Knowledge Identification from Requirements Specification

Eduardo Barra(✉) and Jorge Morato

Universidad Carlos III de Madrid,
Avda. Universidad 30, 28911 Leganés, Madrid, Spain
ebarra@kr.inf.uc3m.es, jmorato@inf.uc3m.es

Abstract. One of the main artifacts in Requirements Engineering is the Requirements Specification (RS). Throughout the life cycle of the RS arises the need of extracting knowledge in order to facilitate communication with stakeholders. However, this process is not usually efficient. In the different proposals for the representation of an RS conflicts often arise, due to coupling and redundancy of requirements. The Aspect-Oriented paradigm provides principles to address a multidimensional modeling to avoid these conflicts. Knowledge-Engineering is proposed to provide a model of the knowledge needed to allow an efficient extraction of knowledge from the requirements by ontologies. An experimental study has been developed to assess its efficiency when compared with classical methods.

Keywords: Requirements Engineering · Requirements Specification · Aspect-Oriented · Knowledge-Engineering · Ontology · Knowledge extraction

1 Introduction

The most important artefact to be used to transmit knowledge effectively between the different activities of any process of Requirements Engineering (RE) is the Technical Specification Document (TSD). This document is written in natural language, and is often supplemented with graphical models for its better understanding. The Requirements Specification (RS) is the section of a TSD that contains the knowledge specification of a software product.

The internal organization of an RS is aimed at reducing the complexity of its semantics and improving the global understanding of the requirements for the effective transmission of knowledge. However, there is wide agreement on organizing requirements in a simplistic way. In fact, they are frequently classified in just two types: functional and non-functional. In this convention the group of functional requirements is accepted by most analysts, however, the organizing of non-functional requirements is ambiguous and subjective. Different researchers claim that, in fact, they impose restrictions on the functional requirements [1]. Therefore, they should be viewed as properties of future software products [2] or as requirements that indicate quality [3]. We think that the paradigm Aspect-Oriented (AO) [4] provides the resources to develop these models that allow a multidimensional organization without compromising the integrity of its semantics. Therefore, the representation of an RS with this paradigm

J.-L. Sierra-Rodríguez et al. (Eds.): SLATE 2015, CCIS 563, pp. 264–270, 2015.
DOI: 10.1007/978-3-319-27653-3_26

reduces the conflicts, couplings and redundancy between requirements. This reduction in the complexity of a software product facilitates the extraction of knowledge, its reuse and management.

The discipline Ontology Engineering is often applied to Knowledge Engineering (KE). Ontologies allow to create an explicit and formal specification of knowledge-managed through computers that provide reusability and shareability [5, 6]. Therefore, our hypothesis is that the representation of an RS through ontologies is an important factor to obtain an efficient representation of knowledge. With this background, this research proposes guidelines according to the AO paradigm based on principles of KE applied to Requirements Engineering (RE) for the development of an RS. OE provides a multidimensional semantic model with a well-founded organization that facilitates the extraction of knowledge.

In accordance with these ideas, the rest of the paper is structured as follows. Section 2 briefly discusses related work for the modeling of an RS based on the paradigm AO and OE. In Sect. 3, some guidelines for the modeling of an RS are proposed and exemplified. In Sect. 4, an evaluation is developed to demonstrate that the representation with the guidelines is effective and efficient in the extracting of knowledge from RS. Finally, Sect. 5 presents the conclusions and future work.

2 Related Work

Our study is focused on the early stages of the software development process (SDP). SDP is usually split into different stages. In the AO paradigm, we can talk about early, middle and late aspects, where the early aspects comprise the requirement specification and the architectonic logical representation. Late aspects deal with the low codification level, and finally, middle aspects are located between these stages (Table 1).

Table 1. Stages in different software development proposals

Activity	USDP	MDA	AO
Specification	Requirements	CIM	Early aspects
	Analysis		
Development	Design	PIM	
		PSM	Middle aspects
	Implementation		Late aspects

In recent years, many studies have shown the potential of the AO paradigm focused on the early stages of software development. These proposals are grouped under the term Aspect-Oriented Requirement Engineering (AORE) [7, 8]. The first step in this research has been to study the most representative approaches in AORE, in order to find out well-founded approaches to include in our work.

In 2003, Rashid et al. proposed using XML language to index the description of the requirements for the management of concerns [9]. In another proposal, "Multi-Dimensional Separation of Concerns in Requirements Engineering" [10], the suggested solution for RE is the separation of concerns into multidimensional categories, in order to provide a categorization modeled from different points of view. In Yu and Prado's work is [11] proposed aspect-orientation techniques to manage objectives. It suggests mechanisms to avoid conflicts between requirements. Another proposal, that combines aspect-oriented analysis and design [12], highlights the need to decompose the requirements into more elementary parts. Finally, the research carried out by Jacobson and Ng shows that when the concepts of the AO paradigm are related to "use cases" the positive influence for identifying concerns can be observed [13].

In these proposals, not only did we find different contributions to be considered, but also different problems. These approaches only consider the modelling of requirements in descriptions of natural language, where a requirement may belong to different aspects.

The modelling of requirements in these proposals lacks a clear technique for the modelling of knowledge, generating conflict, coupling and semantic redundancy. AORE proposals are aimed at the identification, separation and composition of concerns but they are not oriented to the representation of knowledge in an efficient way.

Additionally, different ontology-driven approaches to support RE have been analyzed [14–17]. Although modeling the knowledge of the domain through ontologies addresses an obvious need, there is a lack of well-established natural language techniques to represent these internal semantic relationships among requirements. Besides, the need of developing organizational structures to represent the Requirements Specification (RS) is usually overlooked.

3 Guidelines for the Semantic Modeling of an RS

The first stage of the guidelines proposed in this work focused on the importance of a sound and adequately substantiated organization in the development of an RS. In this case, we propose the creation of a structure to specify the requirement. This structure is designed in accordance with different viewpoints. The main goal is to provide a way for the development of an RS that improves the understanding of the requirements as a whole. In this regard, we have proposed the use of an Architecture Viewpoint (AVP) that provides a schema for developing the RS. The AVP is modeled on a domain ontology that provides a guide for modelling responsibilities. The aim is to identify the viewpoints that will group both the dominant concerns and the related viewpoints in crosscutting concerns. The high-level viewpoints proposed in the AVP will act as containers of viewpoints of lower level in a recursive nesting to reach the last level viewpoint.

In the second stage, the early concerns of a software product are modeled according to the AVP base, in an organized knowledge representation. The goal of these guidelines is to model the early knowledge using ontologies, and leaving as a secondary activity elaborating a description in natural language, in order to facilitate the understanding by non-expert stakeholders. The modelling of concerns through the Ontology Engineering has its major support in the Ontology Web Language (OWL). This is a recommendation introduced in 2004 by the W3C for building the Semantic Web, which is the most popular language for the semantic description of ontologies. Accordingly, OWL has been the language selected in these guidelines for the modelling of an RS. The application of the guidelines involves modeling of knowledge of the Requirements, typical of an RS with the specification of the properties between concerns. There are two main types of properties that OWL represents as relationships: "Object Properties" and "Data Properties". The most important relationships in these guidelines are the "Object Properties" to model the semantic relationships. The modelling of early knowledge of a software product assisted by these guidelines involves the use of natural language in order to allow the developer a richer description of the concerns, in the same way that it is done with typical requirements. This information is added to the "Annotation Properties".

An efficient representation of a specification requires split its components in elements like concerns, information entities, roles, conditions, and so on. Usually specifications are expressed in natural language (NL). Unfortunately NL is complex and prone to ambiguities. An example may explain some of these difficulties of modeling processes in an RS. In Fig. 1 the decomposition of concern expressed in a NL is shown. A multidimensional modeling to split its components in a well-grounded way is necessary to avoid non-atomic and overlapped requirements.

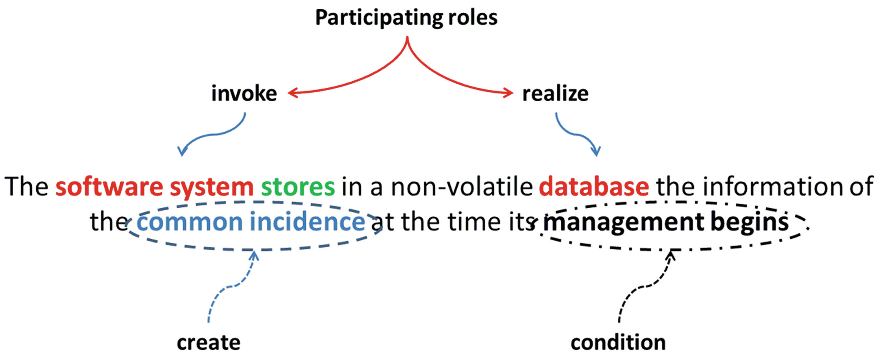
Participating roles
invoke
realize
The software system stores in a non-volatile database the information of the common incidence at the time its management begins.
create
condition

Fig. 1. Constraints related to the concern "Incidence Data Register"

4 Evaluation

In order to show the efficiency of the guidelines for the RS creation, we developed an experiment to compare our guidelines with a classical development. The case study proposed was to develop an application to manage a "stadium for athletics events".

At the step 1 of the experiment, a solution was developed with the method described (hereafter referred to as specification B). In addition, we gave to 32 teams a description of the case study. The project was given to students in Software Engineering in the final year of the degree in computer science. Each team was comprised of 5 members. The teams had to specify the TSD of a software product under a classical development process. Teams were allowed to base their solution on IEEE 830 or ESA PSS-05 standards.

The specifications developed under a classical methodology were analyzed by five researcher teachers, all of them experts in RE. The experts selected the two best ones, which we call specifications A and C for short.

The second step of the experiment involved to extract shared crosscutting-concerns, dominant-concerns and common information entities from the different solutions. Due to its importance in this proposal, the classification of crosscutting concerns is focused on the type of query. The category "information entity" comprises the name of classes and view points, while information related to information entities refers to class properties and its values. Thus, the concerns were identified for the experiment referee on every of the three solutions developed.

At third step, we asked to six software analysis experts from different software factories to manage the three specifications when facing a major update in the software. Therefore they were required to obtain the same knowledge from the three specifications. In order to avoid a biased result due to the learning of the domain, the specifications were given to the expert in a specific order. In so doing, the first specification given is not at a disadvantage in terms of effectiveness of change when comparing with the other specifications. The first specification given was developed with the classical methodology, named as A. When the update was finished, the specification B, made with the guidelines, was given to the experts, and finally the specification C. In the part of the experiment about the extraction of knowledge of the specifications developed under the classical development A and C, every expert analyst received the Technical Specification Document (TSD) with its corresponding digital archive of the RS to be

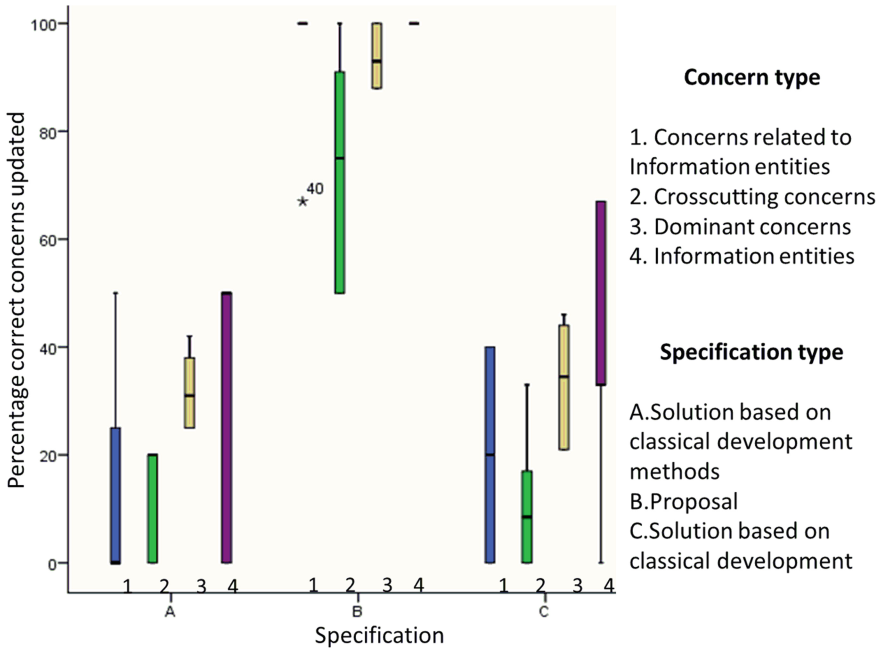

Fig. 2. Percentage of correct concerns identified according its type

used in a management of requirements. In the part of the experiment about the extraction of knowledge of the specifications developed under our guidelines, the six expert analysts received an OWL archive. Tools for ontologies, such as editors, reasoners and a search engine were provided to the experts to carry out the update. The time to work in every specification was limited to 30 min.

At the step 4, results have been compared and analyzed. The evaluation was made quantifying the concerns to update correctly identified by the experts analysts from the total knowledge proposed in each specification.

The result is shown in Fig. 2. We can clearly see that the specification of a software product on the early stages developed with the guidelines allows the knowledge extraction more efficiently.

5 Conclusions and Future Works

The first proposal of the presented approach is to use the AO paradigm concepts to get a multidimensional modeling. This step allows us to avoid conflicts, such as couplings and redundancy in representing RS. These conflicts appear in a recurrent way in modelling RS under classical methodologies. An additional goal is to prevent problems due to ambiguities when representing in natural language RS. In this sense, the KE concepts allow to solve the problem about the knowledge representation in the early stages of a software product.

Several approaches exist for RS modeling on AO and KE. They typically lack knowledge organization and structure. Therefore, they are neither efficient nor effective in defining the knowledge that different stakeholders require.

One of the contributions of our approach is the proposal of guidelines that improve the RE process, creating an RS model that facilitates the knowledge extraction. The guidelines are based on the improvement of the semantic representation of the requirements, with a multidimensional knowledge organization.

In this methodology, regular updates are properly incorporated thanks to the way we split the requirements and user stories. For this reason, the proposal is well-suited to work with agile methods for managing product development.

We developed an experiment with the proposed guidelines to study the effectiveness of the approach. The result of this evaluation proved that the innovative solution proposed by this work improves substantially the knowledge extraction in the early stages avoiding investing many resources trying it.

As a future work we are developing guidelines for the creation of the AVP in a systematic way, which allows us to structure and organize those aspects that support the evolution of a software product. In this regard, the automation of the guidelines will provide the support for the creation of the AVP needed by the software factories, where the resultant AVP could be reused in the same line product. To complement this work, we intend to create a tool to manage concerns. This tool must provide the resources that a typical tool offers to manage requirements, but respecting the concepts of the conceptual AO model in a clear way. That is, support will be given through a virtual guide to describe the requirements, but also modelling the attributes with the necessary concerns that describe their knowledge.

References

1. Sommerville, I.: Software Engineering. Pearson Education, Boston (2005)
2. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley, Reading (1999)
3. Doerr, J., Kerkow, D., Koenig, T., Olsson, T., Suzuki, T.: Non-functional requirements in industry – three case studies adopting and experience-based NFR method. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering, pp. 373–382. IEEE, New York (2005)
4. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C.-V., Irwin, J.: Aspect-oriented programming. In: Akşit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241. Springer, Heidelberg (1997)
5. Berners-Lee, T., Hendller, J., Lassila, O.: The semantic web. Sci. Am. **284**(5), 29–37 (2001)
6. Brewster, C., O'Hara, K.: Knowledge representation with ontologies: the present and future. IEEE Intell. Syst. **19**(1), 72–81 (2004)
7. Grundy, J.: Aspect-oriented requirements engineering for component-based software systems. In: IEEE International Conference on Requirements Engineering, p. 84. IEEE, New York (1999)
8. Rashid, A., Sawyer, P., Moreira, A.M.D., Araújo, J.: Early aspects: a model for aspect-oriented requirements engineering. In: Proceedings International Conference on Requirements Engineering, pp. 199–202. IEEE, New York (2002)
9. Rashid, A., Moreira, A., Araújo, J.: Modularisation and composition of aspectual requirements. In: Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, pp. 11–20. ACM, Boston (2003)
10. Moreira, A., Rashid, A., Araujo, J.: Multi-dimensional separation of concerns in requirements engineering. In: 13th IEEE International Conference on Requirements Engineering, pp. 285–296. IEEE, New York (2005)
11. Yu, Y., do Prado Leite, J.C.S., Mylopoulos, J.: From goals to aspects: discovering aspects from requirements goal models. In: 12th IEEE International Requirements Engineering Conference, pp. 38–47. IEEE, New York (2004)
12. Baniassad, E., Clarke, S.: Theme: an approach for aspect-oriented analysis and design. In: Proceedings of the ICSE 2004, pp. 158–167. IEEE, Washington (2004)
13. Jacobson, I., Ng, P.-W.: Aspect-Oriented Software Development with Use Cases. Addison-Wesley, New Jersey (2005)
14. Kaiya, H., Saeki, M.: Using domain ontology as domain knowledge for requirements elicitation. In: 14th IEEE International Requirements Engineering, pp. 189–198. IEEE, Los Alamitos (2006)
15. Jureta, I.J., Mylopoulos, J., Faulkner, S.: Revisiting the core ontology and problem in requirements engineering. In: 16th IEEE International Requirements Engineering, RE 2008, pp. 71–80. IEEE, Los Alamitos (2008)
16. Velasco, J.L., Valencia-Garcia, R., Fernandez-Breis, J.T., Toval, A.: Modelling reusable security requirements based on an ontology framework. J. Res. Pract. Inf. Technol. **41**(2), 119–133 (2009)
17. Souag, A., Salinesi, C., Wattiau, I., Mouratidis, H.: Using security and domain ontologies for security requirements analysis. In: Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual, pp. 101–107. IEEE, Los Alamitos (2013)