Towards Social Network based Approach for Software Engineering Ontology Sharing and Evolution

Pornpit Wongthongtham, Elizabeth Chang, Ahmed A. Aseeri

Digital Ecosystems and Business Intelligence Institute
Curtin University of Technology, Perth, Australia
{Pornpit.Wongthongtham, Elizabeth.Chang, Ahmed.Aseeri}@cbs.curtin.edu.au

Abstract. In this paper, we present a multi-agent social network approach that integrates the software engineering ontology and expert recommendation facilities for communities of software engineers remotely working on related software engineering projects. The software engineering ontology enables an active ecology of agents to convey, consume and act on project information (semi) autonomously, according to explicit software engineering domain knowledge. Recommendation techniques are addressed to make progress – ability to recommend useful project information, solution(s) for project issues that arise as experts.

Keywords: Software Engineering, Ontology, Social Network based Approach.

1 Introduction

Recently, there is an explosion of interest in ontologies as artefacts to represent human knowledge and as a critical component in several applications. One unique area of research is the software engineering ontology. Over the last three years, we have developed the world's first Software Engineering Ontology which is available online at www.seontology.org. The software engineering ontology defines common sharable software engineering knowledge including particular project information [1]. Software engineering ontology typically provides software engineering concepts – what they are, how they are related, and can be related to one another – for representing and communicating over software engineering knowledge and project information through the internet [2]. These concepts facilitate common understanding of software engineering project information to all the distributed members of a development team in a multi-site development environment. This should not be confused with the distributed systems, such as CORBA where the development is centralised but deployment is distributed. The ontology enables effective ways of sharing and reusing the knowledge and the project information for remote software engineers and software developers. Reaching a consensus of understanding is of benefit in a distributed multi-site software development environment. Software engineering knowledge is represented in the software engineering ontology whose instantiations are undergoing evolution. Software engineering ontology instantiations

signify project information which is shared and has evolved to reflect project development, changes in software requirements or in the design process, to incorporate additional functionality to systems or to allow incremental improvement, etc. This evolution of instances provides many new challenges to an ability to design and deliver project information.

In this paper, we present a multi-agent based recommender approach that integrates the software engineering ontology and expert recommendation facilities for communities of software engineers and software developers remotely working on related software engineering projects. The integration of the software agents and the software engineering ontology is an innovative technology that can significantly improve recommender approach for multi-site distributed software development. Particularly, the software engineering ontology enables an active ecology of agents to convey, consume and act on project information (semi) autonomously, according to explicit software engineering domain knowledge. Recommendation techniques are addressed to make progress – ability to recommend useful project information, solution(s) for project issues that arise as experts.

The aims of the project are summarised as follows.

1) The development of a new multi-agent based recommender system architecture that facilitates meaningful communication, discussion, negotiation, and information exchange through collaborative agents.

2) The development of the notion of collaborative agents enabling semantic interoperability.

3) The development of a new project management approach of sharing project artefacts, experts, progress, document and so on.

4) The development of recommendation techniques to manage project issues that arise as experts.

5) The implementation of a new prototype that is a realisation of system architecture and utilises the software engineering ontology and the multi-agent based system.

In the next section, we discuss the related work. In the third section, we highlight agents for remote collaboration. The forth section is dedicated to multi-agent social network approach. In the section five, we describe the current state of the implementation of a prototype of the system. The paper ends by drawing conclusions.

2 Related Work

Some work has been carried out in the areas of expert recommendation including development of tools and systems to support e-business, e-learning and the like. Most of the proposed systems are applied to the recommendation of web pages or documents e.g. GroupLens [3], Adaptive Web Site Agent [4], Remote Assistant for Programmers [5]. The systems recommend documents using different criteria: similarities among users, user preferences for the subject area, similarity between document, frequency of citation and frequency of access, etc. A referral system which is based on the idea of social networks [6] is an agent based system. The agents who

preserve the privacy and autonomy of their users build social network learning models with each other in terms of expertise (ability to produce correct domain answers), and sociability (ability to produce accurate referrals), and take advantage of the information derived from such a social network for helping their users to find other users on the basis of their interests [5]. Additionally, there is some work in the development of tools and systems for supporting e-learning e.g. Web Based Teaching (WBT) [7], I-MINDS [8]. These systems enable students to actively participate in a virtual classroom which is centred on a website containing teaching materials and an electronic bulletin board for question answering. The systems are based on different kinds of agents e.g. teacher agents, student agents, remote proxy agents. Agents interact with humans and are responsible for disseminating information, maintaining student profiles, generating individual quizzes and exercises, filtering students' questions and messages to reduce traffic and manage classroom sessions progress.

However, these current researchers do not focus on a multi-agent social network approach to support software engineering ontology sharing and management. There has not been any expert approach expertise in a field of software engineering applied to the recommendation of particular software engineering project information. Tackling the disadvantages associated with remote communication over software engineering project information is, therefore, still a longstanding problem in remote collaborative software development. Awareness of what work is being done according to the plan, what work is being done to co-operate between teams, what issues have been raised, what issues have been clarified and how to get together to discuss and to make a decision on those issues, are challenging. Different teams might not be aware of what tasks are being carried out by others, potentially leading to problems such as two groups overlapping in some work or other work not being performed due to misinterpretation of the task. Wrong tasks may be carried out due to ignorance of who to contact to get the proper details. If everyone working on a certain project is located in the same area, then situational awareness is relatively straightforward but the overheads in communications to get together to discuss the problems, to raise issues, to make decisions and to find answers in a distributed environment can become very high. Consequently, these problems cause developmental delays, as outstanding issues are not resolved and issues cannot be discussed immediately or in time over a distributed team environment.

3 Agents for Remote Collaboration

In this paper, we present a multi-agent social network approach to support software engineering ontology sharing and management for remote software developers during common software projects or activities. The ability to make use of software engineering knowledge described in the software engineering ontology enables agents to have the capability of extraction and conveyance semantic rich project information from online dedicated repositories. This assists remote members to have a clear understanding of project information conveyed by the agents. Automatic reasoning capacity of autonomous agents helps manage project issues that arise. Agents are not

only able to convey any involved project information according to explicit software engineering domain knowledge but also recommend solution(s) for any project issues as experts on a constant and autonomous basis.

Consider the following scenario from the not too far distant future: Matthew, who is a programmer working in a team in Australia, realises that the project design does not make sense. Matthew then tells his software agent to raise this issue with the design team and request changes. Matthew's agent goes to online project repositories to find out who to contact. After verifying that changes Matthew has requested are conceptually correct according to software engineering domain knowledge described in the software engineering ontology, the agent then raises the issue with the involved design agents. The design agents inform their clients residing in the US and the UK and start collecting their clients' ideas and pass on to a recommender agent. After all, the recommender agent makes a tentative solution for the project issue, informs everyone and updates project information repositories. Team members residing at different sites cannot have a face-to-face meeting. Team members' agents make a virtual meeting on behalf of their clients. As we see from this scenario, in the near future there will be an active ecology of agents that will collaborate with each other and access information sources to fulfil users' needs. Having efficient use of information, organisations can operate virtually and collaborate across huge distances in Australia and around the world. This creative approach is among the fastest growing sectors of the new economy and needed to exploit the huge potential in the multi-site distributed software development industry.

There are currently two sources of knowledge: 'SoftWare Engineering Body of Knowledge (SWEBOK)' and 'Software Engineering Ontology'. The SWEBOK [9] developed by SEI, Canigei Melon University, USA, is a glossary of terms. It provides a definition of each term. It does not define the concepts and the relationship between concepts. We have the Software Engineering Ontology which is a big extension of the SWEBOK. It defines the concepts, and inter-links between the concepts and the relationship between each other and the organisation of the whole body of knowledge, rather than definition of each discrete term.

Nevertheless, either the Software Engineering Ontology or the SWEBOK will provide a benefit for a team of people working together, particularly if a person wants to find the terms and has a good querying facility to do that. These are still passive structures. The passive nature arises from the fact that the user has to come to the ontology with enough knowledge to know what to look for. We move beyond this point and are looking at creating an active support using ontology as the background. We now create Agents situated in the foreground that interact and mediate between the ontology and human agents. The innovation lies in that we are creating an active support for software engineers, especially for multi-site teams working on one project, so that when a software engineer comes to this platform, he/she would get an advisor or a recommender rather than just a Dictionary or the organisation of Knowledge.

For example, if a software engineer knows the terms, he can go ahead and look for it from software engineering body of knowledge to get clarification of the terms. If he/she wants to know the relationships between the terms, he can use the Software Engineering Ontology. On the other hand, if he/she wants advice on what to do in a

certain situation, then we need some active components which are intelligent enough to utilise the ontology to advise the user. It is the creation of these active components that is the subject of this proposal. This approach represents a big shift from the previously available approach to software development in a multi-site environment. This represents a significant break-through for Software Engineering studies.

In order to achieve this, we will have to extend the existing knowledge on interaction between agents and ontologies and between agents and software engineers. However, we will do it purely on the context of supporting the Software Engineering Ontology. Generally, this approach can be used in a generic sense for any ontology. The result of this project should be capable of being used in any other ontologies. These are the classes of Agents that we need: user agents, ontology agents, and recommender agent.

Note, an important thing to realise that we are NOT creating agent-oriented software, which is software developed by utilising the Agent Paradigm as a primary construct of the software, which is legitimate work and quite a lot of work has been done e.g. [10], [11] and so forth. But what we are doing here is that we are using multi-agent paradigm in conjunction with ontology to support software developers and engineers, no matter what development methodology they use, whether they use Object-Oriented, Agent-oriented, or classical Data flow techniques. This is an important step forward.

Our particular approach is the integration of the agent technology with the semantic web technology. Our approach is based on a dynamic network of platforms managing teams of geographically members.

4 Multi-Agent Social Network Approach

A key feature of the system is that it is an agent-based system integrated with another key emerging technology - semantic web. It is an innovative agent-based system supporting remote collaboration applications.

4.1 Multi-Agent based Recommender System Architecture

The multi-agent based system architecture is grounded on the notion of a set of software agents: User agents, Ontology agents and a Recommender agent. Fig. 1 represents main components in the social network based system architecture.

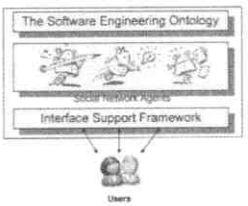

Fig. 1 – Main components in the social network based system architecture.

User agents are agents that allow the interaction between the member and the different parts of the system and among the members themselves. This agent serves as the communication media for members of the team. The agents are initiators based on the member actions to the system and responsible for building the member profile and maintaining it while the member is online. Interactions are performed when the member is active in the system through a web-based interface. The user agents carry out the specific operations accordingly; all the operations have different logic involved. The user agents' life cycle depends on completeness or satisfaction of their task objectives. In other words, the user agents are created for a purpose and once the purpose is achieved, the agents are terminated i.e. when the member is offline.

Ontology agents are responsible for maintaining the software engineering ontology and for serving the right pieces of project information to members' queries. This agent is responsible for updating the software engineering project information into the software engineering ontology and forwarding the update to user agents for updating the right members.

In a software engineering project, the project information over a period of time needs to be modified to reflect project development, changes in the software requirements or in the design process, in order to incorporate additional functionality to systems or to allow incremental improvement. Since changes are inevitable during project development, the software engineering ontology is continuously confronted with many versions of them. Thus, each version is taken care of, by each ontology agent. Such changes are maintained by ontology agents. The ontology agents facilitate updating tasks and ensure reliability and consistency. Note that it does not change the original concepts and relationships in the ontology, rather instantiations of the software engineering ontology change conforming to the ontology change.

Recommender agent is responsible for recommending tentative solutions on project issue. The decision recommended by the agent is based on members in the team agreeing to vote, along with the reputation of each individual member involved in the project.

4.2 Ontological Support for Agent Communication

A major aim of our approach is to enable the integration process from syntactic interoperability to semantic interoperability. The FIPA standard organisation produced a comprehensive set of specifications for interoperable multi-agent systems [12, 13]. It defines the Agent Communication Language (ACL) which provides the syntax for interoperability enabling agents to manage ontologies. The project information in our project is represented as instances of the software engineering ontology defining concepts and their relations. The software engineering ontology is used as an agreement between agents.

The agent platform used i.e. JADE [14] offers a general support for ontologies. It is necessary that the system is not only agent based but also is support for ontologies. Agents convey project information according to the explicit software engineering ontology. We have chosen a compound tool called OWLBeans [15] that allows the

use of software engineering ontology described in OWL DL. The software engineering ontology is used by ontology agents for performing their tasks in cooperation with other agents. OWLBeans aims to provide software engineering ontology management. Dedicated ontology agents acting as ontology servers are able to use and manage the complete software engineering ontology described in OWL and provide services to the other agents that need it. These ontology servers use the Jena toolkit [16, 17] to load, maintain, and reasoning facilities. These ontology servers provide facilities to load maintain and reason the software engineering ontology. They also provide a set of simple actions for querying and manipulating the software engineering ontology. Additionally, proper authorisation mechanisms are taken into account. In particular, the underlying JADE security support has been leveraged to implement a certificate-based access control. Only authenticated and authorised members will be granted access to manage the software engineering ontology.

After some analysis of possible query languages, we have come to the decision of utilising OWL-QL [18], a formal query language targeting the Ontology Web Language (OWL). The reason is simply because the software engineering ontology is represented in OWL and semantic-aware agents use knowledge represented in the OWL software engineering ontology.

4.3 Member Profile Management

An important requirement that has guided the design of our approach has been the support for distributed multi-site team members. The management of experts and project information can provide a great advantage for new members or new teams to catch up with the current project. The team is distributed in multiple sites. Each team can exist and operate isolated sharing of the experts and the project information repositories. The multi-site distributed development nature provides the best conditions for sharing and retrieving project information but also entails some significant techniques in member profile evaluation.

Every member in the teams involved in a given software engineering project has a right to vote for proposed solutions. Everybody's vote is worth points. Below is a list of requirements:

1) A member can work on a project or multiple projects at the same time
2) A member can work in a team or in multiple teams
3) A member can work in different teams in different projects
4) A project involves multiple teams and multiple members
5) A member has a reputation value for a particular area or domain in a given project
6) A member can have a different reputation value for a different area or domain in a different project
7) The reputation value of the team member continues to increase if the team member votes for the chosen (or correct) solution and vice versa
8) The reputation value of the team member decreases if the team member did not vote for the chosen (or correct) solution and vice versa.

The vote cast by each team member is mathematically weighted by the factor of which 'members who actually work on a task have the best understanding of that task'. In other words, if a member votes on an issue which arises within the area he/she is working on, presumably this falls within his/her area of expertise, then his/her vote carries more weight than that of a member who does not have expertise in the issue area, or who does not really work on it. It is assumed, for example, that the designers of a project who work on the project design, have expertise in project design, or know more than others do about this aspect of a project. Thus, if the project issue relates to project design, the votes of members in the design team carry more weight than others.

The reputation value of members may change with time. In other words, over a period of time and in a particular area or domain of expertise, the reputation value of a given member may increase or decrease or in some instances may remain the same. We need a means to account for this dynamic changes in the reputation value of a given user over a duration of time, when taking into account the vote cast by that user [19]. In order to the account for the dynamic nature of reputation, we propose to use the Markov Model. Additionally, using the Markov Model, we can determine what could be the most probable future reputation value of a given team member in the category of the issue at a time in which the decision has to be made. Additionally it may the case the past sequence of reputation values of a given user may exhibit a trend (upwards trend or downward trend or a steady trend) or seasonality pattern (periodical seasonality pattern or non-periodical seasonality pattern) or random noise. The Markov Model and its variants, provide a very good means in order to model the trend components or seasonality components or noise in a given reputation series, and predict the future reputation value while accounting for these components.

The adaptation of the Markov Model for modelling reputation is based on a finite state process. There a total of seven trust states, with one trust state denoting ignorance. We make use of the integers in the domain of {0, 6} to represent the trust states, with 0 denoting ignorance. The values from 0 to 6 represent the trust states. The past duration of time over which the reputation of a given user is to be modelled is termed as time space. The time space is divided into non-overlapping, mutually exclusive and non-intersecting finite number of slots of time, each of which is termed as time slot. In order to model the dynamic nature of reputation and in order to predict the future reputation value of a given user, its reputation value of over the sequence of the past time slots is determined, resulting in the reputation series.

Firstly, the Markov Model determines if the reputation series is a non-stationary reputation series. It does so by determining if the reputation series exhibits (a) seasonality components (b) trend components or (c) noise components. Broadly speaking, two scenarios arise here: (a) the given reputation series exhibits just one of the aforesaid components and (b) the reputation series exhibits more than one of the aforesaid mentioned components.

We would develop algorithms to handle both of these scenarios. In this case (when the reputation series exhibits one or more than one of the above components listed above), we would develop an algorithm that would be capable of modelling, the dynamic nature of reputation taking into account the corresponding components

present in the reputation series. Additionally, the developed algorithm would be capable of predicting the future reputation value of the given user, taking into account the corresponding components present in the reputation series. Furthermore, we develop specialized algorithms in order to address the scenario where in the reputation series exhibits only one of the above mentioned components. The above process is used when the reputation series is basically a non-stationary reputation series. However in case the reputation series is stationary (which is rarely the case) [19], then the process of modelling the dynamic nature of reputation and the process of predicting the future reputation value of a given user is straightforward, relative to the process of modelling a non-stationary reputation series and subsequently predicting the future reputation value of a non-stationary reputation series. In such a case it is based on a discrete time state process and would be capable of modelling the stationary aspect of the reputation series and subsequently predicting the future reputation values well.

4.4 Reputation based Recommendation technique

Whenever issues arise, the recommender agent sends a message to every involved member through user agents requesting for an opinion. Subsequently, the agent gathers and stores the possible solutions. Of all the possible solutions, one solution is recommended for one of the proposed solutions. Each vote is weighted by the expertise of the member casting it in the area of the problem. The reputation value of individual member who votes is weighted.

Assume that a weighting value for member who his/her expertise is not in the area of the issue is 0.2 and a weighting value for member who his/her expertise is in the area of the issue is 0.8. For example, three possible solutions named A, B, and C on the project design issue. Let's assume that for the design area, the reputation point of a member who votes for solution A is 1. Since this member's expertise is in the area of design, which is the area where the issue is raised, (i.e. project design), this member's vote would have a value of 0.8 (1 multiplied by 0.8). If the reputation value of a member who votes for solution B is 2, then this member's vote would have of value 0.4 (2 multiplied by 0.2) because this member's expertise area is requirement (this member is an analyst) while the issue is about project design. Similarly, if the reputation value of a member who votes for solution C is 1, then this member's vote would have a value of 0.2 (1 multiplied by 0.2) because this member's expertise area is construction (this member is a programmer) while the issue is about project design.

For a particular issue, whichever solution has the highest vote value will be recommended. Therefore, from this example, solution A that has the highest vote value is recommended as a final solution. Once a solution has been recommended and finalised, the project data in the software engineering ontology will be updated along the lines of the recommended solution. The system advises all team members of the final decision and records the event. The users' reputation points are also updated for future use.

5 System Implementation

A first prototype of the multi-agent based recommender system is under development using JADE. JADE (Java Agent Development framework) is a software framework that simplifies the implementation of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems [20, 21].

Given the distributed nature of the JADE based multi-agent systems, teams can be distributed through multiple sites connecting usually through the internet and situated in different parts of the globe. Each system platform can be distributed on different computation nodes and it is connected to a web server where agents reside allowing direct interactions with the members. In the web server, there might be more than one user agent and ontology agent but there is a unique recommender agent. In order to cope with performance due to the number of the members to be managed, each user agent is assigned to each member. For reliability reasons, ontology agents manage versioning of the software engineering ontology. It is necessary to develop an ontology server that maintains, uploads and versions of the ontologies but can work in conjunction with the multi-agent system. This requires an extension of an existing single agent ontology server.

6 Conclusion

In this paper, we have defined the multi-agent based collaboration architecture and analysed the roles of the three types of software agents within the architecture. We have also defined the nature and mechanisms of the interaction between the software agents and ontology within collaboration architecture. The reputation based recommendation technique is determined. A first prototype system is under development and evaluation.

References

1. Wongthongtham, P., et al., *Ontology-based multi-site software development methodology and tools.* Journal of Systems Architecture, 2006. **52**(11): p. 640 - 53.
2. Wongthongtham, P., *A methodology for multi-site distributed software development*, in *School of Information Systems*. 2006, Curtin University of Technology: Perth.
3. Resnick, P., et al. *GroupLens: An open architecture for collaborative filtering of netnews*. in *Computer Supported Cooperative Work*. 1994: Chapel Hill.
4. Pazzani, M. and D. Billsus, *Adaptive Web Site Agents.* Autonomous Agents and Multi-agent Systems, 2002. **5**: p. 205-218.

5. Mari, M., A. Poggi, and P. Turci. *Ontology-Based Remote Collaboration for the Development of Software System*. in *the 2nd Italian Semantic Web Workshop (SWAP 2005)*. 2005. University of Trento, Trento, Italy.
6. McDonald, D.W. *Evaluating expertise recommendations*. in *International ACM SIGGROUP Conference on Supporting Group Work*. 2001. Boulder, CO.
7. Ishikawa, T., H. Matsuda, and H. Takase. *Agent Supported Collaborative Learning Using Community Web Software*. in *International Conference on Computers in Education*. 2002. Auckland, New Zealand.
8. Liu, X., et al. *I-MINDS: An Application of Multi-agent System Intelligence to On-line Education*. in *IEEE International Conference on Systems*. 2003. Washington DC, USA.
9. Bourque, P. *SWEBOK Guide Call for Reviewers*. 2003 [cited 29 May 2003]; Available from: http://serl.cs.colorado.edu/~serl/seworld/database/3552.html.
10. Henderson-Sellers, B. and P. Giorgini, *Agent-Oriented Methodologies* 2005: Idea Group Publishing 413.
11. Zhang, Z. and C. Zhang, *Agent-Based Hybrid Intelligent Systems: An Agent-Based Framework for Complex Problem Solving*. 2004, Germany: Springer-Verlag.
12. *FIPA Specifications*. 2005 [cited; Available from: http://www.fipa.org.
13. Bellifemine, F., A. Poggi, and G. Rimassa, *Developing multi-agent systems with a FIPA-compliant agent framework*. Software Practice and Experience, 2001. **31**: p. 103-128.
14. *JADE home page*. [cited; Available from: http://jade.tilab.com.
15. Bergenti, F., et al. *An Ontology Support for Semantic Aware Agents*. in *Seventh International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2005 @AAMAS)*. 2005. Utrecht, The Natherlands.
16. Carroll, J.J., et al., *Jena: Implementing the Semantic Web Recommendations*. 2004, Digital Media Systems Laboratory, HP Laboratories Bristol.
17. McCarthy, P., *Introduction to Jena: use RDF models in your Java applications with the Jena Semantic Web Framework*. 2004, SmartStream Technologies, IBM developerWorks
18. Fikes, R., P. Hayes, and I. Horrocks, *OWL-QL - A Language for Deductive Query Answering on the Semantic Web*, in *Technical Report KSL-03-14*, Knowledge Systems Lab, Stanford University: CA, USA.
19. Chang, E., T. Dillon, and F.K. Hussain, *Trust and Reputation for Service Oriented Environment: Technologies For Building Business Intelligence And Consumer Confidence*. 2006: John Wiley and Sons.
20. Bellifemine, F., *JADE Java Agent DEvelopment Framework*. 2001, Telecom Italia Lab: Torino, Italy.
21. Bellifemine, F., A. Poggi, and G. Rimassa. *JADE: a FIPA2000 compliant agent development environment*. in *The fifth International Conference on Autonomous Agents*. 2001. Montreal, Quebec, Canada: ACM Press, New York, USA.