Social Interest E-Club Configuration and Change Management Report

1 Introduction

In this document we explain fundamentals of the configuration and change requests as it stated with a report template format. It involves the process of controlling the changes that are made to correct errors recording and reporting with the change requests and verifying the completeness and correctness of the program.

Configuration is the way components are arranged to make up the computer system. Configuration consists of both hardware and software components and it thereof as set forth in technical documentation and ultimately achieved in a product. Therefor "Configuration Management" is a technique for managing the development, manufacturing, and maintenance of configuration items that provides a technical and administrative structure of the system. Configuration management enables the construction of a system, subsystem, or configuration item in a systematic manner.

About changes, software systems, programs are always in need for changes to fix the problems, errors and the bugs of the system. Especially in their development stage of the system. For this reason, systems must be suitable for the changes in the future and in the current problems. Furthermore, it is necessary for a system to be prepared for new features and changes in terms of innovations that should be added in the future.

2 Purpose

The purpose of this report is to archive and document how the configuration and change management issues were or will be handled by our project group during the whole development of the project itself. First off, we started with choosing to use “GitHub” for controlling and managing the progress of the project by each member of the group. Through this site, we are able to share our coding progress with each other by pushing it to our own shared group repository. Also, other developers can view other developers’ version of the code in the repository very easily. For this usability reasons, GitHub was the first choice for this project. Apart from GitHub, messaging with developers and working together throughout this development process creates better working space for this project since information about the project can be communicated thoroughly.

3 Configuration and Change Management Specifications

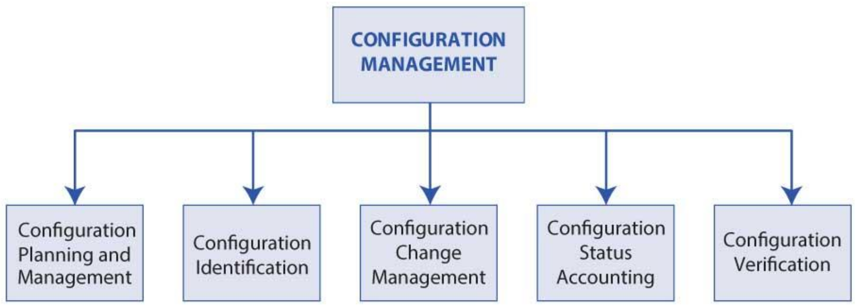

CONFIGURATION MANAGEMENT
Configuration Planning and Management
Configuration Identification
Configuration Change Management
Configuration Status Accounting
Configuration Verification

FIGURE 6.5-2 Five Elements of Configuration Management

The process of maintaining computer systems, servers, and software in a desired, consistent working state is known as configuration management is also a major component of change management here. Change management is the action of choosing which changes to encourage, which to allow, and which to prevent, according to project criteria such as schedule, cost etc. between hundreds of changes that could be applied in the project.

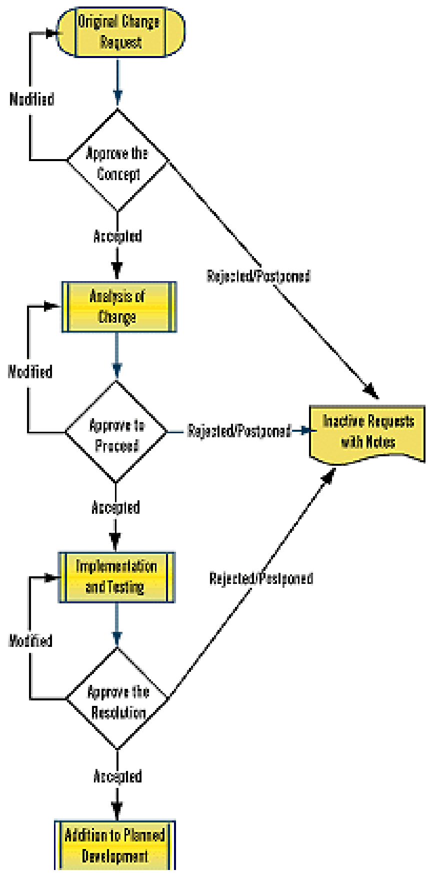
Both managements maintain a consistent set of work products as they evolve to the version that is planned for the project in the end. Three main phases of the system developing life cycle(SDLC) applied for this project were implemented as follows:

Development Phase:

In this phase, changes, especially the major changes, that should be take place in the project, cannot be decided by only one person in the group since it is a teamwork project. For our project, Team have done the plans about the software configurations at this phase. We made sure that all these changes discussed with all the members to choose the best and most effective ones for the project in every way. (Choosing main server system vs program language)

System Testing Phase:

In this phase, a version of the system that is wanted to be tested by the developers gets tested to see the possible issues that can come to the surface while running it. These system tests are the most important part of change management considering that the results of the tests are the big decision-maker for the changes that we should make in our project. Results of the tests get analyzed by the tester of the group and all the other members of the project team used these to draw a better path. This path is drawn with the changes that are going to be applied to the project to make it the version we planned for the release. Every change approved by the project manager of our team ultimately. For example, we started writing the code using MySQL for the database system but then after the tests and the meetings we had as a group, we decided to change the database system to PostgreSQL. PostgreSQL has so many better options for java spring writing and more efficient with comparing it to MySQL and all the other database system we knew.



For these kinds of changes, we followed the flow chart that is stated as follows:

Approve the Concept: This is the first key decision point in any change management process. Change requests can come from tester that made tests on the program or other members of the project development group with identifying the problems they came upon. Also, it could be from the stakeholders' adding or changing requirements. In our group, with every change request, we made discussions in meeting and if this change idea accepted with approval from the developers and project manager, next decision point would be the "analysis of the change".

Approve to Proceed: Once we have accepted a change request, we need to make sure that the changes we make must fulfill our project's current requirements, specifications, and designs. Also, these changes should not make big differences on our project budget. In our project, we do not have a big budget, so we had to go with the affordable one always. Sometimes, we as the team will discover that a complex problem has an easier solution than we expected or that several bugs have a common resolution. Analysis and management make a big part in the process of the changes here.

Approve the Resolution: A change request is completed when it is implemented for testing and this change fulfill the prompt version with the result of the tests. During our coding part, we always must do testing on new changes to verify the results of it make our coding better or suitable. If these changes cause any new errors or problems, we go back to implementation and testing part with modifying it for new tests.
If everything goes accordingly to our plan, we add these changes to our project. If not and it gets rejected, or postponed, we write those down to our project notes to look through them to guide us through if we came across the new ideas like that again.

These notes help us to remember that why we decided against it or postponed it o future time before. And, if circumstances change, or the time for these postponed requests arrive, we can go through these notes again to create new change request from beginning again.

Deployment Phase:

In this phase, we deploy our project's final version that is decided to release for the stakeholders for the use. But even after the deployment, stakeholders can still find and report any kind of bugs in the system, so developers always need to be ready for new changes again. These changes would be depended on stakeholders' feedback. Four our project, we released our first demo and with the feedbacks we get, we go back to the development and system testing phase again to modify our codes to new versions. These new versions are upgraded versions of our previous (demo) version of our code.

4 Key Considerations

We are using GitHub to keep track of the changes we made in the code by each members of the group. GitHub also provides a tool that called "Git tool" that is hosting for controlling versions over GitHub. With this tool, a team member can pull the version to their workspace to work and them and push it again to the shared repository with the Git tool in the easiest way.

As a team of four, two of us develop the frontend while two of us develops the backend part of the program here. We first planned this shared repository with multiple branches for each member but then we reduced by two branches as feedback and main for now. For the first demo, we merged and pushed codes to the master branch. We wrote codes collectively with helping each other with any kind of issues or error that got in while writing it. But we also try to write codes easily readable and understandable so everyone in the group or anyone, who is reading the code or working with the program, can work with it more effortlessly. So, feedbacks can be provided by GitHub and it is understandable and far more effective than any other options.

Working as a team with every change and every modification was easier with the meetings and the discussions we make for analyzing the project and solving the errors we came across. As a group of developers together, we make sure that finding every change we need to make our project better and closer to the final version we desired for the release of our project.

5 References

https://www.nasa.gov/seh/6-5-configuration-management/

https://www.drdobbs.com/software-change-management/184415707