July 1-3, 2003
San Francisco, California, USA

SEKE

Proceedings of the Fifteenth International Conference on Software Engineering & Knowledge Engineering

Knowledge Systems Institute

Knowledge Systems Institute
Graduate School of Computer & Information Sciences
3420 Main Street
Skokie, Illinois 60076

July 1-3, 2003
San Francisco, California, USA

SEKE

Printed In USA, June 2003
ISBN 1-891706-12-8

PROCEEDINGS

SEKE 2003

The 15th International Conference on Software Engineering & Knowledge Engineering

Sponsored by

Knowledge Systems Institute, USA

Technical Program
July 1-3, 2003
San Francisco, California, USA

Organized by

Knowledge Systems Institute

ISBN 1-891706-12-8 (paper)

Additional Copies can be ordered from:
Knowledge Systems Institute
3420 Main Street
Skokie, IL 60076, USA
Tel:+1-847-679-3135
Fax:+1-847-679-3166
Email:office@ksi.edu
http://www.ksi.edu

Printed in the United States of America

SEKE 2003

Fifteenth International Conference on
Software Engineering and Knowledge Engineering

Conference Organization

Conference Co-Chairs

C. V. Ramamoorthy, University of California, Berkeley
Shi Kuo Chang, Knowledge Systems Institute, Chicago

Program Co-Chairs

Kang Zhang, University of Texas at Dallas, USA
John Debenham, University of Technology, Sydney, Australia

Steering Committee

Vic Basili, University of Maryland, USA
Bruce Buchanan, University of Pittsburgh, USA
Shi-Kuo Chang, University of Pittsburgh and Knowledge Systems Institute, USA
C. V. Ramamoorthy, University of California, USA

Program Committee

Silvia Teresita Acuña, Argentina
Vincenzo Ambriola, Italy
Anneliese Amscler Andrews, USA
Tim Arndt, USA
Egidio Astesiano, Italy
Juan Carlos Augusto, UK
Antonia Bertolino, Italy
Leopoldo Bertossi, Canada
Alfs Berztiss, USA
Lionel Briand, Canada
Augusto Celentano, Italy
William Chu, Taiwan
Paolo Ciancarini, Italy
Gennaro Costagliola, Italy
Honghua Dai, Australia
John Debenham, Australia
Yi Deng, USA
Gregor Engels, Germany
Filomena Ferrucci, Italy
Antonio Fresa, Italy
Alfonso Fuggetta, Italy
Carlo Ghezzi, Italy
Athula Ginige, Australia
Angela Guercio, USA
Qinming He, China
Maolin Huang, Australia
Letizia Jaccheri, Norway
Erland Jungert, Sweden
Natalia Juristo,
Latifur Khan, USA
Robert Laurini, France
Stefano Levialdi, Italy
Jay Liebowitz, USA
Huimin Lin, China
Mikael Lindwall, USA
Jiming Liu, Hong Kong
Vincenzo Loia, Italy
Andrea De Lucia, Italy
Maurizio Martelli, Italy
Deependra Moitra, India
Sandro Morasca, Italy
Lakshmi Narasimhan, USA
Paolo Nesi, Italy
Mehmet A. Orgun, Australia

David Robertson, UK
Guenther Ruhe, Canada
Ioana Rus, USA
Walt Scacchi, USA
Phil Sheu, USA
Margaret-Anne Storey, Canada
Scott Tilley, USA
Genny Tortora, Italy
Jeffrey Tsai, USA
M. Vazirgiannis, Greece
Sira Vegas, Spain
Giuseppe Visaggio, Italy
Giuliana Vitiello, Italy
Christiane Gresse von Wangenheim, Brazil
Stefan Wermter, UK
Xindong Wu, USA
Yiyu Yao, Canada
Marvin Zelkowitz, USA

Workshop on Data Mining for Software Engineering and Knowledge Engineering

Workshop Chair/Co-chairs

Honghua Dai, Deakin University, Australia
Geoff Webb, Monash University, Australia

Workshop Publicity Chair

Gang Li, Deakin University, Australia

Workshop Program Committee

Keith Chan, Hong Kong
Lai, Chih, USA
Honghua Dai, Australia
Gang Li, Australia
James Liu, Hong Kong
Zongtian Liu, China
Walcélio L. Melo, USA
Hiroshi Motoda, Japan
Claudio Nehme, Brazil
Chinya Ravishankar, USA
Javier Segovia, Spain
Simeon J. Simoff, Australia
Zhihai Wang, Australia
Geoff Webb, Australia
Xindong Wu, USA
Min Yao, China
Aoying Zhou, China
Zhi Hua Zhou, China

Workshop on Software Engineering Decision Support

Workshop Chair

Günther Ruhe, University of Calgary, Canada

Workshop Program Committee

Stephan Biffl, Austria
Lionel Briand, Canada
Khaled El-Emam, Canada
Ross Jeffery, Australia
Sandro Morasca, Italy
David Raffo, USA
Günther Ruhe, University of Calgary, Canada
Iona Rus, USA
Claes Wohlin, Sweden

Workshop on Web Engineering

Workshop Chair

Athula Ginige, University of Western Sydney, Australia

Workshop Program Committee

Athula Ginige, University of Western Sydney, Australia
Martin Gaedke, University of Karlsruhe, Germany
Bebo White, Stanford University, USA
Paul Dantzig, IBM T J Watson Research Center, USA
Steve Hansen, University of Western Sydney, Australia
Yogesh Deshpande, University of Western Sydney, Australia
San Murugesan, Southern Cross University, Australia
Filomena Ferrucci, Università di Salerno, Italy

Conference Publicity Chair

Eric Wong, University of Texas at Dallas, USA

Conference Finance Chair

Judy Pan, KSI, USA

Conference Secretariat

C. C. Huang, KSI, USA
Rex Lee, KSI, USA

SEKE'2003 Preface

On behalf of the Program Committee of the Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'2003). We would like to welcome you to San Francisco, USA. The conference aims at bringing together experts in software engineering and knowledge engineering to discuss on relevant results in either software engineering or knowledge engineering or both. Special emphasis is on the transference of methods between both domains.

It is our pleasure to announce that by the deadline of 10 March 2003, the conference and workshops received 159 submissions from 26 countries. All the papers have been rigorously reviewed by at least 2-3 members of the international Program Committee. Based on the review result, 71 papers have been accepted as regular papers, with an acceptance rate of 44.7%, and 35 accepted as short papers. We would like to thank all the authors for their contributions.

This year, we have a rich collection of activities in the technical program, including two keynote speeches, one tutorial, three workshops, and 20 technical sessions. The keynotes, tutorial, workshops, and technical sessions cover a wide range of topics in software engineering and knowledge engineering, including:

Data Mining for Smarter Project Management,
Web Engineering,
Software Engineering Decision Support,
Data Mining for Software Engineering and Knowledge Engineering,
Software Maintenance, Software Measurements,
Software Process and Architecture,
Software Visualization and Comprehension,
Agent Systems,
Formal Methods,
Software and Knowledge Reuse,
UML,
Knowledge Acquisition,
Knowledge Retrieval and Management,
Integrity, Security and Fault-Tolerance,
Distributed Software Development, and
Spatial Reasoning and Search.

We are very grateful to the two keynote speakers, Athula Ginige and Gio Wiederhold, the tutorial and workshop organizers, Tim Arndt, Juan Carlos Augusto, Gary D. Boetticher, Honghua Dai, Erland Jungert, Tim Menzies, Guenther Ruhe, and M. Vazirgiannis. The Publicity Chair, Eric Wong, and members of the Program Committee should be congratulated and specially thanked for their publicity effort and timely reviews of the submitted papers.

Finally, we would like to thank Shi-Kuo Chang for his guidance and leadership throughout organization of this conference. The assistance of the staff at KSI is also greatly appreciated. C. C. Huang and Rex Lee at KSI handle the registration and organization of the proceedings. Special thanks go to Yu Qian, a PhD student at UT-Dallas, for his effective and efficient assistance in working with the paper submission and review system, which has made the review process smooth and timely. Finally, we would like to thank Judy Pan at KSI, who carefully arranged the details of the reception, the boat tour and dinner that have greatly enhanced this conference.

John Debenham
University of Technology, Sydney, Australia

Kang Zhang
University of Texas at Dallas, USA

Table of Contents

Towards an Ontology for Software Measurement

Marcela Genero, Francisco Ruiz, Mario Piattini, Félix García 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

As software measurement is a relatively young discipline does not exist yet a common agreement about the concepts and terminology used in this field. We believe that in order to software measurement become a well established engineering discipline it is necessary to start reaching a consensus among software measurement researchers and practitioners about what we are talking about when we refer to "software measurement" or "software measurement process", "measure", "metric", "measurable attribute", "measurable object", etc. Therefore the main focus of this paper is to present a practical and semi-formal ontology for software measurement which was designed using the Unified Modeling Language (UML) as an ontology representation language and an adaptation of the Representation Formalism for Software Engineering Ontologies (REFSENO). Moreover, a software measurement metamodel which captures the ontology has been implemented in the MANTIS environment, a software maintenance engineering environment.

Keywords: Software measurement, software measurement process, metrics, ontology, metamodel, software maintenance, REFSENO.

1. Introduction

Software measurement suffers from several symptoms of any relatively young discipline [3]. Software measurement is currently in the phase in which terminology, principles, and methods are still being defined and consolidated. In spite of this, nobody can discuss the relevant role that software measurement plays in software engineering. Software measures could be used to build prediction systems for database projects [15], to understand and improve software development and maintenance projects [2], to maintain system quality, highlighting problematic areas [5], and to determine the best ways to help practitioners and researchers in their work [19], 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.

We believe that a good starting point to contribute to software measurement become a mature discipline is to create a common agreement about the concepts and terminology used in this field. For this purpose it is necessary that software measurement researchers and practitioners reach to 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 build a software measurement ontology, i.e. a common conceptualisation of software measurement domain, where objects, concepts, entities and their relationships were explicitly represented. 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 [6].

Our background in software measurement [4], [7], [9], [20] helped us to detect the main 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" built within the MANTIS environment [23], a software maintenance engineering environment [28][1].

Moreover, an 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.

We believe this ontology in conjunction with others proposals such as Kitchenham et al. [14], Olsina et al. [17] and Briand et al. [3] allow to get and agreement in the building a repository that contains all the knowledge relevant to measurement projects.

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

- The software measurement process of the ISO 15939 [13].
- The conceptual model for representing collections of software data provided by Kitchenham et al. [14].
- The software measurement conceptual model proposed by Becker-Kornstaedt and Webby (1999).
- The terminology provided in the ISO 9126 [12].

Therefore, the main focus of this work is to present a practical and semi-formal ontology for software measurement which was designed using UML [16] as an ontology representation language and an adaptation of the Representation Formalism for Software Engineering Ontologies (REFSENO) [25].

Although other sets of techniques exist for developing ontologies [10], [24], [26], [27] they have not been applied to software engineering as REFSENO has (An example of a GQM ontology developed using REFSENO appears in [25]).

REFSENO is an improved adaptation of Methontology [8], [10] which imitates the software life-cycle proposed by the IEEE 1074 standard [11]. REFSENO provides constructs to describe concepts where each concept represents a class of experience items. Besides concepts, its properties (called terminal attributes) and relationships (nonterminal attributes) are represented. One relevant feature of REFSENO is that incorporates integrity rules such as: cardinalities and ranges of values for attributes, assertions and preconditions. REFSENO extends the formalism of Ostertag, et al. [18] by additional integrity rules and by clearly separating the schema definition and characterisation.

The rest of this paper is organized as follows: The conceptual description of software measurement is presented in Section 2. The proper definition of an ontology for software measurement comes in Section 3. A software measurement metamodel which allows ontology implementation is presented in Section 4. Finally, the last section presents some concluding remarks.

2. Conceptual Description of Software Measurement

In this section we present a conceptual description about what "to measure" means in this ontology for software measurement, 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 the obtaining 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 with 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 the obtaining the value of an attribute in one instant or point of time for one of the associated metrics. Each metric has an specific measurement unit which belongs to an 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 an 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 ore 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.

This ontology defines concepts as generic as possible in order to achieve the goal of being applicable to different measurement models and measurement methods. In this way, the data representation model proposed by Kitchenham et al. [14] or the conceptual model for the metric domain proposed by Olsina et al. [17] are usable within this ontology.

3. An Ontology of Software Measurement

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 [29].

[1] Vizcaíno et al. [28] gives 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.

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

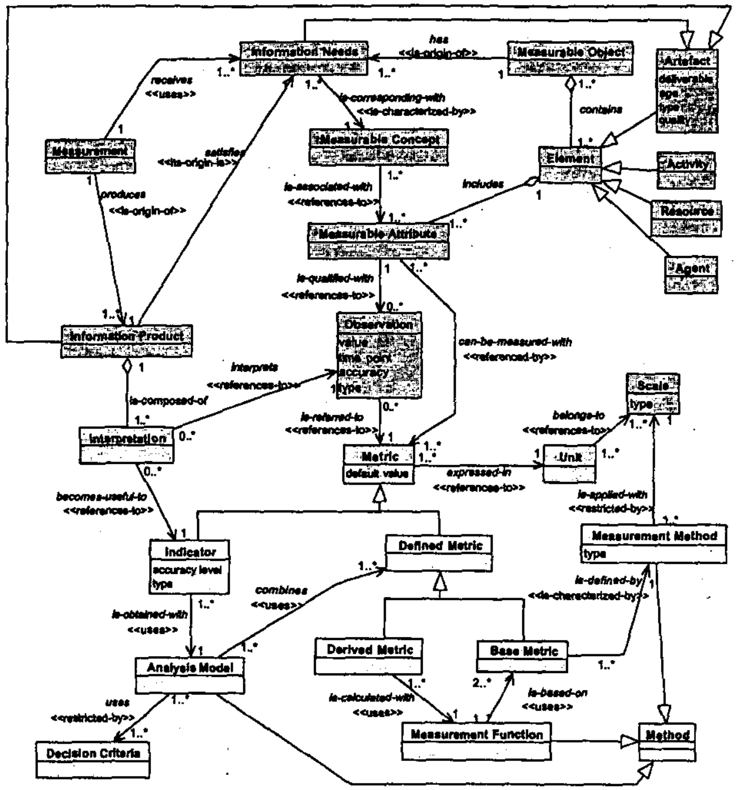

Figure 1. A UML class diagram for the ontology for software measurement

Hereafter the tables of the ontology are presented (an adapted version of REFSENO is used): Table 1 presents the glossary of concepts, table 2 is a table of attributes and finally table 3 presents the relationships.

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

Concept	Super-Concept	Description	Purpose
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 one 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 analyse the measurable object. The value of and 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 one 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 logic sequence of operations used to quantify an attribute respect to a certain scale. Two measurement methods exist: subjective (based on human judgement) and 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 one or more activities.	To define activities
Metric	Concept	Variable associated to one ore 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
Resource	Element	Something of which is not 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 2. Software measurement ontology: Glossary of concepts

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 characteristic 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 objectives (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 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 3. Software measurement ontology: Table of attributes

Name	Description
Becomes-useful-to	An interpretation provides utility on a concrete necessity to an indicator.
Belong-to	One measurement unit belong 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	One 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 an 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.
Is-corresponding-with	A concrete information need of a measurable object is corresponding with one 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 an specific analysis model.
Is-referred-to	An observation obtains the value of one attribute referred to a metric.
Produces	The measur ement 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.

Table 4.Software measurement ontology: Table of relationships

As we mentioned above this software measurement ontology is part of a "super-ontology" built within the MANTIS environment, a software maintenance engineering environment which 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).

Relationship class	Inverse Name	Purpose
Characterise-to	Is-characterised-by	An instance of a concept identify 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.

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

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 [22]. 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.

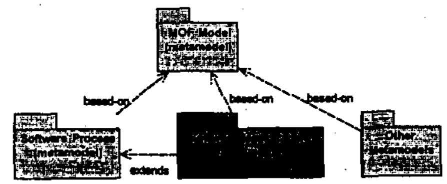

Figure 2. General metamodeling architecture of the MANTIS environment

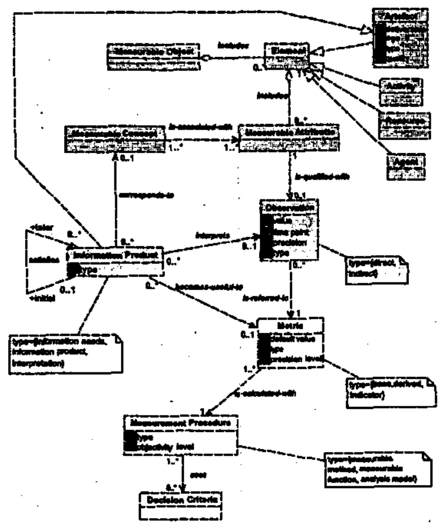

Figure 3. Package of the software measurement metamodel

5. Conclusions

This paper presents a semi-formal ontology for software measurement contributing thus to reach a consensus about the concepts and terminology used in this field. 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 [23]. This gives the ontology a "practical" character.

To develop the ontology, a suitable methodology, REFSENO [25], has been used.

We hope this first proposal of an ontology for software measurement and its corresponding metamodel had a good diffusion within researchers and practitioners. We are aware that this proposal can and must be refined after a discussion process trying to integrate others approaches and ideas suggested by most of the 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 of the Junta de Comunidades de Castilla-La Mancha and the CALDEA project supported by Subdireccion General de Proyectos de Investigación, Ministerio de Ciencia y Tecnología (TIC2000-0024-P4-02).

References

[1] Becker-Kornstaedt, U. and Webby, R. (1999). 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/.

[2] Briand, L., Morasca, S. and Basili, V. (2002). An operational process for goal-driven definition of measures. IEEE Transactions on Software Engineering, 28(12), 1106-1125.

[3] Briand, L., Morasca, S., and Basili, V. (1996): Property-Based Software Engineering Measurement, IEEE Transactions on Software Engineering, 22(1).

[4] Calero, C., Piattini, M. and Genero, M. (2001). Empirical validation of referential integrity metrics. In Information and Software Technology, 43, 949-957.

[5] De Champeaux, D. (1997). Object-oriented development process and metrics, Upper Saddle River, Prentice-Hall.

[6] Deridder, D. (2002). A Concept-Oriented Approach to Support Software Maintenance and Reuse Activities. KBOSSE Workshop. ECOOP, Málaga (Spain).

[7] Díaz, O., Piattini, M. and Calero, C. (2001). Measuring triggering-interaction complexity on active databases. Information Systems Journal, 26(1), Elsevier Science.

[8] Fernández, M., Gómez-Pérez, A. and Juristo, N. (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering. AAAI Spring Symposium. University of Stanford; Palo Alto, California (Estados Unidos), 33-40.

[9] Genero, M., Olivas, J., Piattini, M. and Romero, F. (2001). 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, 388-401.

[10] Gómez-Pérez, A. (1998). Knowledge sharing and reuse. In Jay Liebowitz, editor, The Handbook of Applied Expert Systems. CRC Press.

[11] IEEE (1995). STD 1074-1995. IEEE Standard for Developing Software Life Cycle Processes.

[12] ISO 9126 (2001). Software Product Evaluation-Quality Characteristics and Guidelines for their Use, ISO/IEC Standard 9126. Geneva.

[13] ISO/IEC (2002). FDIS 15939. Software Engineering - Software Measurement Process (draft).

[14] Kitchenham, B., Hughes, R. and Linkman, S. (2001). Modeling Software Measurement Data. IEEE Transactions on Software Engineering. 27(9), 788-804.

[15] MacDonell, S., Shepperd, M. and Sallis, P. (1997): Metrics for Database Systems. An Empirical Study, Proc. Fourth International Software Metrics Symposium – Metrics'97, Albuquerque. IEEE Computer Society, 99-107.

[16] Object Management Group. (1999). UML Revision Task Force, OMG Unified Modeling Language Specification, v. 1.3, document ad/99-06-08.

[17] Olsina, L., Bertoa, M., Lafuente, G., Martín, M., Katrib, M., and Vallecillo, A. (2002). 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, 189-199.

[18] Ostertag, E., Hendler, R., Prieto-Díaz and C. Braun. (1992). Computing similarity in a reuse library system: An AI-based approach. ACM Transactions on Software Engineering and Methodology, 1(3), 205-228.

[19] Pfleeger, S. (1997). Assessing Software Measurement, IEEE Software, 25-26.

[20] Piattini, M., Genero, M. and Jiménez, L. (2001). A metric-based approach for predicting conceptual data models maintainability. International Journal of Software Engineering and Knowledge Engineering, 11(6), 703-729.

[21] Ruiz, F. (2003). MANTIS. Definición de un entorno para la Gestión del Mantenimiento de Software. Ph.D. Thesis, University of Castilla-La Mancha (In Spanish).

[22] Ruiz, F., Piattini, M., and Polo, M. (2001). An Conceptual Architecture Proposal for Software Maintenance. 13th International Symposium on System Integration (ISSI'01). Baden-Baden (Germany), VIII, 1-8.

[23] Ruiz, F., Piattini, M. and Polo, M. (2002). An Integrated Environment for Managing Software Maintenance Projects. In "The Guide to IT Service Management, volume I", chapter 31, Addison-Wesley, 567-588

[24] Staab, S., Schnurr, P. and Sure, Y. (2001). Knowledge Processes and Ontologies. IEEE Intelligent Systems, 16(1), 26-34.

[25] Tautz, C. and Von Wangenheim, C. (1998): REFSENO: A Representation Formalism for Software Engineering Ontologies. Fraunhofer IESE-Report No. 015.98/E, version 1.1, October 20.

[26] Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods, and applications. The Knowledge Engineering Review 11(2), 93-136.

[27] Van Heijst, G., Falasconi, S. and Abu-Hanna A., Schreiber, T. and Stefanelli, M. (1995). A case study in ontology library construction. Artificial Intelligence in Medicine 7, 227-255.

[28] Vizcaíno, A., Ruiz, F. and Piattini, M. (2003). An ontology for software maintenance. Submitted to the Journal of Software Engineering and Knowledge Engineering.

[29] Wang, X. and Chan, C.W. (2001). Ontology Modeling Using UML. 7th International Conference on Object Oriented Information Systems Conference (OOIS'2001), 59-68.