An Ontology of Software Evolution

Author	**Xiaowei Wang**
Supervisors	John Mylopoulos, Nicola Guarino
Studies/Stage	1st year PhD student
Affiliation	University of Trento
E-Mail	xwang@disi.unitn.it

Aims and Objectives

This paper aims at clarifying and formalizing the concepts relating to software evolution, creating shared knowledge of software evolution. To achieve this goal, we propose an ontology of software evolution, for the first time in the research community, based on the analysis of the previous work. With this ontology, we hope to make the implicit knowledge explicit to people, providing guidance for the work of researchers, practitioners and managers.

Problem Statement

In modern society, people heavily rely on software in almost every aspect of human life. Because of the high speed of change in the society, software is forced to change rapidly to adapt to the new environment. To survive in this environment, software systems have to evolve, and consequently the work of maintenance and refactoring of the software systems occupies a large amount of the human and financial resource. Although several taxonomies were proposed, software engineering still depends on stakeholders' intuition and experience, and the concepts used in this domain are still ambiguous. Trying to bridge this gap, we aim to build such a new ontology of software evolution, based on which a lot of terms misinterpreted in the community could be clarified and unified.

Research Questions

To build an ontology of software evolution, initially we have to define several core concepts showed as the following list:

1. the concept of artificial
2. the concept of software as sub-concept of artificial
3. the concepts of species and individual
4. the concepts of software species and software individual
5. the concept of software evolution happening in the software species level
6. the concept of software maintenance happening in the software individual level
7. other related concepts (e.g. software adaptation, software refactoring) happening in software species or individual level

Theory Fundamentals

We take DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) as our starting point [1], which is a fundamental ontology describing basic concepts and knowledge. Besides this, we borrow the ideas about species from Mahner's work [2], and the ideas about artifactual system from Guarino's work [3]. Furthermore, our work will be based on analyzing and reusing history taxonomies and ontologies in this domain. Finally, to develop a formal ontology, the OntoClean methodology may be used as guidance [4].

Preliminary Research Results

Intuitively, software may be interpreted as a tool. Comparing with a hammer, software is only a different tool processing different functions. To discuss about the evolution of software, we have to study the whole life cycle of software covering five main activities in software engineering, and these activities are communication, planning, modeling, construction and deployment. From this perspective, this paper redefines the concept of software by reusing and refining these five activities into five components showed in Table 1.

Abbreviation	**Software component**
D	Domain knowledge
R	Requirement
S	Specification
Des	Design
I	Implementation

Table 1. Software components

According to the view of requirement engineering, requirements are some desired properties that a user may want, and specifications are the characteristics of a machine, together with the domain knowledge, which could fulfill the requirements. This relationship may be represented in a formula as “$D, S \vdash R$” [5]. However, it is missing the developing dimension without which we will lose the capability of describing the activities relating to the developing process (e.g. maintenance, refactoring and other activities). To compensate this missing, the formula may be enlarged into:

$$(D, S \vdash R) \wedge (Des \vdash S) \wedge (I \vdash Des)$$

In the sub-formula “$(Des \vdash S)$”, “Des” means the technical design which can realize the functions defined by the specifications; on the other hand, in the sub-formula “$(I \vdash Des)$”, “I” means the implementation with source codes which can realize the technical design.

In this paper, we insist that software evolution only happens in class level, hence it could be described by the differences between the new and old version of the software. We take a version as a species which defines the laws for its members. These laws are represented in specifications in requirement engineering, hence when we talk about software evolution, we are really talking about the change in specifications. According to this view, software evolution can be represented as S is changed into S’ according to the formula we have already defined.

The changes in specifications may be caused by changes in domain assumption, or by changes in requirement. According to the changes in specifications, new design and source code may be developed in the next iteration in software developing, and the new formula may be rewritten into “$(D', S' \vdash R') \wedge (Des' \vdash S') \wedge (I' \vdash Des')$”.

References

1. Masolo, C., et al., *WonderWeb Deliverable D17: The WonderWeb Library of Foundational Ontologies (preliminary report)*, 2003, Laboratory For Applied Ontology - ISTC-CNR Via Solteri, 38 38100 Trento Italy. p. 38-38.
2. Mahner, M., *What is a species?* Journal for General Philosophy of Science, 1993. **24**(1): p. 103-126.
3. Vieu, L., S. Borgo, and C. Masolo, *Artefacts and Roles: Modelling Strategies in a Multiplicative Ontology*, in *Proceedings of the 2008 conference on Formal Ontology in Information Systems: Proceedings of the Fifth International Conference (FOIS 2008)*2008, IOS Press. p. 121-134.
4. Guarino, N. and C. Welty, *Evaluating ontological decisions with ontoclean.* Communications of the Acm, 2002. **45**(2): p. 61-65.
5. Zave, P. and M. Jackson, *Four dark corners of requirements engineering.* ACM Trans. Softw. Eng. Methodol., 1997. **6**(1): p. 1-30.