Towards a Framework for Requirement Change Management in HealthCare Software Applications

Arash Shaban-Nejad
Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd. W,
Montreal, QC H3G1M8, Canada
+1 (514) 848-2424 ext. 7122
arash_sh@cs.concordia.ca

Volker Haarslev
Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd. W,
Montreal, QC H3G1M8, Canada
+1 (514) 848-2424 ext. 3049
haarslev@cs.concordia.ca

Abstract

Requirements volatility is an issue in software development life cycle which often originated from our incomplete knowledge about the domain of interest. In this paper, we propose an agent-based approach to manage evolving requirements in biomedical software applications using an integrated ontology-driven framework.

Categories and Subject Descriptors K.6.3
[**Management of Computing and Information Systems**]: Software Management – *Software Maintenance.*

General Terms Management, Design, Standardization.

Keywords Software Requirement; Change management; Ontology; Agent

1. Introduction

With advances in health science many features and functionalities required to be added/removed to/from existing software applications in biomedical domain. Requirements change frequently in a software implementation life cycle. This typically forces the requirements of application frameworks to be altered, as well. This include modification and adaptation of both functional and non-functional requirements. A deep and common understanding of the requirement framework is essential for application developers to maintain the changes successfully. In order to provide a common understanding for the framework components, ontologies can be used as the conceptual backbone of a software application. Ontologies can describe software architectures and requirements, which are difficult to model with object-oriented languages [1]. In this paper we propose our approach based on RLR framework for employing agents in an integrated ontology-driven application in order to capture the pattern of changes and validate the results.

2. The RLR Framework

We propose our framework called RLR to Represent, Legitimate and Reproduce the changes and their effects (Fig. 1). This framework helps to capture, track, represent

and manage the changes in a formal and consistent way which enables the system to create reproducible results. In this framework all changes in requirements should be represented in either formal or diagrammatical ways. The change then should be legitimated and validated logically and it should be approved publicly and by experts. In order to reproduce the results of changes and automating the change management process, agents can be used to learn the pattern of changes and their consequences.
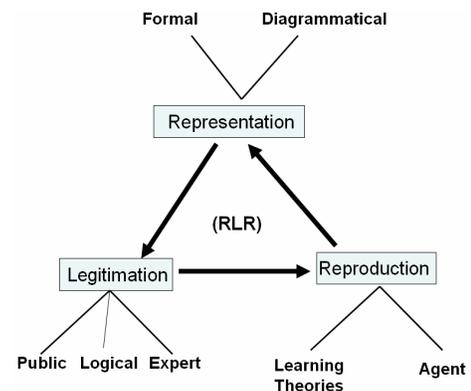

Figure 1. The RLR framework.

2.1 Representation

This phase is responsible for updating the representations of the new knowledge in a consistent manner. Many of the problems in software evolution are basically problems about the nature of change and its representation. For formal representation of changes we use Description Logics and for diagrammatical representation we employ a method based on discrete state model and category theory [2].

2.2 Legitimation

Legitimation in our context is defined as checking the legitimacy and consistency of a change in the domain of interest. This phase assesses the impact of a potential requirement change before the change is actually made. A change should be studied based on its consistency with the whole design by experts and logical reasoners in various

degrees of granularities. Then the final approval is needed by end-users. Logical legitimation can be obtained by a reasoning agent.

2.3 Reproduction

One of the problems in current change management methodologies is too much reliance on human factor. Despite the advantages of human intervention in the process of software maintenance which can increase the overall rationality of the system, it does not guarantee reproducibility of the result of a change. In order to overcome this issue we propose using intelligent agents which enable to discover patterns for different changes and their consequences.

3. Agents and Pattern of Change

Intelligent agents have the ability to identify, search, and collect desired information about various actions from multiple resources under changing conditions [3]. Agents are able to work rationally in order to capture changes in dynamic and heterogeneous environments and respond properly to those changes [4]. In our approach we use three types of agent: Learner agent, Reasoning agent and Negotiation agent. Fig. 2 demonstrates the interactions between these agents.

3.1 Learner Agent

As an application is used and evolved over time, the change logs can accumulate invaluable data and information about various types of changes. A learner agent can use these historical records of changes that occur over and over in a change process to create a pattern of change. It means after several requirement changes (that may happen in various releases), it would be feasible to predict the change direction and change rate for a system with a degree of uncertainty. The learning agent which starts with limited knowledge of the domain - and tries to improve itself - relies on adaptive learning based on semantics provided by the ontological backbone. The learner agent plays an important role in reproduction phase, where we look for patterns for automating the process of change management.

3.2 Reasoning Agent

A reasoning agent is a software agent that controls and checks the logical validity of a system and reveals inconsistencies, hidden dependencies, redundancies and misclassifications. It then automatically gives notice to users or other agents when new information about the system becomes available. We use RACER [5] as a description logic reasoaner agent in the RLR framework.

3.3 Negotiation Agent

In RLR framework the negotiation agent acts as a mediator which allows ontology engineer and other autonomous agents negotiate about the proper implementation of a specific change. A human expert may then browse the results and proposed actions after the negotiation process and decide to confirm, delete, or modify the proposals in accordance to the intention of the application.
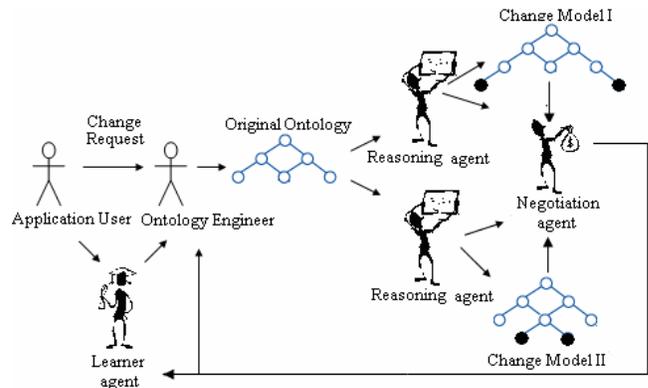

Figure 2. The change management process using agents.

4. Research Direction

We are currently working on implementation of an autonomous system based on the proposed framework to manage the evolving structure of the FungalWeb Ontology [6] and its related artifacts.

References

[1] Deridder, D. and D'Hondt, T. A Concept-centric approach to software evolution. *In Proceedings of the OOPSALA '04 workshop.* (Vancouver, Canada, 2004).

[2] Shaban-Nejad, A. and Haarslev, V. Managing conceptual revisions in a temporal fungi taxonomy. *In proceedings of the 20th IEEE Intl. symposium on computer-based medical systems (CBMS'07)* (Maribor, Slovenia, June 20-22, 2007). 624-632.

[3] Devedžic, V. Knowledge modeling - State of the art. *Integrated Computer-Aided Engineering*. 8, 3 (November 2001), 257-281.

[4] Li, L., Wu, B. and Yang, Y. Agent-based approach for dynamic ontology management. *In Proceedings of the 9th Itl.l Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'05)* (Melbourne, Australia, September 14-16, 2005). 1-7.

[5] Haarslev, V., Möller, R. RACER system description. *In Proc. of the first intl. joint conference on Automated Reasoning (IJCAR'01)* (June 18-23, 2001). 701-706.

[6] Baker, C.J.O., Shaban-Nejad, A., Su, X., Haarslev, V. and Butler G. Semantic web infrastructure for fungal enzyme biotechnologists. *Journal of Web Semantic. 4, 3* (September 2006), 168-180.