Indian Journal of Education and Information Management, *Vol 3 (4), 6–12, April 2014*
ISSN (Print): 2277-5366
ISSN (Online) : 2277-5374

Adding Semantic Aspects to Social Software Engineering for Improve Software Debugging and Codification Software Engineer's Educational Program

Nematallah Ghanavati*

Department of Computer, Mahshahr Branch, Islamic Azad University, Mahshahr, Iran;
ghanavatin@gmail.com

Abstract

Quality assurance in software development is still a challenging process. There are a lot of methods to improve software quality such as software quality assurance techniques, or conducting formal technical reviews. In this research, we try to design and implement a social software and reuse ontologies and integration of some of them and provide an environment to store and analyze the feedback related to functionality of individuals involved in the project, detect the errors and defects reported during the software development process. It also provides an opportunity to share the data with the professionals and get their assistance and contributions. With their knowledge about quality assurance, methods for debugging and an appropriate training curriculum to improve the performance of the individuals involved in the project can be developed, which enhances the verification level and validation of the software which inturn leads to the improvement of its quality.

Keyword: Software Quality Assurance, Semantic Web, Ontology, Debugging

1. Introduction

Software development is usually a social activity that involves people from different domains. They have their particular backgrounds and act under different conditions. The 'social aspects' of Software Engineering is discussed for a long time. But it has been considered again with a new approach these days. The new interest is mainly driven by recent phenomenon in online communities and social interactions on the web that is often known under titles such as 'social networking', 'social software' or 'Web 2.0'[14]. Understanding these phenomena and ways of their utilization to increase the support of social interaction in Software Engineering is currently a main goal in different research efforts.

In this context, the term 'Social Software Engineering' (SSE) has been devised to emphasize the importance of social aspects in software development. One of the main observations in the SSE is that the concepts, principles, and technologies built for social software applications are applicable to software Engineering. Social Software Engineering focuses on the development of systems in highly uncertain domains, with evolving goals, frequent changes and much user involvement. It is focused on community-centered collaboration and uses online environments to share artifacts and knowledge about a software product.

In this research, we based on the social software engineering idea and its combination with pair programming practice in Xp, that its use will improve the quality

*Author for correspondence

of software, provide a social environment so that more specialists or stakeholders can provide their advice and assistance together in your technical tasks with the easiest way, so that designer and programmer could share your product likes models and programming code with specific people for technical reviews and debugging and receive their comments or suggestions. In fact, in this social environment instead of scheduling a technical review in the time period, continuous review of them is possible as creation of products (design or code). So its advantage is early detection of errors and finally produce a high quality product. The role of pair programming on Production of high quality software and high productivity products has been studied in research conducted by the William and Kessler[19]. Also in this project, in order to achieve more efficiency of the designed social environment, performe analysis of the data collected in the environment and get ideas from the experts. So based on them, codification of appropriate learning resources and curriculum of courses for individuals involved in the project can be developed in order to improve their job quality. People can prevent unauthorized persons from gaining their products based on privacy approaches in social networks, the opportunity for sharing their products are limited to certain individuals or groups. The implementation of such decisions as well as access to the discussions would be possible by project managers. Because, firstly they are required such data to perform their management tasks. Second, given that the project manager and team leaders usually are indirect contact with the people involved in the project, they can act as the most appropriate reference to determine the educational content for your staff.

In addition, this environment makes it possible to know programmer or designer that his work reviewed and is discussed by others more widely, to focus more on their work. Notably, some of the software engineers argue that the high participation can waste in debugging and slow the project progress. But the environment is designed so that developers just by a simple copy of their works or products in an edit box, use others help to solve their problems faster in most cases.

In this system, considering the entities (resources) such as errors specifications, discovered defects, training courses, proposed topics, requirements list and stakeholders, we can obtain the necessary feedback during the formal technical reviews, quality and configuration audits, documents reviewing, and different SQA activities. To determine whether the expected functions of the software are properly prepared, and has provided needs of customers and users or no.

Naturally, after finding bugs and defects, they must be corrected. But correcting a problem itself can cause additional errors and consequently will cause more disadvantage. Here, will do items such as finding bugs causes, finding other bugs that may arise due to this debugging, and adoption appropriate debugging technique. In this regard, in this project we have tried to through the design of a semantic web environment, In addition to semantic description of errors and defects in the project provide a social environment to establish possibility of participating wide variety of people including analysts, application developers, customers and other experts in the field of project area in this environment. So that, the role of human factors will become more prominent in the debugging process and generally software process improvement. As already been demonstrated, any method, tools and attempt for debugging without the presence of other people which often act as powerful allies, will not lead to the desired result. As mentioned earlier, using pair programming practices that have been proposed as part of the xp programming models and social programming[4] leads to improve software quality. So we provides facilities in this social environment so effectively uses capabilities of experts and stakeholders in the process of debugging and overall software quality assurance.

2. Software Quality Assurance

Software quality assurance is an activity in throughout the software process and its emphasis is on maintaining the quality of the software. To reduce duplication work in software engineering activities. This leads to lower costs and accelerate delivery time to market. Software quality assurance includes items such as Technical reviews, test strategies, methods and tools, software product management and ensuring compliance with software development standards in our designed environment, collect data about errors and defects and then interrelated and analyzed them to determine which software engineering activities or debugging method is best to eliminate them. In fact it is considered one of the most important SQA[a] tasks. Technical reviews are considered

[a] *Software quality assurance*

as the most effective mechanism to find errors early in the software process. The main purpose of a technical review is find errors before being passed onto the next activity or reach the end user.

Other things that are done in this research is that according to feedback from performance of software engineers and programmers and also took assistance from other knowledge, particularly project managers and individuals are directly involved in project activities and supervising the performance of designers and programmer, Trying to provide solutions such as advice and proper training codification they will need. Indeed, here the aim is development the headlines of training courses more accurate according to individuals capabilities, weaknesses and training requirements that have been obtained from their feedbacks in the project. Until finally can be seen the enhance of effectiveness software projects individuals efficiency and increase their productivity.

3. The Role of Ontologies in Software Engineering

So far, many efforts and activities have been conducted on the use of ontologies in various phases of software engineering. Omghas provided a new technology called the ODM[b] which includes a series of Metamodel to establish a mapping between the semantic web languages structures (such as RDF or OWL) and object-oriented modeling languages structures such as uml.

Other idea that can noted in this field is ontology-based architecture[18] (ODA[c]). Ontologies have the different application areas in Software Engineering. Ontologies are used during the software development process and during run-time. In fact, they are based on two categories. As an example of the first category can be cited the conceptual model of the problem domain, formal description of software artifacts such as components and definitions of mappings between different modeling languages. At run-time, ontologies are used as part of the program logic or as a conceptual layer that supports system dynamic behavior such as adaptations or composition of web services and automatic discovery. As well as, they differ between approaches that use ontologies to model a domain or system's context and approaches where ontologies are part of the system or development infrastructure itself.

[b] *Ontology Definition Meta Model*

[c] *Ontology Driven Architecture*

In this research ontologies are used to support the development of community-oriented quality assurance.

3.1 Related Research

Gerald M. Weinbergis, one of the earliest promoters of software development as a social activity in the 1971[8]. He was focused many of his works on engineering software development processes from the stand point empowering people. In other words, he believes software development is a human driven activity. Because effective communication can play a key role in software development[6,7]. Luiand Chan also have human-centered approach to software engineering. In[10] they offer different ways to collaborative programming approach and show how to combine and coordination these methods. So they can be used to fixing common problems of software management such as motivating software developers, discover the solution patterns, managing software teams, and deliver IT projects.

So far, several approaches have been proposed in which the ontology has been used in software development. Existing approaches defining its ontology, provide reuse of ontologies related to software, or provide a framework or infrastructure that helps for developing the ontology as a software product. It is also kind of community oriented relationship and collaboration that similarly are implemented, when intending to use dominant patterns at various online environment[15].

In this project, our work closely associated to applications of semantic wikisin software engineering. Semantic wikis provides an architecture for the establishment possibility of participation and simultaneously structure preservation and recognition. They can consider as an ideal tool for the software engineering community. However, this area of research is in its in fancy and young. So far, there are only a limited number of approaches that have been tried to use the Semantic Wiki in software engineering[13,20].

Recently in this context, in the rise project is being completed some tasks. Decker and his colleagues proposed the idea of 'self-organized reuse'. That Software products in the semantic wiki it can be jointly create and structured[5]. Xiao et al.[20] present Galaxy Wiki, a wiki based environment in which programmers can collaboratively write source code, build projects, and debug defects.

Also in other studies Lohmann and Riechert have been proposed a semantic wiki based techniques in which have

been useda number of well-known ontologies to represent activities and products emerged in an online environment for community-oriented software engineering[12]. In research conducted by Bertramand colleagues[2] discussed a bug tracking social system through which helps customers, project managers, quality assurance personnel, and programmers to share knowledge and continuous communicating. In addition, investigation has also been conducted to design systems for improve awareness of the project condition and progress and their participants in inside and outside of the organization, Such as research performed by Kadenbach[9].

4. Ontologies for Community-oriented Software Quality Assurance and Debugging

In Debug wiki project has developed a web-based environment for community-oriented software quality assurance and debugging. That several features of social software and ontology is used to support collaboration between stakeholders who are geographically distant from each other. It aims is promote more direct interaction with the larger group of stakeholders, experts, designers and programmers in a collection and then the discussion, develop, and comment on the software errors and defects and related solutions with focus on software quality assurance. In Figure 1 shows an image of Debug Wiki project web environment where the user interface is conceptually divided into different parts and each section is described by a set of ontology concepts. In fact, these concepts are used to show the achievements related to each section. This web environment is realizable as on to wiki plugin. Through its, needed ontologies imported and have been integrated in a common upper ontology based on Semantic web languages RDF(s) and OWL[1].

5. Representing Bugs and Training Courses Metadata via Dublin Core

Errors, defects and training courses can be viewed and edited participatory by all registered users of the site. In fact, a bug can only be seen as information source with well-known properties, such as title, description, and programmer name that has created the bug.

A collection of fifteen often used properties for the description of information resources is defined by the

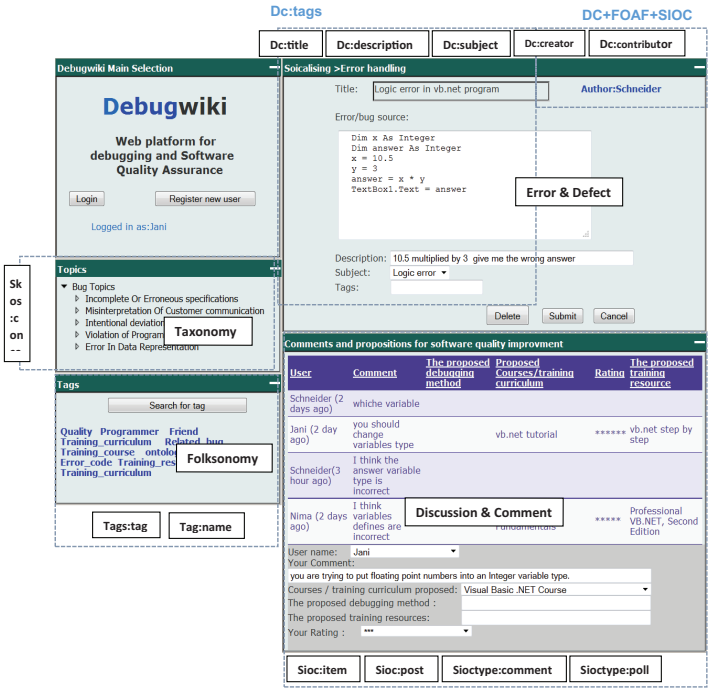

Figure 1. Web environments for quality assurance and community–oriented debugging.

'Dublin Core Metadata Element Set'. We used several of these properties to semantically annotate Bugs and training courses in our web environment including title, description, creator, contributor, subject, and source. Because such metadata is often valuable in the analysis, refinement and prioritization of Bugs and training courses (e.g., to get back to the cause(s) of a Bug), maintaining it across different tools is usually of high interest. Using Dublin Core instead to describe basic metadata a wide variety of tools that can be used for accessing and refining the Bugs, defects and training courses to all tools that are capable to read and interpret this standard, including non-CASE tools. Therefore through it, in the web software designed by us will be possible acquiring and refinement of existing bugs in the software development process easily. Also, through the Dublin Core will be provided storage and display some basic information to control debug operation and maintain their traceability and record the necessary information to describe the required educational details to improve the efficiency of the human factors involved in the project development. It should be noted, the metadata properties defined by Dublin Core are not sufficient to represent all information that might be captured about a bug in all possible cases. Nevertheless they provide at least a major metadata subset that can be valuably reused in Software Engineering.

To display the extracted data on RDF format should be used the appropriate vocabulary. Of course, in this project due to diversity of target data used several vocabularies to describe them.

6. Representation of Stakeholders, Developers and Professionals and Discussions via FOAF and SIOC

Here, we imported the FOAF ('Friend of a Friend') vocabulary for representation Personal information of Stakeholders, developers, specialists and their interrelations and also Expression of their social.

Communication for example, familiarity with other people involved in the project[1, 3]. In particular to achieve these objectives, we use the FOAF class agent and its subclasses Person, Group and Organization. However, further FOAF concepts might be linked in order to represent additional information about stakeholders, such as contact details or bugs assigned to specialists. As well as, the FOAF structure can be exported and visualized as social graph with appropriate tools. This enables new possibilities for social network analysis which maybe useful for Social Software Engineering[11]. We have imported the SIOC ontology in order to provide these online discussions. The account of each stakeholders will be displayed Via SIOC class User, comment is a subclass of Post and ratings are represented by the class Poll. Also Doap ontology has been used for semantic description of software projects.

7. Integration and Align Ontologies in the Upper Ontology

How Integration of semantic web ontologies with the designed upper ontology Seen in Figure 2. A scan be seen in this Figure, in this Web environment Possibility of Specialist participation will be provided in Different activities in the software development process using SIOC ontology. This Specialist can include project managers, stakeholders, Educational Consultants, etc. These people through your posts and tagging them, Attempt to discuss the bugs in project and offer solutions and Suggested training subheadings to improve their performance.

In this ontology, some Subtypes are defined to prioritize or rankings bugs in terms of importance. To provide the possibility for advanced reviews of problems.

8. Bugs and Specialists Classification via Skos and Tags

Skos ontology will be used for description and classification software bugs and educational courses information. In this web environment, this will be done in two different ways. In the first method, they can either be assigned to a concept of a pre-defined taxonomy or be equipped with an arbitrary number of freely chosen keywords (so-called 'tags'). These tags are aggregated to a 'folksonomy'[16] that is visualized as 'tag cloud'[16] in the user interface under the taxonomy tree (see Figure 1) and can be used for navigation and filtering.

Today's tagging mechanism was eagerly accepted, and it has become a significant part of many formal processes of software engineering[19]. Different types of tags are used by various stakeholders to help the classification and organize the elements or software products. Tags are used to support finding duties, express jobs and exchange

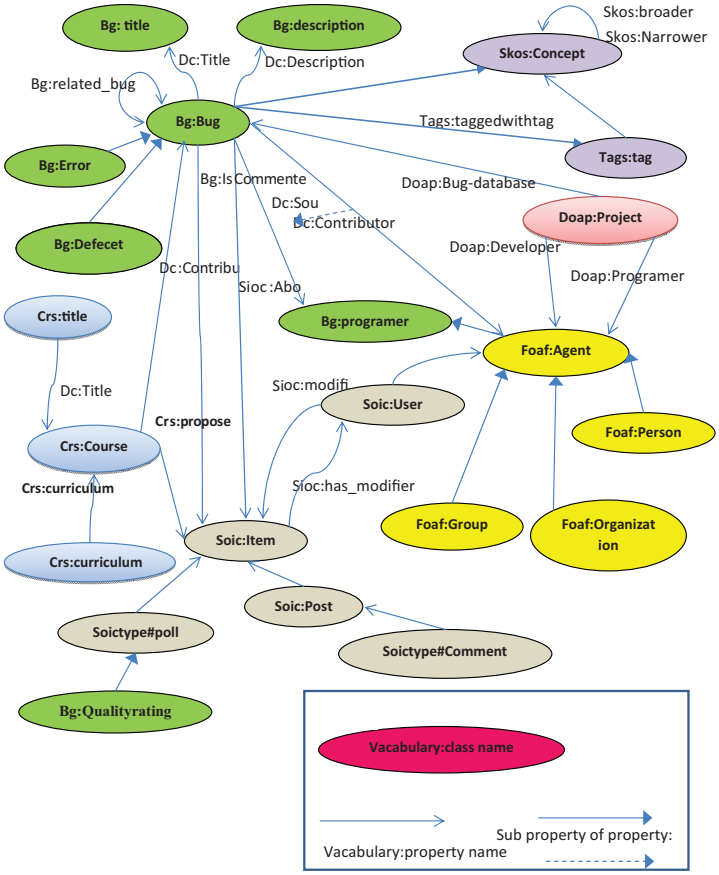

Figure 2. The main classes and properties of the designed upper ontology with integrated and aligned some semantic web ontologies.

information between users. The taxonomy is represented through the 'Simple Knowledge Organization System (SKOS)'. We use the broader and narrower properties to represent the hierarchical taxonomy structure and the class definition to add definitions for concepts. SKOS provides a facility to allow separately display A Classification of the common problems in the software development process. A standardized vocabulary like SKOS extends the range of projects and applications that can easily work with the taxonomies. Here, the tags of the folksonomy are represented by the 'TAGS ontology'. Since we have defined Tag as a subclass of SKOS Concept, a transformation between both is easily possible.

9. Conclusion

Many of software engineering aspects especially domain-independent ones are often already formally described by well-designed ontologies. Reuse of this ontologies can be valuable, as we tried to point out in this article. We have designed an online environment for community-oriented software quality assurance and debugging. This general idea is also applicable to other areas of software engineering.

The key advantage of reuse cross-domain ontologies is the providing interoperability with further tools. Including tools that have not been designed specifically for software engineering. This leads to new opportunities for utilize, enhance, and analyze artifacts and metadata. The modular structure of the presented Ontology with its clear conceptual separation additionally facilitates access to single, integrated ontologies. However, according to high degree of ontologies to describe their capabilities (such as typed links, classes and their characteristics, etc.), usually all instance data can be accessed by more generic Semantic Web tools (For example, ontology editors, inference engines, etc.) and Possibility of further processing is provided.

Reusing ontologies that have been developed by experts in a particular field, Can lead to avoid the extra work and modeling efforts and reduce the risk of incorrect interpretations and misconceptions. It is noteworthy, all aspects of software development cannot be covered by existing ontologies. The ontology presented in this research describes only a small subset of the many aspect of software quality assurance. While software quality assurance has many aspects. However, our goal is not to describe all aspects of the field. Rather development of an ontology to formally show the artifacts emerging in an online environment for society oriented software quality assurance. However this ontology can be considered to develop more comprehensive ontologies in the field of software quality assurance. Thus, future work includes an extension of our ontology by integrating further Semantic Web ontologies where applicable and define new domain-specific concepts into it.

10. References

1. Allemang D, Hendler J. Semantic web for the working ontologist: effective modeling in RDFS and OWL. San Francisco: Morgan Kaufmann; 2008.
2. Bertram D, Voida A, Greenberg S, Walker R. Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. CSCW'10. Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work; 2010. p. 291–300.
3. Brickley D, Miller L. FOAF Vocabulary Specification Version 0.98. 2010. Available from: ontogenealogy.com
4. Dabbish L, Stuart C, Tsay J, Herbsleb J. Social coding in GitHub: transparency and collaboration in an open software repository. Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, ACM; 2012. p. 1277–1286.
5. Decker B, Ras E, Rech J, Klein B, Hoecht C. Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis. Proceeding of SWESE'05; 2005.
6. Gerald MW. Becoming a technical leader: an organic problem–solving approach. Dorset House Publishing; 1986.
7. Gerald MW. Quality software management (volume 1 to 4) systems thinking, first order measurement, congruentaction, anticipating change. New York: Dorset House Publishing; 1994.
8. Gerald MW. The psychology of computer programming. Silver Anniversary Edition (anl sub edition). Dorset House Publishing Company Incorporated; 1998.
9. Kadenbach D, Kleiner C. Project awareness system–improving collaboration through visibility. Online Communities and Social Computing. 2013; 8029:164–173.
10. Lui KM, Chan KCC. Software development rhythms: harmonizing agile practices for synergy. John Wiley & Sons. 2008.
11. Kramer T, Hildenbrand T, Acker T. Enabling social network analysis in distributed collaborative software development. Software Engineering Workshops. 2009; 150(LNI): 255–266.
12. Lohmann S, Riechert T. Adding Semantics to Social Software Engineering: (Re) Using Ontologies in a Community-oriented Requirements Engineering Environment. Software Engineering (Workshops), 2010-subs.emis.de.
13. Lohmann S, Rashid A. Fostering Remote User Participation and Integration of User Feedback into Software Development. Proceeding of I-USED'08; 2008.
14. Porter J. Designing social web applications. New Riders; 2008.
15. Schummer T, Lukosch S. Patterns for computer-mediated interaction. Chichester, UK: John Wiley & Sons; 2007.
16. Sinclair J, Cardew-Hall M. The folksonomy tag cloud: when is it useful? J Inform Sci. 2008; 34(1):15–29.
17. Tetlow P, Pan J, Oberle D, Wallace E, Uschold M, Kendall E. Ontology driven architectures and potential uses of the semantic web in systems and software engineering. 2006. Available from: http://www.w3.org/2001/sw/BestPractices/SE/ODA/060103/
18. Treude C, Storey MA. Work item tagging: Communicating concerns in collaborative software development Software Engineering. IEEE Transactions on Software Engineering. 2012 Jan–Feb; 38. Available from: ieeexplore.ieee.org
19. Williams L, Kessler, RR. All I really need to know about pair programming I learned in kindergarten. Communi-cations of the ACM. 2000 May; 43(5):108–114.
20. Xiao W, Chi C, Yang M. On-line Collaborative software development via wiki. WikiSym '07: Proceedings of the 2007 International Symposium on Wikis. Montreal, Quebec, Canada: ACM; 2007. p. 177–183.