Proceedings

DEXA 2004

Fifteenth International Workshop on

Database and Expert Systems Applications

30 August–3 September 2004
Zaragoza, Spain

SAACS sponsored by

IBM®

Proceedings

15th International Workshop on

Database and Expert Systems Applications

30 August – 3 September 2004

Zaragoza, Spain

IEEE COMPUTER SOCIETY

IEEE

Los Alamitos, California

Washington • Brussels • Tokyo

IEEE Computer Society Order Number P2195
ISBN 0-7695-2195-9
ISSN 1529-4188

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: reprints@computer.org

Editorial production by Deeber Azada

Cover art production by Joe Daigle/Studio Productions

Printed in the United States of America by The Printing House

IEEE COMPUTER SOCIETY

IEEE

Table of Contents

15th International Workshop on Database and Expert Systems Applications – DEXA 2004

Workshop 1 – NLIS '04

4th International Workshop on Natural Language and Information Systems

Session 1: Document Analysis

Session 2: Multilinguality and Morphology

Session 3: Linguistic Knowledge Representation

Workshop 2 - PEH '04

3rd International Workshop on Presenting and Exploring Heritage on the Web

Session 1

Session 2

Session 3

Session 4

Session 5

Workshop 6 - BIDM '04
2nd International Workshop on Biological Data Management

Session 1

Session 2

Session 3

Workshop 7 - MIW '04
5th International Workshop on Management of Information on the Web

Invited Paper

Workshop 8 - TAKMA '04
5th International Workshop on Theory and Applications of Knowledge Management

Session I: Concepts of Knowledge Modelling

Session II: KM Frameworks

Session III: KMS Examples

Workshop 9 - GLOBE '04

1st International Workshop on Grid and Peer-to-Peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems

Invited Paper 1

Session 1: Grid and P2P Systems

Session 2: Semantics for P2P Systems

Invited Paper 2

Session 3: Query Processing and P2P Systems

Invited Paper 3

Session 4: Semantics for Grid Computing and Distributed Algorithms

Workshop 10 - FEIDSS '04

1st International Workshop on Forest and Environmental Information and Decision Support Systems

Invited Presentations

Session 1: Fire Information and Decision Support Systems

Session 2: Holistic Environmental Information and Decision Support Systems

Session 3: Other FEIDSS Papers

Workshop 11 - MDDS '04
7th International Workshop on Mobility in Databases and Distributed Systems

Session 1: Location Data Management

Invited Talk

Session 2: Mobile Agents and Services

Session 3: Transactions and Sensor Data

Workshop 12 - SAACS '04
2nd International Workshop on Self-Adaptable and Autonomic Computing Systems

Session 1

Session 2

Session 3

Panel I

Session 4

Session 5

Session 6

Panel II

Workshop 13 - GIM '04

1st International Workshop on Geographic Information Management

Session 1: Semantic Interoperability

Session 2: Representation of Geographic Information

Session 3: GeoInfo Services 1

Session 4: Spatial Data Infrastructures

Session 5: GeoInfo Services 2

Workshop 14 - SIUFDB '04

1st International Workshop on Supporting Imprecision and Uncertainty in Flexible Databases

Invited Talk

Session A: Toward Flexible Database Systems

Session B: Flexible Relational Databases

Session C: Semi-Structured Data

Session D: Vague and Indeterminate Spatial Data

Workshop 15 - PDMST '04

1st International Workshop on Peer2Peer Data Management, Security and Trust

Session 1: Mobile P2P Systems

Session 2: Trust in P2P Systems

Using REFSENO to Represent Knowledge in the Software Maintenance Process

Aurora Vizcaíno, Francisco Ruiz, Mario Piattini, Félix García

Alarcos Research Group, Universidad de Castilla-La Mancha, Ciudad Real (Spain)
{Aurora.Vizcaino, Francisco.RuizG, Mario.Piattini, Felix.Garcia}@uclm.es

Abstract

Many papers describe ontological designs but few of them explain how the ontology may be implemented. This paper describes how an ontology to represent software maintenance knowledge was specified by using the REFSENO Methodology. The paper also explains the use of similarity functions to compare products and requirements in order to reuse previous solutions and lessons learned.

1. Introduction

Knowledge is a critical resource and an essential element for any business activity as well as for supporting an enterprise's strategy [13]. However, on many occasions, organizations have plenty of documents which have not been catalogued and nobody uses. Most organizations still do not use techniques to acquire workers' knowledge and expertise obtained through their work in the company. This fact has already been commented on by other authors such as Szulanski in [24] who found that the number one barrier to knowledge sharing was "ignorance". Sometimes the organization itself is not aware of the location of the pockets of knowledge or expertise [17]. This implies that the companies have to re-invert time and effort searching for information that has already been used or researching solutions to problems that have previously been solved.

A plausible technique to prevent this problem is to store good solutions to problems or lessons learned to avoid repeating mistakes and to increase productivity and the likelihood of further success [22]. Based on this idea KM-MANTIS, a system in charge of managing the information generated in the stage that originates most expenses in the software engineering (the maintenance process stage) was designed.

The core of KM-MANTIS is an ontology about software maintenance concepts. This ontology was developed combining a theoretic and a pragmatic approach in a very similar manner to the Helix-Spindle Process Model for Ontological Engineering [14]. The ontology was developed using the REFSENO methodology. Therefore, the first step was to define an ontology that formalized and related the different concepts that KM-MANTIS had to deal with. The following step was to implement that ontology. This paper explains the advantages of using REFSENO to implement an ontology in a knowledge management system specialized in software engineering projects management and also illustrates how it was carried out, stressing the use of similarity functions to compare products and requirements. The rest of the paper is structured as follows: Section two outlines the need of managing knowledge in the software maintenance process. Section three explains the methodology and describes the advantages of using it. After that, section four explains how our ontology was implemented by using REFSENO. Finally, in section five the conclusions are presented.

2. Knowledge in Software Maintenance

Software engineering in general, and software maintenance in particular, are activities that generate important amounts of knowledge. This knowledge comes not only from the expertise of the professionals involved in the processes, but is also intrinsic to the product being maintained and, in the case of software maintenance, to the reasons that motivate maintenance (new requirements, defects detected, etc.). Moreover, software maintenance is a constantly changing process since maintenance results from the necessity of adapting software systems to an ever changing environment [19].

Furthermore, software maintenance involves many activities in which different people intervene. Each person has partial information that is necessary to other members of the group but if a software maintainer is the only person who has this knowledge and there is no system in charge of transferring the implicit knowledge (which the employees have) to explicit knowledge (stored on paper, in files, etc) when this maintainer leaves the organisation part of the intellectual capital and of his/her expertise go with him/her. Therefore, companies lose important intellectual capital which is difficult to recover.

Another well-known issue that complicates the maintenance process is the scarce amount of documentation that usually exists in relation to a specific software system. And even if detailed documentation was produced when the original system was developed, it is seldom updated as the system evolves. For example, legacy code from other departments often does not have documentation which describes the features of the software. For all these reasons, maintenance organizations frequently have problems identifying the resources of their knowledge and as a result they do not reuse it.

Techniques and tools are needed to help software practitioners apply past knowledge to current projects [10]. Using a knowledge management system new knowledge might be produced, thus obtaining the maximum performance from the current information. Furthermore, by reusing information and producing relevant knowledge the high costs of software maintenance could also be decreased [2].

3. REFSENO: Advantages and Description

The issues explained above motivated us to design a knowledge management system for acquiring, managing, and disseminating knowledge in a software maintenance organisation with the goal of increasing the workers' expertise, the organisation's knowledge and its competitiveness while decreasing the costs of the maintenance process.

Before constructing the system, modelling, structuring and generalising the information that is generated during the software maintenance process was vitally important. In order to attain this goal we decided to construct a common conceptualisation of the domain, where objects, concepts, entities and their relationships were explicitly represented. Since ontologies enable explicit specification of a conceptualisation [8] and they represent a certain view of an application domain in which the concepts that live in this domain are defined in an unambiguous and explicit way [3], this technique was chosen. Moreover, as is explained in [15] ontologies facilitate enterprise knowledge management, knowledge sharing [16], and knowledge integration [4]. All of these were very important requirements for KM-MANTIS.

To design and implement an ontology it is advisable to follow a methodology which is suitable for this aim. Different methodologies and representations have been proposed. For instance, [11] uses a representation based on first-order predicate logic. Other authors prefer frame-based approaches, such as those that are used in Ontolingua [5], one of the most frequently used ontologic languages. And other authors are using F-Logic and Description Logics.

We chose an improved adaptation of Methontology called REFSENO (Representation Formalism for Software Engineering Ontologies) [25] for the following reasons:

As the name of the own methodology indicates it was specifically designed to develop software engineering ontologies.

REFSENO uses different representations to model knowledge (such as tables and tree structures) which are more intuitive and easy of understand for stakeholders involved in software projects than other approaches used in previous works such as [5], [7], [11], [23], based on first-order predicate logic or similar. This point was very relevant for us, as in the development team there were people who worked in software maintenance companies but who did not know of formal representation approaches.

REFSENO distinguishes different levels of knowledge: conceptual and context-specific knowledge. On the contrary, the above approaches represent a high level of abstraction. Consequently, they represent a lesser level of granularity than REFSENO does.

The methodology proposes different techniques to check the consistency of the ontology and, what is more, has methods of controlling the consistency of the instances to an implementation level, a feature that other methodologies do not consider.

Because KM-MANTIS should detect problems that have already been solved in order to reuse the same solution and avoid effort, the system needed intelligent artificial techniques. Case-Based Reasoning (CBR) is often used to find the best solution for problems dealing with selecting a solution from many existing ones [18]. Thus, it was one of the techniques chosen. Fortunately, REFSENO provides constructs that facilitate the use of CBR as will be illustrated in section 4.

REFSENO provides epistemic primitives to describe concepts where each concept represents a class of experience items. Besides concepts, its properties (called terminal attributes) and relationships (nonterminal attributes) are also represented.

Moreover, REFSENO incorporates integrity rules such as: cardinalities and value ranges for attributes, assertions, and preconditions that the instance must fulfil. REFSENO extends the formalism of [20] by additional integrity rules, and by clearly separating the schema definition and characterisation.

In REFSENO, the detailed information of the ontology is represented by means of a collection of tables: concepts glossary, table of attributes, of relationship classes, etc.

Terminal concept attributes are described by a 9-tuple formed from the following items:

- Name: The name is used for reference purposes.
- Description: A narrative text which defines the meaning of the attribute.
- Cardinality: A range specifying the minimum and maximum number of values the attribute may have.
- Type: Each terminal concept attribute is given a type, and the types are viewed as an epistemistic primitive. REFSENO has some predefined types such as Boolean, Integer, Real, Text, Identifier or Date. New types can be described by users.
- Default value: This is related to the insertion of new instances. If the user entering a new instance does not specify a value for this attribute, the default value is used.
- Mandatory: This is also related to new instances. It indicates whether an attribute value of an instance has to be specified.
- Value inference: This component defines how to calculate the attribute value automatically (if possible) based on the values of other attributes.
- Inferred attributes: This component lists all the attributes whose value is inferred using a value of this attribute. There is a mutual dependence between value inferences and inferred attributes, thus inferred attributes can automatically be derived from the value inferences.
- Standard weight: This weight may be used by the similarity functions (explained later) of the concept this attribute belongs to. A weight of 0 denotes an attribute whose value will not be used for querying.

REFSENO distinguishes three layers to which attributes may belong. These are artifact, interface and context. The attributes of the artifact layer characterise the instances themselves. Attributes of the interface layer characterise how a particular instance can be integrated into the system. Attributes of the context layer characterise the environment in which the instance has been applied and the quality of the instance in the specified environment.

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 retrieval components.

In order to calculate the similarity functions between two instances i and i’ the different layers should be taken into account, since there is a similarity function for each layer. For a concept c these are simartif(c), simI/F(c) and simctxt(c) and this is based on the local similarity functions of the concept’s attributes. The values of similarity functions for a concept c between two instances i and i’ are combined to a single similarity value as follows:

Sim(c)(i,i’)=Wartif*simartif(c)(i,i’)+WI/F*simI/F(c)(i,i’)+
Wctxt*simctxt(c) (i, i’),

where Wartif, WI/F, Wctxt are weights with which the similarity functions can be adjusted to the needs of the users. The sum of the weights is always 1. A similarity value equal to 0 means total dissimilarity between i and i’, and a value equal to 1 indicates total similarity (equivalence). The concept’s similarity functions are of a global nature because they are based on the local similarity functions of the concept’s attributes. An example of how a similarity function is calculated in KM-MANTIS is described in the next section. Besides similarity functions, attributes tables may also have assertions which are conditions expressed as a formula, and that all instances must fulfil and preconditions which must be fulfilled before instances are inserted or changed.

The nonterminal attributes, those that represent how a particular entity is related to other entities, can be represented in the same concept attribute table used for the terminal concept attributes. REFSENO allows other possible representations for nonterminal attributes. For example graphically, by using a tree structure. However, in this paper only the first representation (tables) is used.

4. Specifying the Ontology with REFSENO

The Software Maintenance Projects ontology is made up of a set of three ontologies (see Figure 1), which represent static and dynamic aspects. In order to represent the static aspects, we defined an ontology called Maintenance, which is formed of four subontologies. They describe the concepts related to maintenance and consist of a subontology for products, another for activities, a third for the process organization and the fourth for describing the different agents involved in the software maintenance process. The number of static ontologies coincides with those proposed by [12]. Nevertheless, we have extended and formalised them.

The dynamic part is represented by an ontology called a Workflow Ontology, where three relevant parts of maintenance are defined:

- Decomposition of activities.
- Temporal constraint between activities (this being the order in which the activities must be performed).
- Control of the execution of activities and projects during the process enactment.

A third ontology called the Measure Ontology represents both static and dynamic aspects. An example of a dynamic aspect is the measurement actions [6].

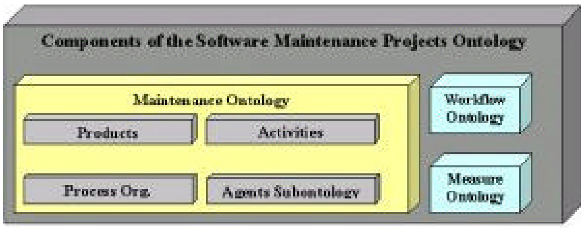

Figure 1: Structure of the software maintenance projects ontology

The ontology and subontologies are described in [21] in detail. This paper focuses on how they are implemented in KM-MANTIS by using REFSENO. In order to illustrate this the products subontology is briefly explained.

This subontology defines the software products that are maintained, their internal structure, composition and the existing versions of each product. Figure 2 shows the ontologic diagram by using a UML class diagram, where the product is stressed since it is the most important.

As Figure 2 shows, one software product can have different versions, which are formed from a set of artifacts. For instance, for a product called "Sales", different versions of this product may exist, and each version is made up of several artifacts. The concept version has its own attributes, such as: number, date, etc. To simplify, they are not represented in the diagram. The previous diagram only shows a summarized view of the referred ontology.

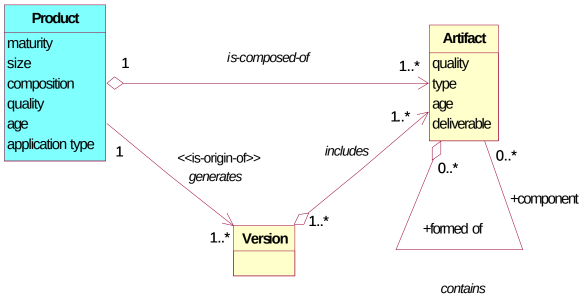

Figure 2. Products subontology diagram

The first step to implement the ontology by using REFSENO is to define the concept glossary which provides a general description and the purpose of the concepts previously represented in the subontology diagram. Each row of the table corresponds to one concept. There is one concept glossary per ontology and subontology represented in figure 1. Here only the product concept glossary is represented.

The second step is to construct a terminal attribute table for each concept defined in the glossary table. In this paper they are omitted by limitations of space, for more detail see [21].

Table 1 Products subontology: Concept Glossary. NOTE: The super-concept "Concept" is the root

Concept	Super-Concept	Description	Purpose
Artifact	Element	This is a software product, part of which is created or modified by the activities. It can be a document (text or graphic), or a code module. Examples: requirement specification documents, quality plan, class module, routine, test report, user manual. Synonymous: software element, work product, product item.	To define the internal structure and software composition.
Product	Concept	Software application, which is being maintained. It is a conglomerate of different artifacts. Synonymous with: Software.	Maintenance.

Concept	Super-Concept	Description	Purpose
Version	Concept	This is a change in the base line of a product. It could be an upgrade, release or actualisation.	To implant the configuration management process.

4.1 Uses of Similarity Functions in KM-MANTIS

KM-MANTIS mainly uses similarity functions to compare software products, and maintenance requests. One goal of comparing products is to predict new clients' demands since what a company has done before tends to predict what it can do in the future [9]. Therefore, products with similar features often demand the same modifications. As [1] claim, if changes can be anticipated they can be built in by some form of parameterisation and in this way costs and efforts are decreased. Moreover, studies show that the incorporation of new requirements is the core problem for software evolution and maintenance and supposes, along with adaptive maintenance, around 75 % of the maintenance effort.

The finality of comparing maintenance requests is to reuse previous solutions to similar problems and also avoid the repetition of mistakes. Storing and reusing solutions that have worked correctly in previous maintenance situations helps to avoid that companies being forced to reinvent new practices due to the limited transfer of knowledge, resulting in costly duplication of effort. Frequently, the best practices linger in companies for years unrecognised and unshared.

We are going to illustrate how KM-MANTIS calculates the similarity function when it needs to compare two instances of product, for example, i and q. First of all, the similarity functions for each layer, artefact and I/F (the context layer is omitted because in this case there are no attributes of this layer) should be calculated. They are stressed in the formula below.

Sim (product) (i, q)=Wartif*simartif(product)(i, q)+ WI/F*simI/F(product) (i, q)

The local similarity functions are calculated by computing the sum of the similarity function of each type of attribute belonging to this layer. Finally, each local similarity function is normalized resulting in a value in the range [0,1]. Thus, in the case of the artifact layer of the concept product, it is necessary to know the similarity function of its types. They are: "TypeMaturity", "MeasureSize", "TypeComposition", "TypeApplication", "MeasureQ" and "Integer" in order to obtain their sum.

REFSENO provides several predefined types and their similarity functions. For instance, the "Integer" type has the following similarity function to compare two instances i and q:

$$Sim(i,q) = 1 - \frac{|i-q|}{(\max value - \min value)}$$

where minvalue and maxvalue are respectively the lower and upper bound of the value range.

In the case of using own types, such as "TypeMaturity", their similarity types should also be described. For instance, type maturity is a taxonomy formed of four labels: initial, evolution, service and retired and its similarity function is the following:

Sim(i,q): 1 if i=q
0.5 if i= initial and q=evolution or vice versa
0.25 if i= initial and q=service or vice versa
0 if i= initial and q= retired or vice versa
0.5 if i= evolution and q= service or vice versa
0 if i= evolution and q= retired or vice versa

After calculating the local similarity functions simartif(product)(i, q) and simI/F(product) (i, q) the global similarity function should be calculated by assigning values to Wartif, and WI/F, depending on what the user's needs are. For instance, if the system wants to compare the similarity between two products according to their own features, the value of Wartif should be maximised and WI/F decreased since the sum of the weights is always 1. Therefore, the system adapts the weights according to the convenience of giving more priority to one layer or to another.

5. Conclusions

Software maintenance generates huge amounts of knowledge that should be processed and managed in order to decrease costs and effort. However, before managing it, the different types of information and their relationship should be specified. Ontologies are the best way to carry out this specification. Many papers describe ontological designs but few of them explain how to implement them. In this paper we have explained why REFSENO methodology was chosen to implement our software maintenance ontology, and how the implementation was performed. Moreover, the use of the functions of similarity that REFSENO provides in KM-MANTIS has also been described.

Acknowledgements

This work 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, SPAIN.

References

[1] K. H. Bennett and V. T. Rajlich, "Software Maintenance and Evolution: a Roadmap," In proceedings of the The Future of Software Engineering, ICSE 2000, Limerick, Ireland, pp. 75-87, 2000.

[2] L. A. de Loof, *Information Systems Outsourcing Decision Making: a Managerial Approach*, 1990.

[3] D. Deridder, "A Concept-Oriented Approach to Support Software Maintenance and Reuse activities," In proceedings of the Workshop on Knowledge-Based Object-Oriented Software Engineering at 16th European Conference on Object-Oriented Programming (ECOOP 2002), Málaga (Spain), 2002.

[4] R. A. Falbo, C. S. Menezes, and A. R. Rocha, "Using Knowledge Serves to Promote Knowledge Integration in Software Engineering Environments," In proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99), Kaiserslautern (Germany), pp 170-175, 1999.

[5] A. Farquhar, R. Fikes, and J. Rice, "The Ontolingua Server: A Tool for Collaborative Ontology Construction," *International Journal of Human-Computer Studies*, vol. 46, pp. 707-728, 1997.

[6] M. Genero, F. Ruiz, M. Piattini, and C. Calero, "Towards and Ontology for Software Measurement," In proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE'2003), San Francisco, (USA), 2003.

[7] A. Gómez-Pérez, "Knowledge Sharing and Reuse," in *The Handbook of Applied Expert Systems*, J. Liebowitz, Ed.: CRC Press, 1998.

[8] T. Gruber, "Towards Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal of Human-Computer Studies*, vol. 43(5/6), pp. 907-28, 1995.

[9] A. Gupta and V. Govindarajan, "Knowledge Flows within Multinational Corporations," *Strategic Management Journal*, vol. 21(4), pp. 473-449, 2000.

[10] S. Henninger, "Tool Support for Experience-Based Software Development Methodologies," *To appear in Advances in Computer*, vol. 59, pp. 29-82, 2003.

[11] T. Hikita and M. J. Matsumoto, "Business Process Modelling Based on the Ontology and First-Order Logic," In proceedings of the Third International Conference on Enterprise Information Systems (ICEIS'2001), Setubal (Portugal), pp. 717-723,2001.

[12] M. Kajko-Mattsson, "Towards a Business Maintenance Model," In proceedings of the IEEE International Conference on Software Maintenance (ICSM), Florence (Italy), pp. 500-9, 2001.

[13] L. Kerschberg, "Knowledge Management in Heterogeneous Data Warehouse Environments," In proceedings of the DAWAK, 2001, pp 1-10.

[14] R. Kishore, H. Zhang, and R. Ramesh, "A Helix-Spindle Model for Ontological Engineering," *Communications of the ACM*, vol. 47, Nº 2, pp. 69-75, 2004.

[15] A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz, "Ontologies for Enterprise Knowledge Management," *IEEE Intelligent Systems*, pp. 26-33, 2003.

[16] J. Mylopoulos, "Information Modelling in the Time of the Revolution," *Information Systems*, vol. 23 (3-4), pp. 127-155, 1998.

[17] J. Nebus, "Framing the Knowledge Search Problem: Whom Do We Contact, and Why Do We Contact Them?," In proceedings of the Academy of Management Best Papers Proceedings, pp. h1-h7, 2001.

[18] A. A. Niknafs, M. E. Shiri, and M. M. Javidi, "A Case-Based Reasoning Approach in E-Tourism: Tour Itinery Planning," In proceedings of the 4th International Workshop on Theory and Applications of Knowledge Management. During DEXA'03, Prague, Czech Republic, pp. 818-822, 2003.

[19] K. M. Oliveira, N. Anquetil, D. M.G, M. Ramal, and R. Meneses, "Knowledge for Software Maintenance." In proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'03), San Francisco, pp. 61-68, 2003.

[20] E. Ostertag, J. Hendler, R. Prieto-Díaz, and C. Braun, "Computing Similarity in a Reuse Library System: An AI-based Approach," *ACM Transactions on Software Engineering and Methodology*, vol. 1 (3), pp. 205-228, 1992.

[21] F. Ruiz, A. Vizcaíno, M. Piattini, and F. García, "An Ontology for the Management of Software Maintenance Projects," *Journal on Software Engineering and Knowledge Engineering*, pp. Accepted for publication, 2004.

[22] I. Rus and M. Lindvall, "Knowledge Management in Software Engineering," *IEEE Software*, vol. May/June, pp. 26-38, 2002.

[23] S. Staab, H.-P. Schnurr, and Y. Sure, "Knowledge Processes and Ontologies," *IEEE Intelligent Systems*, vol. 16, Nº1, pp. 26-34, 2001.

[24] G. Szulanski, "Intra-Firm Transfer of Best Practices Project," In proceedings of the American Productivity and Quality Centre,, Houston, Texas, pp. 2-19, 1994.

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