VERSION CONTROL SYSTEM FOR AGILE SOFTWARE PROJECT MANAGEMENT

Hujare, Ameya
College of Computing
Illinois Institute of Technology
Chicago, USA
ahujare@hawk.iit.edu/ (A20545367)

Kulkarni, Sanket
College of Computing
Illinois Institute of Technology
Chicago, USA
skulkarni27@hawk.iit.edu/ (A20537896)

Pawar, Deep
College of Computing
Illinois Institute of Technology
Chicago, USA
dpawar3@hawk.iit.edu/ (A20545137)

Abstract—**Agile software project management is an approach built on the principles of adaptation to change, flexibility, teamwork, and continuous development. It is intended to handle the dynamic nature of modern software development while also addressing the limitations of classic waterfall project management methodologies. It requires coordination between several team members, many of which are spread out geographically. The Agile methodology places a strong emphasis on software development and deployment through iterations. Managing and sharing a shared codebase can be difficult, especially if there are several code versions to maintain. Version Control Systems (VCS) were created to automate these procedures during the software development process. A version control system (VCS) in software engineering is a technique for monitoring and controlling modifications to a software project's source code, promoting better collaboration. It also facilitates the concurrent development of many software versions and provides electronic documentation. In this research, we examine multiple papers to determine the relationship between Version Control Systems and Agile Software Project Management. We intend to illustrate how VCS enhances the Agile methodology by providing scenarios and use cases.**

Keywords: Agile, Version Control, LVCS, CVCS, DVCS, HVCS, Git, Gitlab, GitHub, Bitbucket, Mercurial

I. INTRODUCTION

Version control systems (VCS), sometimes referred to as revision control or Source code management, are essential elements of the software development lifecycle. These systems monitor modifications to files, software, extensive websites, and different information collections. VCS, which may be recognized by features like revision numbers or levels, is essential for promoting effective teamwork, particularly in complex projects. Word processors, spreadsheets, Google Docs, and other kinds of Content Management platforms (CMS), such as the page history on Wikipedia, can all be integrated with these platforms.

In software engineering, version control is primarily used to keep track of source code modifications. This makes it possible to trace alterations, fix errors, prevent destruction, and safeguard against spam. Moreover, VCS facilitates the simultaneous management of numerous versions of the same product since software development frequently requires sophisticated teams operating concurrently across multiple locations. This feature is essential for identifying and resolving defects since it enables developers to go back and examine previous iterations to see which ones cause particular problems.

Traditional methods of version control involved manually safeguarding multiple copies of software versions, which proved to be difficult and prone to mistakes. From these primitive methods, modern version control systems (VCS) have advanced greatly, providing more complex mechanisms that hide version management procedures from end users. The increasing demand for real-time collaboration—which is no longer limited to people sharing a physical workspace but also encompasses geographically distributed teams with a range of responsibilities—has fueled this growth. The VCS tools of today have evolved tremendously, offering intuitive user interfaces and a smooth integration with the current development environments. There are two main architectures available for them: distributed and centralized. Centralized systems, such as Subversion, require client interfaces to retrieve the most recent versions for local work because they store all project data in a single repository. Distributed systems such as Git, on the other hand, enable independent and offline work by enabling each user to possess a complete copy of the repository. This change improves output while also meeting the needs of software development, which is more flexible and accessible these days.

In its most basic form, it was possible to manually store and mark several versions of a program in the past. This method used to be typical for big software projects and was called manual version control. This approach is doable but incredibly labor-intensive and inefficient. It requires a great deal of discipline and may cause developers to make new mistakes. Under such circumstances, the codebase usually stays static and access privileges are managed by an administrative user, protecting the codebase's integrity in the face of increasing complexity. Modern Version Control Systems (VCS) have been created to abstract version control's complexities from end users in order to overcome these issues.

Moreover, version control is required in a number of sectors other than software development where teams that may not be physically present must collaborate to manage documents.

A VCS that can monitor modifications and assign them to particular users might be especially helpful in these situations. Version control systems have evolved and become increasingly important due to the widespread usage of computers in various industries for the purpose of streamlining processes. As such, they are a crucial part of software configuration management. These solutions give projects a safety net by enabling real-time communication at various times and places. These sophisticated VCS tools are widely used, which highlights their significance in the field of software configuration management. The function of VCS is more important than ever because of our continued reliance on technology in many different areas. It ensures the continuity and integrity of software projects while taking into account the changing dynamics of team collaborations.

II. Discussion

1. Empirical Analysis of Version Control System Tools

Version control systems (VCS) are essential to Agile software project management because they promote iterative and collaborative development methods. Teams may integrate contributions from various developers, track changes, and manage code versions with these tools without losing previously completed work. An outline of how well-known VCSs like GitHub, Bitbucket, Git, GitLab, and Mercurial support Agile techniques is provided below:

*A. **Git***

Git is a distributed version control system that Linus Torvalds created in 2005 and is excellent at effectively managing both small and large projects. Different from centralized systems, every developer's working copy functions as a full-fledged repository, including the entire history of all changes. Git's branching and merging features are especially advantageous for agile development since they allow for frequent and adaptable code changes without interfering with the primary project. It is therefore perfect for Agile's quick iterative procedures and frequent code reviews. Research shows that Git manages several active branches effectively, reducing integration conflicts and facilitating continuous integration and delivery which are an essential elements of Agile teams. Furthermore, Git's capacity to quickly handle big codebases fits in nicely with Agile projects, which often get more complex over time.

How Git supports Agile software project management:

- **Scalability:** Agile methods must be scalable in larger teams, and Git effectively manages big projects and several branches.
- **Project Management Integration:** Git connects with JIRA and GitLab to enable complete project management, including task tracking and sprint planning, by directly connecting code changes to project tasks.
- **Collaboration:** Two tools Pull requests along with merge requests that let teams collaborate by allowing code review, discussion, and improvement, are made possible by Git's distributed architecture.
- **Iterative Development:** Git's branching and merging features provide distinct branches for features, bug fixes, or experimentation, which is in line with Agile's iterative development methodology.
- **Continuous Integration and Deployment:** Git works with CI/CD systems to make automated builds, tests, and deployments easier. This makes frequent deployments possible and helps find integration problems early on.
- **Transparency and Traceability:** Git improves the transparency and traceability necessary for Agile environments by keeping track of each change through thorough logs.

*B. **GitLab***

GitLab is a comprehensive DevOps platform that covers the whole software development lifecycle, from planning to deployment, monitoring, and security, going beyond version control. GitLab has integrated issue tracking for Agile project management, which makes work planning and administration more effective. Maintaining high development velocity is made possible by its automated CI/CD pipeline, which is in accordance with Agile's continuous integration and deployment standards. Furthermore, GitLab has integrated Scrum and Kanban boards, allowing for transparent insight into every stage of the development process and incorporating Agile management techniques within the platform. Studies demonstrate that GitLab's centralization and workflow optimization capabilities are essential for sustaining the rapid pace demanded by Agile techniques.

How GitLab supports Agile software project management:

- **CI/CD:** Integrated continuous integration and deployment, which is essential to preserving Agile's continuous update flow, automates testing and delivery.
- **Requests for Merges:** Encourage teamwork and code reviews in order to reinforce Agile's focus on communication within the team and continuous development.
- **Project Management and Issue Tracking:** GitLab's issue and project management capabilities enable efficient sprint planning and backlog management.
- **Integration with External technologies:** Increases workflow flexibility by supporting integration with a variety of external technologies, such as Slack and JIRA.
- **Analytics and Reports:** Offers insightful data via reporting and analytics tools, supporting both past analysis and future planning.
- **Agile Boards:** Visual task management and progress monitoring are made possible by customizable issue boards that support Scrum techniques.
- **Time tracking:** Time tracking features increase the accuracy of sprint planning by assisting in the efficient estimation and management of work time.

C. GitHub

With its user-friendly online interface, strong access controls, and sophisticated collaboration features like pull requests and code reviews, GitHub improves Git's version management capabilities. Agile project management is supported by GitHub through interfaces with programs such as ZenHub, which integrate Agile management tools within the GitHub platform itself. With features like problems, milestones, and labels, this connection helps with efficient team structure and progress tracking. Teams can customize their Agile workflows for more flexibility and efficiency thanks to GitHub's vast ecosystem and community, which also support a variety of third-party integrations.

How GitHub supports Agile software project management:

- **GitHub Actions (CI/CD):** Automate the procedures necessary for Agile's fast release cycles, such as build, test, and deploy.
- **Analytics and Insights:** Provides metrics to assess output and workflow effectiveness which are useful for Agile evaluations.
- **Project Boards:** Agile workflows such as Scrum and Kanban can be accommodated by these visual tools for task organization and progress tracking.
- **Issues and Milestones:** Useful for keeping track of tasks, improvements, and defects; milestones allow you to group them into releases or sprints.
- **Labels and Filters:** Task management is aided by labels and filters, which are useful for classifying and filtering tasks based on their category, priority, or status.
- **Pull requests and code reviews:** Prior to merging changes, support cooperation, peer review, and code quality assurance.
- **Integration with External Tools:** GitHub's functionality is expanded by establishing connections with additional project management and communication platforms.

D. Bitbucket

Bitbucket, a service provided by Atlassian, is a Git repository management solution that easily interfaces with Jira. It is a well-liked Agile project management tool that links code to project management activities. Complete traceability is guaranteed from project planning to deployment thanks to this link. Consisting of continuous improvement and frequent releases, Bitbucket also comes with Bitbucket Pipelines which is an integrated continuous integration and deployment (CI/CD) solution that automates testing and deployment. Furthermore, collaborative reviews and quick iterations are encouraged by Bitbucket's code review tools, which include pull requests with inline comments. These are essential for Agile teams striving for high-quality outputs.

How Bitbucket supports Agile software project management:

- **Pull Requests and Code Reviews:** Pull requests and thorough code reviews, which are necessary to preserve good code quality, facilitate collaborative development.
- **Bitbucket Pipelines:** These tools automate continuous delivery and integration (CI/CD) in accordance with agile standards.
- **Jira Integration for Problem Tracking:** This feature enhances project management by offering extensive tools for reporting, problem tracking, and sprint planning. It does this by integrating Jira with great depth.
- **Snippets:** Facilitates the exchange of code snippets for enhanced project collaboration and reusability.
- **Add-ons and Integrations:** Offers a range of market-sourced third-party add-ons and integrations to improve Agile project management capabilities.
- **Branch Permissions:** Improves code security and integrity by allowing you to control who can make changes to important portions of the codebase by setting permissions on branches.
- **Wikis for Documentation:** Contains an integrated wiki that facilitates the creation, management, and sharing of documentation by teams, keeping them all informed and in sync.

E. Mercurial

Mercurial is free distributed source control software which is well-known for its ease of use and efficiency managing projects of all sizes. Similar to Git, it facilitates a distributed method in which every user keeps a full local repository.

Mercurial is a popular choice in Agile organizations that demand swift scaling and flexibility because of its simple command set, which lowers the learning curve for novice users. Mercurial is regarded for its performance and simplicity, especially in large-scale projects, even though it is not as popular as Git. Research indicates that Mercurial's straightforward setup and ease of use can boost output, which is particularly helpful in Agile environments where teams constantly adapt and grow.

How Mercurial supports Agile software project management:

- **Command Extensions:** Allows for the integration of code review tools and other custom extensions to modify functionality to better fit particular Agile workflows.
- **Distributed Version Control:** Promotes Agile's emphasis on autonomy by enabling each developer to work independently with a complete local copy of the repository
- **User-Friendliness:** Mercurial's well-known simplicity facilitates speedy team onboarding and lowers tool complexity, which is in line with Agile's need for rapid adaption.
- **Integration with Other Tools:** Mercurial connects with Jira, among other systems, to link code changes to tasks and improve project visibility, even though it is not a project management tool in and of itself.
- **Branching and Merging:** Essential to Agile's iterative development process, branching and merging makes it easier to handle several development lines for features, problem fixes, and experiments.
- **Performance and Scalability:** This is crucial for Agile settings that require regular updates and scalability, as it manages big codebases and multiple branches with efficiency.
- **Community and Support:** Provides an ample amount of resources and a strong sense of community, enabling teams to tackle problems in Agile environments fast.

2. Types of Version Control System

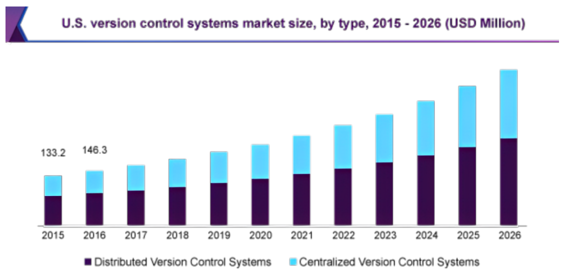

Fig. 1. Market size of DVCS and CVCS

1) ***Local Version Control Systems (LVCS)*** is the simplest type of version control, and it tracks changes just in the local environment. Rather than maintaining complete file copies, these systems preserve changes via a simple database that logs the variations between file versions. Despite its simplicity, this method creates the foundation for understanding more complex systems. Although it seems simple, there are a lot of challenges when using Local Version Control Systems (LVCS) in agile software project management. Agile methods place a high value on teamwork, adaptability, and regular updates which are factors that LVCS does not inherently encourage. Additionally, LVCS lacks the strong capabilities required to maintain several phases of development in agile environments by managing complicated project dependencies and multiple branches. Because there isn't a single repository that offers a coherent version of history, this shortcoming frequently causes serious problems when integrating modifications from different colleagues.

2) ***Centralized Version Control Systems (CVCS)*** like concurrent Versions System (CVS) and Subversion (SVN) are essential to managing software development projects because they guarantee that version control has a single source of truth. CVCS stores all version history on a central server, in contrast to Local Version Control Systems. Because of this centralization, multiple developers can check out and update files from a single repository that logs every change. When using CVCS, developers communicate with the central repository to publish or pull updates while working with their own copies of the files. By offering a unified perspective of project changes, this system improves collaboration and lessens disagreements, but it also has dangers and challenges, particularly in agile contexts where speed and flexibility are essential. Additionally, even though CVCS has branching and merging capabilities, the centralized control of these operations can make these jobs hazardous and complex in large-scale projects. The rigid framework of CVCS may interfere with agile development cycles and the quick adjustments required to satisfy changing project requirements or client input.

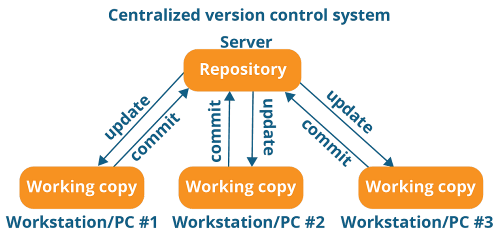

Fig. 2. Centralized Version Control System (CVCS)

3) ***Distributed Version Control Systems (DVCS)*** such as Git and Mercurial are significant developments in version control technology, which offer advantages that are in line with the concepts of agile software project management. With Distributed Version Control Systems (DVCS), every developer can keep a complete local copy of the repository, with all change history, unlike

with Centralized Version Control Systems (CVCS), which rely on a single central repository for version history. Moreover, DVCS improves branch and merge management, which is essential for agile projects that must quickly adapt to changing priorities and requirements. The ability to branch out for particular features or tests gives developers the flexibility to incorporate these changes into the main codebase as needed. Agile teams can run many development tracks simultaneously and increase the speed and agility of their development efforts by having the capacity to manage multiple active branches without depending on a central server. Developers are able to experiment and develop without having an immediate impact on the work of others because to this feature. The robustness of DVCS, which is attained by maintaining many redundant copies of the repository, is an additional advantage in agile environments. There is a significant reduction in the chances of data loss or server outages when every developer's system has a full backup. In order to continue the continuous delivery cycle that is central to agile techniques, this redundancy is required for uninterrupted development.

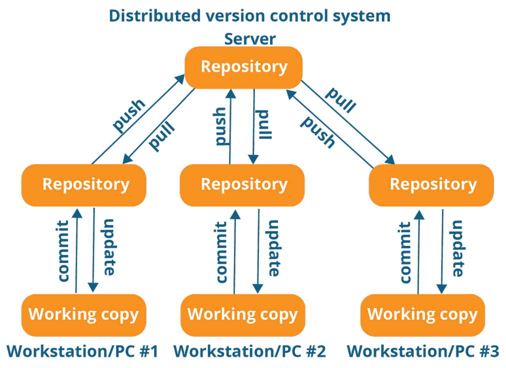

Fig. 3. Distributed Version Control System (DVCS)

4) ***Hybrid Version Control Systems (HVCS)*** is advanced version control systems (VCS) that combine the flexibility and redundancy of distributed version control systems (DVCS) with the centralized control characteristics of centralized version control systems (CVCS). These solutions are designed to provide the best possible results while meeting the intricate needs of large-scale software development projects, which are common in agile project management. Additionally, HVCS addresses some of the shortcomings of strictly centralized or distributed systems, like the scalability issues with CVCS and the difficulties with managing several distributed repositories with DVCS. HVCS provides a flexible and scalable architecture that can be tailored to the unique requirements of each project by empowering teams to determine which project parts need centralized management and which can benefit from dispersed processes. Hybrid Version Control Systems offer a powerful option in settings that prioritize agility in project management, where adaptability and promptness are critical. They provide powerful tools for handling large data sets and complex project frameworks, and they improve team member integration and cooperation. Because of these two features, HVCS is very appropriate for agile approaches, which emphasize collaboration, quick iterations, and the ability to quickly adjust to changing project needs.

3. Recommendation

These are our analysis-based suggestions for the version control system that should be used for various project sizes:

1) ***Git:*** *For Small to Medium-Sized Projects*
Git's strong integration features, user-friendliness, and outstanding performance make it a great choice for small to medium-sized projects. Because of its distributed design, which enables more adaptable development techniques, Agile teams find it very useful to quickly expand or modify their workflows.

2) ***GitLab:*** *For Projects that are either Big or Enterprise-Level*
GitLab is recommended for large-scale or enterprise-level projects. It is perfect for handling larger codebases and teams because of its extensive capabilities, which include strong security and integrated CI/CD tools. Maintaining the high levels of structure and management required in larger projects is made easier by GitLab's all-in-one platform.

3) ***GitHub:*** *For Strict Integration with Development Tools*
For projects that need close connection with other development tools, GitHub is recommended. Its adaptable marketplace and smooth integration with a wide range of well-known tools increase its usefulness in a variety of development contexts. Agile project management relies heavily on improved collaborative experiences, which GitHub provides.

4) ***Bitbucket:*** *Projects with High Security Requirements*
The best choice for projects requiring a high level of security is Bitbucket. It has robust security measures and gains from being integrated with Jira, as well as the rest of the Atlassian ecosystem. This makes Bitbucket especially useful for businesses that need to track and document projects in-depth but also place a high priority on security.

CONCLUSION

The present study has examined the integration of version control systems into agile software project management frameworks, following the evolution from simpler Local Version Control Systems (LVCS) to more complex Distributed Version

Control Systems (DVCS) and Hybrid Version Control Systems (HVCS). For agile techniques, each system type has unique benefits and difficulties. Because of its ease of use, LVCS is good for smaller projects, but it is not as effective at fostering collaboration as agile environments demand. Because CVCS relies on a central server, it tends to cause bottlenecks even while it allows for centralized management and visibility of adjustments, which might hamper the adaptable need for fast adaption. On the other hand, DVCS greatly improves the speed and effectiveness of managing complicated projects by enabling independent, concurrent development tracks and supporting distributed activities, which precisely fit agile objectives. Hybrid models offer a flexible solution that can be tuned to unique project requirements by combining the robust control of centralized systems with the adaptability of scattered ones. The continuous discussion on integrating technologies that support agility, reliability, and efficiency in software development is greatly aided by this topic and the version control system changes that have been recommended. In order to keep up with the advancements in software engineering methodologies and the expanding requirements of global development teams, future initiatives will concentrate on further optimizing these systems. This analysis underlines how important it is for tools used in agile environments to be flexible in order to maintain their compatibility with upcoming technology and project management trends while also supporting current development techniques.

REFERENCES

[1] N. Deepa, B. Prabadevi, L. B. Krithika and B. Deepa, ”An analysis on Version Control Systems,” 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 2020, pp. 1-9, doi: 10.1109/ic- ETITE47903.2020.39

[2] R. Majumdar, R. Jain, S. Barthwal and C. Choudhary, ”Source code management using version control system,” 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2017, pp. 278-281, doi: 10.1109/ICRITO.2017.8342438

[3]] S. S. Ragavan, M. Codoban, D. Piorkowski, D. Dig and M. Burnett, ”Version Control Systems: An Information Foraging Perspective,” in IEEE Transactions on Software Engineering, vol. 47, no. 8, pp. 1644-1655, 1 Aug. 2021, doi: 10.1109/TSE.2019.2931296

[4] S. S. Kasimov and O. N. Djuraev, ”Using version control systems in software development,” 2010 4th International Conference on Application of Information and Communication Technologies, Tashkent, Uzbekistan, 2010, pp. 1-5, doi: 10.1109/ICAICT.2010.5612028

[5] S. O. Dmitriev, D. A. Valter and A. M. Kontsov, ”System for Efficient Storage and Version Control of Arbitrary File Collections,” 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 2020, pp. 295-298, doi: 10.1109/EIConRus49466.2020.9038922

[6] X. Xu, Q. Cai, J. Lin, S. Pan and L. Ren, ”Enforcing Access Control in Distributed Version Control Systems,” 2019 IEEE International Conference on Multimedia and Expo (ICME), Shanghai, China, 2019, pp. 772-777, doi: 10.1109/ICME.2019.00138.