CONFIGURATION MANAGEMENT PLAN

for

Risk Management System

Release 0.1

Prepared by Software Risk Tracking Team

Contents

List of Figures

List of Tables

Revision History

Date	Description	Revised by
3/6/24	Baseline	Software Risk Tracking CM Team

1 INTRODUCTION

1.1 Purpose

This document outlines the plan for Configuration Management (CM) for the Software Risk Tracking product. This document is intended for use by customers and stakeholders.

1.2 Definitions

Baseline	A minimum or starting point used for comparisons
Issue	A problem or challenged that arises during the CM process
Build	The process of assembling software components and source code into an executable or deployable form
CI	Configuration Items

1.3 References

https://cmpic.com/whitepapers/configuration-management-baselines.htm

2 CONFIGURATION MANAGEMENT

2.1 Organization

The CM function within the project will be to control how changes are made to the project, when and how releases are made, and how the changes and releases will be tracked. The CM Plan should be referenced by anyone working as a detailed description regarding how changes to the project are handled. Changes include merges into the master branch and any issues/bugs.

2.2 Responsibilities and Applicability

The CM Plan should be followed by anyone involved in the development process for the Software Risk Tracking project. The CM effort is being lead by the CM/SQA team, which is responsible for maintaining this plan as well as providing quality assurance for the software being produced and merged into the master branch.

3 CONFIGURATION MANAGEMENT ACTIVITIES

3.1 CI Identification

3.1.1 Software Development Library

This project will use Wt and Boost. Wt is a web GUI library in modern C++. It allows developers to quickly create highly interactive web UIs with widgets, without having to write a single line of JavaScript. Wt handles all request handling and page rendering for you, so you can focus on functionality.

Boost is a set of libraries for the C++ programming language that provides support for tasks and structures commonly used in C++ development. It aims to extend the functionality of the C++ standard library and address various aspects of modern C++ programming, including but not limited to smart pointers, concurrency, regular expressions, data structures, mathematical algorithms, and more.

The testing framework being used is CppUnit. This will allow for the creation of unit tests using an object-oriented approach in C++.

3.1.2 Project Baselines

UML Representation:

1. Initial UML Modeling: Begin by creating UML diagrams that represent the software's architecture, design, and key components.

2. Capturing Requirements: Ensure that the UML diagrams accurately capture the requirements and functionalities of the software.

3. Stakeholder Review: Conduct reviews with stakeholders to validate the UML representation and ensure alignment with their expectations and needs.

Construction Phase:

1. Code Implementation: Translate UML models into code through software development.
2. Version Control: Utilize Git to manage changes to the code base and track its evolution.
3. Continuous Integration: Implement continuous integration practices to ensure that code changes integrate smoothly and do not introduce regressions.

Reviews:

1. Code Reviews: Conduct regular code reviews to assess the quality, maintainability, and the adherence to coding standards of the implemented code.
2. Design Reviews: Periodically review the UML representation and its alignment with the evolving software requirements and architecture.
3. Iterative Feedback: Incorporate feedback from reviews into the development process to refine the UML representation and improve the code base.

3.1.3 Software Builds

At this time, the build process is still to be determined.

3.2 Configuration Control

3.2.1 Tools for Change Control

This project will use Git. Git is a version control tool used in software development. Git is a tool for tracking changes in software, allowing several engineers to collaborate in a non-linear fashion.

3.2.2 Procedure for Change Requests

1. Request changes: Anyone involved in the project, like developers or admins, can request changes. Requests can be submitted through a designated system, like an email or project management tool.

2. Reviews Change Requests: Requests are reviewed by technical leads or domain experts to determine their viability and possible consequences.

3. Approves Change Requests: Requests are usually approved or denied by the stakeholders or project sponsors.

3.2.3 Procedure for Changing Baselines

1. Baseline Changes: Baseline modifications are usually requested by all project participants, including developers, project managers, and stakeholders.

2. Reviews Baseline Changes: Requests for baseline changes are reviewed by a team of stakeholders or a designated group.

3. Approves Baseline Changes: Requests are usually approved or denied by the stakeholders or project sponsors.

3.2.4 Review Procedure

Software reviews will consist of documenting the change and creating a test for the new software. Issue tracking will take place within GiTea, using comments under the issues to make any notes or clarifications.

3.3 Status Accounting and Reporting

This section defines the type of events to be reported and accounted for, who they are reported to, how they are reported and recorded for documentation purposes, and the process that is to be expected upon a release.

3.3.1 Status Reporting

Among the three sub-teams collaborating on this project (View, Model, and CM/SQA), any moderate issues that may arise throughout the development of the project that could hinder the release of work products by the intended deadline (i.e. server errors preventing software from working as intended, difficulty in creating code for software increments that will meet stakeholders' expectations, or testing processes result in more frequent than expected software send-backs) must be reported to the sub-team leader or general management leader of the project. These key roles belong to Parminder Singh, Lucas Hasting, and William Reed for the View, Model, and CM/SQA sub-teams, respectively. The leaders can then in turn report to stakeholders in the event that a development increment cannot be completed by the intended deadline, if necessary.

It is the responsibility of not just the leaders but also every single team member to properly communicate any significant status updates in a timely manner though the favored use of Microsoft Teams in order to maintain the effective control of information throughout the lifespan of the project. Microsoft Teams allows the team as a whole to have one to two weekly meetings to report status updates, however, at any point in time any team member can post status updates to the chat log in between meetings.

3.3.2 Issue Tracking

For documentation purposes, tickets in regards to issues that may occur throughout the software engineering process must title in the format x-x.x.x, where x- refers to the number corresponding to the team who issued it (1: View, 2: Model, 3: CM/SQA), and x.x.x refers to the issued date. Tickets must also include the team member who issued it, a brief description of the issue/event that occurred, and a status parameter indicating whether the issue is still open or has been resolved. A log of these tickets can be compiled within a single text document accessible to any member.

Sub-team leaders can assess and manage these issues and communicate

with the sub-team members affected by the issue in order to resolve them. The CM team will persistently monitor the log for newly created tickets to ensure they become resolved in a timely manner.

3.3.3 Release Process

Official releases of software increments will include the current stage of the software, supporting documentation (including installation procedures), and test cases and results. Releases are expected to be deployed every Wednesday through the use of Gitea, importing the materials into a shared repository that can be accessed by stakeholders.

Any issues that have been discovered after a release can be reported to Lucas Hasting, who will redirect it to the team-members the issue pertains to.

3.4 CM Milestones

3.4.1 Baselines

Baselines will be established using spreadsheets. Progress compared to the initial baselines will be updated in team meetings.

3.4.2 Design Review

Design review will ensure that the software is meeting the requirements. It will be tracked and discussed during meetings.

3.4.3 Development Milestones

Development milestones will be tracked using spreadsheets and will be discussed at reported at team meetings. Development milestones include finishing coding and finishing final testing.

3.4.4 Testing Milestones

Testing milestones will be tracked from reports. Tests must meet the acceptance criteria that was discussed before tests were preformed. At a minimum, unit testing, integration testing, and acceptance testing will be performed.

3.4.5 Deployment Milestones

Deployment reports will be made as a record of how deployment went. Deployment must be completely successful, or it must be repeated.

3.4.6 Release Milestones

Release milestones will be tracked with user reviews. Release status will be documented in release reports.

3.5 Training

There will be no training required for the development team.