Social Interest eClub
Configuration and Change Management Report

1 Introduction

Configuration management is the process of managing the configurable components or resources of a system or environment on which a software application runs. Configuration management ensures that these resources and components maintain a consistent state; this consistent state is referred to as a baseline.

Change management is the process of managing the changes that are made to the configurable items in an environment or system.

The major change between configuration and change management is that configuration management focuses on managing the configurable items and the state of the system while change management focuses on managing the changes that affect the configurable items and the system.

2 Purpose

A project changes in parallel to the development. After some time, the developer team might want to (or have to) carry out some changes on the project. Sometimes, even predetermined customer requests might change, and the project team has to handle this situation. These changes are inevitable and considered very common in the development lifecycle. This is where the Configuration and Change Management get involved.

Configuration and Change Management is very useful when maintaining works from multiple people, and continuing the development lifecycle without errors resulting in a stop in development.

In our project, the purpose of implementing Configuration and Change Management is to maintain the development phase, without time loss due to the changes and configurations on our project. For this to happen, all team members will be particularly careful about continuing the development process without any error by informing regularly about the configurations and changes they have made.

The purpose of this document is to define the principles, and factors to consider when maintaining Configurations and Changes in the system without causing an unexpected error.

3 Configuration and Change Management Specifications

Configuration management is concerned with the processes, tools and policies for managing changing software systems. The figure below shows the Configuration Management activities:

(The activities that are in ellipse(not rectangle) are what we will implement in our system. The other ones are sub-implementations.)

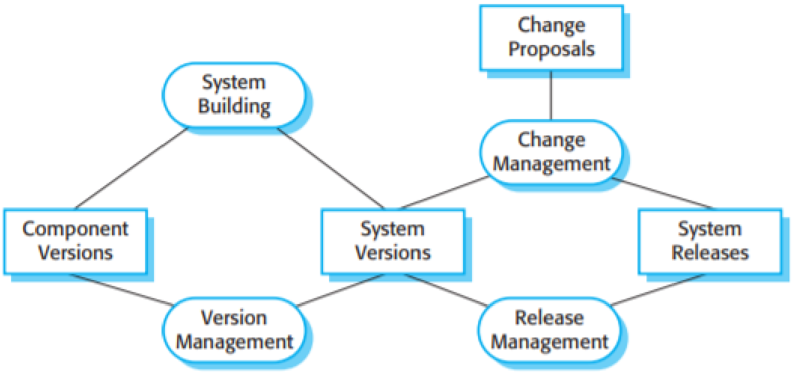


Figure 25.1 Configuration management activities

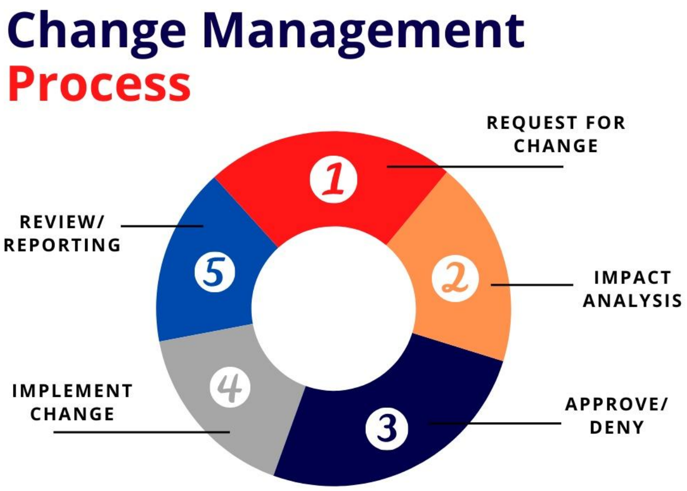
- **Change Management:** It is about keeping track of requests for changes to the software from customers and developers, working out the costs and impact of making these changes, and deciding if and when the changes should be implemented.
  In our system, this will be implemented by GitHub features like issues, commits, pull requests.

- **Version Management:** It is about keeping track of the multiple versions of system components and ensuring that changes made to components by different developers do not interfere with each other.
  In our system, this will be implemented by GitHub branching feature.

- **System Building:** It is the process of assembling program components, data, and libraries, and then compiling and linking these to create an executable system.
  In our system, this will be implemented by Android Studio, Android SDK, etc.

- **Release Management:** It is about preparing software for external release and keeping track of the system versions that have been released for customer use.
  In our system, this will be implemented by team members, by testing, getting feedback on project, issue reporting and bug fixing.

Change is a fact of life for large software systems. Organizational needs and requirements change during the lifetime of a system, bugs have to be repaired, and systems have to adapt to changes in their environment. To ensure that the changes are applied to the system in a controlled way, you need a set of tool-supported, change management activities. The figure below shows the Change Management activities:



- **Request for Change:** Request for Change might emerge either by customers or developers. These requests should be collected and considered to be implemented carefully in later stages.
  In our system, this will be implemented by developer team members. Since our project will not be used by real customers, getting feedback from other people will not be possible.

- **Impact Analysis:** Impact Analysis is about analyzing the difference between the old and the new design of the project. Determining the changed roles between the actors in the system is very crucial.
  In our system, this will be implemented by Brainstorming among team members. The potential consequences of the change will be broadly considered by the project manager and the team members.

- **Approve/Deny:** Approve/Deny of the change is a result of Impact Analysis phase.
  This will be implemented by team members, by interpreting the result of the Impact Analysis. If the change is approved, then it will start to be implemented in the system. If the change request fails, it will either be modified or completely suspended.

- **Implement Change:** This phase is all about implementing the change in software, by coding.
  This will be implemented by the developer team. All the team members will implement changes locally, and then commit their changes to the VCS(GitHub).

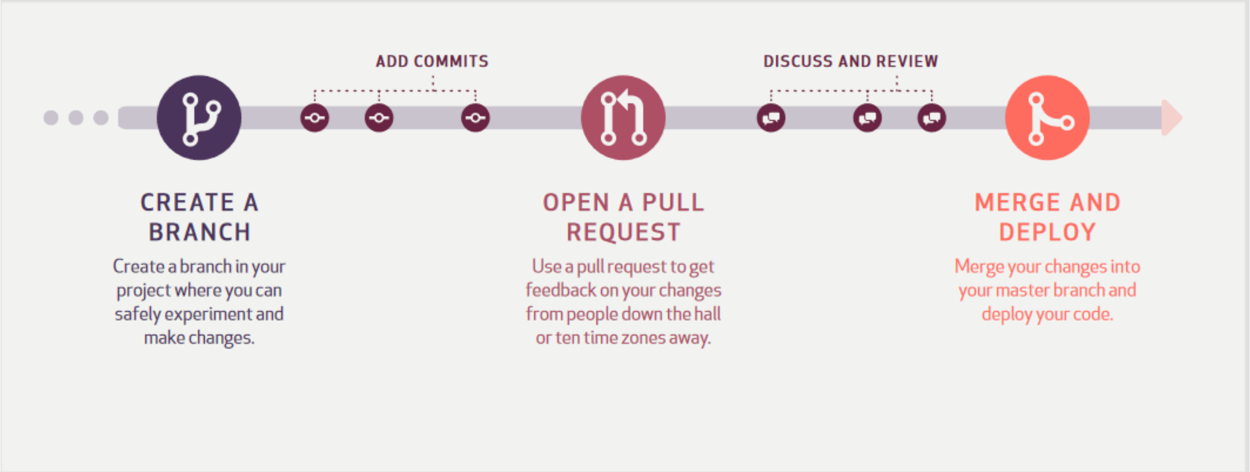
- **Review/Reporting:** This phase will be implemented both during and the end of the “Implement Change” phase. During development, the developer team will report the effects of the changes they have made to the system. After development is finished, tests and reports will be formed to get informed about the current status of the project.
  This will be implemented by the developer team and the configuration manager.

4 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.]

In our project, GitHub VCS (Version Control System) is used in every Configuration/Change operation. GitHub flow is a branch-based workflow that supports teams and projects where deployments are made regularly. In our system, we also use GitHub Desktop since it is very handy and practical to use.

Below figure is the structure of the GitHub Flow:

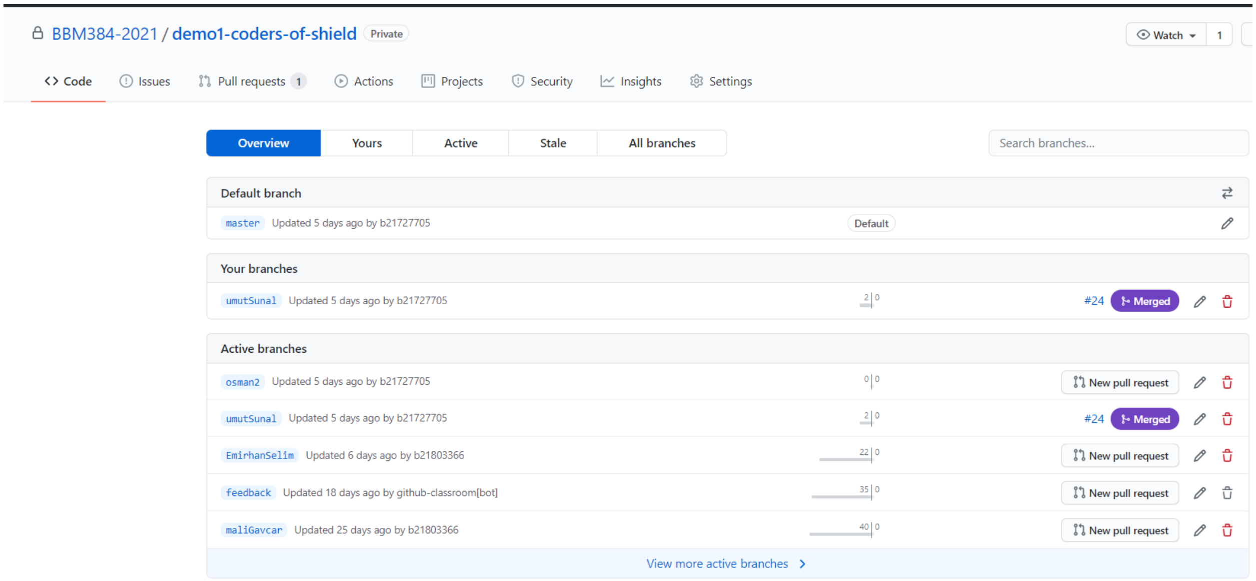


Create a branch in your project where you can safely experiment and make changes.

Use a pull request to get feedback on your changes from people down the hall or ten time zones away.

Merge your changes into your master branch and deploy your code.

1. **Create a branch:** In our project, we develop our code in our own branches. Firstly, we open a local branch with our own names to keep track which team member developed which part.



Default branch

master	Updated 5 days ago by b21727705		Default

Your branches

umutSunal	Updated 5 days ago by b21727705	2 | 0	#24 Merged

Active branches

osman2	Updated 5 days ago by b21727705	0 | 0	New pull request
umutSunal	Updated 5 days ago by b21727705	2 | 0	#24 Merged
EmirhanSelim	Updated 6 days ago by b21803366	22 | 0	New pull request
feedback	Updated 18 days ago by github-classroom[bot]	35 | 0	New pull request
maliGavcar	Updated 25 days ago by b21803366	40 | 0	New pull request

Above figure shows our branch in GitHub. These branches are used to open a pull request.

2. **Add commits:** To keep track of changes, commits are used in the system. Commits are development history logs of a project, and very useful to understand the development phase.

   In our system, commit titles are about the changes made in the system. Commit messages are used to give detailed information about the changes that are mentioned in the title. Since each commit is revertable, the project might be rolled back whenever a bug is encountered in the system.

3. **Open a pull request:** After making changes and adding commits to their local branches, team members could open pull requests to push their changes to the origin(outside local). Project managers, or authorized team members can approve, deny or ask a revision about the request.

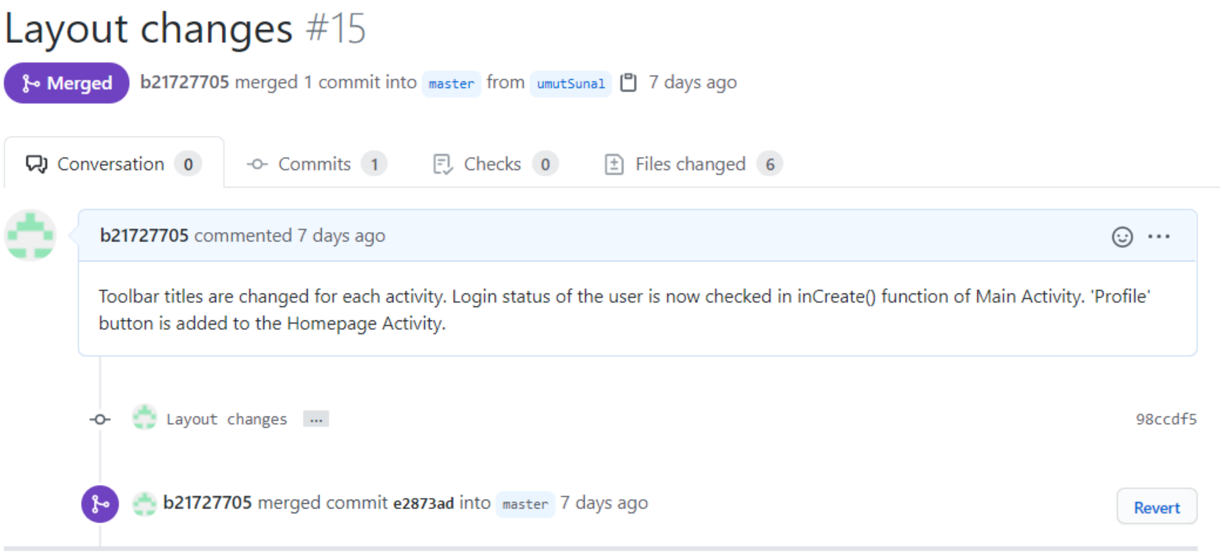
   In our system. team members will open a pull request from their local branch to the origin.

4. **Discuss and review:** Opened pull request will be inspected, and approved if changes are useful.

   In our system, we will discuss and review the changes together with team members.

5. **Merge and deploy:** For each pull request, after the "Discuss and Review" phase, pull requests and its commits will be merged to the master branch if the pull request is appropriate.

   After all development is completed, the code will be deployed to the users.



Layout changes #15

Merged b21727705 merged 1 commit into master from umutSunal 7 days ago

b21727705 commented 7 days ago

Toolbar titles are changed for each activity. Login status of the user is now checked in inCreate() function of Main Activity. 'Profile' button is added to the Homepage Activity.

Layout changes ... 98ccdf5

b21727705 merged commit e2873ad into master 7 days ago

Above figure shows an approved/merged pull request in GitHub. The pull request has 1 commit, and the commit has an informative title and message about the change made in the system.

5 References

- BBM382 Lecture Notes
- https://www.brighthubpm.com/change-management/39825-configuration-and-change-management/
- https://guides.github.com/introduction/flow/
- https://teambuilding.qa/team-coaching/change-management/