Improving Remote Collaboration for Software Testing and Documentation

Jonathan Colón Ruiz
Advisor: Héctor J. Cruzado, PhD

Abstract

The use of distributed engineering teams is becoming increasingly common in software development, which is a modern trend that faces challenges in communication, document sharing, and version control among others. The goal of this project was to enable better collaboration between a Massachusetts-based engineering team and their customer in Alabama by implementing a standardized system that used GitHub for version control and Confluence for document management. The Plan-Do-Check-Act framework was the guiding force throughout the implementation of the project, which involved analyzing current workflows, pilot testing, and assessing the tools' performance. The results from the pilot stage indicated a 40% cut in document-sharing time, elimination of version conflicts, and enhancement of traceability via automated timestamping. User surveys disclosed a considerable degree of satisfaction regarding the new system, though some areas requiring training were also identified. The research demonstrates that the combining of automation and version control has benefited distributed environments in terms of collaboration efficiency and information accuracy and henceforth, the same frameworks can be applied to future projects to ensure that the benefits of training and learning are enjoyed in a continuous manner, through sustainability and improvement.

Introduction

Modern software development projects increasingly relies on distributed engineering teams. Major companies such as Microsoft and IBM manage global projects with distributed teams in North America, Europe and Asia, resulting in difficulties such as time zones differences in coordinating updates, keeping consistent versions and aligning testing schedules [1]. While this allows organizations to utilize specialized skills elsewhere, it also introduces problems in communication, document sharing, and version control.

In the Improving Remote Collaboration project, the engineering team in Massachusetts and its customer in Alabama use emails, video calls, and manual data transfers; none of these provide for traceability or automation. The lack of standardized collaboration tools and workflows leads to inefficiencies, errors, and delays.
The objective of this project is to implement and test a system to store documents, control versions, and carry out automated data transfers for improved communication and fewer errors. The end goal of the project is to set up a complete collaboration framework that controls software version, document management and automation of processes. The adoption of this system is estimated to promote engineering team productivity, lower communication obstacles and clear future project scopes for larger scale engineering projects.

Literature Review

The challenges of remote collaboration in software development are manifold, especially when the teams are geographically dispersed. The most common challenges are communication lags, mismatched requirements, and absence of central documentation, all culminating in rework and errors [1]. These conditions become more exasperated as time-zone differences have to be factored in and reliance on asynchronous modes of communication like e-mail augments the drawback of not being able to get immediate feedback [1].

Document management and version control systems are generally considered critical tools in alleviating such challenges. Thanks to GitHub and Confluence, teams have an integral traceability system, an automated version history, and an access control mechanism to guarantee working always on a consistent validated file [2]. The discipline of version control diminish duplication, prevent data loss, and increase responsibility within distributed teams. Such platforms also integrate with the automated testing environment to give an added advantage for reliability [3].

Recent frameworks and standards emphasize continuous integration and continuous delivery, stressing thereby the importance of automation and repeatability in collaborative projects [3]. Agile and DevOps advocate frequent updates, incremental changes, and excellent feedback mechanisms to sync distributed teams [4]. Scientific as well as industry papers show the advantages of using GitHub in the CI/CD environment to enhance collaboration while eliminating integration errors and supporting documentation workflows [5]. Both findings merge to suggest that the implementation of the existent standards for collaboration platforms is an institutional answer to the inefficiencies faced by distributed-Engineering projects.

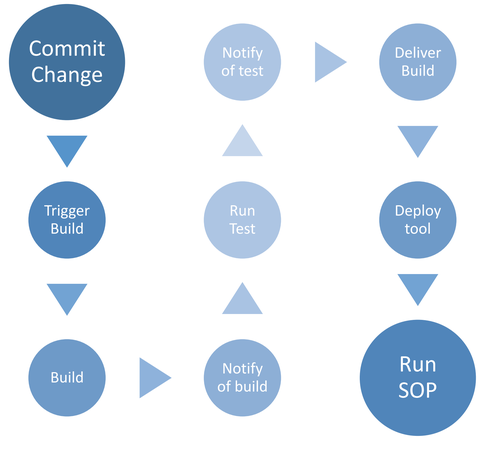

Figure 1
CI/CD Pipeline project example

Methodology

The project applied the Plan-Do-Check-Act cycle to promote better teamwork, document management, and data sharing between the engineering team in Massachusetts and the client in Alabama. In the Plan phase, the workflows that have been in place for testing and documentation were scrutinized for faults like no timestamps, repeated files, and slow processes that were not automated. The criteria were then made for tool selection based on compatibility, automation, and user-friendly operation.

During the Do phase, the setup of GitHub for version control and Confluence for document circulation was performed, both of which were programmed to provide automatic tracking, timestamping, and secure data transfers. Training workshops were held to make sure that the staff of both teams were using these tools uniformly.

The project-evaluated performance through metrics like task efficiency, version conflict reduction, and user adoption rate combined with feedback collected through surveys and observations in the Check phase.

Finally, in Act phase, refinements to the workflow and tool modifications were conducted based on evaluation outcomes, and then the creation of Standard Operating Procedures followed to guarantee long-term sustainability and continual improvement of the system.

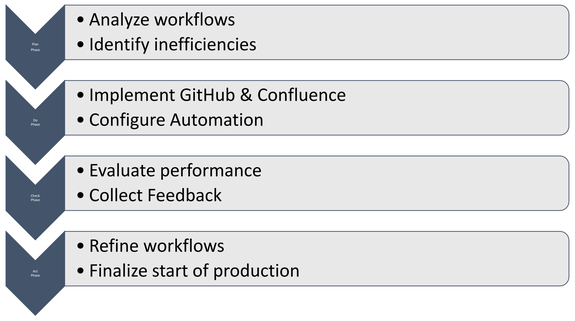

Figure 2
PDCA Framework Phases

Results and Discussion

The pilot implementation has shown a noticeable uplift in communication quality, coherence across versions, and document traceability of the dispersed teams. The introduction of GitHub and Confluence resulted in a 40% reduction in the time required to share updated documents and software versions, which was compared to the average time determined in the planning phase as the baseline. No incidents of duplicate or lost document versions were reported, which clearly indicates that one of the main issues faced by collaborators was successfully addressed. In addition, automated timestamping has increased the ability to authenticate the most recent document versions and keep an eye on revision histories, which has consequently led to an overall transparency increase.

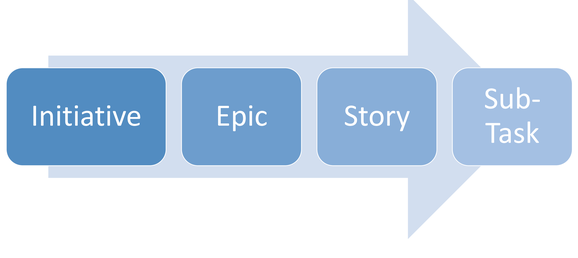

Figure 3
Confluence flow down order utilized

The virtual training sessions were facilitated in the Do phase to get both teams acquainted with the new tools. . The post-training surveys indicated that 85% of the participants found the platforms easier than the previous manual methods, although there was a small group that felt the need for more guidance on the advanced GitHub functions.

Some minor issues still existed such as network latency during the upload of large files and some team members occasionally relying on email out of habit. These problems underscore the need for continuous user support and integration that is incremental to attain full adoption of the system.

Overall, the pilot confirmed that automated document management and version control systems have a large positive impact on the collaboration in distributed engineering settings. Besides cutting manual effort, the tools also made the visibility of the ongoing tasks better, which caused the customer and the engineering team's workflow to be more synchronized. The findings recommend that extending this method in the case of future projects could mean the long-term benefits of getting quality, productivity, and communication reliability increased.

Conclusions

Adopting GitHub and Confluence along with the Plan-Do-Check-Act framework effectively addressed the main collaboration challenges. The system not only brought about better communication but also kept documents uniform and made the tracing of project outputs clearly visible. The use of automated data transfer combined with version control turned out to be very important for the decrease of human error and the increase of efficiency.

It is recommended that both teams continue using the implemented tools while laying down the practices for standard documentation and maintenance. User training should be continuous, and feedback should be collected regularly to maximize tool adoption and feedback from the users collected regularly to maximize tool adoption and improve team performance continuously.

References

[1] R. Druta et al., "A Review on Methods and Systems for Remote Collaboration," Applied Sciences, vol. 11, no. 21, p. 10035, 2021. doi: 10.3390/app112110035

[2] S. K. Devineni, "Version Control Systems (VCS), the Pillars of Modern Software Development: Analyzing the Past, Present, and Anticipating Future Trends," ResearchGate, 2024. [Online]. Available: https://www.researchgate.net/publication/378490782

[3] Document360 Team, "Documentation Version Control: How it Can Improve Collaborations and Workflows," Document360 Blog, Jul. 2, 2025. [Online]. Available: https://document360.com/blog/documentation-version-control

[4] Software Engineering Institute, Carnegie Mellon University, "A 5-Stage Process for Automated Testing and Delivery of Complex Software Systems," 2025. [Online]. Available: https://www.sei.cmu.edu/blog/a-5-stage-process-for-automated-testing-and-delivery-of-complex-software-systems

[5] University of Texas at Dallas, "Documentation Workflow: CIRC User Guide," [Online]. Available: https://docs.circ.utdallas.edu/user-guide/contributing/workflow.html