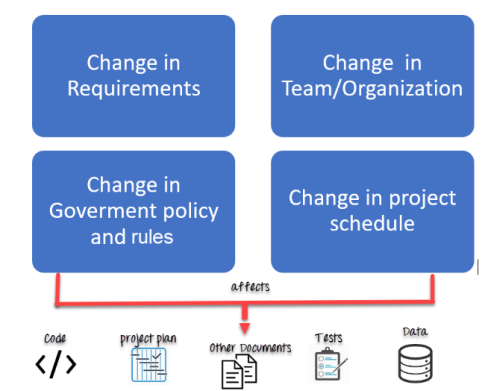
Lecture No. 30

Software Configuration Management

Software Configuration Management is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. It is abbreviated as the SCM process in software engineering. The primary goal is to increase productivity with minimal mistakes.

**Why do we need Configuration management?**

1. There are multiple people working on software which is continually updating
2. It may be a case where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently
3. Changes in user requirement, policy, budget, schedule need to be accommodated.
4. Software should able to run on various machines and Operating Systems
5. Helps to develop coordination among stakeholders
6. SCM process is also beneficial to control the costs involved in making changes to a system



**Software Configuration Management Plan**

The SCMP (Software Configuration management planning) process planning begins at the early phases of a project. The outcome of the planning phase is the SCM plan which might be stretched or revised during the project.

* The SCMP can follow a public standard like the IEEE 828 or organization specific standard.
* It defines the types of documents to be management and a document naming. Example Test\_v1
* SCMP defines the person who will be responsible for the entire SCM process and creation of baselines.
* Fix policies for version management & change control
* Define tools which can be used during the SCM process
* Configuration management database for recording configuration information.

**Activities in Software Configuration Management**

* Change management
* Version Management
* System Building
* Release Management

**Change management**

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

**The processes of change management:**

Identification of needed change: The configurable Items are evaluated to determine the changes they need and where and how those changes can be implemented.

Determining the impact of the change: It is determined how the proposed change will impact the current state of the system.

Change request: After the impact of the proposed changes has been determined, a change request is made the appropriate stakeholders.

Change strategy: Once changes are approved by stakeholders, a strategy is developed for how the changes should be implemented and the time frame. The strategy is aligned with the guidelines defined for configuration management.

Execution of change: After an appropriate strategy has been successfully defined, the changes are implemented, and the state of the system after the changes if stable, becomes the new baseline.

**Version Management**

If a system integrator is used to doing things manually and saving files and other information with a myriad of prefixes, the jargon and ways of using the version control system can be daunting. It is a good idea for system integrators to introduce these concepts step by step so they are not overwhelming.

In the first phase, they could introduce version controls as a backup system. The goal in this phase is to provide a standard way of storing all project data under version control and introduction of the version control paradigm. In this phase, version control is used mostly for sync and backup capability and also its ability to roll back any file to its previous known state. In version control lingo, we would mostly be working with commit and push/pull scenarios. This presents a nice segue to discuss the tools parts of the version control.

There are a lot of options here (Git, TFS, Bitbucket etc.)

**Baseline:**

A baseline is a formally accepted version of a software configuration item. It is designated and fixed at a specific time while conducting the SCM process. It can only be changed through formal change control procedures.

Activities during this process:

* Facilitate construction of various versions of an application
* Defining and determining mechanisms for managing various versions of these work products
* The functional baseline corresponds to the reviewed system requirements
* Widely used baselines include functional, developmental, and product baselines

In simple words, baseline means ready for release.

**Intro to Github**

It allows groups of developers to collaborate on the same documents (often source code) simultaneously and without overriding each other’s work.

Have you ever worked on a Word document with somebody else that may be stored in say, Dropbox, only to discover that you both worked on it at the same time? What happens? Often, you override each other’s changes, download conflicting copies or simply lose your collaborator’s work. Git helps developers alleviate all of these issues.

Git also tracks the history of changes to a project’s source code, including what specifically has been changed, who has changed what and when. This is called **Version Control**.

**Difference between Git and GitHub**

Git is a distributed version control tool that can manage a development project's source code history, while GitHub is a cloud based platform built around the Git tool. Git is a tool a developer installs locally on their computer, while GitHub is an online service that stores code pushed to it from computers running the Git tool.

**Git Repositories**

If we want to start using Git, we need to know where to host our repositories. A repository (or “Repo” for short) is a project that contains multiple files. In our case a repository will contain code-based files. There are two ways you can host your repositories. One is online (on the cloud) and the second is offline (self-installed on your server). There are three popular Git hosting services: GitHub (owned by Microsoft), GitLab (owned by GitLab) and BitBucket. We’ll use GitHub as our hosting service.

**Track changes in your code across versions**

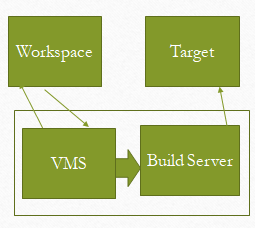
When multiple people collaborate on a project, it’s hard to keep track of revisions — who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository. Much like using Microsoft Word or Google Drive, you can have a version history of your code so that previous versions are not lost with every iteration. It’s easy to come back to the previous version and contribute your work.

**Install git and create a GitHub account**

The first two things you'll want to do are install git and create a free GitHub account(Github accounts are free for public repositories).

**System Building**

* Assembling all s/w components to make a working system
* Check previous s/w build from VM and decide what should be next build
* Keep track of testing enviroment
* Check and balance of Timestamp
* System building and version management systems must communicate
* Configuration file is used for creating baseline
* Jenkins



Development System Build Server Target Enviromint

**Release management**

Release management is a software management process which guides the technology efforts from code development through testing and into production, focusing on coordinating pieces from various product deliverables that must come together to work as an integrated solution, & results into an effective delivery of new and enhanced IT services/functionality required by the business, while protecting the integrity of the existing ones. Needs change management to approve changes and track them throughout the release process. Needs configuration management to assess the impact of changes to CIs and to provide a definitive store for the release package.

* Keep track of releases to customer
* Keep track of all other kind of releases
* Major and Minor Relapses
* Example # V.8.2.1