**HUrricane**

**Configuration and Change Management Report**

[Note: Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document.]

1. **Introduction**

[Briefly describe the content of the Configuration and Change Management.]

Change management is an essential step in a software engineering project. It provides a structured approach to achieving the desired final goal with optimal effort and minimal disruption. Since our project involves the development of a major software system, it requires periodic review to determine which features should be added, modified, or removed.

The Configuration Manager is responsible for overseeing both configuration and change management processes. This includes maintaining consistent versions of project artifacts (such as code and builds), ensuring stable and traceable configurations throughout the development lifecycle. Change management helps identify evolving needs and implement changes systematically. While essential features and critical use cases have already been defined in previous software requirements documents, lower-priority use cases may be revised or removed based on changing needs and project constraints.

1. **Purpose**

[Maintain a consistent set of work products as they evolve. Maintain consistent builds of the software. Provide an efficient means to adapt to changes and issues, and re-plan work accordingly. Provide data for measuring progress.]

The purpose of change management in our project is to ensure consistency with the final project goals. During development, numerous issues and compatibility errors have occurred, and given the nature of the development process, these challenges will continue to arise. One of the key objectives of change management is to maintain a safe and stable working environment for developers, allowing them to focus on delivering quality results.

Change management provides a structured approach to address these issues by enabling the team to identify features that need to be modified, added, or removed. For example, low-priority use cases may be adjusted or removed to accommodate shifting project requirements. Our configuration manager is responsible for ensuring optimal changes to the project by assigning and tracking issues, verifying their resolution, and ensuring that modifications are properly implemented. This process ensures that resources are effectively allocated and that the project remains aligned with its objectives.

1. **Configuration and Change Management Specifications**

[In many organizations, the term "configuration management" implies all of these things. Within the context of this process, configuration management refers to the ability to maintain versions of artifacts and consistent configurations of artifacts, addressing the first two objectives listed above. Change Management refers to the process of managing changes to configuration-controlled artifacts, addressing the latter two objectives listed above. Although it is important to keep up-to-date versions and configurations of all work products, the primary work products of concern are the Artifact: Implementation and the Artifact: Build. Changes are managed via the Task: Request Change and subsequent prioritization and disposition of change requests via the Artifact: Work Items List. This discipline spans the entire lifecycle. Every other discipline relies upon the configuration and change management discipline to maintain a consistent, up-to-date, set of work products -- and to prioritize and track changes to those work products -- throughout the lifecycle. Configuration and change management is performed by everyone on the development team. Because of the importance and pervasiveness of this discipline, configuration and change management guidance is associated with tasks and work products in all other disciplines. ]

While configuration management is about ensuring the most up-to-date and correct versions of project artifacts, change management refers to the process of applying tested and approved changes. Throughout the project lifecycle—from design to deployment—both processes must be continuously active. Since all software processes are interconnected, configuration management, change management, and testing are directly related disciplines.

The requirement analysis process continuously influences configuration updates, while testing activities play a central role in validating and guiding change management. In our project, configuration management focuses on maintaining consistent configurations of critical artifacts such as the implementation and build. Changes are managed through formal requests and are tracked using a work items list to ensure traceability and prioritization.

In our project, configuration management refers to maintaining consistent versions and configurations of key artifacts, primarily the Implementation and Build. It ensures that all work products remain up to date and aligned with the project's structure and goals. Change management, on the other hand, involves tracking and managing modifications to these configuration-controlled artifacts through structured processes like Request Change tasks and the Work Items List, where all change requests are prioritized and resolved.

This discipline spans the entire lifecycle of the project and supports all other processes by keeping artifacts consistent and properly documented. In our team, the Test Manager has defined potential test and issue cases, which serve as inputs for identifying and managing changes. The Configuration Manager leads this process by overseeing the coordination of tests, issue tracking, and implementation of change rules.

Although change management is a shared responsibility across the team, the Software Architect plays a leading role by guiding team members on how to implement necessary changes efficiently and consistently. This collaborative approach ensures that all changes support the integrity and progress of the project.

1. **Key Considerations**

[It is assumed that the project has some form of configuration management system, such as CVS, to maintain version and configuration information, and to enable collaborative system development. Without this, all but the most trivial of development will be virtually impossible.]

Our project’s configuration and change management processes were primarily managed through the GitLab platform. We started by checking our versions and feature deployments. Then, our maintainers—especially our Software Architect and Project Manager—guided us through the approval of deployment and build processes. To distinguish between versions, we used Git and its commit system. This approach kept our teammates synchronized and made the approval process easier.

We created separate branches for each developer in both our backend and frontend repositories to keep the main branch clean, accurate, and stable. All developers contributed to the build and development processes by pushing their changes. Although all our developers had the maintainer role, most approvals and merge requests were handled by the Software Architect and Project Manager.

We established rules for naming and commit messages to ensure clarity and to speed up the approval and build processes. Naming was not allowed to contain dummy values and had to be standardized so it would be understandable by both other developers and machines. Commit messages were required to be short, meaningful, and reflective of the essence of the change. Since all teammates followed these rules, we maintained a peaceful and effective working environment.

**5. Traceability Table**

|  | 1-2 | Spesification | Considerations | Summation |
| --- | --- | --- | --- | --- |
| Yusuf Küçüköner | 1h | 2h | 3h | 6 |
| Salih Eren Yüzbaşıoğlu | 1h | 3h | 1h | 5 |
| Şükriye Öztürk | 2h | 2h | 1h | 5 |
| Mustafa Furkan Ateş | 1h | 1h | 3h | 5 |
| Bedirhan Gençaslan | 1h | 1h | 3h | 5 |

**6. Prompts**

ChatGPT: <https://chatgpt.com/share/67f6f622-a110-800f-bf66-770da4cf1a1f>