase of design in a software system, so it is the ontology itself).
4. Implementation. (In this step is where the previous conceptualisation is represented and stored by using computer tools).

3. Defining a Software Measurement Ontology

In this section we will begin providing a conceptual description of software measurement and an example, which allow a better understanding of the concepts introduced. Finally, in this section we will present our proposal of a software measurement ontology.

3.1 Conceptual Description of Software Measurement

In this section we present a description about what "to measure" means in this software measurement ontology, i.e. the structures and the existing relationships between the information needs of any measurable object (products, projects, processes and services) and its relevant elements (artefacts, activities, resources or agents) which allow to obtain the information products that satisfy such information needs.

- The selection and definition of metrics suitable for satisfying one information need start with a measurable concept, i.e. one idea of which are the measurable attributes related to the information need and how they are related.
- The value of an attribute of one element can be measured via one or more metrics (also called measures). Each observation or measurement[2] allows to obtain the value of an attribute in one instant or point in time, for one of the associated metrics. Each metric has a specific measurement unit, which belongs to a specific scale. Four types of scales exist: nominal, ordinal, interval and ratio. Each metric also has a default value, which is the only value known before performing any observation based on such a metric.
- The observations can be direct or indirect. The direct observations consist of using some of the base metrics associated with one attribute. Each base metric is defined by a specific measurement method (operations which establish a correspondence between the attribute and the scale associated with the metric). The methods can be subjective or objective.
  For performing indirect observations, we have derived metrics and indicators. The derived metrics are not directly usable to satisfy the information needs, while the indicators are usable to satisfy information needs. The value of a derived metric is obtained applying a measurement function to two or more values of base metrics.
- The indicators can have different accuracy levels. For this reason, the value of an indicator can be quantitative or qualitative. The value of an indicator is obtained applying an analysis model, i.e. an algorithm for combining the values of one or more defined metrics (derived or base metrics) using certain decision criteria.
- One interpretation consists of an explanation, understandable for the stakeholders, about the results of one observation of one indicator.
- One information product is the result of the measurement process, which satisfies one information need. It is composed of the collection of suitable interpretations.

3.2 An Example: To Update the Price of a Maintenance Service

We will present a concrete example of the previous conceptual description: One organisation (the maintainer) provides a software maintenance service. The maintainer performs such projects by means of an annual contract, which establishes the responsibilities of each party. When the contract is close to finish the maintainer has to decide if the incomes are in proportion to the effort spent on maintenance service.

The measurable concept is that the maintenance effort is in proportion to the size of the maintained software and with the amount of source code that has been modified. By using some base and derived metrics, an indicator of the amount of maintenance tasks required for product X is obtained. This indicator is used to generate an information product pointing out the maintenance costs during the previous year.

For this example the concepts previously presented can be instantiated in this way:

---

[2] We have chosen the nomenclature "observation" instead of measurement for avoiding the confusion with the concept of measurement as a process.

- *Information needs:* one director of the organization formulates a request by the question "Is the present price of the maintenance service of the product X suitable?"
- *Measurable concept*: "Maintenance effort".
- *Measurable object:* "A service", the maintenance service.
- *Measurable attribute*: "Size" (related to the elements of type "software components") and "complexity" (related to the elements of type "Request of maintenance").
- *Observations*: A direct observation of the "size" attribute using the metric "Lines of source code" must be carried out for the components of product X. Moreover, a direct observation of the "complexity" attribute using the metric "Number of modified lines of source code" must be done for each maintenance request for product X. Three indirect observations have to be performed: two for the two derived metrics and one for the indicator, which are detailed below.
- *Base metrics:* "Lines of source code of the C component" (for measuring the size of each component) and "Number of modified lines of source code for attending to the maintenance request P" (for measuring the complexity of each maintenance request).
- *Measurement methods:* To count the number of lines of source code using the tool SoftMetric (for measuring the size) and to count the modified lines of source code (i.e. those that have been changed, added or deleted) for a maintenance request using a configuration management tool. Both measurement methods are objective methods.
- *Scales:* Both measurement methods are associated with the scale of the integer numbers from 0 to infinity. It is a ratio scale.
- *Measurement unit*: LSC(lines of source code) for the first base metric and MLSC (modified lines of source code) for the second.
- *Derived metrics:* "Size of product X" and the "Amount of source code that have been modified for product X".
- *Measurement function:* Size(product X) = Sum(LSC of the components C of the product X), and Amount-modified-code(product X) = sum(MLSC of requests P for the product X).
- *Indicator:* "Amount of maintenance tasks", which indicates the number of person-hour spent on maintaining a software product. This indicator is an integer number.
- *Analysis model:* Based on the experience that the organization has in previous maintenance services the organization has drawn up a table that allows the calculation of the amount of maintenance tasks required for the product X, expressed as person-hour, based on the following values: "Product type of X", "Size of X" and "amount of modified code for X".
- *Decision criteria:* They are a set of rules that allow the classification of product X in one of the type of product the table has. Otherwise it can be used a weighted mean of the values of more than one type.
- *Interpretation*: The interpretation will include the present average cost for person-hour, the economical costs in Euros of the amount of maintenance obtained, which will be the product of the previous values.

- *Information product:* The economical managers prepare a report explaining in natural language the economical costs and the maintenance tasks that have been performed to the product X (expressed in person-hour) in the last year and the incomes that have been received for that service. By means of this report the directors of the organization can take the decision of modifying or not the price of such a maintenance service.

3.3 A Software Measurement Ontology

In figure 1 we present the UML class diagram of the ontology for software measurement. We have preferred to use UML class diagrams as representation language instead of the diagrams proposed in the REFSENO methodology, because UML is a widely used and well-known object oriented modeling standard that has recently been proposed as an ontology representation language [36].

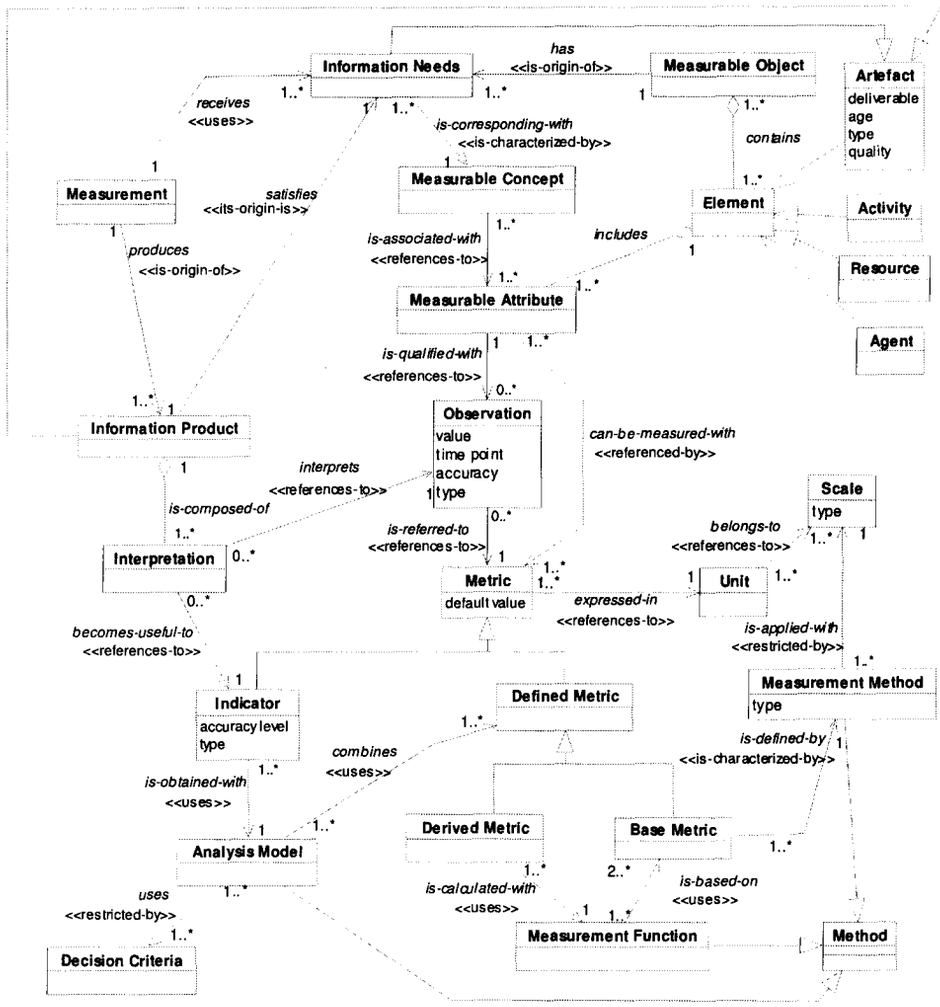


Fig. 1. A UML class diagram for the software measurement ontology

Hereafter the tables of the ontology are presented (an adapted version of REFSENO is used): Table 2 presents the glossary of concepts, table 3 is a table of attributes, and finally table 4 presents the relationships.

Table 2. Software measurement ontology: Glossary of concepts

Concept	Super-Concept	Description	Purpose
Activity	Element	Action that must be performed for achieving the goals of the measurable object.	To describe the task to be done.
Agent	Element	Person, organization or software product, which play an active role in the realization of a measurable object.	To identify the stakeholders of the measurable object.
Analysis Model	Method	Algorithm or calculus that combines one or more metrics defined with certain decision criteria to obtain the value of an indicator and its interpretation.	To establish a way to measure.
Artefact	Element	Part of a measurable object that is created or modified by the activities.	To define the structure of the measurable object.
Base Metric	Defined Metric	Metric defined based on an attribute and the measurement method for obtaining its value. Synonymous: Direct metric.	To measure.
Decision Criteria	Concept	Thresholds values, objectives or patterns used for determining the necessity of an action or for describing the level of goodness of a metric value. It helps to interpret the measurement results.	To measure.
Defined Metric	Metric	Metric for which exists an exact mathematic formula for obtaining its value. It can be a base metric or a derived metric.	To measure.
Derived Metric	Defined Metric	Metric defined based on two or more values of base metrics. It captures information about more than one attribute or about the same attribute for more than one element.	To measure.
Element	Concept	The elements of a measurable object can be artefacts, activities, resources or agents that can be characterised by measuring some of their attributes.	To manage the different types of elements of a measurable object.
Indicator	Metric	Metric that provides an estimation (evaluation) of some determined attributes, which respect to certain information needs. An indicator is the basis for making decisions and analysing a measurable object. The value of an indicator can be quantitative or qualitative.	To measure.
Information Needs	Artefact	Artefact of document type, which describes the necessary information for managing a measurable object.	To manage a measurable object.
Information Product	Artefact	Artefact of document type, which allows satisfying a certain information, need, including the interpretations of the observations of one or more indicators.	To manage a measurable object.
Interpretation	Artefact	Artefact of document type, which consists of an explanation of the result of an observation of an indicator for satisfying an information need.	To manage a measurable object.
Measurable Object	Concept	Object that can be measured, such as projects, processes, products and services. It is an aggregate of elements (artefacts, activities, resources and agents).	
Measurable Concept	Concept	Idea that establishes an abstract relationship between attributes and information needs. Examples: quality, performance, maturity, etc.	To identify objects to be measured.
Measurement Function	Method	Algorithm or calculus performed to combine two or more values or base metrics for obtaining the value of a derived metric. The unit and scale of a derived metric depend on which was the measurement function.	To establish the way of measuring.
Measurement	Concept	Process, which consists of performing the necessary activities for obtaining the information products which satisfies the information needs produced during the management of a measurable object.	To manage a measurable object.
Measurement Method	Method	Method, which consists of a logic sequence of operations, used to quantify an attribute respect to a certain scale. Measurement methods could be subjective (based on human judgement) or objective (based on numeric rules).	To establish the way of measuring.
Measurable Attribute	Concept	Property or characteristic of one element that can be distinguished both qualitatively and quantitatively.	To identify objects to be measured.
Method	Concept	Systematic process which includes the steps and heuristics for allowing the realization of one or more activities.	To define activities.
Metric	Concept	Variable associated with one or more attributes, which allows their qualification through observations (assignment of values). It can be a base metric, a derived metric or an indicator. Synonymous: measure.	To measure.
Observation	Concept	Action that happens in an instant or a point in time, which allows obtaining the value of an attribute, referred to a metric.	To measure.

Concept	Super-Concept	Description	Purpose
Resource	Element	Something, which is not of human characteristics which is necessary for performing an activity.	To manage resources.
Scale	Concept	Ordered set of values, continuous or discrete, or a set of categories to which a metric for an attribute is associated.	To measure.
Unit	Concept	Quantity, defined or adopted by convention, used for comparing quantities of the same type. Only quantities of the same type can be compared directly.	To measure.

Table 3. Software measurement ontology: Table of attributes

Concept	Attribute	Description	Cardinality
Artefact	Deliverable	Indicates if the artefact must be delivered to a client.	1
	Age	Number of years of existence.	1
	Type	Own characteristics of the artefact. For example: document, module, component, file, etc.	1
	Quality	Qualitative measure of the quality.	1
Indicator	Accuracy Level	A quantification of the uncertainty or accuracy inherent to the way of obtaining the value of an indicator.	1
	Type	Own characteristics of the value obtained for an indicator. It can be numeric quantitative or qualitative.	1
Measurement Method	Type	Indicate the own characteristics of the operations used for quantifying an attribute. There exist two types: subjective (quantification influenced by the human judgement) and objective (quantification based on numeric rules).	1
Metric	Default Value	Initial value that a metric has before any observation of this metric was done.	1
Observation	Precision	Indicator of the precision that a value of an observation has.	1
	Time Point	Moment or instant of time when the observation is performed.	1
	Type	Indicates if an observation is direct (associated to a base metric) or indirect (associated to a derived metric or to an indicator).	1
	Value	Quantitative or qualitative result assigned to a metric (base, derived or indicator). Synonymous: Data.	1
Scale	Type	Indicates the own characteristics of the relationship between the values of a scale. It can take four values: nominal, ordinal, interval and ratio.	1

Table 4. Software measurement ontology: Table of relationships

Name	Description
Becomes-useful-to	An interpretation provides utility on a concrete necessity to an indicator.
Belongs-to	One measurement unit belongs to the set of values of one or more scales (all of them of the same type).
Can-be-measured-with	A measurable attribute can be measured via several metrics.
Combines	An analysis model combines the values of one or more defined metrics (derived or base metrics) for obtaining the value of an indicator.
Contains	A measurable object can contain a collection of elements of different types, such as artefacts, activities, resources and agents.
Expressed-in	The results obtained with a metric are expressed in a measurement unit.
Has	A measurable object has some information needs that are necessary to satisfy its correct management.
Includes	A relevant element of a measurable object includes one or more measurable attributes.
Interprets	An interpretation interprets (explains) the result of an observation.
Is-applied-with	A measurement method is applicable with a specific scale.
Is-associated-with	A measurable concept is associated with one ore more measurable attributes.
Is-based-on	The measurement function used to calculate a metric value is based on the value of two or more base metrics.
Is-calculated-with	The value of a derived metric is calculated with a specific measurement function.
Is-qualified-with	A measurable attribute is qualified with one or more observations.
Is-composed-of	An information product is composed of one or more interpretations.

Name	Description
Is-corresponding-with	A concrete information need of a measurable object is corresponding with a measurable concept.
Is-defined-by	A base metric is defined based on the measurement method used to obtain its value.
Is-obtained-with	The value, the accuracy level and the interpretation of an indicator are obtained with a specific analysis model.
Is-referred-to	An observation obtains the value of an attribute referred to a metric.
Produces	The measurement process produces information products as results of its activities.
Receives	The measurement process uses as input of its activities the information needs of a measurable object.
Satisfies	An information product satisfies an information need.
Uses	One analysis model uses one or more decision criteria.

As we mentioned above this software measurement ontology is part of a "super-ontology" built within the MANTIS environment, a software maintenance engineering environment that consists of several sub-ontologies. Therefore it is almost impossible to avoid the duplicity of relationship names if we want to use self-explaining names. This fact does not offer problems if the relationships that have the same name have the same type of relationships (i.e. they have the same semantic).

Table 5 only shows those classes of relationships used in the software measurement ontology (in figure 1 are shown as stereotypes).

Table 5. Table of classes of relationships (Ruiz, 2003)

Relationship class	Inverse-Name	Purpose
Characterise-to	Is-characterised-by	An instance of a concept identifies the instances of other concept.
Is-origin-of	Its-origin-is	An instance of a concept is originated as consequence and after the existence of other instance of other concept.
Refers-to	References-to	The definition of an instance of a concept it is done in reference to instances of other concept.
Restrict-to	Restricted-by	The possible states of an instance of a concept are limited by instances of other concept.
Uses	Is-used-by	An instance of a concept uses instances of other concept.

4. A Metamodel for Software Measurement

The software measurement ontology described in the previous section in conjunction with others ontologies designed within the MANTIS environment describes the most important knowledge of the software maintenance management. Therefore, in order to be able to take advantage of this ontological knowledge for the software tools it is necessary to have suitable metamodels [29]. Figure 2 shows the general architecture of metamodeling of the MANTIS environment, within which it is the software measurement metamodel. This metamodel extends the software process metamodel. The software measurement metamodel consists of only one package that it is shown in figure 3. Further details of the MANTIS metamodels can be found in [28].

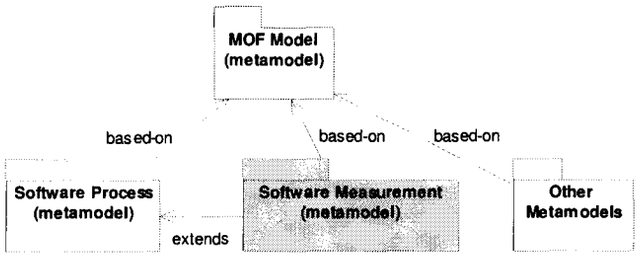


Fig. 2. General metamodeling architecture of the MANTIS environment

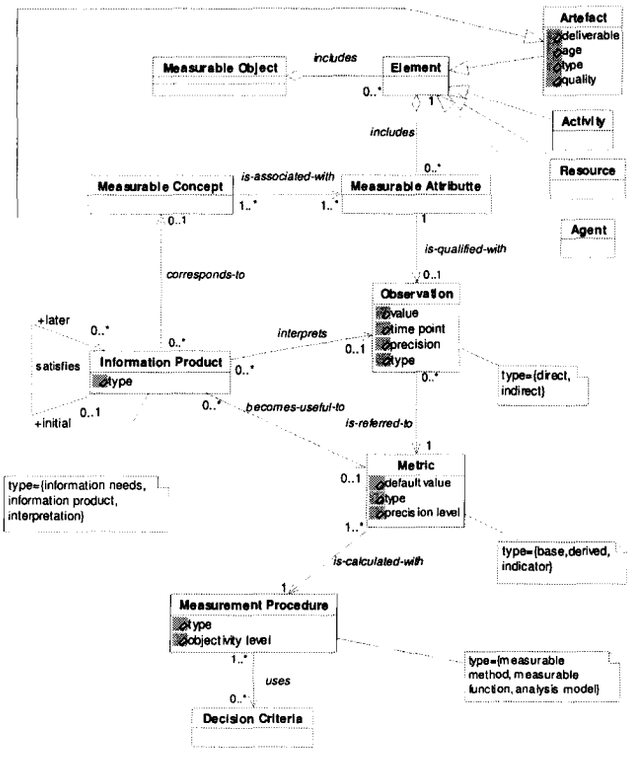


Fig. 3. Package of the software measurement metamodel

4. Conclusions

With the aim of reaching a consensus about the concepts and terminology used in software measurement, in this paper we proposed a semi-formal ontology for software

measurement. This ontology was designed using UML as representation language and an adaptation of the REFSENO methodology. Having a clear and precise ontology was a critical requirement because the ontology was designed with the goal of being implemented inside a software maintenance environment [30]. This gives the ontology a "practical" character.

We believe this ontology in conjunction with other proposals such as [4, 20, 24] allow reaching an agreement in the building of a repository that contains all the knowledge relevant to a measurement project.

We hope this proposal of a software measurement ontology has a good diffusion within researchers and practitioners, and we will be very glad to receive their feedback.

We are aware that this proposal can and must be refined after a discussion process trying to integrate other approaches and ideas suggested by most of the experts in software measurement. For this purpose, we have programmed some meetings with other research groups from Spain and Argentina, which are experts in software measurement.

Acknowledgements

This research is partially supported by the TAMANSI project (grant number PBC-02-001) financed by the Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha, and the CALDEA project supported by the Subdirección General de Proyectos de Investigación, Ministerio de Ciencia y Tecnología (TIC2000-0024-P4-02).

References

1. Althoff, K-D., Birk, A., Hartkopf, S., Mülle, W.: Managing Software Engineering Experience for Comprehensive Reuse. 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99), Kaiserslautern, Germany, (1999)
2. Becker-Kornstaedt, U., Webby, R.: A Comprehensive Schema Integrating Software Process Modeling and Software Measurement. Fraunhofer IESE – Report No 047.99, August 1999. Available at: http://www.iese.fhg.de/Publications/Iese_reports/, (1999)
3. Briand, L., Morasca, S., Basili, V. (2002): An operational process for goal-driven definition of measures. IEEE Transactions on Software Engineering, 28(12), (2002) 1106-1125.
4. Briand, L., Morasca, S., Basili, V.: Property-Based Software Engineering Measurement. IEEE Transactions on Software Engineering, 22(1), (1996)
5. Calero, C., Piattini, M., Genero, M.: Empirical validation of referential integrity metrics. In Information and Software Technology 43, (2001) 949-957
6. De Champeaux, D.: Object-oriented development process and metrics, Upper Saddle River, Prentice-Hall, (1997)
7. Deridder, D.: A Concept-Oriented Approach to Support Software Maintenance and Reuse Activities. KBOSSE Workshop, ECOOP, Málaga (Spain), (2002)
8. Díaz, O., Piattini, M., Calero, C.: Measuring triggering-interaction complexity on active databases. Information Systems Journal 26(1), Elsevier Science, (2001)

9. Farquhar, A., Fikes, R., Rice, J.: The Ontolingua Server: A Tool for Collaborative Ontology Construction, International Journal of Human-Computer Studies 46, (1997) 707-728
10. Fernández, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. AAAI Spring Symposium. University of Stanford; Palo Alto, California (United States), (1997), 33-40
11. Genero, M., Olivas, J., Piattini, M., Romero, F.: Using metrics to predict OO information systems maintainability, CAISE 2001, Lecture Notes in Computer Science (LCNS) 2068, Dittrich, K., Geppert, A. And Norrie, M.C. (eds.) Springer-Verlag, (2001) 388-401
12. Gómez-Pérez, A.: Knowledge sharing and reuse. In Jay Liebowitz, editor, The Handbook of Applied Expert Systems. CRC Press, (1998)
13. Gruber, T.: Towards Principles for the Design of Ontologies used for Knowledge Sharing. International Journal of Human-Computer Studies, 43(5/6), (1995) 907-928.
14. Guarino, N.: Formal Ontology and Information Systems. Formal Ontology in Information Systems (FOIS'98), Trento (Italy), IOS Press, (1998) 3-15
15. Hikita, T., Matsumoto, M.: Business Process Modelling Based on the Ontology and First-Order Logic. 3rd International Conference on Enterprise Information Systems (ICEIS'2001), Setúbal (Portugal), (2001) 717-723
16. IEEE: STD 1074-1995: IEEE Standard for Developing Software Life Cycle Processes, (1995)
17. ISO 9126: Software Product Evaluation-Quality Characteristics and Guidelines for their Use, ISO/IEC Standard 9126. Geneva, (2001)
18. ISO 15939: FDIS 15939: Software Engineering - Software Measurement Process (draft), (2002)
19. Kim, H.: Representing and Reasoning about Quality using Enterprise Models. Ph.D. Thesis. Department of Mechanical and Industrial Engineering, University of Toronto (Canada), (1999)
20. Kitchenham, B., Hughes, R., Linkman, S.: Modeling Software Measurement Data. IEEE Transactions on Software Engineering, 27(9), (2001) 788-804
21. MacDonell, S., Shepperd, M., Sallis, P.: Metrics for Database Systems: An Empirical Study, Proc. Fourth International Software Metrics Symposium – Metrics'97, Albuquerque. IEEE Computer Society, (1997) 99-107
22. Mylopoulos, J.: Information Modelling in the Time of the Revolution. Information Systems, 23(3-4), (1998) 127-155
23. OMG: Unified Modeling Language (UML) Specification, Version 1.4: Object Management Group (OMG). (2001)
24. Olsina, L., Bertoa, M., Lafuente, G., Martín, M., Katrib, M., Vallecillo, A.: Un Marco Conceptual para la Definición y Explotación de Métricas de Calidad. VII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2002). El Escorial , Madrid, (2002) 189-199
25. Ostertag, E., Hendler, J., Prieto-Díaz, R. and Braun, C.: Computing similarity in a reuse library system: An AI-based approach. ACM Transactions on Software Engineering and Methodology, 1(3), (1992) 205-228
26. Pfleeger, S.: Assessing Software Measurement, IEEE Software, (1997) 25-26.
27. Piattini, M., Genero, M., Jiménez, L: A metric-based approach for predicting conceptual data models maintainability. International Journal of Software Engineering and Knowledge Engineering 11(6), (2001) 703-729
28. Ruiz, F.: MANTIS. Definición de un entorno para la Gestión del Mantenimiento de Software. Ph.D. Thesis, University of Castilla-La Mancha (In Spanish), (2003)
29. Ruiz, F., Piattini, M., Polo, M.: An Conceptual Architecture Proposal for Software Maintenance. 13th International Symposium on System Integration (ISSI'01). Baden-Baden (Germany), VIII, (2001) 1-8

30. Ruiz, F., Piattini, M., Polo, M.: An Integrated Environment for Managing Software Maintenance Projects. In "The Guide to IT Service Management, Vol. I", chapter 31, Addison-Wesley, (2002) 567-588
31. Ruiz, F., Vizcaíno, A., García, F., Piattini, M.: An ontology for software maintenance. Submitted to the Journal of Software Engineering and Knowledge Engineering, (2003)
32. Staab, S., Schnurr, P., Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems, 16(1), (2001) 26-34.
33. Tautz, C., Von Wangenheim, C: REFSENO: A Representation Formalism for Software Engineering Ontologies. Fraunhofer IESE-Report No. 015.98/E, version 1.1, (1998).
34. Uschold, M., Gruninger, M.: Ontologies: Principles, methods, and applications. The Knowledge Engineering Review, 11(2), (1996) 93-136
35. Van Heijst, G., Falasconi, S., Abu-Hanna A., Schreiber, T., Stefanelli, M.: A case study in ontology library construction. Artificial Intelligence in Medicine 7, (1995) 227-255
36. Wang, X., Chan, C.: Ontology Modeling Using UML. 7th International Conference on Object Oriented Information Systems Conference (OOIS'2001), (2001) 59-68