Use and Design of Ontology-based Multi-agent System for Multi-site Software Development Environment

Udsanee Pakdeetrakulwonga, Pornpit Wongthongthama
School of Information Systems, Curtin Business School, Curtin University, Perth, Australia
*Corresponding Author: udsanee.pakdeetr@student.curtin.edu.au

ABSTRACT
Large software development projects involve several participants who are distributed geographically without face-to-face communication. To maintain collaborative work through effective communication and coordination, it is necessary to have a common understanding of terminology and methodology to clarify software engineering concepts and enable knowledge exchange and reuse. We consider an ontology designed for distributed software development can be a solution to improve information and knowledge sharing in such scenarios. However, software team members may not be familiar with the use of the ontology by themselves, an active support is needed to proactively deliver knowledge and project information that is semantically defined in the ontology based on appropriate context. In this paper, we propose an ontology-based multi-agent system conceptual framework that can provide intelligent assistance to access and recommend knowledge and project information during multi-site software development. We use UML 2.1 to model interaction among software agents. This framework is expected to improve the effectiveness of the communication and coordination of software teams to reduce the unsuccessful rate of the software development project.

Keyword: Ontology, Multi-site software development, Software engineering, Multi-agent systems

1. Introduction
Due to the globalization of software development and for a number of business reasons, software companies have adopted the global software development approach that enables project team members to work across multiple sites. While on the one hand, a globally dispersed project offers several advantages, on the other hand, it creates additional challenges in regard to communication, coordination and information sharing. In the literature, there are a number of researches that develop ontologies designed for distributed software development environment in order to address such challenges [1]. The Software Engineering Ontology [2] is one of them
that has been developed to provide a common understanding of software engineering domain knowledge and share software project information among dispersed team members.

In this paper, we go step forward by proposing ActiveSEOnt, an intelligent ontology-based multi-agent system conceptual framework that can provide active assistance to access and recommend knowledge and project information to multi-site software development teams who are working in a global context. This framework is expected to improve the effectiveness of the communication and coordination among them to reduce the unsuccessful rate of the software development project.

This paper is structured in the following manner. In Section 2, we provide background and related works. In Section 3, we discuss about the integration of MAS and ontology to support multi-site software development environment. In Section 4, we propose our conceptual framework and the design of interaction among software agents. Prototype implementation is discussed in Section 5 and finally, the paper is concluded in Section 6.

2. Background and Related Work

An agent is a computer program that has relatively complete functionality and cooperates with others to meet its designed objectives [3]. The other characteristic of an agent is its capability of flexible and autonomous action in the environment where it is situated. An agent is also active, task-oriented and is capable of decision-making [4]. Multi-agent systems (MAS) consist of multiple agents communicating and collaborating with each other in one system in order to achieve goals [4]. Paydar and Kahani [5] introduced a multi-agent framework for automated testing of web-based applications. The framework is designed to facilitate the automated execution of different types of tests and different information sources. Ontology-based computational intelligent multi-agent for Capability Maturity Model Integration (CMMI) assessment has been proposed by Lee and Wang [6]. The multi-agent system consists of three main agents interacting with one another to achieve the goal of effectively summarizing the evaluation reports of the software engineering process in regard to CMMI assessment. The CMMI ontology is developed to represent the CMMI domain knowledge. This research does not cover other knowledge areas of the software engineering domain but it specifically focuses on the software engineering process with respect to CMMI assessment only. In addition, the integration of two promising technologies in software engineering which are multi-agent system and Software Product Lines (SPL) is addressed in [7]. It provides the solution of producing higher quality software, lower development costs
and less time-to-market by taking advantage of agent technologies. The ontology is used for modeling the Multi-agent System Product Lines (MAS-PLs) and is represented by UML class diagrams. Monte-Alto, et al. [8] propose a multi-agent context processing mechanism called ContextP-GSD (Context Processing on Global Software Development) that utilizes contextual information to assist user’s task during the software development project. This project applies agent-based technology to process contextual information and support human resource allocation. OntoDiSen is an application ontology exploited in this system representing GSD contextual information. Although this research aims at facilitating the collaboration and coordination in global software development environment and uses ontology to define semantic information which is quite similar to our proposed work, it focuses only on contextual software engineering information, not the whole software engineering domain knowledge. GENESIS [9] is an open source distributed platform supporting cooperative software development. Software agents are utilized to control and monitor activity execution at various sites. In addition, they are used to perform some specific tasks such as metric collection, artefact retrieval, process enactment monitoring as well as supporting the communication among dispersed software development teams.

3. Integrating MAS and Ontology to Support Multi-site Software Development Environment

As stated earlier in Section 2, by considering software agents’ properties, they are expected to be fit to manage working processes and provide effective communication and coordination in multi-site software development environment as followings.

- With the social ability, multiple agents can communicate and collaborative with each other in order to achieve goals. Working in the distributed environment, software team members do not reside at the same place, thus it is hard to have face-to-face communication. Software agents are designed to work proficiently in distributed environment; therefore, they would be able to provide effective communication and coordination among dispersed team.

- Agents can work in flexible manner to provide software engineering domain knowledge and useful project information semantically defined in the Software Engineering Ontology. They can reactively provide knowledge based on user request. Furthermore, they proactively realize what user may expect or what should provide to the user.

The Software Engineering Ontology conceptually organizes and semantically interlinks software engineering domain knowledge and project-related information as
well as support reasoning mechanism. The multi-agent systems can make use of this ontology as a knowledge base to enable the conformity of message exchange and knowledge sharing. This will facilitate the coherent communication among different agents.


We have proposed ActiveSeOnt, an ontology-based multi-agent system for multi-site software development environment to provide an intelligent support to access and recommend knowledge and project information captured in the Software Engineering Ontology. The ontology will work as a knowledge base to support semantically consistent communication and facilitate reasoning mechanism. This research aims at providing the most relevant and precise situational knowledge to support distributed software team in every development phase. The overall architecture of the framework is illustrated in Figure 1.

![ActiveSeOnt conceptual framework](image)

The proposed ontology-based multi-agent system architecture comprises four agents according to their different functions which are:

A. User agents
   - Act as representatives of each user.
   - Build and maintain user profiles.
• Manage semantic annotation service.
• Communicate with recommender and ontology agents.

B. Recommender agent
• Recommend tentative solutions.
• Work with ontology agent to make a decision based on knowledge in the Software Engineering Ontology.
• Notify affected agents in case of ontology update.
• Coordinate with evolution agent in case of unresolved issues/queries.

C. Ontology agents
• Manage and maintain software engineering ontology repository.
• Retrieve information from the ontology to other agents.
• Work with user agents for annotation service.
• Manage ontology population process.
• Notify ontology update to recommender agent.

D. Evolution agent
• Receive update request regarding unresolved issues/queries in existing Software Engineering Ontology and coordinate with the Software Engineering Social Network system (SESN) for the ontology evolution process. More information about SESN can be found in [10] and [11].
• Coordinate with ontology agents for domain knowledge update from SESN process

We use UML 2.1 to model our system because it allows representing more complex scenarios and providing greater details into the modeling process enabling effective capture and representation of MAS actions and interactions [12]. In order to achieve goal, multiple agents interact with each other as per their need. The interaction among software agents can be represented with the sequence diagram as illustrated in Figure 2. There are four types of agents, namely user agent, ontology agent, recommender agent, and evolution agent. A user agent is initialized when each user logs in to the system. It will check with the user profile in order to manage the access level allocation according to user’s role in the software project such as project manager, team leader, requirement engineer, analyst, developer and tester. This process is to make sure that each user has access level to query knowledge or perform operation within his/her responsibility. In case of simple query, the ontology agent will retrieve knowledge from the Software Engineering Ontology and send the result
back to user agent. If the result needs some further process, the ontology agent will pass the retrieved result to the recommender agent. For instance, in case of reusing source code, the recommender agent will rank and order the retrieved candidates according to the number and type of their output and input parameter types to generate an order list of recommended source codes. This will help user to save his/her time in navigating through many retrieved result. In the event that there is an instance update request, the ontology will perform updating and request the recommender agent to send notification to relevant user agents that will have the effect of the update on ontology instance. Regarding software engineering domain knowledge manipulation, it will be done through the evolution agent. The evolution agent will retrieve domain knowledge update information from the repository and request the ontology agent to manipulate the SE ontology domain knowledge. The ontology agent will then inform the recommender agent to send notification to all affected user agents.

Figure 2. Sequence diagram
5. Prototype Implementation

The prototype is used as a proof of concept of this proposed framework. It is currently being implemented in Java and will subsequently be tested. We choose JADE (Java Agent Development Environment) which is considered as an agent middleware that implements an agent platform and provides a development framework. JADE framework is developed in Java and completely based on the FIPA specifications. It also provides a proper set of functionalities for development of MAS without imposing any specific agent architecture [13]. JADE provides several tools, namely FIPA-compliant agent platform, distributed agent platform, multi-threaded execution environment, library of interaction protocol, and GUI administrator to developers who can simply use and write Java code without the need to learn any new special construct [14].

Each agent will collaborate with other agents, so they must be able to communicate and understand messages from one another. Agent Communication Language (ACL) defined by FIPA can be used as the language of communication between agents and the ontology can be used to facilitate the semantic interoperability. The Software Engineering Ontology created from Protégé will be translated automatically in FIPA/JADE compliant ontologies by utilizing a tool called the Java Ontology Bean Generator. This will help to reduce a labor-intensive task of manually translating the ontology from the editor specification into an agent-understandable format.

6. Conclusion

A long distance work in distributed software development environment can cause challenges and some tasks may not be carried out properly due to the difficulty of communication and coordination among team members. This leads to scenarios such as software project fail, delay and budget overrun. This paper proposes an intelligent ontology-based multi-agent conceptual framework that can provide active support to access and recommend knowledge and project information captured in the Software Engineering Ontology. UML 2.1 is used to model social interaction among software agents. It is intended to facilitate effective communication and coordination and provide active support and recommendation for multi-site software development team to reduce the failure rate of software development project.
Acknowledgements

Financial supports for this study are funded by the Australian Government through their provision of the Endeavour Awards program and the Royal Thai Government Scholarship program.

REFERENCES


