ICESHU4
Configuration and Change Management Report

1 Introduction

Throughout the lifespan of a software system, it will be exposed to some changes. These changes can occur during both the development and post-development stages. These changes can be bugs that need to be fixed, new requirements of the system due to customer's request, or new features to add to the system. These changes should be managed carefully and adjustments must be planned.

Configuration Management is the process of identifying, organizing, and controlling the various components and artifacts that make up a system or product. This includes everything from hardware components, software programs, documentation, and other related items. The goal of Configuration Management is to ensure that every component of the system is properly identified, documented, and managed throughout its lifecycle. This process is critical for maintaining the integrity and stability of the system, as well as for ensuring that it remains functional and reliable over time.

One of the key components of Configuration Management is the creation and maintenance of a Configuration Management Database (CMDB). The CMDB is a central repository of information on each component and artifacts that make up the system. It includes information on each component's version, location, and relationship to other components in the system. The CMDB is essential for tracking changes to the system and ensuring that all components are properly identified and managed.

On the other hand, Change Management is the process of managing the changes made to a system or product over time. This process includes identifying the need for change, assessing the impact of the change, planning and implementing the change, and evaluating the results of change. Change Management is important for ensuring that changes are made in a controlled and consistent manner, with minimal disruption to the system or product and its users.

Configuration and Change Management are necessary for the inevitable changes to be implemented correctly and planned well. Lack of management in these adjustments results in a way that increases in cost and time, which can cause cancellation of the project.

2 Purpose

Some inevitable changes and updates can occur during the lifespan of a software system. Requirements change due to customer or environment, design changes to satisfy new and old requirements sufficiently, change in functions or classes to fix arising bugs, code refactoring to improve its performance and quality, new discovered security vulnerabilities that require update or change.

It is important to manage these changes effectively to ensure that the software remains stable, secure, and functional throughout its lifespan. Effective configuration management is a must because it ensures that changes are properly planned, documented and integrated into the software system.

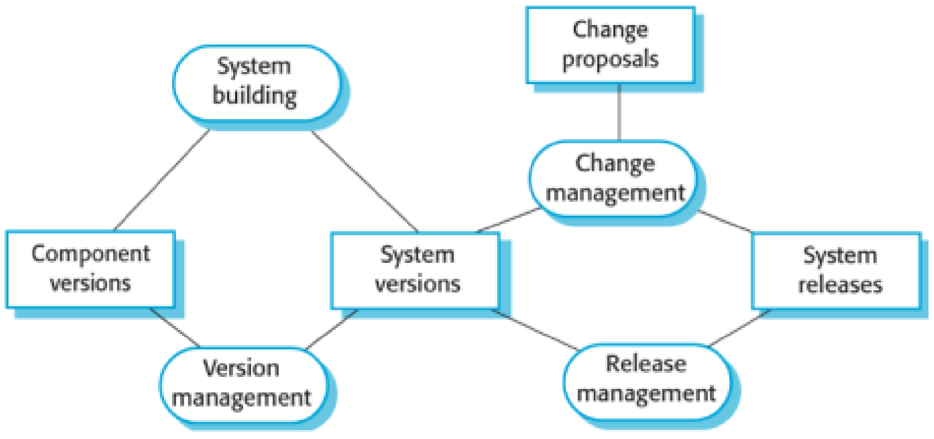
Our project consists of different modules from different technologies and concepts. It is too complex for one person to deal with all the parts of it and make critical choices for every part of it. We overcame this problem by using Github. Github is a web-based version control and collaboration platform for developers and free to use. Github makes it easy for team members to work separately on different parts of the project. By this method we overcome the complexity of the project and variety of technologies since every team member deals with the technology he is used to. It also helps us to work as a team.

Since every team member develops their part of the project, it can be hard to determine the effects of the changes made in the individual parts on the overall project. Also, team members can not be informed about the change. It causes conflicts in further implementations. With effective configuration and change management, we document these changes. It makes every team member informed about the change. The document provides an optimal way to adapt these changes to the system with the help of discussions with every team member. Discussing the changes ensures that every team member knows its effect on their part and how to deal with the change. It also improves team communication and causes strong interaction between team members. It results in an easier development process and a product of high quality.

As a result, the purpose of the document is to establish a set of procedures for managing the configuration and changes of software systems throughout their lifespan. Also it provides a clear and consistent approach to managing the changes to the software system. It will guide the team members to which actions they need to take when the change occurs.

3 Configuration and Change Management Specifications

Configuration Management consists of 4 interrelated activities. These are Change Management, Version Management, System Building, Release Management. We will provide a definition for all of them and describe how we apply them to our project.



Change Management

Change management generally includes the submission and review of change requests, impact assessment and risk analysis, approval of changes, documentation of changes, testing and verification, and deployment of changes. It also involves ensuring that changes are properly integrated with existing system components and that the system remains consistent and stable throughout the change process.

In our project, firstly we defined the change. After that, discussing if it necessary, must change. We take into consideration what happens if we change or which part of the project needs the change. If we determine the effects of change to every part, and we plan to do the change, then we investigate the cost. If the cost is acceptable then we prepare test cases for the change after implementation. Every team member is included in this process.

Version Management

Version management is an activity that involves the management of different versions of software system components, such as source code, documentation, and configuration files. The purpose of version management is to keep track of changes made to these components over time, enabling developers to collaborate on the same code base and manage the development process more effectively.

Since we are working with github, evey team member has own branch. If there is a need of new versions of components, the responsible team member create a new branch for him and update these components in this branch. After implementations and adjustments to the system on the new branch, we test the whole system with the changes due to new version. We decide to take the new version to the main branch if it passes all the testes. Main branch is where the latest stable version of the code is stored.

System Building

System building is an activity that involves the creation and management of software system builds. The purpose of system building is to produce a software system that is ready for deployment to users or customers, by compiling and linking system components, generating executable files, and packaging the system for distribution.

Since we are working on a complex project consisting of different kinds of modules, we separate the project into smaller ones and distribute these parts to each team member. So every team member deals with different parts of the project, thus we overcome its complexity. Since we are using Github, we can merge the work from every team member into one branch. It helps us to work as a team and make development easier.

Release Management

Release management is an activity that involves the planning, coordination, and control of the release and deployment of software system components, such as source code, documentation, and executable files. The purpose of release management is to ensure that software releases are properly tested, documented, and deployed, and that they meet the needs and expectations of users and customers.

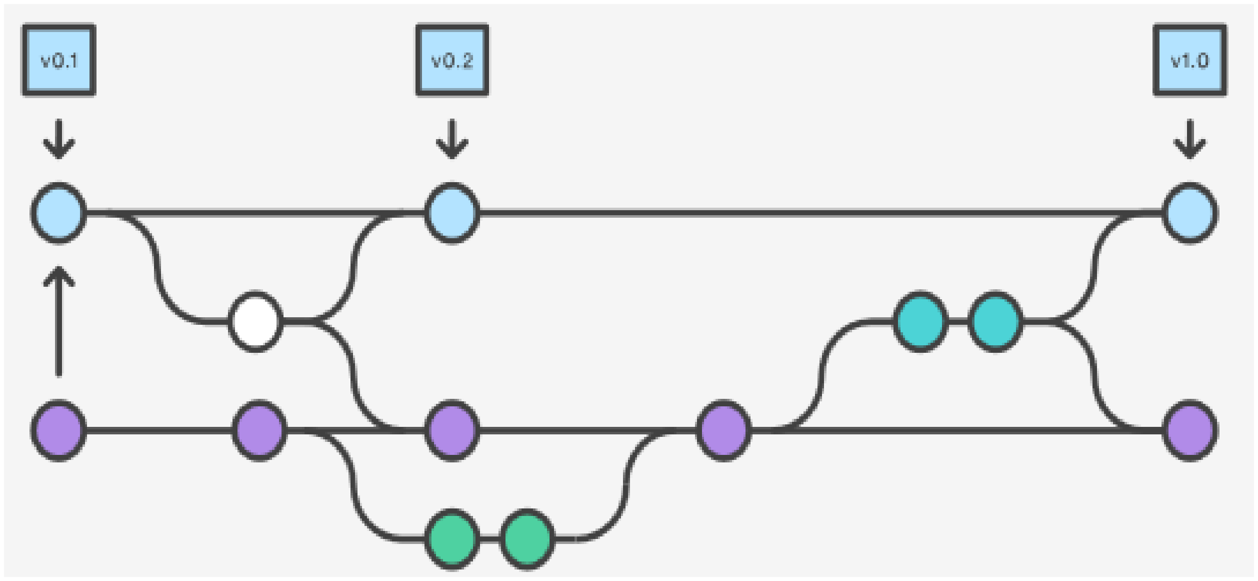
In our project, when we are planning the release we consider demands from customers, system requirements, environment changes, and technical qualıty of the system. If our latest product satisfies the needs of these factors, we decide to release it. Github keeps track of release dates and the changes done. Also we documented the changes and the specific features of release. Since the project is not done yet, we track the release to fix bugs and detect not sufficient parts of the release. We make adjustments according to them in the new release.

4 Key Considerations

Configuration management involves the management of a large amount of information and the relationships between configuration items. To handle this complexity, tool support is essential for storing different versions of system components, building systems using these components, tracking the releases of system versions, and managing inevitable changes. With the help of configuration management tools, developers can ensure that software releases are properly managed, tested, and documented, and they meet the needs and expectations of users.

In this project we are using Github as a configuration management tool. Github allows developers to easily manage different versions of software components, making it easy to track changes, manage different branches of development. Since Github provides a centralized repository for storing code, it is easy for developers to collaborate on projects and share code. Also Github can be easily integrated with a wide range of other development tools and platforms, including project management tools.

We follow the Github flow in order to manage implementation, which is shown below.



This Github flow forces us to create a branch for every team member and make them responsible for their part. But we need to merge this parts and should keep up to date with each branch. There are some processes to handle this in a manageable way.

Create Branch: Branches allow team members to isolate changes they made to the codebase, making it easier to test and review changes without affecting the main codebase. Branches can also be used to collaborate on new features or improvements to the software. Every team member merges their changes to the main branch after consideration. Also, branches can be used to maintain stable releases of software, allowing team members to fix bugs or make security updates without affecting the main branch.

Add Commits: During the development stage, too many changes happen on the project. Commits allow team members to record these changes, it includes bug fixes, feature enhancements, and updates. This makes it easy to track the history of the codebase, and see who made which changes or updates. Commits can also be useful when it comes to reviewing projects and giving feedback.

Open a Pull Request: Pull request allows team members to request code review on the changes or updates he made from other team members or directly from the project manager before merging his code to the main codebase. The detailed comment about changes should be included in the request, so other team members can easily understand changes made.

Review of Code: After a pull request, all the team members or only the project manager review the code and discuss it. The request should be updated or adjusted with the information from feedback. Then the team must decide if the request is acceptable or not. This helps us to ensure that changes are discussed thoroughly and tested before merge operation.

Merge: After all controls are checked, if the team decides the request is acceptable, then the project manager merges the request to the main branch. Since github keeps track of all pull requests, we can go back to the stage before merging if the request causes some problems in the future. Also the detailed comment of request helps us to remember the change and why we made the change.

Deploy: To ensure an organized deployment phase, a release branch is created and used for development purposes. Once development on this branch is finished, the project manager will have the authority to accept the merge and initiate the deployment process. In this way, we minimize the potential risks and provide controlled deployment phases.

In our project, team members follow the flow of Github. Every team member creates his own branch and works on it. After the development finishes in his branch, he tests his code to see if it is working correctly. He created a pull request with detailed comments to merge his code to the main branch. Other team members or just the project manager, review and discuss his code. Pull request owners can be asked to update or adjust his code. If a request is accepted after the review, the project manager merges it into the main branch. After the merge operation, if there are conflicts with other parts of the project, team members discuss the request again and responsible ones solve the conflicts. When all the issues are handled, the project manager makes the deployment. Every team member should update their branch with the main branch. This process helps us to organize the development stage and cause effective communication between team members.

5 Traceability Table

Traceability Table	Umut Güngör	Numan Kafadar	Mustafa Çağrı Korkmaz	Yunus Emre Terzi	Osman Faruk Derdiyok
Configuration and Change Management Report				X	