Computers in Industry 65 (2014) 1053–1064

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

An exploratory study on ontology engineering for software architecture documentation

K.A. de Graaf [a,*], P. Liang [a,b], A. Tang [c], W.R. van Hage [d], H. van Vliet [a]

[a] VU University Amsterdam, Amsterdam, The Netherlands
[b] Wuhan University, Wuhan, China
[c] Swinburne University of Technology, Melbourne, Australia
[d] SynerScope B.V., Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 20 June 2013
Received in revised form 11 April 2014
Accepted 16 April 2014
Available online 27 May 2014

Keywords:
Ontology engineering
Software architecture
Software ontology
Ontology-based documentation
Knowledge acquisition
Knowledge management

Abbreviations:
SA, software architecture
AK, architectural knowledge
HTML, hypertext markup language
WYSIWYG, what you see is what you get
GUI, graphical user interface
CF, contextual factor

ABSTRACT

The usefulness of Software Architecture (SA) documentation depends on how well its Architectural Knowledge (AK) can be retrieved by the stakeholders in a software project. Recent findings show that the use of ontology-based SA documentation is promising. However, different roles in software development have different needs for AK, and building an ontology to suit these needs is challenging. In this paper we describe an approach to build an ontology for SA documentation. This approach involves the use of typical questions for eliciting and constructing an ontology. We outline eight contextual factors, which influence the successful construction of an ontology, especially in complex software projects with diverse AK users. We tested our 'typical question' approach in a case study and report how it can be used for acquiring and modeling AK needs.

1. Introduction

For a Software Architecture (SA[1]) to be useful, it needs to be understood [1] and effectively communicated between its designers and its users [2]. Documenting the Architectural Knowledge (AK) in a SA facilitates its communication and understanding. AK can be defined as "*the integrated representation of the software architecture of a software-intensive system (or a family of systems), the architectural design decisions, and the external context/environment*" [3].

It is common for AK to be documented in file-based documents such as text files, diagrams, source code, and meeting notes. This however introduces practical limitations as it is hard to describe AK unambiguously and comprehensively for all of its different uses. Additionally, AK is highly interrelated, making it hard to support the needs of different users by a single document indexing structure.

In order to address these limitations, we study ontology-based SA documentation, in which AK is made explicit and unambiguous by applying a semantic structure. An ontology refers to a formal domain model describing its concepts and relationships among concepts [4]. In [5] de Graaf et al. report on a controlled experiment that was conducted in a large and complex industrial software project. The results provided empirical evidence that ontology-based SA documentation is more effective and efficient for AK retrieval than file-based SA documentation.

* Corresponding author. Tel.: +31 6 18 47 91 32
E-mail addresses: ka.de.graaf@vu.nl (K.A. de Graaf), liangp@cs.vu.nl (P. Liang), atang@swin.edu.au (A. Tang), willem.van.hage@synerscope.com (W.R. van Hage), hans@cs.vu.nl (H. van Vliet).

[1] See 'Abbreviations' under Article Info for a list of abbreviations used in this paper.

http://dx.doi.org/10.1016/j.compind.2014.04.006
0166-3615/

The authors of this paper had to implement ontology-based SA documentation in the software project in which this industrial experiment was executed. To do so, we had to consider what ontology engineering approach would be suitable. This paper is about building the ontology structure for ontology-based SA documentation. It is not about instantiating (or 'populating') this ontology or about the experiment in which the ontology was populated, as described in [5].

Developing an ontology for a software project in industry should be economically feasible, i.e. it should be efficient and accurate, for organization and individual users [6]. If the ontology is inaccurate, documentation users will not retrieve or understand the AK that they need. Users would lose interest and confidence in ontology-based SA documentation, and revert to other means to get the required AK.

In the context of large software projects it can take much time and effort to develop an accurate ontology. Knowledge acquisition from many diverse stakeholders, each having their own needs and views [1] of AK, is required to build an accurate ontology. These users of AK are generally pressed for time and their primary interest is seldom about making documentation. Moreover, the AK needed by users in large software projects is often domain specific and complex.

Developing the ontology is not a one-time effort because the AK needs of SA documentation users shift over time. For example, during development users will be interested in AK that relates requirements to the components implementing those requirements, while during integration testing (other) users might be interested in relations between software releases and requirements coverage. Shifting AK needs necessitates a regular evaluation and revision of the constructed ontology.

In this paper we describe eight contextual factors in software projects. These contextual factors influence ontology engineering for SA documentation, especially in large and complex projects. We devised a 'typical question' approach for ontology construction that takes the contextual factors in large and complex software projects into account. In this approach typical questions about AK are acquired from SA documentation users and used to build an ontology. These typical questions are frequently asked by AK users[2] during their everyday tasks, i.e. questions that represent their everyday AK needs.

We applied the 'typical question' approach in an exploratory industrial case study which provided several insights. In the case study we explored how well our 'typical question' approach was applied to construct a useful ontology by acquiring and modeling AK needs of many diverse users that use SA documentation in different projects and product lines. The 'typical question' approach can continuously refine the AK ontology when AK needs evolve.

This paper is motivated by the lack of applied ontology engineering approaches for constructing an ontology for SA documentation. We make the following contributions:

- Illustrate a 'typical question' approach for constructing the ontology used in ontology-based SA documentation.
- Outline important contextual factors that influence ontology engineering for SA documentation in software projects.
- Demonstrate how the 'typical question' approach can be applied through an exploratory case study in an industry software project.

This paper is organized as follows. Background of ontology-based SA documentation, ontology engineering, knowledge acquisition, and Grounded Theory is given in Section 2. Section 3 describes our 'typical question' approach for ontology construction. Section 4 describes the contextual factors that influence ontology engineering for SA documentation in software projects. Section 5 reports the exploratory case study and the lessons learned from it. Section 6 presents related work and Section 7 concludes this paper.

[2] We consider AK users the same as SA documentation users in this work.

2. Background

2.1. Ontology-based SA Documentation

Many relationships exist between AK in SA documentation, e.g., between requirements, decisions, and components. Consider a single decision recorded in a document. A software engineer needs to know how the decision impacts components and interfaces s/he is working on. When evaluating the decision a software architect is interested in its rationale, alternatives, related decisions, and related requirements. A quality assurance manager might need to know all quality attributes that the decision impacts or vice versa.

The linear organizational nature of file-based documentation makes it hard to provide a structuring of AK that is suitable for every user. This structuring can be done using views [1], perspectives, or some other sectioning in a Table of Contents. However, once written down, the structuring of AK becomes static and linear in file-based documentation causing difficulties for users that want to retrieve AK that is unsupported by the structure. Furthermore, it can be troublesome to describe AK unambiguously in documentation, i.e. clearly, consistently, with explicit notations for AK and AK interrelations, especially when documentation and AK evolves.

On the other hand, an ontology provides a type of AK structuring which facilitates AK retrieval by different types of documentation users. Knowledge in an ontology can be searched by concepts and reasoned with. The type (or 'class') of AK becomes explicit in an ontology and the relationships between AK have explicit semantics, e.g., '*realized by*' and '*results in*'. This improves the efficiency and effectiveness of AK retrieval [5].

Users of ontology-based SA documentation can retrieve AK using a semantic wiki [7] or plug-ins in text-editors [8]. Semantic wikis allow for web-based visualization and management of (ontology and its instances in) knowledge bases and semantic-enhanced search facilities such as graph-like exploration, faceting, and filtering of knowledge instances based on semantic interrelations.

An advantage of ontology-based documentation in a semantic wiki over traditional wiki and hypertext systems is the use of semantic relationships. Hyperlinks provide pointers to information but the pointers do not specify the meaning of relationships. Whereas semantic relationships can be specified in ontology-based documentation, each with an explicit name, type, and meaning. Moreover, one can assign formal properties to semantic relationships, e.g., transitivity and symmetry, which allow for automatic inference and reasoning.

The ontology-based SA documentation that was used in the experiment in [5] consists of a semantic wiki (see http://archimind.nl/archimind/) in which fragments, sections, and diagrams from file-based SA documentation are stored as HTML in wikipages. Basic version control, a WYSIWYG editor to update wikipage content, and change history, support maintenance activities. Using semantic annotation, the AK in wikipage content (e.g., a description of '*GUI*') is highlighted and instantiated as belonging to a class of AK (e.g., component) and having relationships (e.g., 'satisfies') to other AK (e.g., requirement '*login*').

When an AK user views an AK instance, e.g., *GUI*, s/he can clearly see that *GUI* is a component, with relationship '*satisfies*' to requirement *login*, and what SA documentation text (in wikipages)