Merging Software Maintenance Ontologies: Our Experience

Aurora Vizcaíno[1], Nicolas Anquetil[2], Kathia Oliveira[2], Francisco Ruiz[1], Mario Piattini[1]

[1]Alarcos Research Group. University of Castilla-La Mancha, Spain
{aurora.vizcaino| francisco.ruizg|mario.piattini}@uclm.es
[2]University Católica de Brasilia, Brazil
{anquetil|kathia}@ucb.br

1. Introduction

Ontologies capture consensual knowledge of a specific domain in a generic and formal way, to allow reusing and sharing it among groups of people. However, there is not a unique possible ontology to model for a particular domain. For example, in the case of the software maintenance domain there are several published ontologies (Kitchenham et al., 2001), (Deridder, 2002), (Dias et al., 2003), (Ruiz et al., 2004), each one dealing with the maintenance process with a different focus. After reviewing the different ontologies defined for maintenance we decided to merge two ontologies that seemed to be complementary, these are the ontology described in (Dias et al., 2003) which identifies what knowledge is needed during maintenance process and the ontology proposed by Ruiz et al. (2004) focused on the management of software maintenance projects. As both ontologies are based on Kitchenham's work (1999) and have a very similar objective, we believed that merging them would be a relatively easy task. In the next section we describe our experience trying to do so. Finally, conclusions are outlined.

2. Merging Ontologies

Noy and Musen (1999) define ontology merging as the generation of an unique ontology (hereafter termed the new ontology) from different original ones (hereafter termed the sources ontologies). There are different methods to merge ontology: ONIONS (Steve et al., 1998), FCA-Merge (Stumme and Maedche, 2001) or PROMPT (Noy and Musen, 2000). We used a method similar to that proposed in PROMPT. The first activity of this method consists in making a list of the concepts considered in all the sources. This stage is very useful to detect concepts that are defined in only one ontology, and concepts that are common to two (or more) sources ontologies. The second activity is to detect what terms with different nouns or label

represent the same meaning (synonyms) and what terms which have the same name in both ontologies actually represent different concepts. At this stage, the ontology designers must also decide what they want to focus on and what information is considered important or irrelevant (Gómez-Pérez et al., 2004). The third activity, when the new ontology has been developed, is to find conflicts, for instance, more than one term with the same noun or redundancy.

Before explaining how we carried out these activities to merge two source ontologies for software maintenance (Dias and Ruiz) we will shortly describe the two ontologies. Dias' ontology is composed of five sub-ontologies: the Sytem sub-ontology, the Skills sub-ontology, the Modification Process sub-ontology, the Organizational Structure sub-ontology and the Application Domain sub-ontology. Ruiz's ontology is formed of four sub-ontologies: The Products sub-ontology, the Activities sub-ontology, the Process Sub-ontology and the Agents sub-ontology. As the decomposition in sub-ontology presents some intersection, we decided to work iteratively, merging sub-ontologies instead of the whole ontologies. In the first iteration, we merged the System sub-ontologies (Dias, see Figure 1) with the Product ontology (Ruiz, see Figure 2).

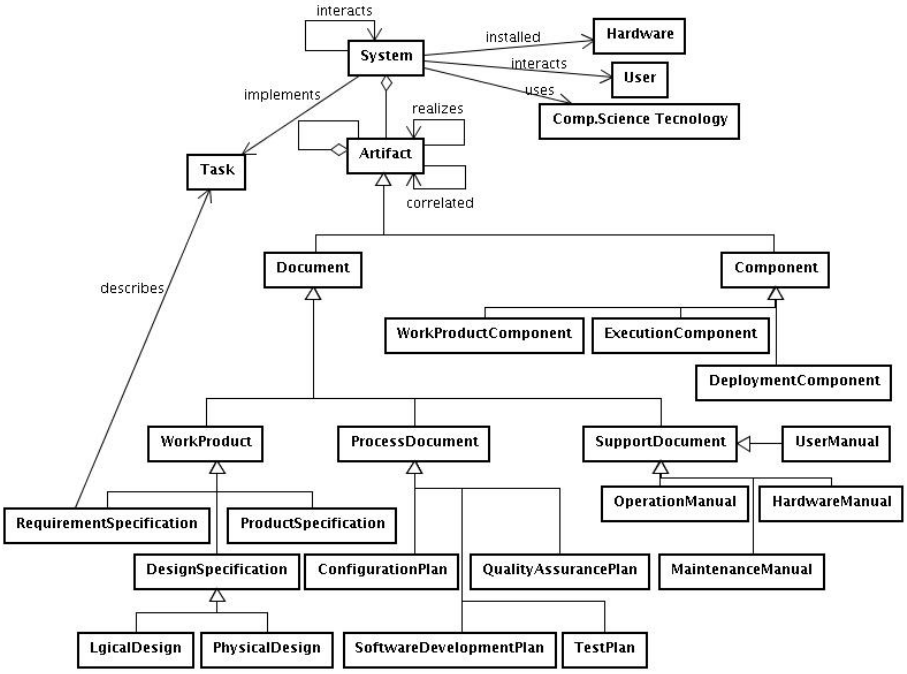

Figura 1. System sub-ontology (Dias)

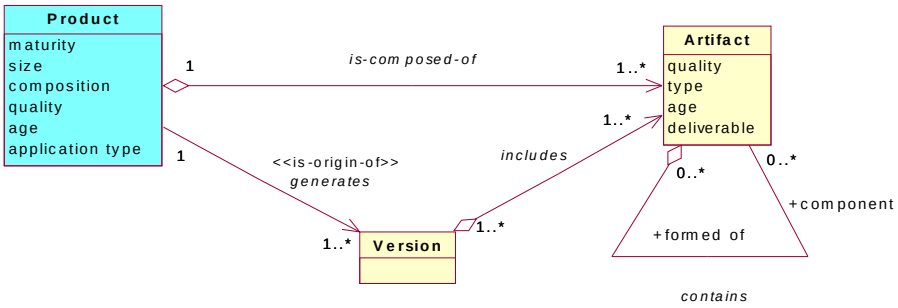

Figure 2. Product sub-ontology

Following the method described above, we created the list of concepts considered in each ontology. In the Product sub-ontology only three concepts were defined: *product*, *artifact* and *version* while in the case of the System sub-ontology twenty-three concepts were represented. We realized that the System sub-ontology (Dias) took into account a lot of information that was ignored in the Product sub-ontology (Ruiz). Our conclusion was that the level of granularity of the two sub-ontologies was very different. Ruiz's sub-ontology represents a Product as a composition of Artifacts and explicits the notion of Version. Dias' sub-ontology represents a Software System as a composition of Artifacts, it does not represent the notion of Version, but it describes in more details what Artifacts are (Documents or Software Components that are further decomposed in different types of documents and components). It also states some relations a System may have, for example, that a System interacts with Users and other Systems, implements some Domain Tasks and is installed on some Hardware.

With this difference of granularity in mind we decided to split the second activity into two parts, first we asserted what common concepts had different names (synonyms); second we discussed what level of granularity our ontology should have and what concepts should be merged or ignored in the new sub-ontology.

At first glance, we realized that only one concept (Artifact) was present in both sub-ontologies. Then, we considered the Product and System concepts. After some discussion, it became apparent that the different meanings of “system” (operating system, software system, etc.) led the two groups on differing tracks. It was decided that the concept was the same in both sub-ontology and that it would be named Software Product in the final sub-ontology and it represents “the software that a maintenance company should maintain”. It was also agreed that the concept Version should be included in the new sub-ontology.

The following step was to determine the granularity of the new ontology. To do it, it was proposed to choose what information could be omitted at first glance. Although, everybody agreed that a user interacts with a system, and a system is installed on some hardware and that a system uses technology it was consider that this information was not directly related to the product ontology. Therefore, these three relationships were deleted. On the other hand, the Component concept might be only divided into Execution Component and Source component, since Deployment Component and Work Product Component may be considered into Source Component. This proposal was also accepted, because the new subdivision was

simpler and clearer. The most difficult point was that related to the division of Documents. As Figure 1 shows, the System sub-ontology gives a lot of details about the types of documents. However, some persons claimed that it was not convenient to propose such deep categorization for the documents since the documents originated depend on the methodology used, as each methodology recommends a type of documents. Other persons said that the classification proposed in the system sub-ontology was the same that other experts in software maintenance there were proposed and it could be considered a general classification for maintenance documentation. Figure 3 shows the provisional sub-ontology called Software Product. The new sub-ontology that arose from these discussions and months of work is presented in Figure 3.

In the second iteration, two other sub ontologies were merged: Dias' Modification Process sub-ontology, Ruiz's Activities sub-ontology, and Ruiz's Process Sub-ontology. The problems encountered were similar but aggravated by the fact that the mapping between the sub-ontologies was not exact (as for the Product and System sub-ontologies). First Ruiz decomposes the process sub-domain in two (process and activities) when Dias only uses one sub-ontology. Second, some concepts in one group were found in the other ontology but in yet another sub-ontologies, for example the Technology concept belongs to Ruiz's Process sub-ontology whereas it is in Dias' Skills sub-ontology. This fact shows that the sub-domains have borders in common which one may decide to include in one or the other sub-ontology.

3. Conclusions

In this paper we propose to share our experience merging two software maintenance ontologies. We have adopted an iterative approach where two sub-ontologies have already been agreed upon and work is continuing on merging the rest of the sub-ontologies. As an example, we explained how we merged two sub-ontologies (Dias' System and Ruiz's Product). This example was chosen because it is very simple, one sub-ontology having only three concepts. However simple this example, it showed that from only three concepts in Ruiz's sub-ontology, one (Version) was not present in the other source and had to be added in the final sub-ontology; one was in both sources (Artifact) but at different granularity levels; and one was in the other sub-ontology but with another name.

From this simple example, we conclude that the different proposals to automate, even a part of, the merging process is almost impossible since it requires good knowledge of the domain, understanding of each ontology point of view, and even the use of negotiation strategies between the designers of the different ontologies in order to make proposals, discuss them and to reach an agreement. The difference in point of view appears as a strong complication factor as it may include cultural aspects where to each ontology designer, his-her own point of view seems extremely natural and straightforward whereas the other's point of view appears at first, if not wrong, at least very unnatural. Finally, this process may be further complicated by the expressing language. In our case, both ontologies were expressed in English as a scientific inter-lingua, however, neither of the two groups are native English speakers which led to

some difficulties as each group translated its ideas from its own mother tongue back and forth to English.

Acknowledgment

This work is partially supported by the MAS project (grant number TIC2003-02737-C02-02), Ministerio de Ciencia y Tecnología, Spain, and the ENIGMAS project (grant number PBI-05-058), Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, Spain.

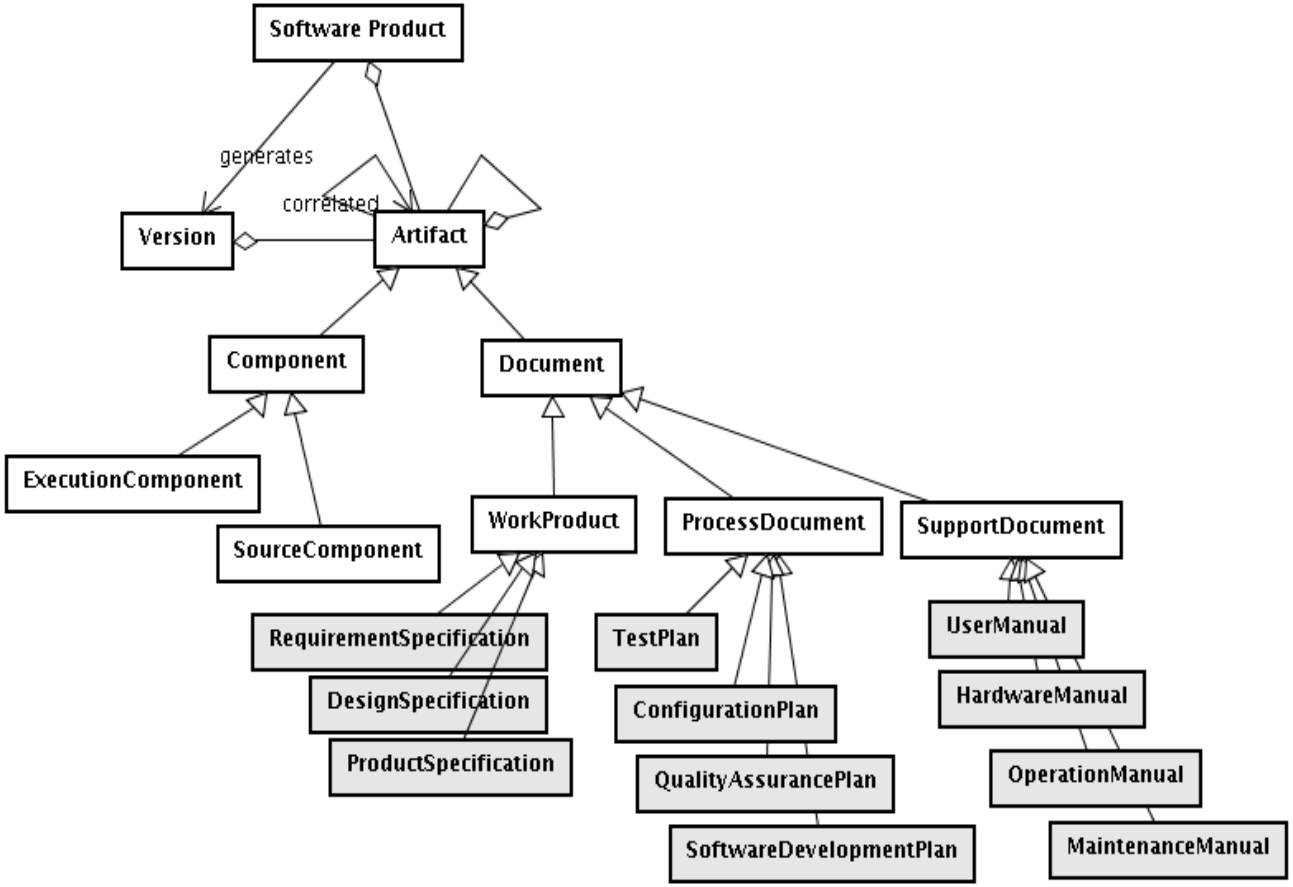

Figure 3: Merged sub-ontology

REFERENCES

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

Dias, M. G. B., N. Anquetil, et al. (2003). "Organizing the Knowledge Used in Software Maintenance." Journal of Universal Computer Science 9(7): 641-658.

Gómez-Pérez, A., M. Fernández-López, et al. (2004). Ontological Engineering.

Kitchenham, B. A., G. H. Travassos, et al. (1999). "Towards an Ontology of Software Maintenance." Journal of Software Maintenance: Research and Practice **11**: 365-389.

Noy, N. and M. Musen (1999). SMART: Automated Support for Ontology Merging and Aligment. 12th Banff Workshop on Knowledge Acquisition, Modeling, and Management, Banff, Alberta (Canada).

Noy, N. and M. Musen (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. Workshop on Ontologies and Information Sharing, Seattle, Washington.

Ruiz, F., A. Vizcaíno Barceló, et al. (2004). "An Ontology for the Management of Software Maintenance Projects." International Journal of Software Engineering and Knowledge Engineering **14, Nº 3**: 323-349.

Steve, G., A. Gangemi, et al. (1998). Integrating Medical Terminologies with ONIONS Methodology. Information Modeling and Knowledge Bases VIII, IOS Press.

Stumme, G. and A. Maedche (2001). FCA-MERGE: Bottom-Up Merging of Ontologies. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001), Seattle, Washington, Morgan Kaufmann Publishers.