International Journal of Digital Information and Wireless Communications (IJDIWC) 3(4): 29-42
The Society of Digital Information and Wireless Communications, 2013 (ISSN: 2225-658X)

State of the Art of a Multi-Agent Based Recommender System for Active Software Engineering Ontology

Udsanee Pakdeetrakulwong [1] and Pornpit Wongthongtham [2]
School of Information Systems, Curtin Business School
Curtin University
Kent Street Bentley WA 6102, Australia
[1] udsanee.pakdeetr@postgrad.curtin.edu.au, [2] ponnie.clark@curtin.edu.au

ABSTRACT

Software engineering ontology was first developed to provide efficient collaboration and coordination among distributed teams working on related software development projects across the sites. It helped to clarify the software engineering concepts and project information as well as enable knowledge sharing. However, a major challenge of the software engineering ontology users is that they need the competence to access and translate what they are looking for into the concepts and relations described in the ontology; otherwise, they may not be able to obtain required information. In this paper, we propose a conceptual framework of a multi-agent based recommender system to provide active support to access and utilize knowledge and project information in the software engineering ontology. Multi-agent system and semantic-based recommendation approach will be integrated to create collaborative working environment to access and manipulate data from the ontology and perform reasoning as well as generate expert recommendation facilities for dispersed software teams across the sites.

KEYWORDS

Software engineering ontology, multi-agent based systems, recommendation systems, multi-site software development, ontology development

1 INTRODUCTION

Due to the emergence of the Internet and the globalization of software development, there has been a growing trend towards the traditional centralized to the distributed software development form which means that software team members work on the same project but they are not co-located. They are distributed across cities, regions, or countries. For example, the requirement specification and design are done in Austria, the development is done in China and Brazil and the testing is done in Russia. There are several terms used for this approach, for example, Global software development (GSD), Distributed software development (DSD), or Multi-site software development (MSSD). Ågerfalk et al. [1] discussed the reasons why organizations consider adopting distributed development of software systems and application models which include utilizing larger labor pool, accessing broader skill base, minimizing production costs and reducing development duration from round the clock working. Conchúir et al. [2] also mentioned other advantages like market proximity, local knowledge accessibility and adaptability to various local opportunities. However, this type of long-distance collaborative work is not without problems. It can cause challenges such as communication difficulties, coordination barriers, language and cultural differences [3]. 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 and lead to scenarios such as software project delay and budget overrun. Many researches were proposed to overcome these issues. Thissen et al. [4] discussed the communication tools and collaboration processes that were used in globally distributed projects to facilitate team communication and interaction. Biehl et al. [5] proposed a framework for supporting collaboration in multiple display environments called IMPROMPTU. It enabled team members to

discuss software development tasks through shared displays. Salinger et al. [6] presented Saros which was an eclipse plug-in for collaborative programming activities between distributed parties.

Since the Semantic Web emerged, ontologies have been widely used as a means of providing the semantics to support the retrieval information based on the intended meaning rather than simply match the search terms [7]. Since then, they have now applied to several fields including 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 as well as provide semantics and mechanisms for communication and structuring of knowledge. In addition, ontologies also have a great potential for analysis and design of complex object-oriented software systems by using them to create object model for object-oriented software engineering [8].

In multi-site software development environment, ontologies have played an important role to support working context. There are several tools, techniques, models and best practices that utilizing ontologies to facilitate collaboration, communication, project knowledge management including software engineering processes activities and it is proved that ontologies can bring benefits such as communication within remote teams, knowledge sharing and effectiveness in information management [9].

Wongthongtham et al. [10] introduced the "Software Engineering Ontology" which was 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 multi-site development environment. They defined the software engineering ontology as a formal, explicit specification of a shared conceptualization in the domain of software engineering [11]. Formal implies that the software engineering ontology should be machine-understandable to enable a better communication and semantically shared knowledge between humans and machines (i.e. in the form of software application or software agents). Explicit implies that the type of software engineering concepts and their constraints used are explicitly defined. Shared shows that the consensual knowledge of software engineering is public and accepted by a group of software engineers. Conceptualization implies and abstract model of having identified the relevant software engineering concepts.

The software engineering ontology comprises two sub-ontologies: the generic ontology and the application specific ontology [11]. The generic ontology contains concepts and relationships annotating the whole set of software engineering concepts which are captured as domain knowledge. Application specific ontology defines some concepts and relationships of software engineering for the particular software development project captured as sub domain knowledge. In addition, in each project, project information including project data, project understanding, and project agreement that specifically for a particular project need are defined as instance knowledge. Remote software teams can access software engineering knowledge shared in the ontology and query the semantic linked project information to facilitate common understanding and consistent communication.

However, the current software engineering ontology has the same passive structure as other ontologies [12]. 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.

In this paper, we propose a novel approach that can offer active support to the software engineering ontology users. Two main key technologies will be used which are agent technologies and recommendation systems.

This paper is organized as follows. In section 2, we discuss the motivation of this work. Background and related work are reviewed in section 3. In section 4, we propose our conceptual framework. Section 5 demonstrates some scenario examples of multi-agent based recommender system providing active support through software

engineering ontology. Finally, the conclusion and future work are discussed in Section 6.

2 MOTIVATION

The potential benefits of this work are significant as follows.

2.1 Report in the literature [13] mentions that not all globally distributed projects can benefit from working in the global context. Twenty to twenty-five percent of all outsourcing relationships fail within two years and fifty percent fail within five years. One of the main reasons for this failure rate is the communication barrier across multiple sites. The proposed work is intended to support effective communication within projects in order to reduce the failure rate of geographically distributed software development projects.

2.2 The proposed recommender approach integrating with automatic reasoning capacity of autonomous software agents will provide active support to multi-site software teams by recommending useful project information and solutions for project issues that arise as experts.

2.3 With the proposed framework, software companies can take advantage of developing software in a global context, the benefits of which are: reduction in development costs, access to a large skilled labor pool, effective utilization of time zones etc. This will enable them to be more competitive when bidding in the software development market.

3 BACKGROUND AND RELATED WORK

3.1 Agent Technologies

The evolution of Web technologies started from Web 1.0 which was considered as the traditional information web. Then it moved to Web 2.0, focusing on user-generated contents or community-oriented information gathering. However, with the problem of the substantial amount of data and unstructured content generated, web users have difficulty searching for the contents. Therefore, Web 3.0 also known as Semantic Web has emerged to alleviate this issue. The underlying structure is that data should be well-organized to support information exchange and enable a machine or software agent to understand, process and reason to produce a new conclusion. Web 3.0 is the combination of existing Web 2.0 and the Semantic Web which integrates ontology, intelligent agent, and semantic knowledge management together [14].

A software agent is a computer program that has relatively complete functionality and cooperates with others to meet its designed objectives [15]. The other characteristic of an agent is its capability of flexible and autonomous action in the environment where it is situated [16]. An agent is also active, task-oriented and is capable of decision-making [17].

Multi-agent system (MAS) consists of multiple agents communicating and collaborating with each other in one system in order to achieve goals [17]. It is used to solve complex problem that cannot be done by individual agent. MAS is appropriate for domains that are distributed such as global manufacturing supply chain network [18, 19], distributed computing [20, 21], software collaborative developing environment [22, 23], etc. It can increase the efficiency and effectiveness of working groups in distributed environments. Implicit [24] was a multi-agent recommendation system for web search intended to support groups or a community of people with similar but specific interests. Romero, Viscaino and Piattini [25] introduced a multi-agent simulation tool to support training in global requirement elicitation process. They used 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 [26]. Each agent will collaborate with other agents, so they must be able to communicate and understand messages from one another. MAS has been widely used in several researches to support software collaborative systems in distributed software development environment. For example, (Col_Req) was the multi-agent based collaborative requirements tool that supported requirement engineers for real time systems during the requirement engineering phase [27]. Distributed stakeholders (e.g. software

teams, customer, etc.) worked on the system for collaborative acquisition, navigation and documentation activities.

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. There are several existing researches that integrate the use of ontologies and MAS. Paydar and Kahani [28] introduced a multi-agent framework for automated testing of web-based applications. The framework was 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 was proposed by Lee and Wang [29]. The multi-agent system consisted of three main agents interacting with one another to achieve the goal of effectively summarizing the evaluation reports of the software engineering process regarding CMMI assessment. The CMMI ontology was developed to represent the CMMI domain knowledge. This research did not cover other knowledge areas of the software engineering domain but it specifically focused on the software engineering process with respect to CMMI assessment only. The integration of two promising technologies in software engineering which were multi-agent system and Software Product Lines (SPL) was addressed in [30]. It provided the solution of producing higher quality software, lower development costs and less time-to-market by taking advantage of agent technologies. The ontology was used for modeling the Multi-agent System Product Lines (MAS-PLs) and was represented by UML class diagrams. MADIS [21] was a multi-agent design information system aiming at supporting the distributed design process by managing information, integrating resources dispersed over a computer network and aiding collaboration processes. The MADIS ontology was developed to formally conceptualize the engineering design domain to enable knowledge sharing, reuse and integration in a distributed design environment. Monte-Alto et al. [31] proposed a multi-agent context processing mechanism called ContextP-GSD (Context Processing on Global Software Development) that utilized contextual information to assist user's task during the software development project. This project applied agent-based technology to process contextual information and support human resource allocation. OntoDiSen was an application ontology exploited in this system representing GSD contextual information. Although this research aimed at facilitating the collaboration and

Table 1. Review of some multi-agent system applications

Methodologies/ Tools/Authors	Purpose of using multi-agent systems	Focus	Make use of ontologies	
Implicit	Supporting web search for groups or communities of people	Web search		✓
Romero et al.	Being a simulation tool to support training in global requirements elicitation process	E-learning		✓
(Col_Req)	Supporting software engineers during the requirements engineering phase for collaborative acquisition, navigation and documentation activities.	Requirements engineering activities		✓
Paydar and Kahani	Performing automated test process	Software testing	✓	
Lee and Wang	Summarizing the evaluation reports for the CMMI assessment	CMMI assessment	✓	
Nunes et el.	Supporting mass customized software production	Software product lines	✓	
MADIS	Supporting the distributed design process by managing information, integrating resources dispersed over computer network and facilitating collaboration processes.	Distributed collaborative engineering design	✓	
ContextP-GSD	Processing context information and supporting human resource allocation	GSD contextual information	✓	

coordination in global software development environment and used ontology to define semantic information which was quite similar to our proposed work, it focused only on contextual software engineering information, not the whole software engineering domain knowledge.

The summary of the reviewed multi-agent system applications is presented in Table 1. It is evident that many researches have exploited multi-agent technology in various applications and a number of them utilizes 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.

3.2 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 [32]. The initial academic work on implementing recommendation systems was first conducted in the mid-1990s. Park et al. [33] 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 showed that publications related to this topic had increased significantly, especially after 2007 and also extended to fields other than movies and shopping. They conclude from their review that 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.

Recommendation systems are normally classified based on how recommendation is implemented as following [34].

- Content-based approach recommends items which resemble the ones that a specific user formerly preferred.
- Collaborative filtering approach recommends items to the users based on the similarity between users.
- Hybrid approach combines collaborative filtering and content-based techniques.

Content-based approach has the main strength that it can provide accurate recommendations to a user without knowing others' preferences. However, due to the syntactic similarity metrics employed, it suffers from the overspecialization problem whereby only those items similar to those the user already knows are recommended [35].

Collaborative filtering approach mimics human behavior for sharing opinion with others. It offers recommendation based on not only user's interest but also on others' preferences; therefore, it can produce more unexpected or different items than content-based technique. However, collaborative filtering also suffers from some severe drawbacks such as data sparsity, gray sheep, and synonymy [34]. The data sparsity issue means that a recommender is unable to make meaningful recommendations because of an initial lack of ratings such as new user and new item. The gray sheep problem refers to the users whose interests do not match any group of people so they do not benefit from this approach. The synonym challenge causes poor quality of recommendations because the collaborative filtering approach cannot discover items that have different names but have the same meanings.

From critical weaknesses of content-based and collaborative filtering recommender systems, hybrid approach has been introduced by combining these two approaches to resolve certain problems associated with those two approaches. Nevertheless, hybrid recommender system is still limited by the syntactic matching but semantic mismatching [35]. The syntactic matching techniques relate items from common words not from their meaning, so the result of recommendations is 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) [36]. 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 [37]. Various applications in several fields have been proposed which include a semantic reasoning mechanism in their recommendation systems, for instance, Blanco-Fernández et al. [38] presented a methodology to overcome the overspecialization problem and improve the effectiveness of content-based recommendation approaches by applying semantic descriptions of the items and including semantic reasoning technique in them. They claimed that the proposed methodology had the potential to enhance the quality of recommendations better than the traditional recommendation systems did and it could be applied in various domains. This model was realized through the implementation of the prototype, AVATAR, a recommender system of personalized TV content. Cantador et al. [39] explored a model of an enhanced semantic layer for hybrid recommendation systems. Different methods were integrated for different purposes in order to improve the accuracy and quality of recommendations such as ontology-based knowledge representation concept, spreading activation algorithm and three recommendation techniques which were personalized, semantic context-aware and content-based collaborative recommendation systems. The authors illustrated the use of their methodology in a news recommendation system, News@Hand. An ontology-based semantic recommendation for programming tutoring system called Protus 2.0 was a research in education domain proposed by [40]. It was an adaptive and personalized web-based tutoring system that used recommendation approaches during the personalization process. Web Ontology Language (OWL) was used to represent context knowledge while Semantic Web Rule Language (SWRL) was exploited to deal with semantic reasoning. Although semantic-based recommendation systems were employed in several domains, none of them was specifically intended to create recommendations to manage queries or project issues raised in software development teams through the use of ontologies in software engineering.

3.3 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 [41]. RSSEs have become an active area of research for the past several years and they have been proven to be effective and useful to software developers to cope with the huge amount of information when they are working on software projects. They can provide recommendations for development information (i.e. code, artifacts, quality measures, tools) and collaboration information (i.e. people, awareness, status and priorities) [42].

Here are some reviews of recommendation systems that focus mainly on recommending expert or relevant people. Codebook [43] was a social network web service that linked developers and their work artifacts and maintains connections with other software team members. Conscius [44] was a recommender system that located a source code expert on a given software project by using communication history (archived mail threads), source code, documentation and SCM change history. Steinmacher et al. [45] proposed a recommendation system that could assist newcomers to discover the expert who had the skill matching the selected issue to mentor the regarding technical and social aspects of a particular task. Ensemble was a recommender application that helped software team members to communicate in the current works by recommending other people when developer does any updates on related artifacts such as source code or work items [46]. These recommendations could help to locate related people and save time when seeking their expertise during software development process. They increased the accuracy of recommendations by exploiting user context, workspace information and social information.

Some other RSSEs focused on supporting developers while they were coding or debugging program. Fishtail was a plugin tool for the Eclipse IDE which automatically recommended source code examples from the web to developers that were relevant to their current tasks [47]. Cordeiro et al. [48] proposed a context-based recommendation to support problem-solving in software development. They developed a client/server tool to integrate recommendation of question/answering web resources in the developer's work environment to provide automatic assistance when the exception errors occured. DebugAdvisor [49] was proposed as a search tool for debugging which supported fat query, a query with all contextual information of the bug issue. Developers could do a bug report search from multiple software repositories with a single query. The system returned a bug description raked list that matched the query and then used it to retrieve recommendation of the related artifacts such as source code and functions from the generated relationship graph. Jaekel et al. [50] developed a Semantic Helper component which was one of the modules of the FACIT-SME project, a three-year project intended to assist IT SMEs to select and use quality business process models and software engineering methods in their software development projects. Dhruv [51] advised software developers on relevant software artifacts and bug reports. Semantic web technology was explored in this research in order to facilitate problem-solving in the open-source software community. It exploited ontologies to identify where related artifacts were located and their description including relevant bug information. A Semantic Helper component aims was intended to assist other components by filtering information and doing automatic matching between the models which were stored in semantic format in FACIT-SME repositories. This recommender system also provided ranking lists of the most relevant models from a given query.

All the described applications had 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 systems except for Dhruv and Semantic Helper used traditional knowledge representation and syntactic matching techniques so they lacked integrated and shared information and could not support a semantic reasoning mechanism.

4 CONCEPTUAL FRAMEWORK

This section presents the proposed conceptual framework of multi-agent based recommender approach for active software engineering ontology. The users of software engineering ontology will be provided intelligent support to access and recommend knowledge and project information captured in the software engineering ontology. Intelligent agents will work collaboratively to facilitate the software project teams who are working together irrespective of their geographical location. The aims of the multi-agent based recommender system are:

1) to extract and convey semantic rich project information described in the software engineering ontology to team members,
2) to manage project issues that arise by utilizing the agent's ability of automate reasoning,
3) to recommend solutions for any project issues as experts on a constant and autonomous basis,
4) to support work of adding semantic project information automatically into the software engineering ontology instantiations during the refinement process.

The proposed conceptual framework of multi-agent based recommender system is shown in Figure 1. It comprises four types of agents with the short descriptions of their roles as following.

1) User agents

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

2) Semantic recommender agent

- Recommend tentative solutions including affected software artifacts and users.
- Work with ontology agent to make a decision based on knowledge in software engineering ontology.
- Notify affected agents in case of ontology update.
- Coordinate with evolution agent in case of unresolved issues/queries.

3) 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.

4) 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

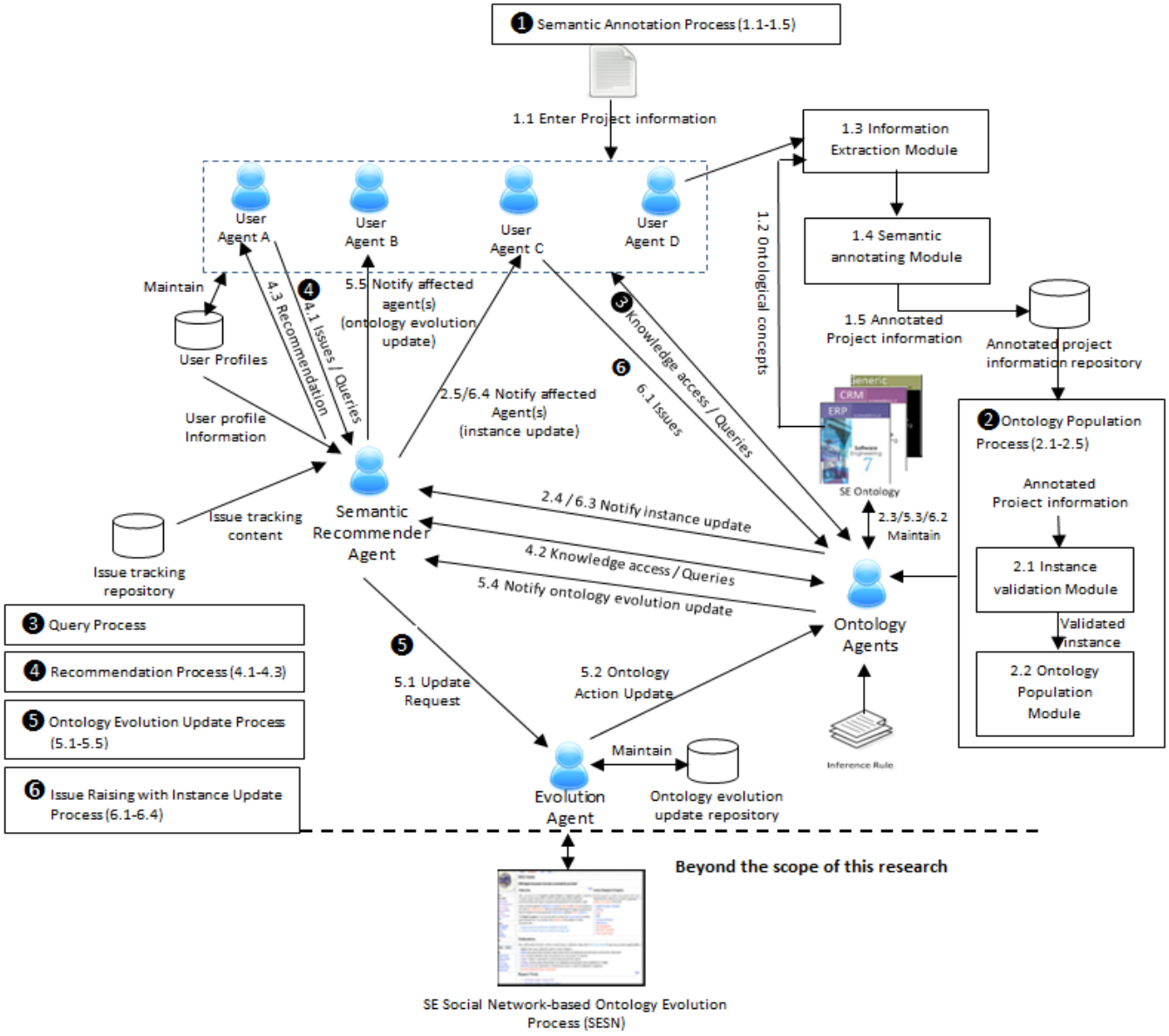

Figure 1. Multi-agent based recommender system conceptual framework

ontology evolution process.
- Notify ontology agents for update from SESN process.

The agents will work collaboratively throughout six processes as following.

1) Semantic Annotation Process

As mentioned, in the software engineering ontology, 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.

This process starts from user agents receiving project information from software team members. User agents will perform information extraction process with references to classes and instances in the software engineering ontology retrieved by ontology agents. The RDF annotation is then generated by semantic annotating module and stored in the repository containing the annotation of other project information.

2) Ontology Population Process

Ontology population is a process of adding new instances into an existing ontology. When project information is successfully annotated, it is ready to populate into the software engineering ontology.

In this research, ontology agents will be responsible for managing ontology population process. The annotated project information is identified as candidate ontological instances and will be validated for the consistency between incoming instances and those already stored in the ontology. It is then inserted into the software engineering ontology as new instances.

3) Query Process

User agents will send their queries to ontology agents. Ontology agents will retrieve and provide information from the software engineering ontology in accordance with their queries.

4) Recommendation Process

User agents will send their issues or requests to the semantic recommender agent. The recommender agent then cooperate with ontology agents to make a recommendation based on knowledge explicitly described in the software engineering ontology and other resources, e.g. user profiles or issue tracking systems. Semantic recommendation techniques will be employed during the recommendation process to improve the accuracy of recommendation and to provide the tentative solutions as well as the most relevant knowledge according to user request.

5) Ontology Evolution Update Process

In case that the 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 Software Engineering Social Network System (SESN) for the ontology evolution process. Nevertheless, this is beyond the scope of this research but more information can be found in [52] and [53]. When the evolution process is completed and agreement regarding changes has been reached, the evolution agent will notify ontology agents to merge these concepts with the existing software engineering ontology. When ontology agents complete the update, it will tell the recommender agent to notify all affected agents. This change will cause some particular concept and relationship to be adjusted and leads to the change of generic concepts in the

ontology. This is called 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.

6) Issue Raising with Instance Update Process

This process is different from ontology evolution update process. Ontology evolution update process is a process of an evolution at concept level that changes will be made to the underlying software engineering domain knowledge while instance update process is a process of an evolution at instance level that deals with changes in refinement process or in the conceptualization. This process starts from software team member raises an issue to his personal user agent to make a change of instance in the software engineering ontology. Ontology agents will check any instance, component, or people who will be affected from this change and notify the user. He or other members can propose their opinions to the change until the final agreement has been discovered. Ontology agents will then update related instance in the software engineering ontology repository and inform the semantic recommender agent about the update. The recommender agent will notify only those team members who should be advised about the changes and their effects.

5 SCENARIO EXAMPLES OF MULTI-AGENT BASED RECOMMENDER SYSTEM PROVIDING ACTIVE SUPPORT THROUGH SOFTWARE ENGINEERING ONTOLOGY

Here are some scenarios that can explain how the proposed system works. Suppose that Globeware Company is a US multinational company which has three software development sites located in US, Australia, and India. They are currently working on a mobile application project. All requirement gathering and software specification are done in US while software design and implementation are done in Australia and India. Globeware utilizes the agent-based recommendation system for software engineering ontology framework in this project to facilitate effective remote communication and coordination. The software engineering ontology instantiations for this project have been derived from populating software project information, project agreement, and problem domain from each phase in SDLC which are mapped into the concepts defined in the software engineering ontology. Here are some examples showing how this methodology can provide active support to team members when working on software development project.

First example: Member A 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 Figure 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 Figure 1 – ❷ontology population process). The recommender agent will take responsibility for notifying all affected agent(s) about this ontology instance update.

Second example: Member B is a new member who has just joined this project as a developer. He would like to learn more about project information such as output from the design phase that only relates to his work and catch up with the current status of the project. He can query ontology agents via his user agent to access project information and status. The agent will autonomously consider retrieving only particular project information stored as instance knowledge in the specific software engineering ontology that is related to his work so it assists him to start working quickly

with the most relevant and precise situational knowledge (refer to Figure 1 –❸query process). If he doubts the output from the design phase, he can raise a query or an issue through his user agent who will communicate with the recommender agent to reason knowledge published in ontology repository to find a possible solution or recommend the most suitable person who can clarify his issue (refer to Figure 1 –❹recommendation process).

Third example: Member C finds out that there is a bug in the new released system so he informs his user agent. Before the bug issue is filed, the recommender agent and ontology agents will try to locate related problems from the project issue tracking system based on its associated concepts defined in the software engineering ontology and its instances. The benefit is to avoid a bug duplicated report from other developers which may create confusion and unnecessary information overload. Ontology agents will then attempt to link the bug symptoms to related software artifacts that are all annotated using the software engineering ontology in order to help the developer quickly diagnose which part of the software artifacts might be causing the problem. Additionally, before the developer fixes the bug, Ontology agents will inform him of the classes or components that might be affected. Furthermore, with a full record of mappings between previously reported bugs and people who resolved those bugs, the recommender agent will be able to recommend potential people to consult or to resolve some particular bug issue (refer to Figure 1 –❹recommendation process).

Fourth example: Member D 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. He or other members can propose their opinions by working on the diagram itself and do tracked changes. Ontology agents 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 agents 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 software development can proceed more clearly and effectively in the multi-site environment.

6 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. The roles of four types of software agents are analyzed and identified. The interaction between software agents and ontology within collaboration framework are defined into six processes. This work is intended to facilitate effective communication and coordination for remote software development teams to reduce the unsuccessful rate of multi-site software development project.

For future work, semantic annotation will be implemented to annotate project information such as user requirements, source codes, etc. and then populate it into the software engineering instantiations. We will then design a semantic-based recommendation system based on the software engineering ontology and integrate them with multi-agent implementation. We will evaluate and validate our work in accordance with a framework for evaluation in design science research addressed by Venable, Pries-Heje and

Baskerville [54]. The prototype will be developed and evaluated by two groups of multi-site software development teams in order to obtain feedback to measure the usability and effectiveness of the system to solve the problem. In addition, to evaluate the system performance, simulation will be used by executing a prototype with artificial data.

7 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.

8 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. S. Islam, M. M. A. Joarder, and S. H. Houmb, "Goal and risk factors in offshore outsourced software development from vendor's viewpoint." pp. 347-352.
4. 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.
5. J. T. Biehl, W. T. Baker, B. P. Bailey, D. S. Tan, K. M. Inkpen, and M. Czerwinski, "Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy, 2008, pp. 939-948.
6. S. Salinger, C. Oezbek, K. Beecher, and J. Schenk, "Saros: an eclipse plug-in for distributed party programming," in Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, Cape Town, South Africa, 2010, pp. 48-55.
7. T. S. Dillon, E. Chang, and P. Wongthongtham, "Ontology-based software engineering- software engineering 2.0." pp. 13-23.
8. Y. Blanco-Fernández, J. J. Pazos-Arias, A. Gil-Solla, M. Ramos-Cabrer, M. López-Nores, J. García-Duque, A. Fernández-Vilas, and R. P. Díaz-Redondo, "Exploiting synergies between semantic reasoning and personalization strategies in intelligent recommender systems: A case study," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2371-2385, 2008.
9. A. Borges, #233, r. Soares, S. Meira, Hil, #225, r. Tomaz, R. Rocha, and C. Costa, "Ontologies supporting the distributed software development: a systematic mapping study," in Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, Porto de Galinhas, Brazil, 2013, pp. 153-164.
10. 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.
11. P. Wongthongtham, E. Chang, T. S. Dillon, and I. Sommerville, "Ontology-based multi-site software development methodology and tools," *Journal of Systems Architecture*, vol. 52, no. 11, pp. 640-653, 2006.
12. P. Wongthongtham, T. Dillon, and E. Chang, "State of the art of community-driven software engineering ontology evolution." pp. 1039-1045.
13. C. Ebert, "The dark side: challenges," *Global Software and IT*, pp. 19-25: John Wiley & Sons, Inc., 2011.
14. H.-C. Chu, and S.-W. Yang, "Innovative semantic web services for next generation academic electronic library via web 3.0 via distributed artificial intelligence," *Intelligent Information and Database Systems*, Lecture Notes in Computer Science, pp. 118-124: Springer Berlin Heidelberg, 2012.
15. H. Qingning, Z. Hong, and S. Greenwood, "A multi-agent software engineering environment for testing Web-based applications." pp. 210-215.
16. N. R. Jennings, "On agent-based software engineering," *Artificial Intelligence*, vol. 117, no. 2, pp. 277-296, 2000.
17. V. N. Marivate, G. Ssali, and T. Marwala, "An intelligent Multi-Agent recommender system for human capacity building." pp. 909-915.
18. J. Jiao, X. You, and A. Kumar, "An agent-based framework for collaborative negotiation in the global manufacturing supply chain network," *Robotics and Computer-Integrated Manufacturing*, vol. 22, no. 3, pp. 239-255, 2006.
19. W. T. Goh, and J. W. P. Gan, "A dynamic multi-agent based framework for global supply chain." pp. 981-984 Vol. 2.
20. Z. Zhong, J. D. McCalley, V. Vishwanathan, and V. Honavar, "Multiagent system solutions for distributed computing, communications, and data

integration needs in the power industry." pp. 45-49 Vol.1.
21. C. Chira, "A multi-agent approach to distributed computing," *Computational Intelligence Report No*, vol. 42007, 2007.
22. A. Y. AHamo, and M. A. Aljawaherry, "Constructing a collaborative multi-agents system tool for realtime system requirements," *International Journal of Computer Science (IJCSI)*, vol. 9, no. 4, 2012.
23. Z. Chuan, "A software collaborative developing environment based on intelligent agents." pp. 1-4.
24. A. Birukou, E. Blanzieri, and P. Giorgini, "Implicit: a multi-agent recommendation system for web search," *Autonomous Agents and Multi-Agent Systems*, vol. 24, no. 1, pp. 141-174, 2012/01/01, 2012.
25. 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.
26. V. Iordan, A. Naaji, and A. Cicortas, "Deriving ontologies using multi-agent systems," *WSEAS Transactions on Computers*, vol. 7, no. 6, pp. 814-826, 2008.
27. K. Giri, "Role of ontology in Semantic web," *DESIDOC Journal of Library & Information Technology*, vol. 31, no. 2, 2011.
28. S. Paydar, and M. Kahani, "An agent-based framework for automated testing of web-based systems," *Journal of Software Engineering and Applications*, 2011.
29. 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.
30. 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.
31. 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.
32. 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.
33. 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.
34. A. Y. Hamo, and M. A. Aljawaherry, "Constructing a Collaborative Multi-Agents System Tool for Real Time System Requirements," *International Journal of Computer Science*, vol. 9, 2012.
35. "Semantic Annotation, Indexing, and Retrieval."
36. Q. Gao, J. Yan, and M. Liu, "A semantic approach to recommendation system based on user ontology and spreading activation model." pp. 488-492.
37. 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.
38. 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.
39. 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.
40. 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.
41. M. Robillard, R. Walker, and T. Zimmermann, "Recommendation systems for software engineering," *Software, IEEE*, vol. 27, no. 4, pp. 80-86, 2010.
42. H. J. Happel, and W. Maalej, "Potentials and challenges of recommendation systems for software development." pp. 11-15.
43. A. Begel, K. Yit Phang, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories." pp. 125-134.
44. 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.
45. I. Steinmacher, I. S. Wiese, and M. A. Gerosa, "Recommending mentors to software project newcomers." pp. 63-67.
46. 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.
47. 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.
48. J. Cordeiro, B. Antunes, and P. Gomes, "Context-based recommendation to support problem solving in software development." pp. 85-89.
49. 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.
50. 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.
51. A. Ankolekar, K. Sycara, J. Herbsleb, R. Kraut, and C. Welty, "Supporting online problem-solving communities with the semantic web." pp. 575-584.
52. A. A. Aseeri, "Lightweight community-driven approach to support ontology evolution," School of Information Systems, Curtin University, 2011.
53. N. Kasisopha, and P. Wongthongtham, "Semantic wiki-based ontology evolution." pp. 493-495.
54. J. Venable, J. Pries-Heje, and R. Baskerville, "A comprehensive framework for evaluation in design science research," *Design Science Research in Information Systems. Advances in Theory and Practice*, Lecture Notes in Computer Science, pp. 423-438: Springer Berlin Heidelberg, 2012.