Using Semantic Annotations for Supporting Requirements Evolution.

Conference Paper · January 2011

Source: DBLP

CITATIONS
4

READS
64

3 authors, including:

Lucas de Oliveira Arantes
Universidade Federal do Espírito Santo
5 PUBLICATIONS **15** CITATIONS

SEE PROFILE

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS

SEE PROFILE

Using Semantic Annotations for Supporting Requirements Evolution

Bruno Nandolpho Machado, Lucas de Oliveira Arantes, Ricardo de Almeida Falbo

Ontology and Conceptual Modeling Research Group (NEMO),
Computer Science Department, Federal University of Espírito Santo, Vitória, Brazil
{brunonandolpho, lucasdeoliveira}@gmail.com, falbo@inf.ufes.br

Abstract — **Requirements change for a variety of reasons and at different stages of the project development. Changes to items in a requirements document must be propagated to other items that depend on the changed items in order to maintain their consistency. This paper explores the use of semantic annotations in requirements document templates to support requirements evolution.**

Keywords – Requirements evolution; requirements documentation; semantic documentation.

I. INTRODUCTION

The Requirements Engineering (RE) process plays a key role to ensure that software products will fully support and evolve with the business processes. It is the process by which requirements are gathered, analyzed, documented, and managed throughout the software lifecycle [1]. During the RE process, elicited requirements need to be precisely specified and documented. Moreover, requirements change for a variety of reasons. Such changes may be required at various stages of the software lifecycle, and must be propagated through other items that depend on the changed items. In order to maintain their mutual consistency, we need to manage traceability links among items and to propagate changes along such links [2].

Requirements are usually recorded in one or more documents, which are used to communicate requirements to different stakeholders. There are many different ways to structure requirements documents, depending on, among others, the type of the system being developed, the target audience, the level of detail to be considered, and organizational practices. To ensure that the essential information is included in each document, organizations should define their own standards for requirements documents. If an organization works with different types of requirements documents, it should define an appropriate template for each requirements document type [3, 4].

Despite its shortcomings, structured natural languages, augmented with graphical models, remains the most practical way for most software organizations to document their requirements [4]. Moreover, despite the current advances in electronic documentation along with the boom of collaborative text edition tools (such as wiki engines), desktop text editors are still the most frequently solution used by software organizations when it comes to electronic documentation [5,6]. Whether on the use of wiki engines or the use of desktop text editors, documents produced by these tools are still the main vehicle for knowledge dissemination [5,7,8]. This is the case of software engineering in general, and RE in particular.

Requirements documents hold a considerable amount of information that are to be mainly interpreted by human readers, such as requirements statements, use case descriptions, and so on. Managing requirements evolution requires reading different versions of different documents, in a task that is often dull and error prone. In addition, gathering relevant information contained in different documents spread through the organization's repositories demands a considerable effort and, because of that, this activity is often skipped [5].

Requirements traceability can be more easily achieved if the semantic content of the requirements documents could be exposed in order to allow visibility of the data and the relationships embedded in the document, and if the semantic content of each document version is extracted and registered into a version control system. In order to make these scenarios possible, it is essential to allow semantic metadata annotation into documents, turning requirements documents into semantic requirements documents.

For dealing with theses problems, Arantes and Falbo [9] developed an Infrastructure for Semantic Document Management (ISDM) [9] that presents the following features: semantic annotation of document templates; traceability support; searching based on extracted semantic content; and change notification subscription.

This paper discusses how this infrastructure is used to support requirements evolution, and is organized as follows: Section II talks briefly about requirements evolution, and semantic documentation; Section III shortly presents the ISDM and its main components; Section IV addresses the use of ISDM in the requirements management context, and presents some preliminary results from using ISDM in practical situations. Section V compares our work with some related ones. Finally, Section VI presents our conclusions.

II. REQUIREMENTS EVOLUTION AND SEMANTIC DOCUMENTATION

Two important activities of the RE process are requirements documentation and evolution. The results of requirements development should be documented for latter

agreement between customers and developers about the software to be built, and to serve as basis for further development and evolution.

As said before, requirements can be recorded in one or more documents, each one devoted to different classes of stakeholders. Pfleeger and Atlee [10], for instance, suggest the use of two different types of documents: the requirements definition, which is written in terms of the customer's vocabulary, and the requirements specification, which is written in terms of the system's interface.

There are many different ways to structure these documents. Ideally software development organizations should define templates for different types of requirements documents, imposing a standard structure on them [2,4].

With regard to the requirements definition document, it should specify the agreed requirements statements in a language that supports communication with stakeholders. The most obvious option is to document these statements using free prose in natural language. However, this approach is prone to several defect types, such as ambiguity. To overcome these problems, we may adopt a disciplined approach for documentation, using structured natural language. In such approach, stylistic rules on how statements should be written, and predefined statement templates, among others, are used to discipline writing the statements [2].

The agreed statements posed in a requirements definition document are subject to change. In order to maintain consistency, these changes must be propagated through other items that depend on the changed item. Since evolution is inevitable, we must prepare for change, and traceability is one of the most important ingredients for this. Consistency maintenance requires managing traceability links among items. However, traceability management is not an easy task. To take full advantage of its benefits, we need to reduce the complexity and cost of establishing and maintaining the traceability graphs [2].

A way of addressing this issue is applying a semantic documentation approach to requirements documentation. Allowing users to add metadata annotations to documents can improve the understanding and accessibility of the data contained on it. Features such as document annotation, data extraction from metadata, and data indexing and searching, are somehow the basis for semantic documentation [7,8,9].

III. An Infrastructure for Semantic Document Management

In essence, the Infrastructure for Semantic Document Management (ISDM) provides: (i) a way to semantically annotate document templates; (ii) a mechanism for controlling versions of the semantic content extracted from semantic document versions, and therefore providing a way for tracking the evolution of the data embedded inside a semantic document; and (iii) data visibility to end-users, allowing searches and data-change notification subscription, to aid developers to get an up-to-date information about something he/she is interested in [9].

As shown in Figure 1, the ISDM architecture is composed by two main elements [9]: the Semantic Document Repository (SDR), which is responsible for storing semantic documents; and the Main Module, which is composed by three sub-modules: (i) the Semantic Annotation Module is responsible for allowing users to semantically enrich a document template; (ii) the Data Extraction and Versioning Module is responsible for extracting the semantic content from an annotated document whenever a new version of that document is checked into the SDR. After extraction, the semantic content of the version is stored in another repository, called Data Repository, that is also part of this module; (iii) the Search and Traceability Interface Module is responsible for providing an API (Application Programming Interface) that allows users and other systems to perform ontology-based searches and data traceability towards the Data Repository.

In a nutshell, document engineers annotate document templates using the Semantic Annotation Module. Later on, developers and analysts may instantiate that template, generating semantic documents. These semantic documents are checked into the SDR. When a new version of a semantic document is available at the SDR, the Data Extraction and Versioning Module unwraps the semantic content of that version and stores it into the Data Repository, making the information of that version available. At this point, it is possible to further enhance the integration of information scattered throughout various semantic documents contained in the SDR. This is done by merging the graphs corresponding to the last versions of the documents contained in the repository. Finally, users and other systems may interact with the Search and Traceability Interface Module in order to perform queries about the evolution of a particular semantic content stored in the Data Repository.

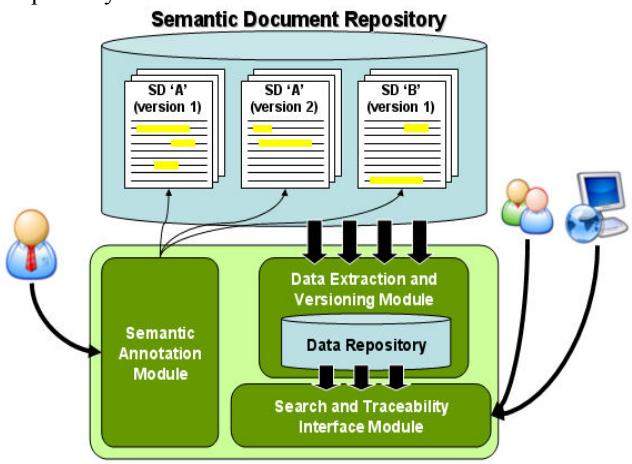

Figure 1. ISDM Architecture.

For annotating document templates, domain ontology-based annotations are used. ISDM works with templates and

documents written in Open Document Format (ODF) [11], and Open Office [12] was chosen as the document editor for composing annotations in document templates. In order to allow template annotation, specialized instructions for annotating text fragments and tables were developed. These kinds of annotations encapsulate annotations directed for document instances, allowing document engineers to add a set of instructions directly in these document elements (tables and text fragments). These instructions are processed when a document instance is analyzed by the Data Extraction and Versioning Module that will ultimately generate instances and relations accordingly. For more details, see [9].

IV. SUPPORTING REQUIREMENT EVOLUTION WITH SEMANTIC ANNOTATIONS

During the software life cycle, changes in requirements are quite common. New requirement dependencies are discovered, requirements statements are rewritten and priorities change. If a change is propose in a requirement, we need to identify how this change affects others requirements. In this scenario, the traceability matrix is an important artifact that is used to analyze the impact of changing requirements. Defining a network of requirement dependencies is the starting point for creating the requirements traceability matrix. Moreover, the use of a requirements management tool can support developing traceability matrices.

To deal with this, we are currently using the ISDM for supporting some tasks of the RE process at NEMO (Ontology & Conceptual Modeling Research Group). In NEMO, we use two types of documents for documenting requirements: a Requirements Document (RD) and a Requirements Specification (RS). The first is directed to clients and users, and captures user requirements. It is written in natural language, following rules defined for writing requirements statements. The second details the user requirements into systems requirements, and serves as basis for further development. It is mainly composed by models (use case diagrams, class diagrams, state diagrams, among others), although there are also some textual parts, especially the ones related to use case descriptions. Due to space limitations, in this paper we only discuss how we are using semantic annotations in RDs.

A. Annotating the Requirements Document Template

The NEMO's RDs are composed by some preliminaries, and four sections. For elaborating a RD, developers should follow the RD template shown in Figure 2.

First of all, there are two important information placed in text fragments: the name of the project, and the names of the responsible for the document. For capturing them, the ISDM provides annotations for annotating text fragments. Following we present the text fragment annotations added to the RD template. These annotations are done based on the requirements ontology proposed in [13].

Requirements Document

Project : <<project name>>
Responsible: <<names of the responsible analysts, separated by commas>>

1. Introduction

This document presents the user requirements of the <<system name>>. It is organized as follows: Section 2 describes the system purpose; Section 3 presents a description of the problem domain; Section 4 presents the user requirements elicited from clients and users.

2. System Purpose

<<one paragraph describing the system purpose>>.

3. Domain Description

<<a free text briefly giving an overview of the domain, describing the problem to be solved and business processes to be supported>>

4. User Requirements

Functional Requirements

Id	Statement	Priority	Depends on
FRXX	<<sentence following defined pattern>>	<<possible values: High, Medium, Low>>	<<ids of the requirements on which the requirement depends>>

Business Rules

Id	Statement	Priority	Depends on
BRXX	<<idem FR>>	<<idem FR>>	<<idem FR>>

Non Functional Requirements

Id	Statement	Priority	Category	Depends on
NFRXX	<<idem FR>>	<<idem FR>>	<<type of the NFR>>	<<idem FR>>

Figure 2. Requirements Document Template

[[completeText]];instance({content},http://localhost/ontologies/SE/onto.owl# Project,$project);

[[break with ',']];
instance({slice},http://localhost/ontologies/SE/onto.owl#Person, $person);

property($person,http://localhost/ontologies/SE/onto.owl#involvedIn,$project);

The first annotation is added in the place marked with the <<project name>> tag; the second is added in the place marked with the <<names of the responsible analysts, separated by commas>> tag. The third annotation is done based on the requirements ontology, and says that the people informed as responsible analysts are involved in the project. Other tags shown in text fragments, such as the one related to the system purpose, are annotated in a similar way.

ISDM also provides annotations for annotating tables, which were used in the RD template for annotating the Domain Description table, the Functional Requirements (FR) table, the Business Rules (BR) table and the Non Functional Requirements (NFR) table. In the case of FR table, as shown in Figure 2, the first column refers to the FR id; the second to the FR statement, the third to its priority, and the last one to a list of the ids of the requirements (FRs, BRs and NFRs) on which it depends, separated by comma. Annotations in tables allow that each column has a different set of instructions.

The following annotations were added to the FR table in the semantic template:

[[ignorerow0]];

[[at0]];instance({content},http://localhost/ontologies/SE/onto.owl#FunctionalRequirement,#req);
property(#req,http://localhost/ontologies/SE/onto.owl#artifactProducedIn,#project);

[[at1]];property(#req,http://localhost/ontologies/SE/onto.owl#description,{content});

[[at2]];property(#req,http://localhost/ontologies/SE/onto.owl#priority,{content});

[[at 3 break with ',']];instance({slice},http://localhost/onto.owl#Requirement,#reqline);
property(#req,http://localhost/ontologies/SE/onto.owl#relatedWith,#reqline);

The first annotation only says to ignore the first row in the table, since it does not contain data (it is a header). The annotations that begin with *[[at]]* are used for annotating the content from a column. The second annotation indicates that the content of the first column is a FR and that this FR is produced in the project informed before. The third annotation points out that the content of the second column is the statement of the FR captured in the first column. Finally, the last annotation says that the current FR depends on the requirements informed in the fourth column.

The other tables and their annotations are quite similar to the FR table. Thus, making use of the semantically anotated template, it is possible to the Data Extraction and Versioning Module to extract the semantic content present in the requirements document, as explained next..

B. Extracting and Visualizing Information from the RDs

Once the RD template is instantiated and a new version of the RD is produced, it should be committed in the corresponding Semantic Document Repository (SDR) (see Figure 1), in order to the Data Extraction and Versioning Module (DEVM) process the semantic document. DEVM generates instances and relations accordingly to the contents of the document in the Data Repository, in an OWL format.

Whenever a new version of a semantic document is commited in the repository, a new version of the document content is generated in the Data Repository, and it is possible to use the services from the platform.

Figure 3 presents part of the tables concerning functional requirements and business rules, produced in the context of the Cargo Delivery Control Project. The versions of the RDs of this project are stored in the the repository /home/nemo/reposg7. Taking the third verson of this RD into account, it is possible to generate the traceability matrix shown in Figure 4. The full matrix is not presented in this paper due to space limitation. It is worthwhile to point out that as the data is stored in OWL format, we can make some inferences using the JENA engine.

Functional Requirements

Id	Statement	Priority	Depends on
FR01	The system shall control types of cargos.	High	
FR02	The system shall control transportation rates.	High	FR01, FR12, BR03, BR05
FR03	The system shall allow the customer to make a request to transport a cargo from a quotation previously made.	High	FR02, BR02, NFR02, NFR06
…	…	…	…

Business Rules

Id	Statement	Priority	Depends on
BR01	Clients who have hired a service can not be excluded.	High	
BR02	The system must generate a unique identifier for each quotation.	High	
BR03	Each cargo type has a procedure to be followed to be transported.	High	

Figure 3. Part of a Requirements Document.

As the dependency relationship (the last column of the requirements tables) is transitive, we can infer, for instance, that if FR03 depends on FR02 and FR02 depends on FR01, then FR03 depends on FR01. Direct dependencies are shown in the figure with the flag "D"; inferred dependencies, on the other hand, are marked in the table with the letter "I". Such information is very important, since it is useful for analyzing the impact of a change. Inferring requirements dependencies in large projects can be difficult, laborious and error prone to perform manually.

Req	FR01	FR02	FR03	...	BR01	BR02	BR03
FR01				...			
FR02	D			...			D
FR03	I	D		...		D	I
…				...			

Figure 4. Part of the Traceability Matrix.

Another service provided by ISDM is to trace the differences between versions of a document. Using this service, a developer may look for the changes made in a specific document. Figure 5 shows the results from a query for providing the differences between versions 2 and 3 of the RD partially presented in Figure 3. This document was modified, for instance, by changing the statement of FR03 and the requirements on which it depends. Such changes are preceded by "(CHANGED)" in the figure. Moreover, there were three FRs added and one removed, which are represented in the figure by lines beginning with the words "(ADDED)" and "(REMOVED)", respectively. Using this service, we can follow up requirements evolution, as they are listed when they are added, removed or modified in a new version of the RD.

On the other hand, during the execution of the project, a developer may want to know the history of a specific requirement. To this end, the developer can use the service for visualizing the evolution of a requirement, illustrated in Figure 6. As we can see in this figure, *FR04* was added in the Revision 1. The requirements on which it depends were changed in the Revision 2, when two related requirements

were added (*BR03* and *NFR01*). Finally, in revision 3, *FR04* was removed from the RD. With this service, a developer can trace the evolution of a specific requirement, detailing how it was related to other requirements and the values of its properties along the project.

Results:

Repository: file:///home/nemo/reposg7, File: /Requirements Document.odt, Revision Start, End: (2,3)

Revision 3

(CHANGED) http://localhost/ontologies/SE/onto.owl#FR03

property: http://localhost/ontologies/SE/onto.owl#description
from: The system must allow the customer to make a request to transport cargo from a quotation made, thus generating a worker order.
to: The system must allow the customer to make a request to transport cargo from a quotation made.

(CHANGED) http://localhost/ontologies/SE/onto.owl#FR03

property: http://localhost/ontologies/SE/onto.owl#relatedWith
from: http://localhost/ontologies/SE/onto.owl#BR13
http://localhost/ontologies/SE/onto.owl#NRF01
http://localhost/ontologies/SE/onto.owl#NRF05
to:

(CHANGED) http://localhost/ontologies/SE/onto.owl#FR08

property: http://localhost/ontologies/SE/onto.owl#relatedWith
from: http://localhost/ontologies/SE/onto.owl#RNF01
http://localhost/ontologies/SE/onto.owl#NRF03
http://localhost/ontologies/SE/onto.owl#NRF05
to: http://localhost/ontologies/SE/onto.owl#BR02
http://localhost/ontologies/SE/onto.owl#BR13
http://localhost/ontologies/SE/onto.owl#NRF02

(REMOVED) http://localhost/ontologies/SE/onto.owl#FR04

(REMOVED) http://localhost/ontologies/SE/onto.owl#FR05

(REMOVED) http://localhost/ontologies/SE/onto.owl#FR06

(ADDED) http://localhost/ontologies/SE/onto.owl#FR13

Figure 5. Results of the query regarding the differences between two versions of a RD.

Repository: file:///home/nemo/reposg7, File: /Requirements Document.odt, Individual: http://localhost/ontologies/SE/onto.owl#FR04

Revision 3 of / Requirements Document.odt

(REMOVED) http://localhost/ontologies/SE/onto.owl#RF04

Revision 2 of / Requirements Document.odt

(CHANGED) http://localhost/ontologies/SE/onto.owl#FR04

property: http://localhost/ontologies/SE/onto.owl#relatedWith
from:
to: http://localhost/ontologies/SE/onto.owl#BR03
http://localhost/ontologies/SE/onto.owl#NFR01

Revision 1 of / Requirements Document.odt

(ADDED) http://localhost/ontologies/SE/onto.owl#FR04

Figure 6- Search Form evolutionary tracing of an individual and results t

C. *A Preliminary Evaluation*

ISDM has been used to trace the requirements of projects performed at NEMO. Up to now, 7 projects used it. In 4 of these projects, the RDs evolved in two versions. In the other 3 projects, there were three versions. The various versions of these documents were added to the Semantic Document Repository (SDR), and each project had its own version control repository. The three tables shown in Figure 7 present data regarding the evolution of the RDs of these projects, including how many requirements were added, changed or removed in each version of each project.

Revision 1

Project	Added	Changed	Removed	Sum
Project 1	29	-	-	29
Project 2	10	-	-	10
Project 3	25	-	-	25
Project 4	29	-	-	29
Project 5	13	-	-	13
Project 6	10	-	-	10
Project 7	24	-	-	24

Revision 2

Project	Added	Changed	Removed	Sum
Project 1	11	16	12	24
Project 2	16	10	-	26
Project 3	3	22	3	25
Project 4	2	27	1	30
Project 5	11	12	-	24
Project 6	14	10	-	24
Project 7	9	17	5	28

Revision 3

Project	Added	Changed	Removed	Sum
Project 4	9	9	4	35
Project 6	13	15	2	35
Project 7	2	11	10	20

Figure 7. Changes of requirements in numbers.

The numbers shown in Figure 7 give an idea of how the use of ISDM for managing requirements has proved to be useful in NEMO. Since many requirements are added, removed and changed, especially at the beginning of the projects, controlling their changes and impacts in other artifacts are essential. The use of ISDM has speeded the impact analysis and, as a consequence, the time spent in performing changes. Before the use of ISDM at NEMO, those analyses were done by humans, using only the functionalities provided by OpenOffice for comparing documents. By showing the changes done and generating traceability matrices, ISDM provided a very useful basis for impact analysis.

V. RELATED WORK

Since evolution is inevitable, there are several works that aim at minimizing evolution efforts. Special attention has being devoted to requirements traceability management.

Rochimah et al. [14] evaluated seven traceability approaches focusing on their contributions to simplify software evolution tasks. Three of them generate traceability

links in an automated way, while other three are semi-automated, in the sense that they combine a manual and automated way in obtaining the traceability links. Regarding the degree of automation, our approach can be classified as semi-automated, since templates are annotated manually and the links between requirements are informed by the analysts when the templates are filled in. On the other hand, the traceability matrices are automatically generated, and the changes done can be queried. Concerning the degree of formality, Rochimah et al. consider that the seven evaluated approaches are semi-formal. This is also the case of our approach, which uses an ontology as basis for the template annotations. Finally, regarding the change type, as the other approaches, ours allows to identify additions, deletions, and modifications. A distinguishing feature of our approach is that analysts work in the same way they have always done, i.e. filling templates in a text editor, and committing them in the project's repository. In this way, our approach is in line with the "support in-place traceability" best practice, defined by Huang et al. [15] as traceability being provided to the artifacts residing within their native environments.

There are several requirements engineering tools, such as TRACE [16], that provide various forms of traceability and support to change. However, at the best of our knowledge, none of them uses a semantic documentation approach, and thus users have to interact with the tool, instead of writing requirements documents in a text editor. In our work, the idea is just to allow the analysts to continue using a desktop text editor and, by means of semantic annotations made in templates, extract and track requirements in a transparent way to the user.

VI. CONCLUSIONS

In this paper we discussed the use of ontology-based annotations in semantic Requirements Document (RD) templates for supporting requirements managing and evolution. Our strategy was to instantiate and to extend the Infrastructure for Semantic Document Management (ISDM) [9], developing functionalities that address important issues for requirements management, such as automatic generation of traceability matrices. The annotations are done in RD templates, and they are based on the conceptualization defined in the Software Requirements Ontology proposed in [13]. It is important to highlight that, since we annotate templates, the effort spent with annotations is very small compared with the benefits obtained.

The resulting tool was used to support requirements management in 7 projects. From this preliminary use, we have already glimpsed some improvements to be done. First, we need to develop a more user-friendly interface for performing searches and for displaying traceability matrices. Second, ISDM provides other features that can be explored in the context of Requirements Management. This is the case of the change notification subscription functionality. This feature allows notifying users when a given individual in a document changes. This could be useful to notify stakeholders when a requirement that they are interested in changes. Third, although our approach has shown to be useful for organizations that use a desktop text editor for documenting requirements, we know that, ideally, software organizations should use requirements management tools for that. Thus, we are working on integrating ISDM with the requirements management tool of ODE (Ontology-based Development Environment) [17], a Software Engineering Environment developed at NEMO.

ACKNOWLEDGMENTS

This research is funded by the Brazilian Research Funding Agencies FAPES (Process Number 45444080/09) and CNPq (Process Numbers 481906/2009-6 and 483383/2010-4).

REFERENCES

[1] A. Aurum, C. Wohlin, Engineering and Managing Software Requirements, Springer-Verlag, 2005.
[2] A. Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.
[3] G. Kotonya, I. Sommerville, Requirements engineering: processes and techniques, John Wiley, 1998.
[4] K.E. Wiegers, Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. 2nd Edition, Microsoft Press, 2003.
[5] T. C. Lethbridge, J. Singer, A. Forward, "How Software Engineers Use Documentation: The State of the Practice", IEEE Software, vol. 20, no. 6, pp. 35-39, Nov./Dec. 2003.
[6] Forward, A., Lethbridge, T.C., "The relevance of software documentation, tools and technologies: a survey". Document Engineering. DocEng'02, 2002.
[7] Bruggemann, B. M.; Holz K.P.; Molkenthin F., "Semantic Documentation in Engineering", Proceedings of the Eighth International Conference on Computing in Civil and Building Engineering , California, USA, August 2000.
[8] H. Eriksson, "The semantic-document approach to combining documents and ontologies", International Journal of Human-Computer Studies, Volume 65 , Issue 7, 2007.
[9] L.O. Arantes, R.A. Falbo, "An Infrastructure for Managing Semantic Documents", Proc. 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010.
[10] S.L. Pfleeger, J. Atlee, Software Engineering: Theory and Practice, 4th edition, Prentice Hall, 2009.
[11] OASIS Open Document Format for Office Applications. *www.oasis-**open**.org/committees/**office**/*
[12] OpenOffice.org – The Free and Open Productivity Suite. Visited in April, 30th 2010. http://www.openoffice.org/
[13] Falbo, R. A., Nardi, J., C., Evolving a Software Requirements Ontology. In: Proceedings of the XXXIV Latin-american Conference on Informatics - CLEI´2008, Santa Fé, Argentina, 2008. p. 300-309.
[14] S. Rochimah, W. M. N. Wan Kadir, A. H. Abdullah, "An Evaluation of Traceability Approaches to Support Software Evolution", The Second International Conference on Software Engineering Advances (ICSEA 2007), French Riviera, France, 2007.
[15] J. Cleland-Huang, R. Settimi, E. Romanova, B. Berenbach, S. Clark, "Best Practices for Automated Traceability", Computer, Vol. 40, Issue 6, pp. 27 – 35, June 2007.
[16] H.P.-J.Thunem, "TRACE: A Generic Tool for Dependable Requirements Engineering". The European Safety and Reliability 2009 Conference, Volume 1, pp. 137–142, Prague, Czech Republic, 2009.
[17] R. A. Falbo, F. B. Ruy, R. Dal Moro, "Using Ontologies to Add Semantics to a Software Engineering Environment", Proceedings of the17th International Conference on Software Engineering and Knowledge Engineering, Taipei, China , 151-156 , 2005.