Software Configuration Management Plan

For Poppin by WEBMB

4/3/24

1. Introduction

This Configuration Management Plan (CMP) identifies the procedures for managing the configurations of the Poppin software during development and maintenance. This plan is intended for use by the development team during the software development activities. The limitations of this plan are a lack of any funding for the development and a significant time constraint of developing an application within one semester including software documentation.

1.1 Purpose

This Configuration Management Plan defines how the configuration items (CIs) will be managed within the Poppin application. It outlines the responsibilities, procedures, and activities for managing modifications and release of configuration items. The plan ensures cohesion and correctness of the application.

The development team will use Git as a configuration management tool for ensuring version control (VC). Git operates with a group repository in which the history of all changes to the project is saved. Git allows developers to make a pull request from a branch of the repository, make changes to the code, and push them once done with a certain task. The developer will then create a merge request to the current development branch, so that the history of the two branches is combined. When a development branch is completed, it can be pushed to the main branch of the repository. Git also allows the history of commit changes to be viewed by all collaborators for efficient tracking of development progress. Therefore, Git is capable of coordinating development progress through collaboration, and will be used to control access to different versions of the Poppin software in the development phase.

The development team will use GitHub for issue tracking during the development phase. GitHub allows developers within a repository to create “issues” which represent bugs and feature requests. Issues are created with a description and label, allowing for features of assignment and categorization. GitHub integrates issues with pull requests so that they can be linked for context and collaborators can reference issues within their commit messages. Therefore, GitHub will be used to track issues of the Poppin software.

1.2 Scope

This procedure is applicable to the personnel involved in the development of new and modified system software. As well as to all configuration items associated with the Poppin application. This plan is applicable to software development life cycle from the Development Phase to the Release phase.

1.3 Key Terms

- **Configuration Management Plan:** A document that outlines the responsibilities, procedures, and activities for managing configuration items throughout the software development life cycle.
- **Configuration Item:** Any component associated with a software project that has been placed under configuration management.
- **Version Control:** The process of tracking and managing changes to software so that all versions of components are stored for the system's lifetime.
- **Baseline:** A reference point in software development in which a configuration item is complete and ready for release. Baselines are controlled so the versions of the components that make up the system cannot be changed.
- **Branching:** The creation of a new codeline from an existing one, in which the two can be developed independently.
- **Merging:** Merging two versions in different codelines and in turn creating a new combined version.
- **Repository:** A shared database of software that stores multiple versions of software components and organizes them.
- **Release:** A version of a system that has been completed and released for use.

1.4 References

Standards: IEEE Std. 828-2005 Standard for Software Configuration Management Plans

Project Documents: Poppin Application Software Requirements Specification

2. SCM Management

2.1 Organization

Organizational context (technical and managerial) within which the configuration management activities are implemented.
The team behind the Poppin app integrates software change management into the development process using GitHub. The configuration management includes:

- Version Management: The Product Owner reviews and merges changes from different developers using Github's pull request system to prevent conflicts.

- Change Management: GitHub tracks software change requests for transparency and accountability throughout the development process.

- Release Management: This involves preparing the software for external release and tracking system versions released for customer use.

These activities are facilitated by GitHub and the structured organizational roles to ensure smooth collaboration and effective software development.

2.2 Responsibilities

Scrum Master:

- Responsible for project management and documentation.
- Facilitates meetings, updates project progress, and coordinates tasks.

Product Owner:

- Responsible for managing the GitHub repository and ensuring code quality.
- Reviews and merges code changes and oversees version control.

Development Team:

- Responsible for feature development and code integration.
- Collaborates on tasks, follows coding guidelines and participates in meetings.

2.3 Applicable Policies, Directives, and Procedures

- N/A

3. SCM Activities

3.1 Configuration Identification

3.1.1 Configuration items include all of our source code (App.js, Post.js, ProfilePage.js, etc.), requirements.txt, README.md, documentation (User Stories, Domain Model, Design Class Diagram, etc.) and sprint reviews.

3.1.2 Each configuration will be labeled with a standardized name, containing our group name, the document's sprint number, and the document's official name. For example, the first version of a SRS document from sprint 2 will be titled

WEBMB_Deliverable_2_SRS.docx If documents have multiple versions, they can be titled as following: WEBMB_SprintNumber_VersionNumber_FileName.FileType

3.1.3 Our configuration items will be on the Github repository WEBMB_Poppin, with separate branches for each week's code and documents. For example, we have the branches deliverable_2_documents and deliverable_2_development for Sprint 2. At the end of each sprint, code is merged to main while documents remain in their original branch.

3.2 Configuration Control

3.2.1 Requesting changes can be submitted through GitHub as issues, including the proposed change, the reasoning behind it, and what impact it will have.

3.2.2 At each meeting, we will set aside time to discuss any recent changes that have been requested on GitHub.

3.2.3 After group discussion, we can vote on change approval if there is contention about the issue. In the event of the tie, that week's Product Manager will get to decide if the change is approved or denied.

3.2.4 Based on what type of change is requested, the Scrum Master will assign the change to the team member(s) whose work is most closely related to the change.

3.3 Configuration Status Accounting

3.3.1 The configuration status will be tracked as the number of opened and closed change requests of GitHub, which can be viewed by any team member and will be assessed by the Project Manager. Also, we track progress on features and subtasks within each feature via the Sprint Backlog.

3.3.2 All team members can view this configuration status data via our shared GitHub repository.

3.4 Configuration Evaluation and Reviews

3.4.1 Before each major release, each configuration item will receive an audit to ensure compliance with the project specifications.

3.4.2 Our Product Manager will lead the group in an audit before a major release. The results of each audit meeting. Any updates to a configuration item to comply with the project specification will be documented in our team's system configuration management software.

3.5 Interface Control

3.5.1 Changes to configuration items are coordinated with interfacing items to ensure compatibility among the system. We will conduct meetings for this as needed.

3.6 Subcontractor/Vendor Control

3.6.1 Our project incorporates MongoDB, via Mongo Atlas Service. Our Product Owner will track how changes to MongoDB impact the quality of our product.

3.7 Release Management and Delivery

3.7.1 Our Product owner will compartmentalize our code and its dependencies for release via Docker.

4. SCM Schedules

4.1 Sequence and coordination of SCM activities

- The Product Owner will plan out when the different SCM activities need to be done and assign those tasks to either the development team or the Scrum Master.

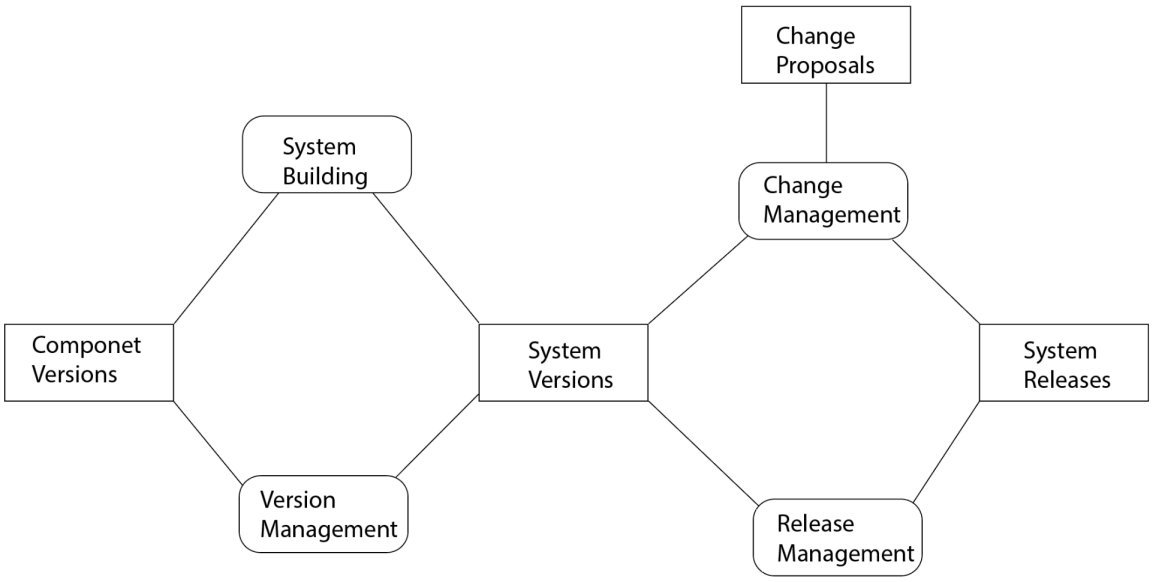
4.2 Relationship of key SCM activities to project milestones or events, such as:

- After the initial launch, the official set of files and code is determined whenever the Product Owner merges all the different branches into the main branch. Once that main branch is established, the software is then packaged up and released.

4.3 Schedule either as absolute dates, relative to SCM or project milestones or as sequence of events.

- The software configuration management activities will be scheduled every 2 to 4 weeks after the first version of the software is finished and released.

4.4 Graphical representations can be used here.



5. SCM Resources

5.1 Identifies environment, infrastructure, software tools, techniques, equipment, personnel, and training.

- All change management will be managed through Github, and MongoDB will handle the database functionality. Javascript is used for the backend of the application, along with the frontend of the application along with HTML and CSS. Training on these systems will be required for the Product Owner and all members of the development team.

5.2 Key factors for infrastructure:

- *Functionality, performance, safety, security, availability, space requirements, equipment, costs, and time constraints.*
- The system shall use MongoDB to handle data storage which allows anyone using the application to view all the posts. The system will be built to handle a post database of approximately 2,000 posts and will scale with the user base. We anticipate that the system will handle approximately 200 users at release of version 1.0.0.

5.3 Identify which tools are used in which activity.

- GitHub (Distribution Control Management System)
- MongoDB (Database)
- Node JS (Backend)
- JavaScript (Backend/Frontend)
- HTML (Frontend)
- CSS (Frontend)

6. SCM Plan Maintenance

6.1 Who is responsible for monitoring the plan?

- The Product Owner is responsible for monitoring the plan.

6.2 How frequently updates are to be performed?

- Updates will be performed every 2-4 weeks or as required.

6.3 How changes to the Plan are to be evaluated and approved?

- Changes to this plan will be submitted to the Product Owner for approval.

6.4 How changes to the Plan are to be made and communicated?

- Any changes made to the plan will be communicated to the team by text messages, and updating documentation stored in the Google Drive.

6.5 Also includes history of changes made to the plan.

- No changes have been made to this plan as of March 7, 2024

Changes	Date	Initials
Initial Plan Created(0.1)	4/7/24	WD