ARCHITECTURE FOR INTEGRATING A WEB-BASED IDE AND A PROJECT MANAGEMENT SYSTEM

Arnold Choa[1], Linda Chua[1], Kevin See[1], Alvin Uy[1], Danny Cheng[2]
[1]Software Technology College of Computer Studies, De La Salle University
[2]Information Technology Department College of Computer Studies, De La Salle University

In a typical software company, a workstation would have an Integrated Development Environment (IDE) such as Eclipse installed used for development and a project management system used to track the progress of the project in the web. In recent years, these binary-based IDEs have become web-based IDEs. Some examples of such systems are CtrlSpace and CodeRun. By virtue of being online and used as a service, the barriers to collaborate on projects have also been drastically reduced. Project management software solutions such as Microsoft Project are typically used in conjunction with a programming IDE and a code versioning tool in order to collaboratively develop a solution. Although web-based IDEs and project management systems are useful tools for software development, the two modules remain to be separate entities. As such, common problems that arise include high amounts of overhead in order to properly utilize both solutions as well as to ensure proper documentation of the project for the purposes of addressing change management requirements. Thus we have developed architecture and its corresponding implementation that integrates a development IDE, a project management system, as well as code versioning solutions in order to address some of these issues. Based on our project, we were able to integrate all three solutions thus allowing the tracking of features and issues defined in a project management system directly to the programming codes that are relevant to them and support an asynchronous mode of collaborative software development process. The advantage of our systems lie in the granularity of the mapping where by the system was able to track up to the programming code level instead of simply ending at the file level. This is done when a developer commits the changes he/she has made with respect to a ticket in the project management system. All changes made are tracked automatically and correlated to tickets even thru the different versions of the code. As such the process aids in the documentation and change management process and a newly assigned developer would be able to quickly find exact program codes based on specific features and issues reported. Based on our test results, the architecture and corresponding implementation was able to track and map tickets to program code with a high accuracy level. Future studies will look into improving the performance and robustness of the system as well as tools for synchronous collaboration for developers and project managers.

Keywords: Cloud computing; web-based integrated development environment; project management system; issue tracking system; software as a service

1. INTRODUCTION

Traditionally, IDEs are used by programmers for everything that involves coding. Over time, programming has evolved from coding simple projects to developing complex applications. With this comes the collaboration of developers as members of development teams to produce such complex systems [1]. These members would function by making sure that the system adheres to the requirements as well as maintaining its quality through constant testing, debugging, and keeping track of bugs. It is also from this that software development methodologies are needed to maintain this very cycle.

In an effort to organize the software development methodologies, project management systems were born. A project management system tracks and organizes the development team's progress and work [1]. This system can be deployed as a web application so that it is accessible by anyone with an Internet connection. The advantages of project management systems include the incorporation of issue- or bug-tracking management [1] as well as version control repositories that allow users to synchronize and version source codes. The tasks are represented in the project management system through the use of tickets or issues. The ticket is the most important element of a project management system as it contains information about a bug, a new feature, or an enhancement related to the project. As such, whenever one makes changes regarding a task, he/she must update the ticket manually in order for other developers in his/her team to view the changes as well, should the need arises.

Although IDEs and project management systems are important tools in software application development, the two modules remain separate entities. A survey conducted on 120 different companies showed that the least satisfactory aspect of project management systems is the lack of integration with other systems [2]. This means that if a developer updates a ticket, there is no convenient way for another developer to view the source codes that are relevant to the ticket. In other words, there is no system to automatically map tickets to relevant source codes. The remaining segments of this paper will discuss CtrlSpace2, a system that seamlessly integrates an existing web-based IDE, CtrlSpace, with functionalities of a project management system with respect to the problem.

2. CTRLSPACE2 PLATFORM

CtrlSpace2 is a platform that will allow for a seamless integration of both web-based IDE and project management system. Through the integration, users of the system will be able to switch from development mode in the IDE to the project management system, back and forth. In addition, it will document the progress of the user as he attempts to complete a ticker or issue, thereby allowing other users to view the progress and relevant source codes should the needs arises.

3.1 Main Features

The following are the functionalities that CtrlSpace2 provides:

1. allows users to create multiple projects in the project management system that represents a software development endeavor.
2. allows users to manage the projects created in the project management system by providing an interface to manipulate project elements.
3. automatically updates fields in tickets to aid users in managing a project.
4. provides a facility to link specific parts of a source code, which can be viewed in the IDE, to a ticket by getting the changes made on each source code every time the user executes a commit operation.
5. provides a way to jump from the project management system while viewing a ticket to a particular line in a linked source code that will be viewed in the IDE by using the details contained in the ticket.

3.2 System Architecture

Because CtrlSpace2 will be deployed through the web, it follows the client-server model of computing which is the standard model for all web-based applications. Figure 1 shows the architecture of the system. As seen from the Figure 1, the architecture follows the model-view-controller (MVC) design pattern. The green regions of the architecture (i.e., FileType, Plugin, and Project Manager) indicate the modules from the previous iteration which was implemented by the previous proponents who worked on the system. The blue regions indicate the modules from the current iteration (i.e., Comments, Events, Issue, PMS Project, Settings, Time Log, and Widget Managers) implemented by the authors of this paper. The red regions indicate the modules (i.e., all the remaining parts in the Server side and the last column in the Client side) where the integration takes place.

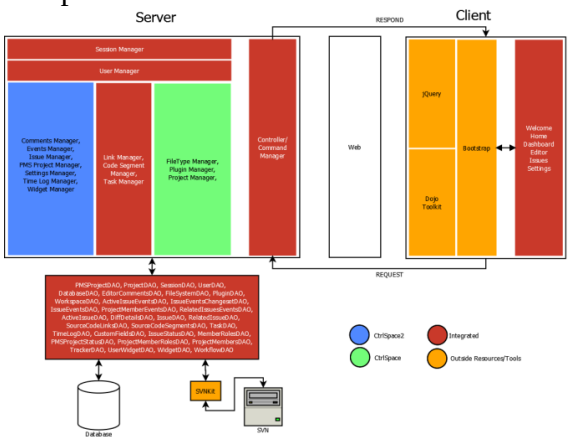

Figure 1. CtrlSpace2 Architecture

3.2.1 Communication Pattern

The client and the server employ two communication patterns for exchanging data. The first pattern involves AJAX (Asynchronous JavaScript and XML) requests. In order to perform AJAX requests, the system uses the Dojo implementation of the Bayeux Protocol which is based on the publish/subscribe model. The second communication pattern, which is used to load web page uses the HTTP Protocol which is a standard protocol followed by web applications.

3.3 Issues Considered in the Implementation

The following lists the different issues that were considered in this implementation. Corresponding algorithms were adapted and implemented to resolve these issues.

3.3.1 Linking Source Codes to Issues and Vice Versa

Algorithms to generate source code links and relating them to issues are the main focus of the system. In response to resolving certain conflicts when generating and relating links, an active ticket has been used to isolate the current task the user is working on.

3.3.2 Measuring Interestingness

The metrics or measurement of interestingness of a particular source code is important in order for the system to properly present information to the user. The occasions at which the interestingness of a source code may be used may be when jumping from an issue in the project management system to the IDE and vice versa

3.3.3 Accuracy of Links

Because determining which source code segments are related to which issue would require the use of artificial intelligence, which will require a lot of time to develop, users of the system are required to follow certain procedures and flows. Because of that, source code metrics and heuristics were used in order to determine the accuracy at which the system links source code segment/s to issue/s. The concept of coupling and cohesion [6] are examples of metrics used for determining the accuracy of links through heuristics. The premise of coupling and cohesion is that the more issues that are related to a particular source code, the lower the accuracy of a particular link will be due to the higher chance of a conflict. For instance, if a link has only one related issue, it would likely be 100% accurate because the probability that the relevance of the link to such issue is high as only one related ticket is considered. On the other hand, if it has more than one, the accuracy would be lower because more than one issue is considered and thus, the relevance to such tickets would be lower.

3.3.4 Conflicts in Linking

Because the system only uses heuristics to determine the accuracy, there will be a possibility that the source code link generated by the system would be incorrect because of the amount of issues related to a particular source code. This would result in conflict in linking. Because of this, the proponents considered various scenarios at which a conflict could emerge.

4. RESULTS AND OBSERVATIONS

The system underwent five testing phases, namely: functional testing, performance testing, usability testing, compatibility testing, and user scenario testing.

4.1 Compatibility Testing

This involved testing the system in different operating systems, particularly mobile operating systems, such as Android and iOS. Findings showed that most system functions were able to execute properly. However, both Android and iOS were touch-based, thus some system functions requiring the use of mouse and keyboard failed to execute.

4.2 User Scenario Testing

This involved testing whether the ways of integration were indeed seamless and effective. This was done by having a group of users simulating a small software project development team and developing a web application using the system. Findings showed that functionality-wise, the system was 100% accurate when linking source codes to issues. However, relevance-wise, the system was only as good as the users working on the source codes since they, not the system, dictated whether related code segments were indeed relevant.

5. CONCLUSION

One of the most prominent complications haunting the IT industry right now is change. Although change can be beneficial, it can trigger a series of complications that, if left unattended, can slow down an entire project. Thus, handling change is crucial to any software development endeavor.

The two primary tools used in the industry are integrated development environments and project management systems. Albeit useful, they remain to be separate tools. This is a major inconvenience according to some of the industry's leading software developers and managers.

With change management, there is a need to integrate both tools. The primary focus of the research was to determine how to integrate the management and development tools and in doing so would help the users manage change. This can be done by relating the two main elements of

both tools: source codes and issues.

The resulting output of this research was CtrlSpace2, a system that provided integration not only on a functional level, but also on a user level where steps to accomplish a certain task are shortened into one.

6. REFERENCES

[1] Whitehead, J. 2007. Collaboration in software engineering: A roadmap. In Future of software engineering, 214-225.

[2] Azizyan, G., Magarian, M., and Kajko-Matsson, M. 2011. Survey of Agile tool usage and needs. In Agile conference, 2011, 29-38.

[3] Lock, D. 2007. Project management. Gower Publishing Ltd.

[4] Tarbox, B. 2011. Making the switch: One team's story of adopting JIRA, Fisheye, Eclipse, and Mylyn, http://www.youtube.com/watch?v=5pJMewJ50rM.

[5] Cheng, L.-T., Souza, C. R. de, Hupfer, S., Patterson, J., and Ross, S. 2003. Building collaboration into IDEs. Queue, 1, 40–50. DOI= http://doi.acm.org/10.1145/966789.966803.

[6] Meirelles, P., Santos, C., Miranda, J., Kon, F., Terceiro, A., and Chavez, C. 2010. A study of the relationships between source code metrics and attractiveness in free software projects. In Brazilian Symposium on Software engineering, 2010, 11-20

[7] Co, R. C., Obaldo, M. A., & Ong, A. E. (2010). An Architecture for a Web-Based IDE. (unpublished)