**A Software Configuration Management System:**

**GitHub, GitHub Actions, and Repl.it**

Samantha M. Hipple

School of Technology and Computing, City University of Seattle

CS 504: Software Engineering

Kendra Schraml

April 24, 2022

**A Software Configuration Management System:**

**GitHub, GitHub Actions, and Repl.it**

Software Configuration Management Systems (CMS) are designed to streamline the process of systematically managing, organizing, compiling, reviewing, testing, versioning and releasing all of the different software components, or artifacts, and their support activities during the development of a software product. Software artifacts include everything from requirements and design specifications, and change management plans to user manuals, source code, executable code, test cases and more. The CMS is an important part of the software engineering process that makes project management simpler, minimizes errors, increases traceability, and improves the overall software quality (Tsui, 2013).

Software Configuration Management (SCM) is useful not only for quality control, but for team management as well. The increased traceability allows teammates to work concurrently on the same project from all over the world, with each contribution tracking the contributor and the specific modifications or additions made. Additionally, so long as the entire team is proficient in the use of the CMS, as new information about changes in areas such as user requirements, policy, budget, schedule needs and more come up, they can be readily distributed in a centralized location accessible to all teammates and stakeholders (Martin, 2022). According to The QA Lead Team (2021), there are five steps involved in SCM: (1) Planning and Identification, (2) Version Control and Baseline, (3) Change control, (4) Configuration Status Accounting, and (5) Audits and Reviews. This paper will detail the SCM process for the creation of a project guided by a freeCodeCamp.org YouTube tutorial called *Code a Discord Both with Python - Host for Free in the Cloud*. The plans for, and progress of, this software project will be explained following the presentation of the rationale behind each of the five steps listed above. Additionally, test cases have been designed and completed for the overall functionality and integration of the custom CMS. Summaries of these test cases will be provided throughout the discussion within the SCM step they most readily relate.

**Planning and Identification**

The first step of the SCM process identifies the scope of the overall project and makes plans for the development process. This is often accomplished via team meetings and brainstorming sessions. Artifact types, organizational categories, naming system, tools, resources, files, documents, and anything else that is needed for a successful project should be identified in this step (The QA Lead Team, 2021).

This project began with a GitHub repository and a student assignment to create five submission folders with README.md files to store and report our progress throughout the engineering process. Thus far, resources that have either been acquired or found potentially useful for our coding project’s success are:

* **CMS:** GitHub, GitHub Actions, Repl.it, Microsoft Teams
* **Software Applications:** VS Code, GitHub Desktop, Microsoft Word, Google Chrome (web browser/internet access), Discord, Command Prompt
* **Technical Resources:** Discord Developer Portal, YouTube (freeCodeCamp), GitHub Actions Documents, Git Hooks Atlassian Tutorial, Repl.it Guides
* **Documentation:** README.md files for each directory’s narrative, notes, plans; five IP submissions detailing the project’s CMS, requirements and design, test cases, software metrics, and documentation; bot code

**Custom GitHub&Repl.it CMS:**

The overall CMS plan for this project is to integrate GitHub with GitHub Actions in order to automate certain actions based on event triggers. Additionally, our GitHub repository for this project has been connected to our Replit account, where the majority of the coding for this project will be written. Replit (2022) is a “free, collaborative, in-browser IDE” according to its homepage and was recommended by the freeCodeCamp guide used to build our Discord bot. Replit has the ability to deploy and host the bot on the cloud for free so that it stays online and available continuously for Discord users to access (see Figure 1).

**Figure 1:**

*Integrated GitHub Project Repository with Repl.it Account*Graphical user interface, text

Description automatically generated

GitHub has also been integrated with Microsoft Teams to provide notifications for specified events within the repository.

Currently, GitHub does not readily recognize changes in Microsoft Word (.docx) files of the same name. When attempting to commit just an updated .docx file via CLI commands, Git will respond that the repositories already match. Git will only accept updated versions of .docx documents that have been saved under a new file name. Due to this, GitHub is also unable to display what changes were made when an updated .docx file is committed to the repository the way it can for code and text files. For now, any and all .docx files must be saved with version numbers so that they are at least uploaded and stored in the project repository with the rest of the project’s artifacts (“View .docx”, 2014).

**GitHub Actions for Automation:**

This project contains Microsoft Word documents (.docx), such as this and future papers, for each of the five IP submissions. There is thought to attempt to configure an automated action that responds prior to all commits that translates .docx files to .txt or .md files. GitHub can track any changes made within updated text and markdown files of the same name during commits (“View .docx”, 2014). Translating .docx files so they are part of the commit would allow their modifications to be tracked within GitHub’s current system. As seen in Figure 2 below, GitHub Actions allows users to customize the automation of their workflow through code by creating actions that occur in response to specific events such as data being pushed to a repository or in response to denying a pull request according to Github (2022b).

**Figure 2**

*GitHub Actions Demo Workflow Commit*

Graphical user interface, text, application

Description automatically generated

While configuring the CMS for this project, a GitHub Actions tutorial was completed as a test case demonstrating the ability to create a *workflow* module within our repository with a code file that displays various details about the repository any time a *push* event is triggered as displayed in Figure 3 below (GitHub, 2022c).

**Figure 3**

*GitHub Actions Output from Push Event*

Text

Description automatically generated

**Version Control and Baseline**

When a software configuration item is in a state where it is determined to be ready for release, that artifact is considered a baseline. Once a formal baseline has been established for an item, modifications and updates must go through formal change control procedures – typically, at a specified and designated time within the SCM process. Common baselines include functional, developmental and product baselines. The functional baseline must correspond with the reviewed system requirements and can be used to create various versions of the application (Martin, 2022).

During this step, mechanisms are often defined for managing the production of alternate software versions. Some of these tasks include: (1) the identification and classification of all project artifacts, (2) establishing a hierarchal tracking system for the various software versions, (3) identifying essential relationships between the software components, (4) the establishment of product, developmental, and functional baselines, and (5) the development of a project-wide standard for the labeling all of the products, revisions, and files to simplify collaboration efforts (The QA Lead Team, 2021).

According to Atlassian (n.d.) Git is a modern, open-source, version control system created by the inventor of Linux OS kernel, Linus Torvalds, that can assist with all of the above tasks. Git is also what powers GitHub. GitHub calls itself “the largest and most advanced development platform in the world” that has enabled millions of developers and organizations build, ship, and maintain their software products (GitHub, 2022d). Github provides each user the ability to create repositories for their projects to: store and organize their artifacts as they are developed, track revision history, enable collaboration, collect user feedback, report bugs, organize tasks, share information, propose changes, and much more (GitHub, 2022a). The CMS test cases implemented for this section demonstrate our GitHub version control system’s code check-out/check-in process.

**Code Check-Out:**

Cloning a GitHub repository is an example of using our CMS to check-out code we wish to review or update. This project’s repository is already downloaded to our local files so, for this test case, we installed the GitHub Desktop API and screened the cloning of the remote repository to the API. Figure 4 demonstrates the simple process of downloading of our project’s remote GitHub repository to the GitHub Desktop Application.

**Figure 4**

*Cloning Remote Project Repository into GitHub Desktop API*

A screenshot of a computer

Description automatically generated with medium confidence

**Code Check-In:**

Committing modifications to artifacts and the addition of new artifacts is an example of code check-in for our CMS. Our test simply involved creating new configuration items and updating previously committed items, creating a commit, pushing said commit, then reviewing the changes tracked for the overall repository. Figure 5 displays these results.

**Figure 5**

*Reviewing Previous Commit of IP01 Updates*

**A screenshot of a computer

Description automatically generated**

**Change Control**

Change control is the method within SCM that focuses the most on quality control. Change control is designed to ensure that any changes made to baseline components integrate smoothly with the rest of the project. This process also helps with the approval and release of new baselines. In this step, change requests are submitted by team members and approved or denied by the software configuration manager – the most common change requests by a team are related to adding/editing code for various artifacts and changes to user permissions (The QA Lead Team, 2021).

During this step, the software configuration manager becomes responsible for controlling any necessary changes requested by the client or team. This is done by determining the technical merit of the changes via evaluation of potential side effects and overall impact on the project. Lastly, configuration managers are responsible for committing the approved changes or explaining why a change request was denied (Martin, 2022). This section will demonstrate test cases concerning pull requests and merging branches.

**Reviewing a Pull Request:**

**Graphical user interface, text, application, email

Description automatically generated** Pull requests announce changes pushed to a branch in a repository in GitHub. Pull requests can only be made when there are differences between one or more branches. In order to run our test case demonstrating pull requests with Git, after the creation of our GitHub Actions example code, the change was saved to a new branch called **Hipples-patch-1**. Figure 6 displays the pull request upon review. Figure 7 displays a successful merging of the Hipples-patch-1 branch with our main branch.

**Figure 6**

*Reviewing Pull Request to Merge main Branch with Hipples-patch-1 Branch*

**Figure 7**

*Merged Hipples-patch-1 Branch with main Branch - Adding the GitHub Action*

**Graphical user interface, text, application, email

Description automatically generated**

**Merging Pull Requests:**

After a commit to a feature branch has been evaluated to not have any open conflicts, the option to merge the branch with the main branch is displayed. For our test case to demonstrate this functionality, we merged the GitHub Actions update pushed to the Hipples-patch-1 branch with our main branch – then deleted the feature branch. The above screenshot demonstrates the end result of this test.

**Configuration Status Accounting**

This penultimate step focuses on tracking the release of configuration items and their subsequent versions. Configuration status accounting is accomplished by verifying that the project is progressing according to plan via test cases based on the predetermined baselines, release notes, and other related documents – ensuring that all functional requirements have been met. Each version released during the process is assessed for what is new compared to the prior release and why the changes were necessary (The QA Lead Team, 2021).

During this step, a complete listing of all of the changes made to the previous baseline to reach the new baseline is documented; the status and resolution of all change requests are monitored, and the related documentation is maintained; all configuration items are identified; previous releases/versions are extracted for analysis and testing; and the tracking of progress towards the next baseline begins (Martin, 2021). The CMS test case planned for this step of the SCM process include creating a snapshot of release to establish an initial baseline for our Discord bot. The process of publishing a release with notes is illustrated in Figure 8.

**Figure 8**

*Publishing Release Snapshot for Baseline of Dissident Bot Code*Graphical user interface, application, Teams

Description automatically generated

**Audits and Reviews**

The final step of the software configuration management process is a technical review of each stage of the project’s development life cycle. Everything that has gone into the development of each baseline while building the software is audited and reviewed including configurations, workflows, change requests, processes, documentation, and more. Multiple reviews of the product are performed by the team to confirm its integrity and compile essential documentation such as release notes, user manual, and installation guides (The QA Lead Team, 2021).

Configuration auditing is designed to ensure that all of the SCM baseline requirements have been met by the final deliverable and that defined processes are being followed. SCM auditors check for compliance with configuration control standards, that traceability is maintained during the building process, that configuration status reports are up-to-date, and, in the end, that the completeness and consistency of the overall software product is valid by comparing the end results with the goals of the project (Martin, 2022). Our project did not create any test cases for this final step as it is still in the beginning phases.

**Conclusion**

GitHub is a brilliant platform for software developers to manage builds, both solo and with a team. However, GitHub is mainly a version control system – not a fully loaded configuration management system. Luckily, due to the extreme popularity of GitHub, integration is often swift and easy with the vast majority of software developer applications. Therefore, the compilation of a custom headless CMS that integrates GitHub into a variety of other management systems is a common approach many developers take.

**References**

Atlassian. (n.d.). *What is Git*. Tutorials. Retrieved April 24, 2022, from <https://www.atlassian.com/git/tutorials/what-is-git>

GitHub, Inc. (2022a). *About repositories.* GitHub Docs. Retrieved April 23, 2022, from <https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories>

GitHub, Inc. (2022b). *Automate your workflow from idea to production*. Features • github actions. Retrieved April 23, 2022, from <https://github.com/features/actions?gclid=CjwKCAjwjZmTBhB4EiwAynRmD9yOTI9DXFWdv3GgAiWW8rSdtZA6RfETQ7rDsFs0-05gKeJxvb0EYBoCqfkQAvD_BwE>

GitHub, Inc. (2022c). *Quickstart for github actions.* GitHub Docs. Retrieved April 22, 2022, from <https://docs.github.com/en/actions/quickstart>

GitHub, Inc. (2022d). *Where the world builds software*. GitHub. Retrieved April 24, 2022, from https://github.com/about

Martin, M. (2022, February 26). *Software configuration management in software engineering*. Guru99. Retrieved April 22, 2022, from <https://www.guru99.com/software-configuration-management-tutorial.html>

Replit, Inc. (2022). *Code, create, and learn together*. replit. Retrieved April 23, 2022, from <https://replit.com/>

Tsui, F., Karam, O., & Bernal, B. (2013). *Essentials of software engineering* (3rd ed.). Jones and Bartlett Learning.

*View .docx file on github and use git diff on .DOCX file format*. Stack Overflow. (2014, March 16). Retrieved April 22, 2022, from <https://stackoverflow.com/questions/22439517/view-docx-file-on-github-and-use-git-diff-on-docx-file-format>