International Journal of Information Management 34 (2014) 704–710

Contents lists available at ScienceDirect

International Journal of Information Management

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

Ontology for knowledge management in software maintenance

Edgar Serna M.[a,*], Alexei Serna A.[b]

[a] Faculty of Engineering, Instituto Tecnológico Metropolitano, ITM, Medellín, Colombia
[b] Instituto Antioqueño de Investigación, IAI, Medellín, Colombia

ARTICLE INFO

Article history:
Available online 25 July 2014

Keywords:
Knowledge management
Ontology
Software maintenance

ABSTRACT

The concept of ontologies has been widely reported in the literature, and has been described various ontological designs, but few works try to explain in practical terms how to design an ontology. On the other side, although it is widely accepted the importance of maintenance of software products, few ontological designs focus on applying techniques to manage knowledge to get it. This paper analyzes the ontologies proposed to address this need, in order to present information that can help to software development organizations in similar works. Besides, is described a methodological process to structure an ontology that can be applied to manage knowledge in software maintenance.

1. Introduction

In order to perform efficient work in Software Engineering models and apply them to each of its phases, the required knowledge is diverse, large-scale and constantly increasing. Research in Software Engineering is oriented towards knowledge management, trying to make the best decisions and providing companies the information they need for the developing these phases (Lindvallv, 2002).

Software maintenance phase is an activity in which knowledge plays an important role; knowledge level of those in charge of realizing that is complex, voluminous and intensive in areas such as: the domain of the program, the company which uses it, the past and present practices of Software Engineering, the several programming languages, the programming methodology used, the relations between the different modules, the necessary tools, among others (Pigoski, 1996). Often, the information needed for developing this role is not found or is very difficult to locate and rebuild; therefore, in order to do their work, managers must query the limited documentation available or consult through colleagues, which causes that part of the existing knowledge in maintenance groups is lost or not used (Walz, Elam, & Curtis, 1993).

Knowledge management provides techniques and methods that help to reduce the loss or waste of knowledge, and allows that software maintainers can share knowledge (Rodríguez, Martínez, Favela, Vizcaíno, & Piattini, 2004); for organizations of software development and maintenance it assures them benefits like quality improvement of their products and processes, and reduction of costs and errors (Dingsøyr & Conradi, 2002). However, before starting development processes of knowledge management systems, is important to identify the knowledge that is to be managed, the place where is to be stored and where it is required. Moreover, because the organization generally does not know how to locate the person or does not know the person having the knowledge needed in order to solve the problem, this also becomes work to be performed (Nebus, 2001).

In order to help managing this knowledge, researchers have been working in the conceptualization of ontologies as domain models, which currently emerge as one of the more appropriate management tools for supporting knowledge representation, processing, storage and retrieval, and this tool begins to be extended to all the phases of Software Engineering. In relation to maintenance, and considering the types of knowledge involved in ontologies supporting it, the ontological developments nourish from diverse experiences and contributions, but few achieve modeling and implementing a representative ontology of this area.

This article is a review paper on the subject and is structured as follows: (Section 2) definition of software maintenance; (Section 3) definition of ontologies; (Section 4) analysis of some ontologies around software maintenance; (Section 5) description of a methodological proposal to create ontologies enabling management of software maintenance; and (Section 6) conclusions and future work which ensure continuing this research.

2. Software maintenance

Software maintenance is defined as "any modification of a software product, after its delivery, to correct errors, improve

* Corresponding author. Tel.: +574 312 7294593.
E-mail addresses: edgarserna@itm.edu.co (E. Serna M.), alexei.serna@fundacioniai.org (A. Serna A.).

http://dx.doi.org/10.1016/j.ijinfomgt.2014.06.005
0268-4012/

performance or other attributes, or the action of adapt the product to a changing environment" (Mamone, 1994); "Changes in product management software to keep them updated and fully functioning" (Singh, 1994). In the last decade software development techniques, progressed considerably and new processes, new languages and new tools were proposed and tested; on the contrary, software maintenance apparently fell behind. This subject is very relevant and important in Software Engineering, and it has been relatively marginalized by technical literature (Pressmann, 2014). Systematizing maintenance is difficult, mainly because it is a reactive activity and therefore very chaotic to perform.

While software development projects can take months or years, the maintenance phase usually lasts for many years. Estimating resources is a key element of maintenance planning, and these resources should be included in the budget of project planning. The planning of software maintenance should begin with the decision to develop a new system, and should consider the quality objectives recommended by IEEE (1998). This phase of software life cycle is the result of the need of adapting the system to an ever-changing environment in which, in most cases, delays cannot be avoided. Companies must keep pace of these changes, and this often means they must modify the software which supports their activities. A feasible solution to break this vicious cycle is the development of knowledge management systems for software maintenance and avoiding a common practice for most companies: try to document again their software systems; this is an expensive task, away from the benefit of installed programs for focusing on documentation (Silva & Dasilva, 2012).

3. Ontologies

Recently, the term "ontology" in the field of ICT raises great interest, especially after the World Wide Web Consortium – W3C – considered it as the technology required to facilitate knowledge infrastructure to the emerging Semantic Web or Web 3.0 (Horrocks, 2008; Lefort, Taylor, & Ratcliffe, 2006). It is a technology which aims to clarify, specify and reach agreement about the knowledge related to a particular domain, paradoxically it does not reach a consensus clearly defining what it is or what it should be (Evans, 2008). Lacy (2005) states that the term ontology is overloaded, while its meaning refers to different things depending on who defines it. Traditional axioms allows establishing a difference with the philosophical concept, finding that in philosophy "Ontology" is a field that studies the nature and organization of reality or existence, directly related to epistemology; while one of the most cited definitions in the field of knowledge engineering defines ontology as *an explicit specification of a conceptualization* (Gruber, 1995). An ontology is the conceptual and terminological description of a shared knowledge about a specific domain. Leaving aside the formalization and interoperability of applications, this is just the main competence of the term: improving communication using the same system as terminological and conceptual (Reuver & Haaker, 2009). From these concepts is important to consider that, contrary to the philosophical one, ontology should not be considered as a natural entity that is discovered, instead of that it should be considered as an artificial resource that is created for a specific purpose and for a particular application (Mahesh, 1996). Accepting this is to accept that, in most cases, the application determines the categorization performed.

Taxonomies are designed from a point of view: the explicit criterion of a resulting hierarchy revealed in the other relationships, which define and add expressiveness to the whole set. Because of this, designing an ontology involves making choices and selecting criteria; and since its purpose is acting as a reference for people, applications and organizations, those choices and categorizations must agree and must be acknowledged (Oliveira, Anquetil, de Sousa, & Batista, 2003a). In the case of software maintenance projects is very useful to have defined ontologies on the subject of managing projects, because these misunderstandings are solved and discussions derived from the understanding of "maintenance request" concept are avoided (Ruiz, Vizcaíno, Piattini, & García, 2004).

4. Ontologies around software maintenance

It is important that before structuring a knowledge management system for software maintenance be considered to model, structure and generalize the information that is generated and consulted in such process. In order to efficiently achieve this activity are used ontologies with which, according to Gruber (1993), an explicit specification of a conceptualization is done. The technology of ontologies can be used for sharing organizational knowledge, and to promote the feature of interoperability between systems (Horrocks, 2008). Because it is a branch of knowledge in permanent construction and development, different authors propose their ontologies regarding software maintenance. We describe below some of the most representative works.

4.1. The informal ontology for software maintenance (Kitchenham et al., 1999)

Describing the most important aspects to be considered in empirical studies of software maintenance; although the proposal has a particular objective in identifying the domain factors that influence the results of this study, it proposes other influencing this phase and which serve to modeling an ontology maintenance. The proposal is structured in four sub-ontologies: (1) of products, in which are defined software products that will be maintained, as well as their internal structure, composition and existing versions; (2) of activities, in which are defined the activities and resources, two types of basic elements in the management of a maintenance project –– *types of activity:* an abstraction of "how the work is done"; *types of resources*: the most important is the hardware and software, but also consider other such as local and consumables –; (3) of organizational processes, which defines how to perform the activities and how to organize the maintenance process; and (4) of agents, which includes the hierarchy of types of existing agents in the management of maintenance projects.

Although it is a proposal that serves as a reference for other ontologies, some of which are discussed in this work, this proposal neither becomes a model addressing an application of software maintenance ontology nor takes into account the concept of knowledge management within a context in which could be possible that testers efficiently take advantage of it.

4.2. The concept-oriented approach (Deridder, 2002)

This ontology is based on the abstract definition of Gruber (1993), in which an ontology is an explicit specification of a conceptualization, and reinforces the thesis, which states that it represents a certain point of view about an application domain, in which the concepts that live in it explicitly and unambiguously must be defined. This ontology complements the definition of Gruber as considers the role of concepts of explicit specification. An ontology must be a reference work, therefore it must exist a strict ontological commitment for the users could access the meanings of concepts. The objective of the proposal is considering a light approach in which is desirable, but not necessary, achieving a number of concepts that could be used as standard for a larger community, and the objective is not creating formal and rigorous ontologies to be