Software Configuration Management Plan

For

UMaine Connect by Maineframe (Group 2)

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this plan is to describe the configuration management strategies that will be employed in the development of UMaine Connect, a web and mobile social media application focused on social event coordination. The procedures outlined in this plan will guide development of the application and detail its organization and development process to Maineframe’s partners, clients, and other interested parties.

1.2. Scope

The scope of this configuration management plan is all software hardware and documentation that guides in the creation of UmaineConnect. The configuration management plan will be used to manage and carry out tasks for the development of UmaineConnect.

1.3 Key Terms

UmaineConnect: The name of the application that's being developed.
MaineFrame: The name of the organization that's handling the development of MaineFrame.

1.4 References

- GitHub
- Sprint Backlog
- Sprint Review

2. SCM Management

2.1. Organization

The organization developing UmaineConnect is MaineFrame. MaineFrame is an organization composed of 5 students from the University of Maine software engineering course COS 420. The organization's goal is to develop UmaineConnect over the course of one semester. The organization will be consistent and ensure effective development of UmaineConnect.

2.2. Responsibilities

- Product owner
    - The product owner creates tasks and determines when they are completed.
    - Sets up sprint goals and determines if they are valid.
    - Has the ability to cancel sprint goals if they are not needed anymore.
    - Supervises product backlog and sprint backlog.
    - *Period of effectivity:* One sprint, after which the project team will elect a new product owner.
- Scrum master
    - Establishes the rules and the plan for the sprint.
    - Creates issues in github to distribute work among all team members.
    - Communicate with outside parties if necessary.
    - Creates documents in the shared google drive that are needed for the sprint.
    - *Period of effectiveness:* One sprint, after which the project team will elect a new scrum master.
- Development team
    - Works on what is assigned to them by the scrum master.
    - Implements user stories in the sprint and updates them as the progress is made.
    - Works with the product owner to figure out the specific goals and when they are considered done.
    - *Period of effectivity:* One sprint or more: After each sprint, the team will elect a new product owner and scrum master for the following sprint. All other team members will be part of the development team.

2.3. Applicable Policies, Directives, and Procedures

As of right now UmaineConnect is not operating under external constraints. The only policy that UmaineConnect provides is the course. If any new external and internal policies are added in the future they will be updated here.

3. SCM Activities

3.1. Configuration Identification

3.1.1. Acquiring Configuration Items

There will be many different ways that the configuration items can be changed

- Survey of what's required by the system
- Communication with the team members and users or potential users.
- Will be based on the what the user stories are asking
- Observation of all the documents that have been established or created since the start of development.

All of these items will help ensure proper and effective development of UmaineConnect. These items will be updated in the future as development continues and any of the items that may not be need anymore will be removed.

3.1.2. Anatomy of Configuration Items

To help locate configuration items in the discussion of the developers all configuration items must include the following.

- A name for the item.
- An assigned number for the items.
- An in depth description of the configuration item
- A section for why an item is included.
- An update section to update the items as development continues

3.2. Configuration Control

3.2.1. Requesting Changes

Changes can only be requested by the members of UmaineConnect. Changes can be requested through the discord channel or in a Scrum meeting. A formal change request should include

- A descriptive title.
- Create an issue in github.
- Create a pull request(only for code base).
- An explanation of what should be changed and why it's necessary.
- Submission of background or research information on why that change will be helpful.
- A proposal of when and how the change will be implemented of the change request of accepted by the group
- The name of the person will also need to be provided who will make the changes.

3.2.2. Evaluating Changes

Changes will be discussed and evaluated in the scrum meeting when all the members are present. After the discussion the team will take a vote on if the change should be accepted or can propose an alternative. If it is decided to implement the change it will be added to the product backlog and assigned a Sprint number goal.

3.2.3. Implementing Changes

For a change to take place the majority of the members must agree with it. If the majority of the members don't agree with the change, the change request will be denied. In case of approval the person who is assigned to make the change will complete it. In both cases at the end the issue will be closed in github. If the change request is about a code base then it shall be done through a pull request in GitHub and reviewed by a member. After which it will be merged into the upstream branch and will close the issue.

3.3. Configuration Status Accounting

- Metrics to be tracked:
  - All metrics will be tracked and reported on GitHUb
  - Must be discussed with the team members
  - Implementation progress
    - The members who are given the role creating the configuration management will also be responsible for updating and keeping track of it.
  - Time to implement
    - The members who are working on the configuration management will keep track of the time spent on each sprint and report it to the sprint backlog.
- The access to the metrics will be exclusively only given to the members who worked on creating and updating it. Other members must request to update and must be approved by the members who created the configuration management.

3.4. Configuration Evaluation and Reviews

A formal audit process will occur involving one member from the team that did not work on the configuration management. The creators of the configuration management will pick this member.

Purpose of the audit

- Uncover bugs in the configuration management.
- Test the configuration management on different devices.
- Recommend changes that may be needed to the configuration management.

The audit may occur as many times as needed. Each time it must be done by a different member of the group. Any issues or bugs that are discovered during the audit must be reported and a change request must be created if necessary. Margin of errors should be outlined before the audit process begins. The configuration document should only be released when it's deemed fit by the creators of the configuration document. The release date of the configuration management should be given before the audit and can be changed based on the progress made.

3.5. Interface Control

Any interference of the configuration document outside of the scope will be handled very carefully and will be led by the product owner. As these changes will require different actions no instruction on how to deal with these changes be provided here but the interferences should be dealt with right away. There can be any changes made based on the interface and particular care must be shown so this does not affect the long term functionality of UmaineConnect.

3.6. Subcontractor/Vendor Control

Any development to UmaineConnect that is out of scope will go through the same audit that is described in 3.4. All the development will be managed by the developer and will be looked over by the product owner. The specific will vary on the development so the specific how it will be done will be given to the product owner.

3.7. Release Management and Delivery

Release policies regardless of type:

- The release date must be agreed upon by the members.
- Release will only occur after multiple audits are done and bugs are discovered.
- The release will be done at a low use hours to minimize the down time
- New release code will be put on to a new server instance, and then to “release” it, traffic from the CCNavigator url will be directed to this new server to minimize downtime.

There will be three types of releases each with its own policies.

Major Releases:

- This will be a full release.
- The user will be notified through advertisement.
- The user will receive notification on UmaineConnect about the release.

Minor Releases:

- Advertisements will be created to spread awareness about the release.

- The user will receive notification on UmaineConnect about the release.

Patch Releases:

- No advertisement will be made unless agreed upon by the product owner.
- The users will not be informed through notifications.

Further policies may be added depending on the context and the type of the release.

4. SCM Schedules

4.1. Sequence and coordination of SCM activities

A typical timeline of the project team's workflow and coordination of SCM activities during each two-week development cycle is detailed below. Due to the varying schedules and responsibilities of each team member, UMaine Connect's development process is not highly structured. Progress is made and reported continually throughout the development cycle.

Beginning of sprint (days 1-3)

- Team elects product owner and scrum master for the sprint; remainder of project team becomes the development team.
- Scrum master delegates responsibilities to team members on the Kanban board and sets up the directories and files that will be required for the deliverable.
- The project team assesses the progress achieved in the previous development cycle, as well as the project trajectory and goals, and updates the Kanban board as necessary.

Middle of sprint (days 4-13)

- Development and work on necessary documentation and revisions proceeds. Developers push their work to GitHub throughout the development cycle.
    - **Version management:** The majority of our code is pushed directly to the main branch, which serves as the baseline for the project. The main branch is given a new baseline number after each update, and a version history is maintained in the repository documentation.
        - Developers tend to work on separate components during the development cycle to minimize potential programming conflicts.
        - Separate branches may rarely be created for developing experimental features or features that conflict with previously established application features.
    - **System building:** As UMaine Connect is a web application, it is easy for developers to continually add and test new components. Components are

    - validated and stable component interaction is ensured on each developer's local machine before updates are pushed to GitHub.
- Team members may continually propose, review, and implement changes to the project.
    - **Change management:** Because the development team is small and each team member is typically involved with every aspect of the project in some way, change requests are proposed, reviewed, and implemented by the collective project team. The product owner, scrum master, and development team collaborate to assess and realize change requests.
- Configuration audit takes place.
- Team members may schedule meetings with others to work on tasks they have been delegated together.
- The project team schedules a scrum meeting at a time the entire team can be present to discuss development progress and work together on project artifacts.
- Issues on the Kanban board are closed as they are completed and tasks are moved to different pipelines as necessary.

End of sprint (days 13-14)

- A final scrum meeting is scheduled to assess development progress and ensure all required artifacts for the development cycle have been completed.
- The project team works together to complete any outstanding tasks. They may also move tasks to future development cycles if it is not feasible to complete them in the current cycle.
    - **Release management:** The project team ensures that there is a stable release of the software at the end of each development cycle. Emphasis is placed on steady progress and research; if the team determines a goal cannot be completed for the current development cycle, it will be moved to a future one rather than rushed to completion for the current release.
- The scrum master collects the project artifacts and submits them.

5. SCM Resources

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

- Development tools
    - VSCode
- Database tools
    - Firebase Database - NoSQL
- Environment tools
    - HTML4, Javascript, CSS 2.1
- VCS Tools
    - Git, Github
- Notable libraries (API)

  - Firebase

5.2 Key factors for infrastructure:

- The development tools we chose are considered standard in the industry for developing web applications, such as VSCode, Javascript, and the Firebase API.
- We chose Firebase due to the ease of setting up a NoSQL database and the ability to leverage a session manager and login credentials easily.
- Github was chosen due to our familiarity with it, as well as it being widely used for web applications.

5.3 Identify which tools are used in which activity.

- The Firebase database is used to store our user credentials as well as event data.
- VSCode is used for creating and editing HTML, CSS, and Javascript files.
- GitHub is used for version control and an easy one-stop-shop for sharing files between development members each week.
- Firebase API creates an easy pathway for data to be transferred such as user sessions and event management data.

6. SCM Plan Maintenance

This section describes how the plan will be maintained over the lifecycle of the project, including how revisions will be proposed and approved.

6.1. Plan Monitoring and Maintenance

The scrum master is primarily responsible for monitoring the plan. All team members, including the product owner and development team, will be responsible for maintaining and reviewing updates to the plan.

6.2. Update Frequency

The plan shall be reviewed at least once during every two-week development cycle and updated as necessary to reflect changes to changes in the project's configuration management practices.

6.3. Update Submission and Approval Process

Any team member may make changes to the plan, but they must notify the team and submit their revisions for approval. All revisions to the plan will be reviewed and approved by consensus of the entire project team. This is to ensure that all team members are familiar with and agree with each change.

6.4. Revision History

Date	Contributors	Purpose	Version
04/03/2021	Ayan Tariq, Nicole Cortez, Noah Brooks	Initial draft of the plan	1.0