Towards Active Software Engineering Ontology

An Intelligent Support to Access and Recommend Knowledge and Project Information

Udsanee Pakdeetrakulwong
School of Information Systems, Curtin Business School
Curtin University
Perth, Australia
udsanee.pakdeetr@student.curtin.edu.au

Pornpit Wongthongtham
School of Information Systems, Curtin Business School
Curtin University
Perth, Australia
P.Wongthongtham@cbs.curtin.edu.au

***Abstract*—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 order to address such challenges, in the literature, the software engineering ontology has been developed to provide a common understanding of software engineering domain knowledge and share software project information among dispersed team members. However, the nature of the software engineering ontology is still passive as is the nature of many existing ontologies. Passive means that users of the ontology need the competence to access and translate what they are looking for into the concepts and relations defined in it; otherwise, they may not be able to receive precise knowledge and project information. The purpose of this paper is to develop a methodology to provide active support to access and recommend knowledge and project information in the software engineering ontology. Two main key technologies, i.e., agent-based system and recommendation techniques are exploited in this research. Intelligent agents can cope with the distribution and interoperability of a global software development environment and when they are integrated with recommendation approaches, they can offer automated suggestions to actively support software team members. This aims at providing the most relevant and precise situational knowledge and will lead to improving the effectiveness of communication and coordination of long-distance collaborative work. The scope of the proposed work is not just limited to software engineering ontology; rather, it is domain-independent and can be applied to any other ontology in any domain.**

Keywords—software engineering ontology; multi-agent systems; recommendation systems; multi-site software development

I. INTRODUCTION

The globalization of the software development industry and Internet technology enables software team members to work not only in multiple locations in the same country, but also in different regions and in different countries. Ågerfalk [1] stated the reasons why organizations consider adopting distributed development of software systems and application models. The reasons include utilizing a larger labor pool and a broader skill base as well as cost advantages, e.g., hiring software developers from lower wage countries like India or the Philippines. Conchúir [2] also mentioned other advantages like market proximity, local knowledge accessibility and adaptability to various local opportunities. However, for a distributed team working without face-to-face communication, this can cause misinterpretation of the software project information due to the different cultures and educational backgrounds, temporal and social issues, including diverse working styles. This may result in some tasks not being carried out properly due to the difficulty of communication and coordination among team members located in different geographical areas. This may lead to scenarios such as software project delay and budget overrun. In the literature, many researchers have proposed methods to overcome these issues in multi-site software development. Thissen [3] discussed the communication tools and collaboration processes that are used in globally dispersed projects to facilitate team communication and interaction. Wongthongtham [4] introduced an ontology model of software engineering as a part of a communication framework to define common software engineering domain knowledge and share useful project information for long-distance collaborative work.

In the context of knowledge sharing, ontology is an explicit specification of a conceptualization [5]. Ontologies can be used by a distributed software development team to provide a more precise and clear terminology because of their distinct characteristics in semantics and reasoning ability. Anandaraj [6] stated that, nowadays, ontologies have been attracted and can be applied in software engineering throughout the various stages of the software development life cycle because they can provide a shared conceptualization of fundamental concepts and relationships of software development projects. In addition, Siricharoen [7] proposed the use of ontologies to build object models in object-oriented software development because of the similarity between objects in object models and concepts or classes in ontologies.

Software engineering ontology was first developed to provide efficient collaboration among software team members who are geographically distributed. It is considered an efficient means of clarifying the concepts and project information and enabling knowledge sharing among the software team members [4]. It represents software engineering

knowledge and concepts, software development methodologies, software tools and techniques. The software engineering ontology comprises two ontologies: generic ontology and application specific ontology. Generic ontology contains concepts and relationships annotating software engineering contents which are transformed so as to be machine-readable and man-machine interoperable. Application specific ontology defines explicit specification for a particular software development project and is used to facilitate consistent communication within software team.

However, the current software engineering ontology has the same passive structure as other ontologies [8]. Passive structure means that in order to address the ontology, users need to have competence to translate the issue to the concepts and relationships to which they are referring; otherwise, the user may not be able to obtain precise knowledge and project information. In order to address this drawback, active support is needed that can utilize the ontology to advise users on what to do in a certain situation.

This paper is structured in the following manner. In Section 2, we discuss about the related work. In section 3, we propose our conceptual framework. Scenario examples of our proposed system are presented in Section 4. Finally, the conclusion and future work are discussed in Section 5.

II. RELATED WORK

A. Agent Technologies

An agent is a computer program that has relatively complete functionality and cooperates with others to meet its designed objectives [9]. 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 [10]. Multi-agent systems (MAS) consist of multiple agents communicating and collaborating with each other in one system in order to achieve goals [10]. Romero [11] introduced a multi-agent simulation tool to support training in global requirement elicitation process. They use agent technology to simulate various stakeholders in order to enable requirement engineers to understand and gain experience in acquiring requirement elicitation. Knowledge sharing and exchange is one of key factors in the development of MAS. Each agent will collaborate with other agents, so they must be able to communicate and understand messages from one another. Ontologies can be used to facilitate the semantic interoperability while Agent Communication Language (ACL) defined by FIPA can be used as the language of communication between agents. Existing researches that integrate the use of ontologies and MAS are in [12], [13], [14], and ContextP-GSD [15]. It is evident that many researches have exploited multi-agent technology in various applications and a number of them utilized multi-agent technology along with the use of ontologies to support software development tasks. However, most of them cover only a specific phase or issue in software engineering domain knowledge. Currently, there are no multi-agent system applications that provide active communication and coordination throughout the whole software engineering process.

B. Recommendation Systems

Recommendation systems are techniques or software tools assisting users with suggestions for items, contents or services to be of use in overloaded amounts of information [16]. The initial academic work on implementing recommendation systems was first conducted in the mid-1990s. Park [17] undertook a literature review and classification of recommender systems based on 210 research papers on recommendation systems published in academic journals between 2001 and 2010. The result shows that publications related to this topic had increased significantly and it is highly likely that research in the area of recommendation systems will be active and has the potential to increase significantly in the future.

a) Classification of recommendation system approaches

Many researches have been implemented with well-known recommendation system techniques such as collaborative filtering, content-based filtering, and hybrid recommender systems. The collaborative filtering approach recommends items to the users based on the similarity between users. The content-based filtering technique recommends items which resemble the ones that a specific user formerly preferred. However, these two approaches still have their own drawbacks. Collaborative filtering techniques have the problems of cold start and sparsity while content-based filtering approaches suffer from the overspecialization problem whereby only those items similar to those the user already knows are recommended. Hybrid recommender systems have been introduced by combining these two approaches to resolve the problems associated with certain approaches. Many recommendation systems use syntactic matching techniques that relate items from common words not from their meaning, so the result of recommendations are sometimes limited and poor quality. Semantic-based recommendation systems have emerged to address the limitations of previous recommendation techniques. These recommendation approaches integrate the semantic knowledge in their processes and their performances are based on a knowledge base which contains relations between concepts, normally defined through ontology or concept-diagram (like taxonomy) [18]. Blanco-Fernández [19] stated that semantic-based recommendation systems have been proven to have better performance than previous approaches by applying a knowledge base and semantic reasoning filtering techniques. These two elements can help to improve the accuracy of recommendation systems because semantic descriptions are used, unlike syntactic approaches which consider the word

only. Various applications in several fields have been proposed which include a semantic reasoning mechanism in their recommendation systems, for instance, in [30], [31], and [32]. Although semantic-based recommendation systems have been employed in various domains, none of them is specifically intended to create recommendations to manage queries or project issues raised in software development teams through the use of ontologies in software engineering.

b) Recommendation systems for software engineering

Recommendation systems for software engineering (RSSEs) are software tools introduced specifically to help software development teams to deal with information-seeking and decision-making. [33]. RSSEs have become an active area of research for the past several years and they have proven to be effective and useful to software developers working on software development projects. A comparison of reviewed RSSEs is presented in Table 1.

From Table 1, only a recommender system for requirements elicitation in large-scale software projects proposed by Castro-Herrera [29] has been developed for the software requirement process and it uses a collaborative recommendation approach to build a system that can support collaborations among stakeholders who are involved in software requirement elicitation. Most of other RSSEs researches are conducted during the implementation phase with the aim of locating a domain expert such as Codebook [20], Conscius [21], [22] and Ensemble [23], or focus on supporting developers who are writing source codes or debugging programs, i.e., Fishtail [24], [25], DebugAdvisor [26], Dhruv [27], and Semantic Helper [28]. All the described applications have been developed to improve the productivity of software development projects only for one of phases in SDLC, and most of them focus on the implementation phase in particular. However, software team members mostly need support in every phase of a software development project. Regarding knowledge representation, all RSSEs from Table 2 except for Dhruv and Semantic Helper use traditional knowledge representation and syntactic matching techniques so they lack integrated and shared information and cannot support a semantic reasoning mechanism.

C. Ontology-based semantic annotation

Ontology-based semantic annotation, or semantic annotation for short, is the process of connecting document information items to concepts defined in ontologies by creating individual instances, and constructing a knowledge base by using ontologies as skeletons [34]. In this approach, ontologies are an important main part of an application domain description and are used as a means of identifying semantically-related annotations. Ontology-based semantic annotation is deployed to generate intelligent content and provide a wide range benefit to content-oriented intelligent applications [35]. In the semantic web, it helps to annotate web documents and enables human and machine to understand web contents. Valarakos [36] considered a semantic annotation as the step before ontology population which is the process of adding new concepts or properties to the formal model of the ontology. Semantic annotation is a means of obtaining new instances but not adding new instances to a knowledge base constrained by the ontology. Amardeilh [37] pointed out that one advantage of semantic annotation is that it can be used to supplement ontology populating tasks and the author also proposed the ONTOPOP framework aiming at processing documentary resources to perform the semantic annotation and ontology population tasks.

TABLE 1. A COMPARISON OF REVIEWED RSSES RESEARCHES

Methodologies/ Frameworks/ Authors	Objective	What/Who to recommend	Semantic knowledge representation	Focus (SDLC phase)
CodeBook [20]	Help developers to discover and maintain connections to others	Related people	No	Implementation
Conscius [21]	Locate expert on a given software project	Expert	No	Implementation
Steinmacher [22]	Support newcomers	Expert	No	Implementation
Ensemble [23]	Help developers have better coordination	Related people	No	Implementation
Fishtail [24]	Support source code writing	Source code examples	No	Implementation
Cordeiro [25]	Support problem solving	Question/answering web resources	No	Implementation
DebugAdvisor [26]	Provide a search tool for debugging	Related people, source files, functions to the bug report	No	Implementation
Dhruv [27]	Support online problem-solving communities with the semantic web	Objects and software artifacts, bug resolution	Yes	Implementation
Semantic Helper [28]	Filter information and do automatic matching between models in FACIT-SME repositories and SE methodologies	Ranking lists of the most relevant quality models	Yes	Implementation
Recommendation system for requirement elicitation [29]	Facilitate stakeholder collaboration in a large-scale software projects	The placement of stakeholders into related discussion forums	No	Requirement analysis

III. MULTI-AGENT BASED RECOMMENDER SYSTEM CONCEPTUAL FRAMEWORK

This section will be proposed the multi-agent based recommender system conceptual framework that will integrate the use of collaborative intelligent agents and recommendation techniques to access and recommend software engineering knowledge and project information captured in the software engineering ontology. The framework is developed as shown in Fig. 1. The multi-agent based recommender system framework comprises four types of agents, namely, user agents, semantic recommender agent, ontology agents, and evolution agent working collaboratively.

User agents act as representatives of each user. They are active when a user is online and are responsible for building the member profile and maintaining it. They uniquely identify each team member within the system and enable users to communicate with each other. When users issue queries or raise issues, they can do it through their personal user agents and the agent will pass their requests to a recommender agent or ontology agents. Another important responsibility of user agents is to provide an automatic semantic annotation service. In the software engineering ontology, this consists of abstractions of the software engineering domain concepts and instantiations. As mentioned, there are two types of abstraction: generic software engineering representing a whole set of software engineering domain concepts, and application specific software engineering illustrating the set of software engineering concepts used for particular projects. Instantiations, also known as population, are part of the abstraction of the application specific software engineering ontology. They are used for storing data instances of the projects. Software project information is often updated according to changes in requirements or in design processes; therefore, manually transformation or mapping new changes into semantically rich form and populating them as instances of the software engineering ontology is time-consuming, laborious, tedious and prone to error. With the help of agents which perform semantic annotation process and ontology population, project information can be automatically transformed or mapped into concepts defined in the ontology with a minimum of human intervention.

The Semantic recommender agent, or recommender agent for short, is in charge of providing the tentative solutions and the most relevant knowledge according to user request. After receiving issues from user agents, they will make a decision based on knowledge explicitly described in the software engineering ontology and other resources, e.g. user profiles or issue tracking systems. The recommender agent will work with ontology agents to search for possible solutions from the software engineering ontology repository and to recommend based on the semantic recommendation techniques. If the recommender agent really cannot provide the tentative solutions, it may indicate a different understanding about different categories of systems being developed. Users then can raise issues using the Software Engineering Social Network system (SESN) for software engineering ontology evolution process. This is beyond the scope of this project but more information can be found in [38] and [39].

Ontology agents are responsible for managing the

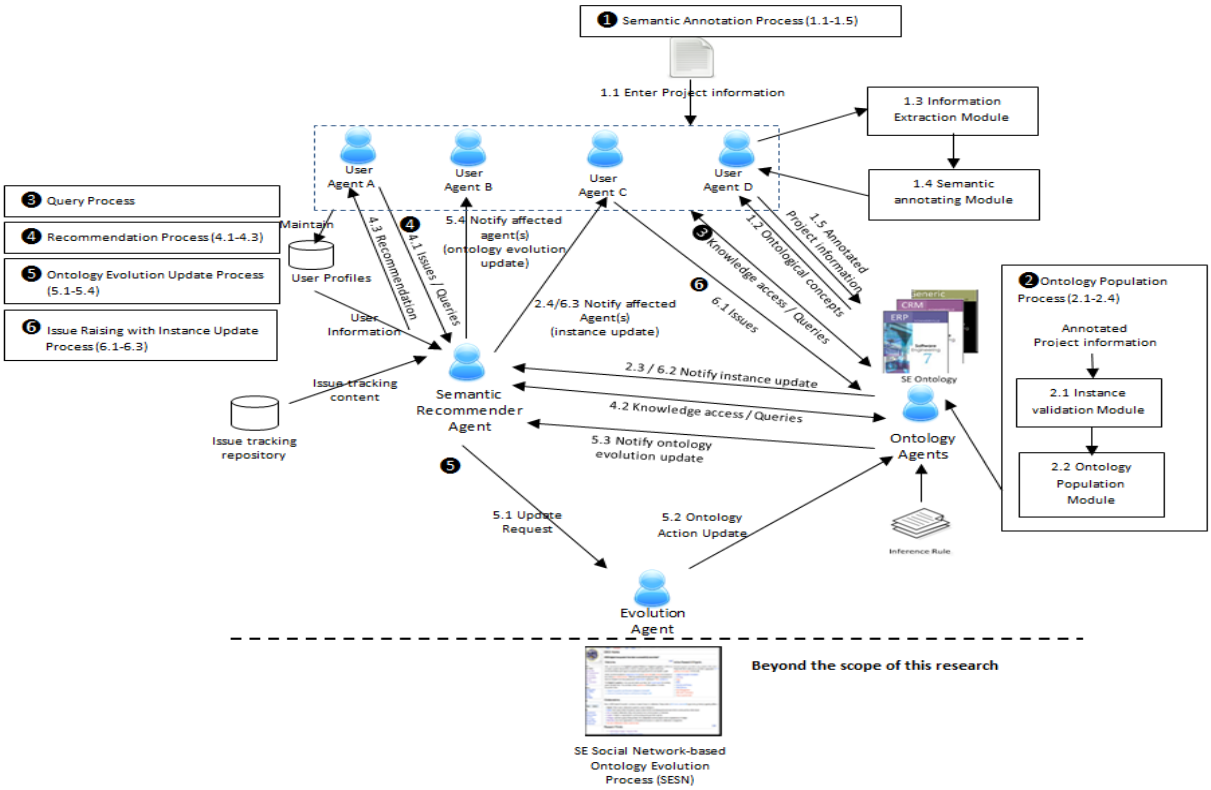

Fig. 1. Multi-agent based recommender system conceptual framework

connection and maintaining the software engineering ontology other agents in accordance with their queries. Another responsibility of ontology agents is the ontology updating task. It can be categorized into two types: instantiation update and ontology evolution update. For instantiation update, the ontology agents will do the ontology population task by receiving the annotated project information from user agents and populate it into the specific ontology as an instance. This change does not modify any original concept or relation in the ontology so it is not an ontology evolution. However, for ontology evolution update, ontology agents will receive notification to update the ontology from the evolution agent. This change will cause some concepts and relationships to be adjusted and leads to the change of generic concepts in the ontology. This is called the ontology evolution and may generate a new version of software engineering ontology. It is to be noted that a version of software engineering ontology refers to a broad category of software applications e.g. software engineering for CRM, ERP or cloud computing rather a specific software development project. Therefore, each version still needs each ontology agent to manage and maintain including ensure reliability and consistency.

The Evolution agent is intended as a means of dealing with the software engineering ontology evolution process. If that semantic recommender agent is not able to recommend solutions due to requests that do not match with the concepts defined in the software engineering ontology or different understandings of project-related information, the evolution agent will coordinate with the social network based software engineering ontology evolution process for the update. Nevertheless, this is beyond the scope of this research. When the evolution process is completed and agreement regarding changes has been reached, the evolution agent will notify the update to the ontology agent to merge these concepts with the existing software engineering ontology and tell the recommender agent to notify all affected agent(s) (refer to Fig.1 – ❺ontology evolution update process). This may create another version of the software engineering ontology.

IV. SCENARIO EXAMPLES OF MULTI-AGENT BASED RECOMMENDER SYSTEM PROVIDING ACTIVE SUPPORT THROUGH ONTOLOGY FOR MULTI-SITE SOFTWARE DEVELOPMENT

This section is to examine scenario examples that show some of capabilities of the system to support communication and coordination in software development project.

A. Example 1

Member A raises an issue about customer class diagram through the information platform in plain text. From the content, the ontology agent will automatically parse software engineering terms by referring to the concept in software engineering ontology and autonomously reason and derive only related instances which are customer class and other relevant classes and relationships. Then it will dynamically draw the diagram from the retrieved information and show this to Member A (refer to Fig. 1 –❸query process). He or other members can propose their opinions by working on the repository. They provide information from the ontology to diagram itself and do tracked changes. The ontology agent will also warn them about affected classes or components from their change proposal. The content in ontology repository will not be updated until the final agreement has been discovered. Then ontology agent will converse the solution diagram and store it back into the semantic format of the specific software engineering ontology. The recommender agent will automatically notify only those team members who should be advised about the changes and their effects (refer to figure 1 – ❻ issue-raising with instance update process). It makes a discussion among team members to propose issues, questions or solution easier than communicating with normal plain texts or just words. So with the support of collaborative agents, long-distance communication which often causes misunderstanding problems during the multi-site software development can proceed more clearly and effectively.

B. Example 2

Member B is a system analyst. Since the user requirement has changed, an additional class has to be added (considered as a new instance) into the specific software engineering ontology in which all project data is generally stored as instances. He contacts his user agent and inputs project information about the additional class. The user agent will automatically annotate it into concepts formed in the ontology through a semantically annotating process. Related concepts, classes, data type, object property and data type property are used as metadata to annotate the content of documents (refer to Fig. 1 – ❶semantic annotation process). The annotated additional class will be in the semantic structure of the software engineering domain and ready to be populated to the ontology by ontology agents (refer to Fig.1 – ❷ontology population process). The recommender agent will take responsibility for notifying all affected agent(s) about this ontology instance update. With the assistance of collaborative agents, the work load of manually annotating a document and populating ontology with project data/project agreement which can be regarded as a time-consuming, labor-intensive, tedious and prone to error task can be reduced and requires the minimum human involvement.

V. CONCLUSION AND FUTURE WORK

This paper proposes the multi-agent based recommender system conceptual framework for providing an intelligent support to access and recommend knowledge and project information captured in the software engineering ontology. 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. It is important to be noted that this paper documents our preliminary research findings. We will continue to develop a methodology and carry out the validation and verification of our proposed work by using a prototype system and obtain feedback from users to measure the performance, usability and effectiveness of the proposed system.

REFERENCES

[1] P. J. Ågerfalk, B. Fitzgerald, H. Holmström, B. Lings, B. Lundell, and E. O. Conchúir, "A framework for considering opportunities and threats in distributed software development." pp. 47-61.

[2] E. Ó. Conchúir, P. J. Ågerfalk, H. H. Olsson, and B. Fitzgerald, "Global software development: where are the benefits?," *Communications of the ACM,* vol. 52, no. 8, pp. 127-131, 2009.

[3] M. R. Thissen, J. M. Page, M. C. Bharathi, and T. L. Austin, "Communication tools for distributed software development teams," in Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, St. Louis, Missouri, USA, 2007, pp. 28-35.

[4] P. Wongthongtham, E. Chang, T. S. Dillon, and I. Sommerville, "Development of a software engineering ontology for multi-site software development," *IEEE Transactions on Knowledge and Data Engineering*, 2008.

[5] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition,* vol. 5, no. 2, pp. 199-220, 1993.

[6] A. Anandaraj, "Study of ontology in software modelling process and life cycle," *International Journal of Research and Reviews in Software Engineering (IJRRSE),* vol. 1, no. 1, 2011.

[7] W. V. Siricharoen, "Ontology modeling and object modeling in software engineering," *International Journal of Software Engineering and its Applications,* vol. 3, no. 1, 2009.

[8] P. Wongthongtham, T. Dillon, and E. Chang, "State of the art of community-driven software engineering ontology evolution." pp. 1039-1045.

[9] H. Qingning, Z. Hong, and S. Greenwood, "A multi-agent software engineering environment for testing Web-based applications." pp. 210-215.

[10] V. N. Marivate, G. Ssali, and T. Marwala, "An intelligent multi-agent recommender system for human capacity building." pp. 909-915.

[11] M. Romero, A. Viscaino, and M. Piattini, "Towards the definition of a multi-agent simularion environment for education and training in global requirements elicitation." pp. 48-53.

[12] S. Paydar, and M. Kahani, "An agent-based framework for automated testing of web-based systems," *Journal of Software Engineering and Applications*, 2011.

[13] C.-S. Lee, and M.-H. Wang, "Ontology-based computational intelligent multi-agent and its application to CMMI assessment," *Applied Intelligence,* vol. 30, no. 3, pp. 203-219, 2009/06/01, 2009.

[14] I. Nunes, C. P. Lucena, U. Kulesza, and C. Nunes, "On the development of multi-agent systems product lines: A domain engineering process," *Agent-Oriented Software Engineering X*, Lecture Notes in Computer Science, pp. 125-139: Springer Berlin Heidelberg, 2011.

[15] H. Monte-Alto, A. Biasão, L. Teixeira, and E. Huzita, "Multi-agent applications in a context-aware global software development environment distributed computing and artificial intelligence," Advances in Intelligent and Soft Computing, pp. 265-272: Springer Berlin / Heidelberg, 2012.

[16] T. Mahmood, and F. Ricci, "Improving recommender systems with adaptive conversational strategies," in Proceedings of the 20th ACM conference on Hypertext and hypermedia, Torino, Italy, 2009, pp. 73-82.

[17] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim, "A literature review and classification of recommender systems research," *Expert Systems with Applications*, 2012.

[18] Q. Gao, J. Yan, and M. Liu, "A semantic approach to recommendation system based on user ontology and spreading activation model." pp. 488-492.

[19] Y. Blanco-Fernández, M. López-Nores, J. J. Pazos-Arias, and J. García-Duque, "An improvement for semantics-based recommender systems grounded on attaching temporal information to ontologies and user profiles," *Engineering Applications of Artificial Intelligence,* vol. 24, no. 8, pp. 1385-1397, 2011.

[20] A. Begel, K. Yit Phang, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories." pp. 125-134.

[21] A. Moraes, E. Silva, C. d. Trindade, Y. Barbosa, and S. Meira, "Recommending experts using communication history," in Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering, Cape Town, South Africa, 2010, pp. 41-45.

[22] I. Steinmacher, I. S. Wiese, and M. A. Gerosa, "Recommending mentors to software project newcomers." pp. 63-67.

[23] P. F. Xiang, A. T. T. Ying, P. Cheng, Y. B. Dang, K. Ehrlich, M. E. Helander, P. M. Matchen, A. Empere, P. L. Tarr, C. Williams, and S. X. Yang, "Ensemble: a recommendation tool for promoting communication in software teams," in Proceedings of the 2008 international workshop on Recommendation systems for software engineering, Atlanta, Georgia, 2008, pp. 1-1.

[24] N. Sawadsky, and G. C. Murphy, "Fishtail: from task context to source code examples," in Proceedings of the 1st Workshop on Developing Tools as Plug-ins, Waikiki, Honolulu, HI, USA, 2011, pp. 48-51.

[25] J. Cordeiro, B. Antunes, and P. Gomes, "Context-based recommendation to support problem solving in software development." pp. 85-89.

[26] B. Ashok, J. Joy, H. Liang, S. K. Rajamani, G. Srinivasa, and V. Vangala, "DebugAdvisor: a recommender system for debugging," in Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, Amsterdam, The Netherlands, 2009, pp. 373-382.

[27] A. Ankolekar, K. Sycara, J. Herbsleb, R. Kraut, and C. Welty, "Supporting online problem-solving communities with the semantic web." pp. 575-584.

[28] F. W. Jaekel, E. Parmiggiani, G. Tarsitano, G. Aceto, and G. Benguria, "FACIT-SME: A semantic recommendation system for enterprise knowledge interoperability," *Enterprise Interoperability V*, pp. 129-139, 2012.

[29] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, "A recommender system for requirements elicitation in large-scale software projects," in Proceedings of the 2009 ACM symposium on Applied Computing, Honolulu, Hawaii, 2009, pp. 1419-1426.

[30] Y. Blanco-Fernández, J. J. Pazos-Arias, A. Gil-Solla, M. Ramos-Cabrer, M. López-Nores, J. García-Duque, A. Fernández-Vilas, R. P. Díaz-Redondo, and J. Bermejo-Muñoz, "A flexible semantic inference methodology to reason about user preferences in knowledge-based recommender systems," *Knowledge-Based Systems,* vol. 21, no. 4, pp. 305-320, 2008.

[31] I. Cantador, P. Castells, and A. Bellogín, "An enhanced semantic layer for hybrid recommender systems: Application to news recommendation," IGI Global, 2011, pp. 44-78.

[32] B. Vesin, M. Ivanović, A. Klašnja-Milićević, and Z. Budimac, "Protus 2.0: Ontology-based semantic recommendation in programming tutoring system," *Expert Systems with Applications,* vol. 39, no. 15, pp. 12229-12246, 2012.

[33] M. Robillard, R. Walker, and T. Zimmermann, "Recommendation systems for software engineering," *Software, IEEE,* vol. 27, no. 4, pp. 80-86, 2010.

[34] J. Liang, S. Tiebing, and Q. Fuzheng, "OMBMDID: A preliminary attempt at automatic utilization of knowledge contained in mechanical design documents," *Concurrent Engineering,* vol. 17, 2, pp. 159-166, June 2009.

[35] K. Yang, "A Conceptual framework for semantic web-based e-commerce," Département d'informatique et de génie logiciel, Université Laval, 2006.

[36] A. Valarakos, G. Vouros, and C. Spyropoulos, "Machine learning-based maintenance of domain-specific application ontologies," *Ontologies*, Integrated Series in Information Systems, pp. 339-372: Springer US, 2007.

[37] F. Amardeilh, "Semantic annotation and ontology population," *Semantic Web engineering in the Knowledge Society*, pp. 135-160: IGI Global, 2009.

[38] A. A. Aseeri, "Lightweight community-driven approach to support ontology evolution," School of Information Systems, Curtin University, 2011.

[39] N. Kasisopha, and P. Wongthongtham, "Semantic wiki-based ontology evolution." pp. 493-495.