Version Control Systems for Agile Software Project Management

Avhad,Kranti
A20473187
kavhad@hawk.iit.edu

Gampa,Mounika
A20473187
mgampa@hawk.iit.edu

Shrivastava,Ansh
A2048142
ashrivastava@hawk.iit.edu

Abstract – *The approach for Agile methodology promotes continuous iteration of development and testing throughout the software development lifecycle of the project. With agile methodology, it becomes easier to adapt to end-user requirements using sprints. Version control systems help software teams to work faster. This paper discusses the background and the related works about Version control systems that have been studied by researchers for Agile Software Project Management. The purpose of this paper is to convey the knowledge and ideas that have been established on Version control systems.*

1. Introduction

Version control, also known as "Revision Control," "Source Control," or "Source Code Management," is a phrase used in Software Engineering to describe a group of systems for managing changes in programming codes, documents, or any other set of data. Version control systems are software tools that assist the team in managing all source code changes.

On the server, the source files are kept. Multiple developers can create a local branch by forking from the main branch. These users then make modifications to these local copies and send them to developers with higher access levels, who decide whether to accept or reject the merge request. If the modifications are accepted, they are pushed to the main repository; otherwise, they are rejected with suitable recommendations on how to improve the file. Agile is open to adaptation, encourages experimentation and welcomes changes of direction, even in later phases of the project. Because of this, the budget tends to be more flexible. Waterfall is a linear project progression, so it's best suited for projects with a defined end goal.

Waterfall Model	Agile Methodology
Linear sequential lifecycle model, follows a sequence.	Continuous iteration of development and testing in development process.
Single release in the Project	Multiple Releases.
Cannot inspect and adapt to end user requirements	Easily adapt to end user requirements using sprints
User can give the feedback at the end of the project.	User feedback is available at the end of each release.
Structured software development methodology.	Flexible
Profit can be availed only once, when the project completes.	Acquire financial benefits in various stages for each release.

Figure (i) Comparative Analysis Table

Version control has been separated into three generations during the last 40 years, based on the varied methodologies used to achieve versioning.

First Generation:
Locks were used to handle concurrency. For example, Revision Control System (RCS), which uses a set of UNIX commands to allow multiple users to generate and use code.
Second Generation:
The Concurrent Versions System (CVS) was part of the second generation of version control, and it handled concurrency utilizing the "Merge before Commit" principle.
Third Generation:
The basic principle for dealing with concurrency has been altered to "Commit before Merge." Tools like Git, Mercurial, and others are examples of third-generation systems.

Benefits of Version Control

Comparing Earlier Versions of the Code: Since then, all code changes and revisions, as well as timestamps, have been adequately recorded for history keeping and issue tracking. These can be utilized in circumstances where you need to roll back to a previous version or if you have a bug in your code in production.

Keeping Track of Modifications: It's a system that keeps track of all revisions and changes done throughout time. It keeps track of the contributions made by various developers as well as the modifications' intent.

Faster Deployment: Version control enables team members to work on the same document at the same time while being independent of one another, without influencing the contributions of other collaborators. "Branches" are the term for these. Thus resulting in faster deployments.

The approach for Agile is a project management that focuses on taking incremental, iterative actions to complete projects. Rather to completing the project in one fell swoop, it is completed in incremental chunks over time.
Continuous Integration (CI) allows agile strategy of delivering minor enhancements that drive development closer to the goal. A User Story is a collection of small enhancements or tasks. When new Commits are

into the main branch, continuous integration places a strong emphasis on testing automation to ensure that the application is not broken

Continuous Deployments (CD) refers to the process of releasing these user stories to production. Because it deploys all code changes to a testing and/or production environment immediately after the build stage, continuous delivery is an extension of continuous integration. This means you have an automatic release process in addition to automated testing, and you can deploy your application at any time by simply pressing a button. As a result, we have a continuous development, testing, and release cycle.

As a result, CICD, or Continuous Integration and Continuous Deployment, is a software development process in which incremental code changes are made often and delivered into production, with the benefits listed below.

Quick delivery with the implementation of CICD, many organizations are able to release features multiple times a day.

Customer engagement is ongoing. The client can track the progress of the whole project with each release. They can also make comments if it falls short of expectations or needs to be improved.

Continuous developments with all the releases scheduled close to one another, the developer team works almost continuously towards the final goal, managing the bugs and updates to the project.

The CICD process is used to develop and deploy these small jobs into production. Agile involves splitting the project into small individual tasks and submitting it progressively.

2.Implementing Agile using CICD

Because the market is changing so quickly, the previous software development life cycle is no longer sufficient to meet the demands of a competitive market. It is a sequential development method that flows like a waterfall through all stages of the project (analysis, design, development, and testing), resulting in a lengthy delivery time because each phase must complete all tasks before going on to the next. Agile software development is an iterative and incremental.

Test Automation Tool Idea behind “test automated pyramid” is to produce a high-quality code base. This pyramid consists of three levels of testing. Unit test, Service & UI test.
Methodology that accepts client modifications at any stage. Tasks required to be split down into smaller modules and delivered within a set deadline in order to implement the modifications.

Agile's first manifesto said, "satisfy the client through early and continuous delivery of valuable software," and based on this CI **(Continuous Integration)** was introduced, which was later expanded to CD **(Continuous Delivery).** CI **(Continuous Integration)** is a development practice in which development teams make small, frequent code modifications. CD refers to the ability to get changes of any kind at any time during the development process, whether it's a new feature, configuration modifications, problem fixes, or experiments. With this, CI added CD to its features, resulting in the CICD pipeline, which increased efficiency and productivity in agile processes.

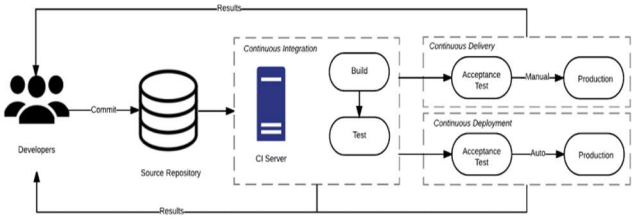


Figure (ii) Continuous Integration Continuous Delivery

CICD tools are a collection of different purpose tools which helps to make the process agile. Some popular CICD tools are:

Jenkins: An open-source CI automation server provides hundreds of plugins to support building and automating any project. Once a project is tested, Jenkins also supports the ability to deploy code with CD.

GitLab: A web-based Git repository manager with wiki, issue tracking, and CI/CD pipeline features, using an open-source license.

Bamboo: An on-premises CI tool integrated into Bitbucket that ties automated builds, tests, and releases together in a single workflow.

Codefresh: An easy-to-use tool to ease the migration of a project to Docker containers and to launch built Docker images to a hosted environment.

AWS CodeDeploy: A fully managed deployment service that automates software deployments to a variety of compute services, such as Amazon EC2, AWS Lambda, and on-premises servers.

The main deployment mechanism is based on the CICD pipeline approach. In the event of a performance issue, the deployment will be split into two phases. Rolling it out to a test-benchmark environment in the first step to

discover and fix performance issues early on, then rolling

it out to production in the second phase. CICD improves the performance of the deployment using the following iterative phases benchmark, load test, scale and provision.

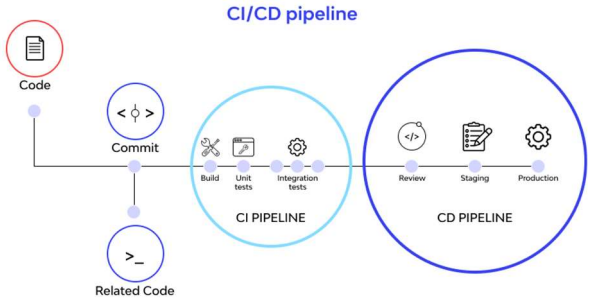


Figure (iii) CICD pipeline

Continuous Integration refers to the practice of development in which development work is integrated and merged by team members. Whereas, Continuous Delivery is the ability to accept any type of changes and apply these changes to production automatically. The whole process of CI/CD ca consist of various small steps which are covered in the form of mini projects/task. There are some terminologies which we should consider while learning about CI/CD pipeline methodology such as -

Benchmark: In which the production capabilities and limitations are checked, and the production level is adjusted before the scaling requirement is defined. Another factor is the Load Test, where a software process is tested to see if it can perform as well as production or if there any performance issues.

Scale Identification: The scaling factor is used to determine whether new software need additional resources to satisfy the performance standard. If further resources are required, they will be provided in the next phase.

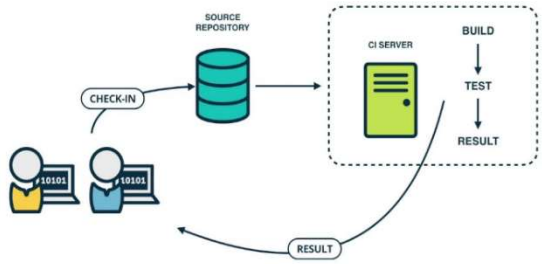


Figure (iv) Continuous Integration

Deployment is the most crucial stage in a CI/CD pipeline as well as in agile lifecycle. There can be numerous deployment methods which can run simultaneously. The successful deployment will be responsible for processing the project to the next phase successfully.

Iterative methods are some key methods which has the extra benefit for elastic scaling, means scaling on demand. Automations in CM tools combined with a CICD pipeline to save time, lower the risk of deployment failure, and ensure a stable release.

Implementation in CI/CD is a set of grouping of tools and services that work together as a single entity. In each stage of the CICD process, different tools might be employed (Source, Build, Test, Deploy).

Source
A modification to an app's source code, configuration, environment, or data causes a new instance of the pipeline to be created. This pipeline includes version control systems like GitLab, GitHub, BitBucket, SVN, Azure DevOps, and other industry technologies

Build
The pipeline builds (compiles) the application, generates redistributable packages from the source code, and ensures the code doesn't have any syntax errors and typos deeming it unusable. Docker, Jenkins, GoCD, CircleCI, Azure Pipelines, and other industry tools are examples.

Test
The code is tested, as well as the binaries, configuration, environment, and data. Selenium, xUnit, Katalon, and other industry tools are examples.

Deploy
Releases the software into the environment and performs a series of functional tests to ensure that it is performing as expected and is secure. Ansible Tower, Atlassian Bamboo, AWS CodeDeploy, CircleCI, Jenkins, and other industry solutions are examples.

3.Agile's Evolution in Software Industry

The second half of the last 50 years of Software Engineering has been dominated by Agile Software Development.

One of the most prevalent agile techniques is retrospectives, which allow for reflection on past performance, discussion of current progress, and planning for future improvement. Agile's growing popularity as a preferred software development approach and an important research sub-domain of software engineering necessitates its own retrospective. We'll talk about how Agile has impacted the overall performance of software businesses, as well as how current tech stacks are changing to support Agile in the workplace.

3.1. F-Secure Corporation - Agile Research – Mobile D

This is a case study where the F-secure Corporation deployed a new agile software development process (Mobile-D) in a pilot project in order to utilize its experiences in developing an organization specific agile process model alongside their traditional F-Secure product realization process.

The organizational goal was to utilize the experiences from the pilot project in establishing organizational agile process. The efficiency of agile SPI was empirically evaluated.

Improvement Category	Improvement Actions	Negative Experiences
Quality Assurance	8	3
Pair-Programming	4	7
Project Monitoring & Management	4	1

Figure (v) Most important improvement categories in Phantom project.

Figure (v), demonstrates the improvement categories in the project. Agile adoption increased the Quality assurance and also helped in the Project Monitoring and Management.

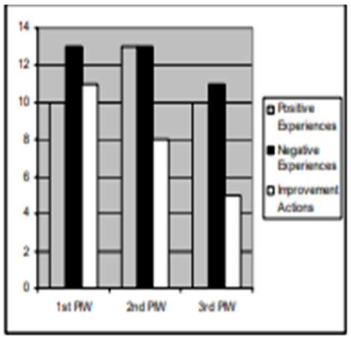


Figure (vi) Quality of project post iteration.

The empirical evidence from the case study illustrates that the organization was able to employ and benefit from the deployment mechanisms suggested in the agile deployment framework. Both the customer and the project team found the PIW method a useful mechanism in iteratively improving the daily working practices.

3.2. Shippable Case Research - Version Control Systems - Amazon Web Services

Many organizations are using cloud into their current software development cycles. Amazon Web Services is one such major provider. They provide a wide range of services, including SaaS, IaaS, containerization, and orchestration for developing, managing, and scaling containers according to needs. "Shippable" was research conducted by Amazon Web Services on one of their clients.

Shippable is a GitHub and Bitbucket-based platform for continuous integration, testing, and development. They offer comprehensive automation to testers and developers, as well as end-to-end automated solutions to make their clients' lives easier. Shippable technology helps developers focus on delivering high-quality code by removing hurdles and challenges from a repeatable Continuous Integration or Continuous Delivery workflow.

The Shippable platform consists of two parts:

Continuous Integration (CI), allowing developers to quickly build and test their repositories for every code commit or pull request.

Continuous Delivery (CD) pipelines, which automate an application's flow from source control to production. Developers may easily deploy containerized apps to container services like Amazon EC2 Container Service using these deployment pipelines (Amazon ECS).

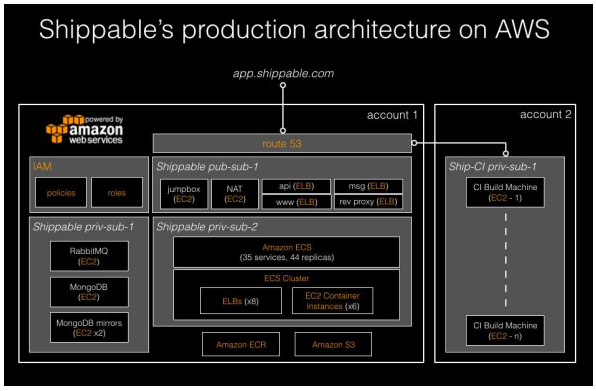


Figure (vii) Amazon ECS implementation in Shippable

Despite the fact that Shippable was able to deliver all of these services, their engineering team was 60-80% occupied with infrastructure management and maintenance, which made it difficult for them to develop new user features because their engineers were burned out from the sheer volume of work. Later, when Amazon Web Services (AWS) released its AWS Amazon Elastic Compute Cloud (EC2) services, Container Services, as a fully managed container orchestration service that makes it easy for one to deploy, manage, and scale containerized applications. Shippable used their service to increase the product's overall quality. The Amazon ECS service scheduler manages multiple copies of each microservice across the ECS cluster, Amazon ECS and Amazon CloudWatch handle all infrastructure logging with minimal effort from the engineering team, and Shippable Docker images are stored on Amazon EC2 Container Registry (Amazon ECR), a fully-managed Docker registry.

3.3. Case Research on Version Control System: A Review

Version Control Systems (VCS) have been used by many software developers during project developments as it helps them to manage the source codes and enables them to keep every version of the project they have worked on. It is the way towards managing, organizing, and coordinating the development of objects. In Software Engineering, software developers need to collaborate with each other to develop a better project. Thus, VCS is very useful because it also supports a collaborative framework that makes it easy for software developers to work together effectively. Without VCS, collaboration is very challenging. This paper discusses the background and the related works about VCS that have been studied by researchers. The purpose of this paper is to convey the knowledge and ideas that have been established on VCS.
Version Control System (VCS) is a system that manages the development of an evolving object. In other words, it is a system that records any changes made by the software developers. There are a lot of uses for VCS in software development that makes the development process easier and faster. VCS is also known as Revision Control System

Centralized Version Control Systems (CVCS) Similarly as with several other programming bundles or ideas, as the prerequisites continue developing, clients feel that the local version control systems constrain their activities 11. People are not ready to work cooperatively on a similar project, as the records with their versions are put away in some person's nearby personal computer and are not open to other individuals who work on similar documents. CVCS is an approach that enable the developers to work cooperatively. It stores a master copy of files history and in order to read, retrieved, commit new changes to a certain version, they need to contact a server 22.

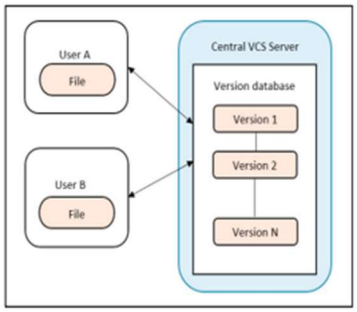


Figure (viii) Centralized version control system

As shown as Fig. 1, CVCS have a single repository which is basically the server2,11. All the file were kept in the server that everybody has access to from their local computer. All the developers make changes against this repository through a checkout taken from the repository but only the last version of the files is retrieved. As the files are stored in the server, it means that any changes made will automatically share with other developers. The developers otherwise known as the contributors and committers, must be the core developers who perform basic tasks such as creating branches, merging branches or reverting changes to previous states 10. In spite of that, this restriction will affect the participation and authorship of new contributors because in order to perform basic tasks, they need to follow some steps in joining the process to become a core developer. Below are the processes that must be followed by the contributor to become a core developer who has the writing permission:
• A software developer will start as a passive user which means his/her activity cannot be traced. At this stage, the developer cannot submit any email message or bug reports as he/she is only allowed to visit the project's website and read the mailing list:
• From a passive user, the developers will become advanced users if they become familiar with the project. At this stage, the developers are allowed to send some messages to the mailing list. This activity can be tracked;
• Developers who have partial knowledge about the project may do some report bugs, submit comments, and make some modifications to the project.
So, the conclusion is Software developers should have a rudimental understanding of what VCS is and which type of VCS suits them. The adoption of a VCS is a must in software development. It helps software developers manage their codes easily because it is common to have a lot of changes involving the addition or deletion of features. In order to adopt a VCS, a software developer must know and perfectly understand which approach should be used as it will affect the whole project and team. It is also important for them to have the knowledge of different approaches of VCS because the various approaches will affect their software development process differently.

3.4. Version Control System Research - Continuous Delivery of Value

The journey of Qentelli - envision automate, a worldwide financial consulting firm, towards CI-CD. This is a huge company in the United States whose main problem was to implement DevOps for themselves. The client has hundreds of large-scale online apps that are widely used and require frequent updates to keep up with their clients' rising business needs. The client set out on a goal to scale their business, embrace DevOps culture, define benchmark practices for specific engineering teams, and automate manual deployment processes in order to align their desired business agility with IT agility. The internal team was having trouble creating effective automated quality gates for the CI-CD pipeline's Continuous Integration section. The purpose of this project is to improve the organization's software delivery methods by implementing Continuous Delivery principles in

order to enable faster releases, better software quality, and more efficient engineering workflows. The following were the solutions presented to improve deployment automation: Employed uniform branching strategy.

Dedicated QA environments.

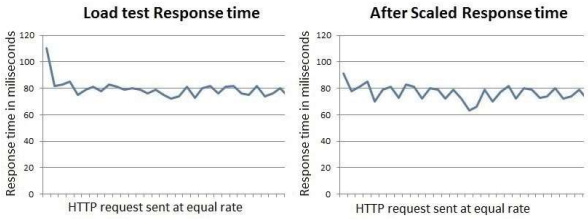
i. Integrated Unit and Smoke tests within the Pipeline.
ii. Automated deployments through ADO Release Pipelines.
iii. Automated Re-deployment Triggers for failed builds/releases

Automated Manual Activities (planned tasks to start/stop services), better legacy code quality through automated code quality checks, and migration of repos from TFS to Git in ADO, as well as interaction with Azure Boards, were among the solutions. Qentelli then migrated their entire TFS infrastructure to Azure DevOps and implemented 'Build and Release' pipelines for the migrated applications, integrating Checkmarx to scan source code early in the SDLC, integrating Sonatype in build pipeline, creating a new environment (QA) and enabling the team to test the code via smoke tests by introducing the shift left logic, where the entire production environment is cloned into a partner environment and tested so as to verify if everything is working. Qentelli was able to deliver deployments four times faster, with 25% fewer technical faults, 50% higher test coverage, 40% less downtime during release deployments, and 90% cost savings through automation.

4.Conclusion

Most software companies hoping to gain a competitive edge are using CI/CD practices today to help improve their DevOps pipeline.

It has become essential for organizations to embrace the world of DevOps and enable frequent delivery of good-quality software through CICD pipeline, to set themselves apart and have a cutting edge over others in the competition. CICD boosts the system's productivity while also making the delivery process more efficient. The CICD pipeline's iterative process aids in the improvement of the deployment process when employing the test benchmark technique.



With its above-the-index results and increased customer-oriented solutions, Agile is dominating the globe. Now, 71 percent of the IT industry is adopting Agile in their organizations, 98 percent of companies have benefited from Agile, 34 percent of companies are adopting containerization, and 21% plan to incorporate in the future, open source platforms are being relied on more since the release of Docker, 60 percent of business organizations have benefited after utilizing CICD tech stacks, overall company revenues have increased, overall customer satisfaction has increased. The use of an agile version control system has resulted in improved product delivery, increased customer satisfaction, a decreased failure rate, increased throughput, and reduced engineer workload.

5.References

[1]Nazal NurlisaZolkifli, AmirNgah, AzizDeraman, “Version Control System: A Review” https://www.sciencedirect.com/science/article/pii/S1877050918314819

[2]W. Cunningham, "Principles behind the Agile Manifesto", http://agilemanifesto.org/principles.html

[3] S.A.I.B.S. Arachchi, Indika Perera, Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software ProjectManagement

[4] J. Humble, and D. Farley, "Continuous delivery: reliable software releases through build, test, and deployment automation"

[5] Taylor Putnam, “A Case Study in Implementing Agile”, agileconnection.com

[6] Amazon Web Services, “Shippable Case Study: On Amazon ECS”, aws.amazon.com

[7] Qentelli, “A Global financial company’s journey to CICD”.

[8] W. Cunningham, "Principles behind the Agile Manifesto", http://agilemanifesto.org/principles.html