Notes Software Engineering

Brendon Mendicino

March 3, 2023

Contents

1 Introduction

Definitions:

Definition 1.1 – Multi Person Multi Version

People coordinating in long period of time.

Definition 1.2 – Software

Is a collection of code and not only digital assets like: rules, documentations, procedures and many more.

Definition 1.3 – Software Types

- **Stand alone**: products used alone, like email, office, calendar;
- **Embedded in software products**: car, smart house;
- **Embedded in business process**: an information system;
- **Embedded in production process**: embedded in factories and classic production pipelines;

1.1 Describe Software

To describe a software we define his properties, they are divided in **functionality**, which express an action to be performed, and **non-functional**, this property describe how a functional property should behave, defining his correctness, and define his reliability, the idea is that correctness it's impossible to obtain, so it tries to limit the number of defect, setting a line that allows the threshold for the number of defects, on the other hand the availability is the percentage, over a period of time, of the system without occurring in any defect. Other non-functional properties are security, safety and deniability. Efficiency is the response time and the amount of resources used.

Every software has process: development, operation, maintenance. During development there are 4 main phases:

- **requirements**;
- **design**;
- **coding**;
- **testing**;

Basic rules:

- keep it simple;
- separation of concerns;
- abstraction

2 Git

Git uses distributed CMS, uses snapshots of the current files, other than that git also has integrity with a checksum. Basic git concepts:

- **repository**: contains all the files and versions of the project;
- **working copy**: it is a snapshot of the repository, the working copy is on the client side;
- **commit**: it is an atomic operator that modifies the repository, all commits are tracked into a log file, to every commit a message is associated;
- **push**: is an operation that updates the modifications from a local server to the online server;
- **update**: updates the working copy by merging the changes;
- **staging area**: it is local dock that stores the changes that are not committed yet;
- **typical workflow**: checkout project, stage/commit changes, push;

Git basic commands:

- `$ git init`: initialize a local repo;
- `$ git remote add origin http://server.com/project.git`: add a new remote repository;
- `$ git status`: status;
- `$ git add`: add a file;
- `$ git diff`: changes;
- `$ git commit -m "..."`: commit changes;
- `$ git commit -am "..."`: commit all changes;
- `$ git rm`: removes a file with git tracking, adding the staging area;

- `$ git mv`: moves a file adding modifications to staging area;
- `$ git log`: logs;
- `$ git pull/push`

Every commit points to a tree which contains a list of modified files, it's possible to reach the **blob** of the changes made to that file via a pointer, every commit is linked to the previous one. The last commit of the current working branch is called **HEAD**, every HEAD points to a branch that we last commit to. If we want to switch branch we need to checkout the branch, and the HEAD will change which branch it is pointing to. If there are many branches we want to bring the changes of a branch to our current one, we need to use `git checkout other-branch`, when merging two branches some **conflict** can be created, when this happens the changes need to solved by hand, in other case git is able to fix them on his own.

Other than merge there is also **rebase**, which is a bit different that merge, it tries to rewrite the history of the branch we are rabasing to and to commit all our changes to the last commit. The difference between merge and rebase, is the rebase it creates a linear commit history, while merge is keeps traks of every commit where done in different branches.

- `$ git reabase -i HEAD~4`: -i $\implies$ interactive mode, $\tilde{4}$ $\implies$ number of commits we want to target;

Example 2.1

Fork and pull is the way to go for open source development, the operation of fork copies the full project in your account and thus becoming the new owner, the two projects are separate unless it is synched to the original one.

Definition 2.1 – Conventional commit message

- **feat**: