Exploring Ontologies to Support the Establishment of Reference Architectures: An Example on Software Testing

Elisa Yumi Nakagawa[1], Ellen Francine Barbosa[1], José Carlos Maldonado[1]

[1]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - USP
São Carlos-SP, Brasil

Software architectures have played a significant role in determining the success of software systems. In particular, reference architectures have emerged, achieving well-recognized understanding in a specific domain, promoting reuse of design expertise and facilitating the development of systems. In another perspective, ontologies have been widely investigated aiming at representing, communicating and reusing knowledge. In spite of their relevance on directly dealing with domain knowledge, reference architectures and ontologies have been separately treated. In this paper we investigate the impact in using ontologies to the establishment of reference architectures. We illustrate our idea using an ontology of software testing to build a reference architecture for the testing domain. Preliminary results indicate that ontologies are an important and viable mechanism aiming at building reference architectures.

Laboratories:
Labes

Publication:
WICSA(2009)

Funding:
FAPESP, CNPq, CAPES,
Ministério da Ciência e Tecnologia,
QualiPSo Project (IST-FP6-IP-034763

Exploring Ontologies to Support the Establishment of Reference Architectures: An Example on Software Testing

Elisa Yumi Nakagawa, Ellen Francine Barbosa and José Carlos Maldonado
Dept. of Computer Systems, University of São Paulo – ICMC/USP
São Carlos, SP, Brazil
{*elisa, francine, jcmaldon*}*@icmc.usp.br*

Abstract

Software architectures have played a significant role in determining the success of software systems. In particular, reference architectures have emerged, achieving well-recognized understanding in a specific domain, promoting reuse of design expertise and facilitating the development of systems. In another perspective, ontologies have been widely investigated aiming at representing, communicating and reusing knowledge. In spite of their relevance on directly dealing with domain knowledge, reference architectures and ontologies have been separately treated. In this paper we investigate the impact in using ontologies to the establishment of reference architectures. We illustrate our idea using an ontology of software testing to build a reference architecture for the testing domain. Preliminary results indicate that ontologies are an important and viable mechanism aiming at building reference architectures.

1. Introduction

Over the last decades software architecture has received increasing attention as an important subfield of Software Engineering [1]. Software architecture is the structure or structures of the system which comprise software elements, the externally visible properties of those elements, and the relationships among them [2]. In this context, reference architectures for different domains – e.g. embedded systems, e-commerce and web browsers – have emerged. A reference architecture captures the essence of the architectures of a collection of systems of a given domain. The purpose of a reference architecture is to provide guidance for the development of architectures for new systems or extended systems and product families. They can be also seen as a knowledge repository of a given domain. Furthermore, it is worth highlighting that software architectures, including reference architectures, play a major role in determining system quality (e.g. performance and maintainability) since they form the backbone for any successful software-intensive system. Considering the relevance of reference architectures, approaches to establish them can be found [3], [4], [5].

In another perspective, ontologies have been widely investigated as a formal explicit specification of a shared conceptualization. Hence, an ontology basically consists of concepts and relations, as well as their definitions, properties and constraints expressed by means of axioms [6]. Ontologies have been applied to describe a variety of knowledge domains, such as medicine, engineering and law. In the field of software engineering, ontologies can also be found [7]. In this field, we established OntoTest [8], an ontology of software testing. Ontologies have diverse applications. For instance, they can be used as a repository of domain knowledge and, therefore, as a basis to develop knowledge-based systems and systems for that domain. Some initiative can be found in order to use ontology to support development of software architectures [9]; however, there is a lack of work that investigate the impact of using ontologies to design reference architectures.

Motivated by this scenario, we have hypothesized that, since ontologies can be explored as repositories of domain knowledge, their use aiming at supporting the definition of reference architectures is an interesting and appropriate approach. Thus, in this paper, we have as main goal to present how ontologies can be used in the context of a process to establish reference architectures, illustrating it using OntoTest in the testing domain. We explore our ideas in the context of ProSA-RA, a process specifically proposed to build reference architectures.

The remainder of this paper is organized as follows. Section 2 presents the related work. In Section 3 we present how an ontology can contribute to the reference architecture establishment. In Section 4 we summarize our results, contributions and discuss perspectives for further work.

2. Related Work

A number of methods for the software architecture design has been proposed; however, regarding to reference architecture design, Muller [5] has proposed recommendations in order to create and maintain reference architectures. In the context of product line development, Bayer [3] presents PuLSE-DSSA, a systematic approach to define reference architectures capturing knowledge from existing system

architectures. Other works have pointed out the need of formalizing processes to design reference architectures [4], since informal processes have still been used. In another perspective, Alkerman and Tyre [9] have investigated the use of an ontology in the establishment of software architectures, in particular, architectural instances. This ontology organizes concepts related to architecture decision domain, such as system, subsystem, concern, component and interface. It is important to notice that the use of ontologies of specific domains has not been widely investigated as basis to develop software architectures, including reference architectures.

3. Establishing a Reference Architecture

To establish a reference architecture for software testing domain, we used OntoTest, an ontology of software testing which supports acquisition, organization, reuse and sharing of knowledge on the testing domain. OntoTest explores different perspectives involved in the testing activity, such as techniques and criteria, human and organizational resources, and automated tools. Specifically, it defines a common well-established vocabulary for software testing. Due to the complexity of the testing domain, OntoTest adopts a layered approach. On the ontology level, the Main Software Testing Ontology addresses the main concepts and relations associated with testing. On the sub-ontology level, specific concepts from the Main Software Testing Ontology – testing process, testing artifacts, testing steps, testing strategies and procedures, and testing resources – are refined and treated into details. OntoTest encompasses 115 concepts and for each basic concept represented in the Main Software Testing Ontology, there is a number of sub-concepts which are refined and treated in the sub-ontologies. Besides graphical representation, a set of axioms to formalize the testing ontology and a set of textual definition of each concept are also available. OntoTest is described in OWL[1], using Protégé[2] ontology development tool. We have also developed a web-based application – the *OntoTestSearchingTool* – to disseminate and harmonize testing concepts.

ProSA-RA is a process to develop, describe and evaluate reference architectures. Although it was conceived to establish architectures which include aspects (from AOP - Aspect-Oriented Programming [10]) in their structures, it is also applicable for other types of architectures. The outline structure of ProSA-RA is illustrated in Figure 1. To establish reference architecture for testing domain using ProSA-RA, the following steps were performed:

Step RA-1: Information Source Investigation: Information sources are firstly selected and investigated. In our case study, information was arisen considering diverse sources: (i) Documents: books, technical reports and articles were

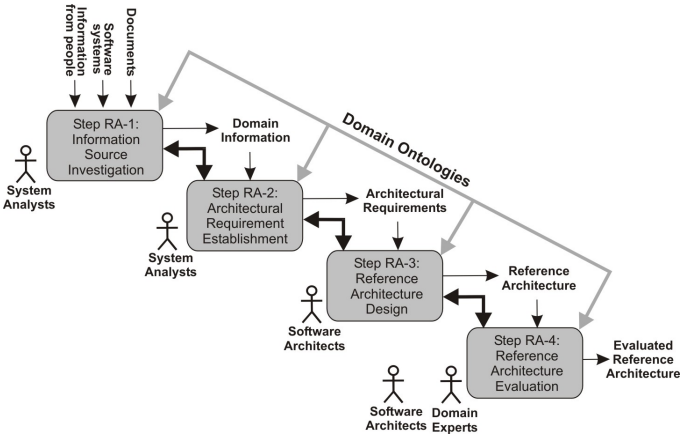

Figure 1. Outline Structure of ProSA-RA

investigated. We concentrated in work that propose software testing processes, investigating and identifying the potential core software testing activities. Furthermore, software architectures proposed in the testing literature were also investigated and the knowledge (structure, modules, and their relationships) was considered; (ii) Software systems: the investigation of testing tools was also an important contribution for this step. Eight known academic testing tools were considered and two broadly known and used commercial testing tools; and (iii) People: our experience in developing testing tools as well as the experience of other developers were also very important.

It is important to highlight that the arisen information is almost always non-structured, non-organized and missing. Thus, ontologies are particularly important to support this step, since they represent the knowledge domain in a well-structured format. Thus, the searching of ontologies for the domain is the first task. Since each ontology can cover certain knowledge aspects and can use a different terminology, connection between the various terminologies must be analyzed. In order to systematize the identification, selection and analysis of ontologies, Systematic Reviews[3] can be conducted. Some characteristics that we consider important in an ontology are: (i) it must encompass the domain concepts in order to really represent the domain; (ii) it must be as detailed as possible; and (iii) it can be automated. For the case illustrated in this paper, we have ourselves developed the ontology; thus, the searching task was not conducted.

In our case, during the investigation of the information sources, OntoTest supported three main tasks: (i) To understand the testing domain concepts and their relationship as well. Although the software testing domain is a consolidated

1. http://www.w3.org/TR/owl-ref/
2. http://protege.stanford.edu

3. A Systematic Review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest, using a trustworthy, rigorous, and auditable methodology.

research area, there is not a consensus about all concepts of this domain; (ii) To group, structure and associate the arisen information. Three types of relation can be used in order to facilitate organizing the information: association, composition and generalization[4]. From a set of information, some relationships are easily identified; however, other relationships are not trivial or are indirect; and (iii) To define the bound of the intended reference architecture. A good and complete ontology, when available, supports the establishment of the architecture scope, i.e. which elements/modules must or not be inserted into architecture. Even if the ontology is not complete, we believe that its use is relevant, since at least the main and more important concepts should be considered in the ontology. Ontologies help to answer questions, such as: Are we conscious about what we will consider in the architecture? Are we certain about the bound (scope) of the architecture?

For an effective use of the ontology in this step, the availability of the ontology in the automated way, like in Protégé or tools like *OntoTestSearchingTool*, is interesting. This reduces the efforts to find and understand the domain concepts and their relationship. In this step, information about the domain are still informally registered and, sometimes, in the mind of involved people.

Step RA-2: Architectural Requirement Establishment: Based on the arisen information, we wrote down a document containing the architectural requirements of the intended architecture. It is observed that these requirements are more general than the requirements of an architectural instance, since they must describe the set of architectural instances of testing tools. Table 1 presents part of these requirements, grouped in three categories – (i) general requirements, (ii) requirements related to core of testing tools, and (iii) requirements related to testing supporting modules – aiming at characterizing a first identification of the great modules of the architecture. After that, we mapped each requirement to a testing concept in OntoTest. Results of this mapping are listed in the third column of Table 1. The identification of the related concepts is important, since these concepts will guide the module identification in next step. In future the use of the architecture, these modules must satisfy and automate the related requirements.

Step RA-3: Reference Architecture Design: Software architects translate the architectural requirements into an architectural design. If in one hand these architects usually have a deep knowledge in software architecture area, on the other hand they lack knowledge in specific domains. Thus, ontologies constitute an important mechanism to be consulted by the architects during the architectural design. In our example, OntoTest played this role.

Based on the architectural requirements, a reference architecture, named RefTEST (Reference Architecture for Testing Tools), was designed. Intending to adequately design and describe RefTEST, we used UML 2.0 techniques and three architectural views: module, runtime, and deployment views. For the sake of space, only the module view that shows the structure of the software in terms of code units is presented in Figure 2. In order to accomplish the requirements in Group 1 (in Table 1), RefTEST is based on SoC (Separation of Concerns) and well-established architectures of interactive systems (MVC and three tier architecture). The package `testingToolCore` contains the core of testing tools, encompassing requirements in Group 2 and exactly corresponding to concepts identified through OntoTest. Other packages (`supportingTools`, `organizationalTools` and `serviceTools`) contain modules that support the testing activity, satisfying requirements in Group 3. We highlight that these packages automate activities considered as crosscutting ones, i.e. activities spread throughout or tangled with others activities, in the same idea adopted by AOSD community [10]. For instance, testing documentation is performed along test suites; hence, it is considered a crosscutting activity. That is indicated using `<<crosscuts>>` stereotypes.

We observed that only traditional architectural views are not sufficient to completely represent reference architectures, since they do not provide elements to explain each concept/term that is present in the architecture. Thus, we have used ontologies to represent an additional view: the conceptual view. Indeed, we used OntoTest as the conceptual view of RefTEST, contributing to a more complete architectural description of RefTEST.

Step RA-4: Reference Architecture Evaluation: We have adopted checklist inspection approach in order to evaluate

Table 1. Architectural Requirements

N.	Requirement	Concept
Group 1: General Requirements		
1	The architecture must support development of both stand-alone and web-based *testing tools*.	Testing tool
2	The architecture must support incremental development of *testing tools*.	Testing tool
...	...	...
Group 2: Requirements Related to Core of Testing Tools		
20	The architecture must support development of *testing tools* that import *test cases*.	Test case
...	...	...
31	The architecture must support development of *testing tools* that generate *test requirements*.	Test requirement
...	...	...
Group 3: Requirements Related to Testing Supporting Modules		
93	The architecture must support development of module to *document testing activity*.	Test documentation
...	...	...

4. Association, composition and aggregation were based on the same understanding adopted by object-orientation software development approach.

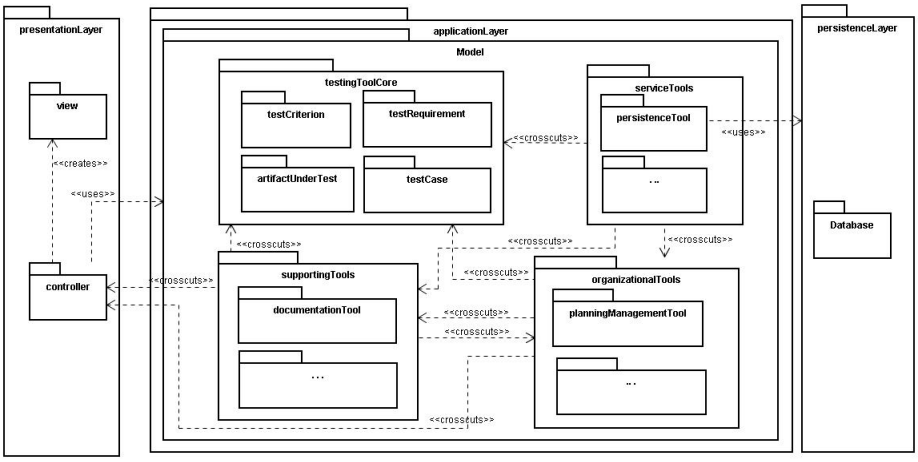

Figure 2. Module View of RefTEST

the quality of reference architectures. A checklist corresponds to a list of questions that guides reviewers on detecting defects in documents and, specifically in our work, defects in documents related to reference architecture. In our example, reviewers were software architects or people from software testing domain. By using this checklist, we have evaluated quality characteristics (maintainability, performance, security, usability, portability and reuse) as well as the architectural description of RefTEST itself. In this step, OntoTest was a relevant mechanism consulted when reviewers had difficulties to understand the reference architecture. OntoTest was even more important when reviewers were software architects that lack knowledge in testing area. For instance, questions contained in the checklist, such as "Was the architecture developed based on terms widely understood and used in the domain?", were better answered before architects have looked up the ontology.

4. Conclusion

The establishment of reference architecture is not a trivial task. We observed that ontologies can affect directly the establishment of reference architectures. In our example, the ontology was essential in all four steps of ProSA-RA, contributing in different perspectives. The more significant impact was observed during the information source investigation (Step RA-1 of ProSA-RA), the first and fundamental one in order to achieve a broad understanding of the domain. Thus, in this paper, we propose the use of ontologies as a central element to the building of reference architectures. The results obtained provide a preliminary evidence on the practical use of ontologies. Thus, we believe that the same results could be achieved for other domains that have mature and complete ontologies. However, more systematic and controlled experiments must be conducted in order to validate our idea and to observe the effectiveness of ProSA-RA in other situations, such as when one or more partial ontologies are available.

Acknowledgment: This work is supported by Brazilian funding agencies (CAPES, FAPESP and CNPq), the INCT-SEC and QualiPSo European Research Projects.

References

[1] P. Kruchten, H. Obbink, and J. Stafford, "The past, present, and future for software architecture," *IEEE Software*, vol. 23, no. 2, pp. 22–30, 2006.

[2] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Addison-Wesley, 2003.

[3] J. Bayer, T. Forster, D. Ganesan, J. F. Girard, I. John, J. Knodel, R. Kolb, and D. Muthig, "Definition of reference architectures based on existing systems," Fraunhofer IESE, Tech. Rep. 034.04/E, 2004.

[4] U. Eklund, Örjan Askerdal, J. Granholm, A. Alminger, and J. Axelsson, "Experience of introducing reference architectures in the development of automotive electronic systems," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–6, 2005.

[5] G. Muller, "A reference architecture primer," [*On-line*], *World Wide Web*, 2008, available: http://www.gaudisite.nl/ (Access: 06/19/2009).

[6] M. Uschold and M. Grüninger, "Ontologies: Principles, methods and applications," *Knowledge Engineering Review*, vol. 11, no. 2, June 1996.

[7] R. A. Falbo, C. S. Menezes, and A. R. Rocha, "Using ontologies to improve knowledge integration in software engineering environments," in *ISAS'98*, Orlando, USA, July 1998, pp. 1–8.

[8] E. F. Barbosa, E. Y. Nakagawa, and J. C. Maldonado, "Towards the establishment of an ontology of software testing," in *SEKE'06*, San Francisco Bay, USA, July 2006.

[9] A. Akerman and J. Tyree, "Using ontology to support development of software architectures," *IBM Systems Journal*, vol. 45, no. 4, pp. 813–825, 2006.

[10] G. Kiczales, J. Irwin, J. Lamping, J. Loingtier, C. Lopes, C. Maeda, and A. Menhdhekar, "Aspect-oriented programming," in *ECOOP'97*, Jyväskylä, Finland, 1997, pp. 220–242.