Software Release Collaboration System

[1]Angeline Hazel Henry [2]Chandhini.G [3]Mariesha Justin [4]Ms. Ramya Laurraine.U
[1,2,3]Student [4]Assistant Professor
[1,2,3,4]Department of Information Technology
[1,2,3,4]Loyola-ICAM College of Engineering and Technology
Chennai, India

Abstract

The key to web based collaboration between multi-site software developers and team members is how effective these distributed teams share development work products and project status. Recently systems have evolved to aid information sharing in a more flexible manner. These systems support video tools, media tools with centralized repository tools for effective communication. Most previous studies have thrown light into the obstacles of using these highly technological media tools. Some of the major impediments are different time-zones and cultures, IT infrastructure in mostly Agile Global Software Development (AGSD). This paper proposes a system where developers can remain offline ensuring a simple, user friendly guiding route map in the developmental environment. The software ensures organizing of builds for easier tracking of completed modules.

Keyword- Agile Global Software Development, Distributed Web Collaboration, Globally Distributed Teams, Builds, Synchronous Communication

I. INTRODUCTION

Communication and coordination have always been an issue in large software engineering projects (e.g., [3], [5]). Coordinating work across sites has become more challenging due to the spanning national, language, and cultural barriers (see, e.g., [4]). Recent research [12], [14], suggest that Communication and coordination between cross-site causes loss of developmental speed. This reduction in speed would in turn affect the project schedule and project estimation in cost.

During development of project, every single developer works simultaneously on their modules and the project manager decides a single date of release for the builds developed till date. On this date the builds are sent to the testing team. Further, if the testing team is globally distributed to the development team, then cost of project development increases. This cost can be checked if the builds are sent for testing by pre- approving the changes made at the developer end.

This paper proposes software which helps in verifying and approving builds that enhance real-time collaboration among distributed teams.

II. COMMUNICATION INFRASTRUCTURE

The team "communication infrastructure" refers to the software, hardware, and installations needed to enable communication among Distributed Teams. Such software that helps to collaborate the co-located teams focuses on media tools with centralized repository for phone calls, video conferencing and interactive software [17]. These software are mostly sentient and require synchronization.

Fig. 1: Communication-Key to good software development

Highly sophisticated software becomes complex to use for a small team or company. This might also deviate from the main aim of verifying the builds before testing team receives it. To overcome this, the developed software GLORELEASE is simple to the point and enables to approve the builds released from the development team before it reaches the testing team.

III. ARCHITECTURE OF SOFTWARE RELEASE COLLBORATION SYSTEM

The most essential part of development is a build. Development team codes small parts of every module called builds. Each build has to be tested for errors and any changes that have to be made are sent back to the developer. This software enables the developer to upload his builds along with its application name, version, type of release and the description of the changes made to it.

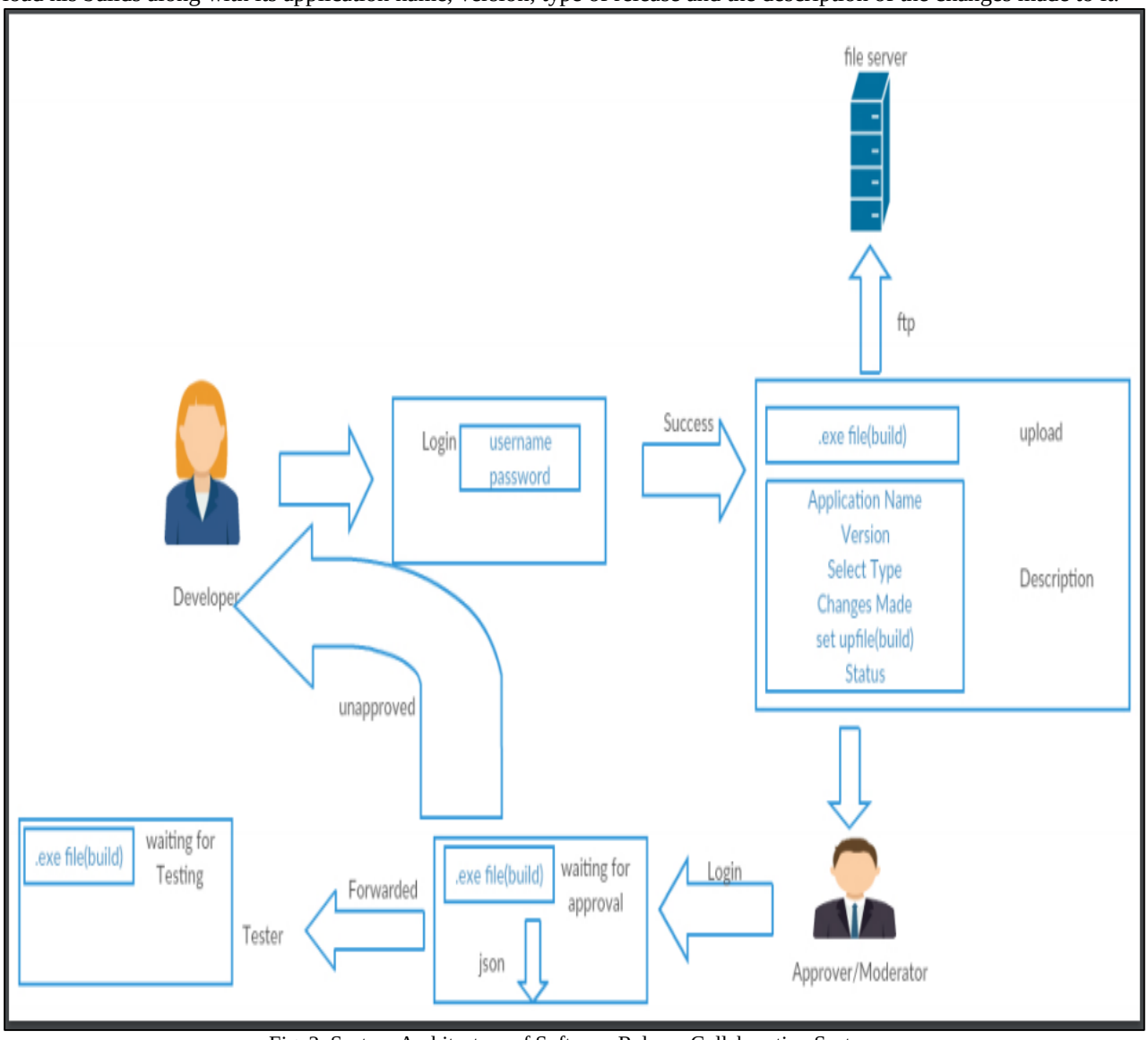

Fig. 2: System Architecture of Software Release Collaboration System

A. Software Requirements

- 32 bit Windows 7 Operating System
- visual studio community 2015

B. Hardware Requirements

- 1.6 GHz or faster processor
- 1 GB of RAM (1.5 GB if running on a virtual machine)
- 10 GB of available hard disk space
- 5400 RPM hard disk drive
- DirectX 9-capable video card that runs at 1024 x 768 or higher display resolution

IV. WORKING OF GLORELEASE

The overall working of the project consists of three main tasks, which are mentioned below

A. Login

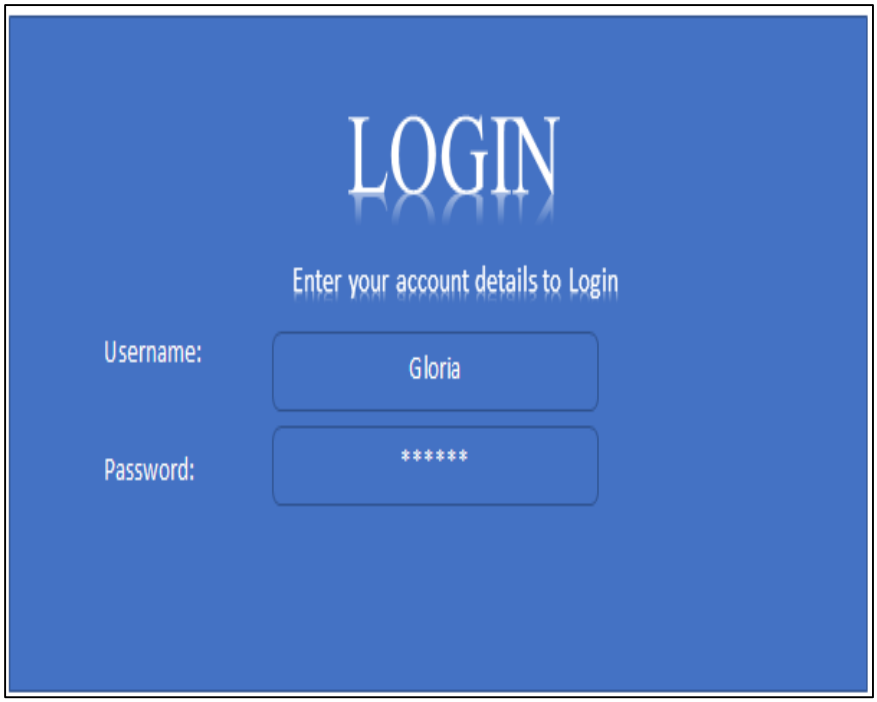

Fig. 3: Login Page

Initially the developer has to login before uploading his builds. This is to ensure that the developer can only upload builds and cannot download them. When the approver logs in he has the freedom to download the builds.

B. Upload Builds

This module is used by the developer to upload the details of the build which includes its version, type of release, changes made to it and the build in the format of .exe file.

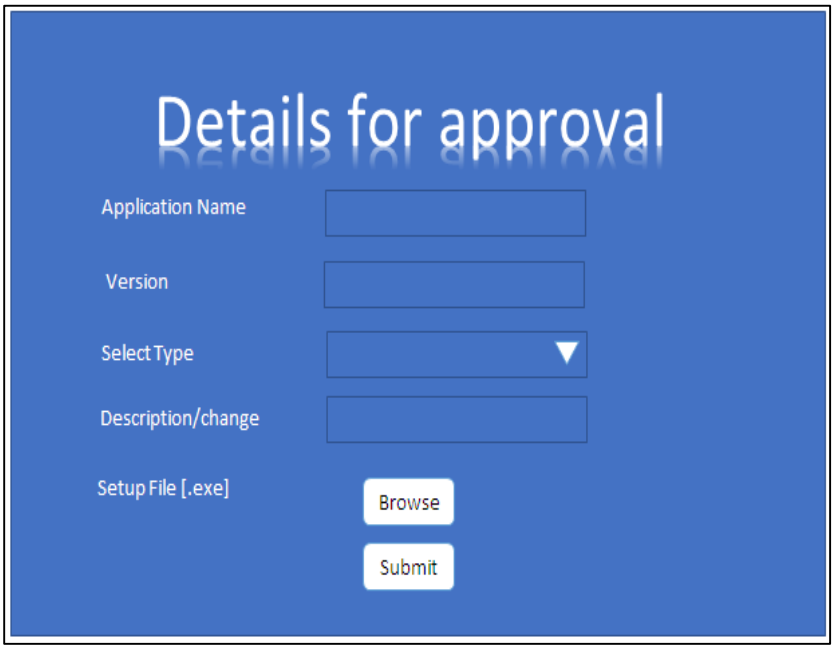

Fig. 4: Upload Details Page

When the .exe file of the build is uploaded through the FTP server, a HTTP Link is automatically generated.

The submit button will direct the developer to the next page with a preview of the details. The developer can now select the files that he wants to send for verification and approving to the project manager.

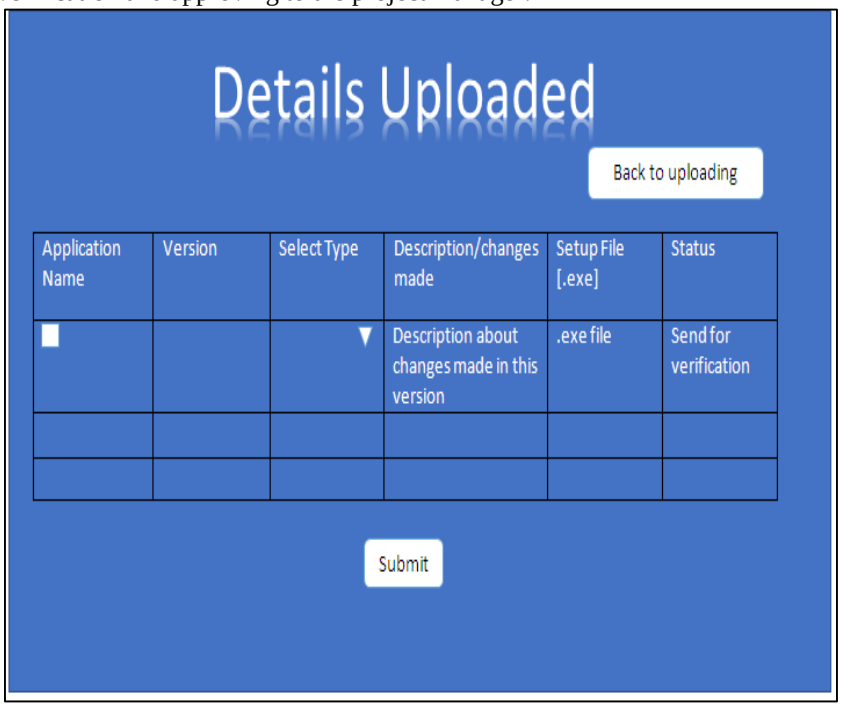

Fig. 5: Upload Build Page

The details that are selected for approving are then serialized into a json file and stored. The json file along with the .exe build file will be stored in an FTP server.

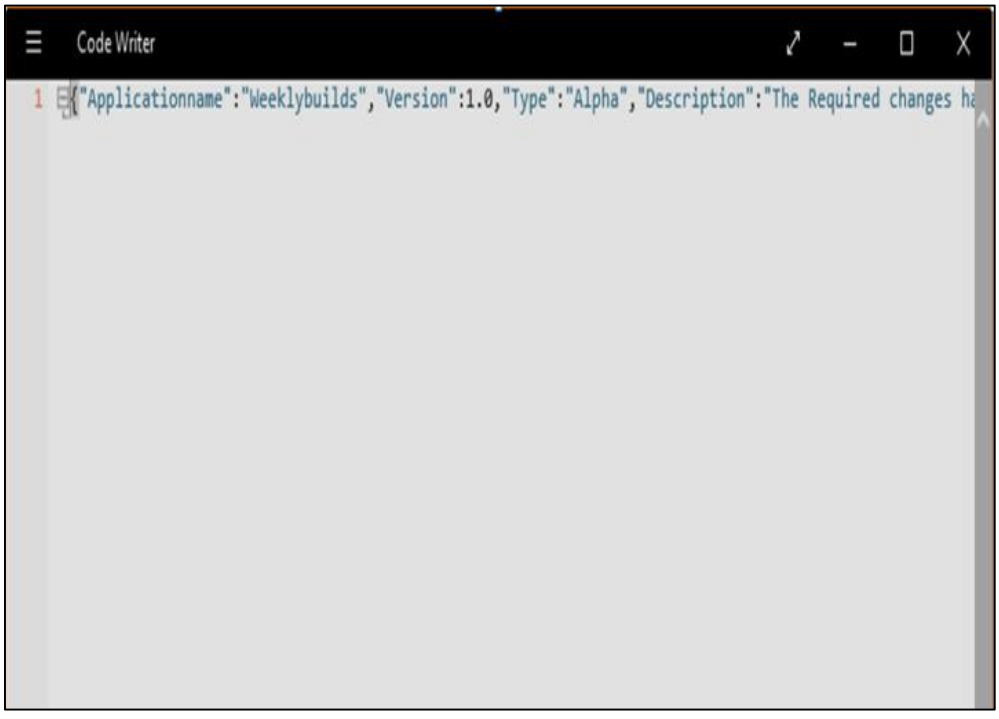

Fig. 6: Sample json File

C. Approve Builds

When the builds are successfully uploaded, an e-mail will be sent to the approver who might be the PROJECT MANAGER. The email contains the link that will direct the approver to the software. This does not require the approver and developer to be synchronously online, overcoming the limitation of communication in distributed web collaboration.

The approver will provide his login credentials and enter into the software. There he views the list of builds uploaded for approving represented in a tabular format. The table shows the following details:

- Application Name
- Version
- Description or changes made
- .exe file
- Status

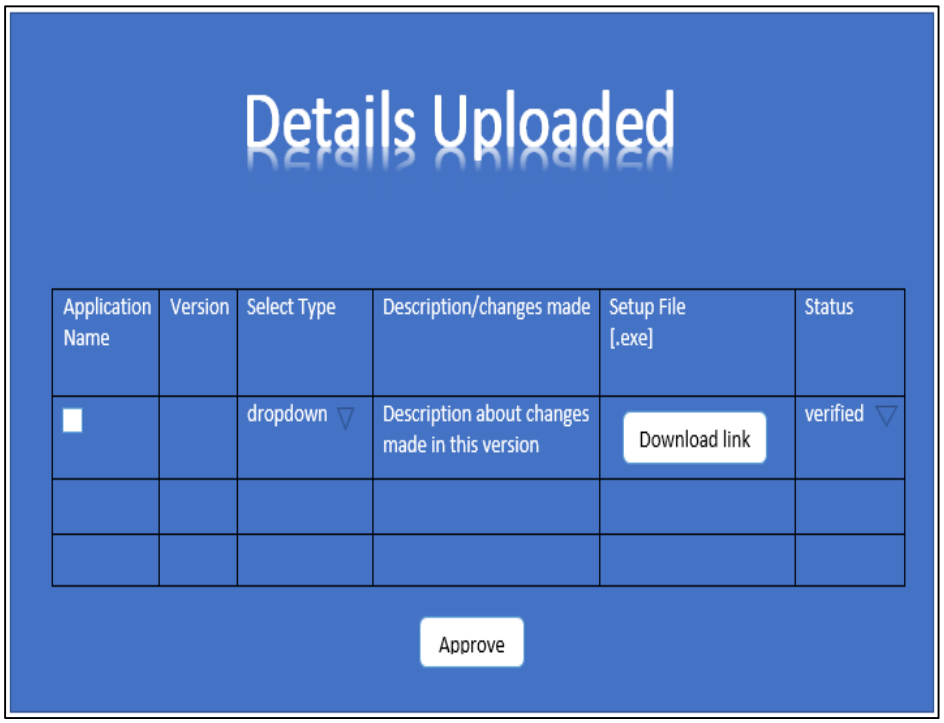

Fig. 7: Approver Page

The .exe file has a download button attached to it. The approver downloads the executable files and runs the code. He can give inputs to check for the expected output. If he gets the expected results he approves the file by clicking the approve button, this will forward the release to the testing team else he enters the changes to be made in the description column and sends it back to the development team for revaluation or modifications of the software. This is done to cut down the possibility of the software being sent back to the development team and thus it curtails the project cost.

The approver can also remove the unwanted contents, add what is needed in the description and forward it to the testing team. This is done to prevent miscommunication or wrong information being transferred.

V. EVALUATING THE BUILDS BEFORE TESTING

In addition to curtailing the possibility of the software being sent back to the development team and cutting down the increase of project cost there is a need for a moderator to monitor mistakes like sharing the URL of the source code's location, leaking of source code or sensitive information. The approver here acts like a moderator and reviews the message and file sent between the developers and testers. Before the message goes to the testing team, the project manager (approver) should approve it to be sent to the globally distributed testing team. The approver should also be eligible to edit the code and the message if he is not satisfied else he sends it back to the developing team. This cuts down 20% of the project's estimation delay and reduces redundant errors [14].

VI. CONCLUSION

This paper gives a perspective of communication infrastructure while developing products in start-up software development companies. This will also be helpful in the case of AGSD environment.

Most of the previous studies have dealt support of communication with a media tool from a technological perspective but that will be expensive and not satisfactory for small teams in a software development company. For this project, the observations were obtained from three aspects: communication between the team member and project manager, communicating the status of the development process, and communicating the rate of progress of the product under development.

REFERENCES

[1] M.J. Abel, "Experiences in an Exploratory Distributed Organization," Intellectual Teamwork: Social Foundations of Cooperative Work, J. Galegher, R.E. Kraut, and C. Egido, eds., pp. 489-510, 1990.
[2] R.D. Battin, R. Crocker, J. Kreidler, and K. Subramanian, "Leveraging Resources in Global Software Development," IEEE Software, vol. 18, no. 2, pp. 70-77, Mar./Apr. 2001.
[3] F.P. Brooks Jr., The Mythical Man-Month: Essays on Software Engineering. Addison Wesley, 1995.
[4] E. Carmel, Global Software Teams. Prentice-Hall, 1999.
[5] B. Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," Comm. ACM, vol. 31, no. 11, pp. 1268-1287, 1988.
[6] P. Dourish and S. Bly, "Portholes: Supporting Awareness in a Distributed Work Group," Proc. ACM Conf. Human Factors in Computing Systems, pp. 541-547, 1992.
[7] C. Ebert and P. DeNeve, "Surviving Global Software Development," IEEE Software, vol. 18, no. 2, pp. 62-69, Mar./Apr. 2001.
[8] R.S. Fish, R.E. Kraut, and R.W. Root, "Evaluating Video as a Technology for Informal Communication," Proc. ACM Conf. Human Factors in Computing Systems, pp. 37-48, 1992.
[9] J. Galbraith, Organizational Design. Addison-Wesley, 1977.
[10] R.E. Grinter, J.D. Herbsleb, and D.E. Perry, "The Geography of Coordination: Dealing with Distance in R & D Work," Proc. Int'l ACM SIGROUP Conf. Supporting Group Work, pp. 306-315, 1999.
[11] M. Handel and J.D. Herbsleb, "What Is Chat Doing in the Workplace?" Proc. ACM Conf. Computer Supported Cooperative Work, pp. 1-10, 2002.
[12] J.D. Herbsleb and R.E. Grinter, "Splitting the Organization and Integrating the Code: Conway's Law Revisited," Proc. Int'l Conf. Software Eng., pp. 85-95, 1999.
[13] J.D. Herbsleb, H. Klein, G.M. Olson, H. Brunner, J.S. Olson, and J. Harding, "Object-Oriented Analysis and Design in Software Project Teams," Human-Computer Interaction, vol. 10, nos. 2-3, pp. 249-292, 1995.
[14] J.D. Herbsleb, A. Mockus, T.A. Finholt, and R.E. Grinter, "An Empirical Study of Global Software Development: Distance and Speed," Proc. Int'l Conf. Software Eng., pp. 81-90, 2001.
[15] "Global Software Development," IEEE Software, J.D. Herbsleb and D. Moitra, eds., vol. 18, no. 2, Mar./Apr. 2001.
[16] R.E. Kraut and L.A. Streeter, "Coordination in Software Development," Comm. the ACM, vol. 38, no. 3, pp. 69-81, 1995.
[17] Agustin Yagüe , Juan Garbajosa, Jessica Díaz, Eloy González," An exploratory study in communication in Agile Global Software Development" in Computer Standards & Interfaces , xxx edition. Madrid, Spain: Elsevier, 22 June 2016, 184–197.

