Software Configuration Management

Software Configuration Management Activities

Software configuration management (SCM), in general, is the monitoring and management of configurations in network devices such as routers, switches, & firewalls. The consistent performance of a network is highly dependent on how effectively its devices are managed and maintained using SCM.

With the growing business needs, enterprise networks are subject to frequent configuration changes and these changes can easily get out of hand. Network admins find it difficult to manage as there is no track of the changes happening in the network, and it becomes impossible to backtrack the change that caused a network issue. This is where software configuration management becomes a necessity for every business.

An efficient software configuration management tool keeps track of all the configuration changes that happen in the network, and also lets the admin know if there are any faulty configuration changes. It actively monitors the changes and their nature, so any possible glitch in the network is notified to the admin even before the change is applied to the device configuration.

NCM - A Software Configuration Management tool.

Network Configuration Manager (NCM) is a powerful and easy-to-use software configuration management tool that enables you to get the best of your network by administering all of your network configurations. With NCM, you get all the features that are the need of the hour:

Automating network configuration backups
Tracking and monitoring configuration changes
Detecting config changes in real-time using syslog messages
Achieving compliance with Cisco Policy, HIPAA, SOX, PCI, and more.
Scheduling reports for monitoring and auditing.

Change Control Process

Change control is manual step in software lifecycle. It combines human procedures and automated tools.

Change control process is illustrated in below figure

Change request submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration object and system function, and project cost of change.

The result of the evaluation are presented as a change report, which is used by the change control authority(CCA) – A person or group who make final decision on the status and priority of the change.

Different factors of Change Control process

- Change request initiation and Control
- Impact Assessment
- Control and Documentation of Changes
- Documentation and Procedures
- Authorized Maintenance
- Version Control etc

Change control process Flow

Software Version Control

Software Version Control is a system or tool that captures the changes to a source code elements: files, folders, images or binaries.
Version Control Tools track these changes and allows manipulation of versions and baselines.
Many tools do very similar tasks

Types of Version Control Software
Source Control Tools
Software Configuration Management Tools
Configuration Management Tools
Document Management Systems
Product Life Cycle Management Systems, PLM
Content Management Systems, CMS

Version Control Functionality However, Version Control, or Software Configuration Management Tools do this specific job superbly.
Think of it this way, software developers who use these tools day-in and day-out, write these tools.
So of course, they do a really good job of providing these functions:

Check Out, Edit, Check In
Conflict Resolution
File Merging
Conflict Resolution
File Revision History
Source Code Branching

. Version Control Software is an overview of the features and concepts of a few of the more commonly used Open Source SCM Tools, Subversion, Bazaar and Git.

An Overview of CASE Tools

CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering. It means, development and maintenance of software projects with help of various automated software tools.

CASE Tools

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

Central Repository: CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.

Upper Case Tools - Upper CASE tools are used in planning, analysis and design stages of SDLC.

Lower Case Tools - Lower CASE tools are used in implementation, testing and maintenance.

Integrated Case Tools - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

Constructive Cost Models (COCOMO),

- COCOMO Model. Cocomo (Constructive Cost Model) is a **regression model** based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.
- The initial version was published in 1981 and was known as **COCOMO-81.** It was developed considering a waterfall process would be used and that all software will be developed from scratch. The COCOMO model is one of the most popular models cost estimating in software engineering domain.

- There are a lot of cost estimation techniques in software engineering such as:

- COCOMO 1 cost estimation
- COCOMO 2 cost estimation

COCOMO Model 1

- The COCOMO 1 model is a regression-based model that considers various historical programs software size and multipliers.
- The most important fundamental calculation is the use of effort equation to find the number of Person-Months required in developing a project.
- This model estimates the cost by considering the size and other quality aspects of the similar type of historical programs. It calculates the Efforts i.e. estimation of the number of Person-Months required to develop a project.
- Cocomo model is having 3 models (Basic model, intermediate model, advance model)
- ***Estimated Cost = Number of Person Months * Loaded Labour Rate***
- Estimates of requirements, maintenance are derived from the quantity. COCOMO requires input as the project's estimated size in Source Lines of Code.
- The COCOMO model in software engineering is based on the relationships between the two formulas:
- **Formula 1: Development Effort is based on system size:**
- ***MM = A * KSDI ^ B***
- where,
- MM is the effort measured in Man per Months
- KDSI is the number of Source Instructions Delivered in a Kilo (Thousands)
- **Formula 2: Effort (MM) and Development Time**
- $T_{DEV} = C * MM ^\wedge D$
- where,
- T_{DEV} is the development Time
- Coefficients A, B, C, and D depending upon the mode of development that can be categorized into following 3 distinct classes of software projects defined above.(organic, semi detached, embaded)

COCOMO Model 2

- The COCOMO 2 model in Software Engineering is tuned to modern software life cycles. COCOMO 1 model has been very successful. However, it doesn't apply to newer software development practices as well as it does to traditional practices.
- This model targets modern software projects and will continue to evolve over the next few years. The sub-models of COCOMO 2 model are as follows:
- **Application Composition Model**: Used when software is developed from existing parts.
- **Early Design Model**: Used when system requirements are collected and concluded but designing has not yet started.
- **Reusable Model**: Used when software is developed using the reusable components. It is used to compute the efforts taken to integrate these components.
- **Post-Architecture Model**: Used once the system designing is completed and when further system information is collected.
- **Multiplier Affect**:
  The capability of the developers, functional requirements, familiarity with the development platform, etc.
- The COCOMO 2 formula to estimate the calendar time required to complete a project when staff will be required.
- ***TDEV = 3^(PM)***$^{(0.33 + 0.2 * (B-1.01))}$
- where,
- **PM** is the effort computation and B is the exponent (B = 1 for the early prototyping model). This computation predicts the nominal schedule for the project.
- The time required is independent of the number of people working on the project.

Resource Allocation Models,

- Resource allocation systems are a type of information system that have the same four-layer generic structure. Resource allocation systems manage a fixed amount of some given resource, such as tickets for a concert or a football game, that must be allocated to users who request that resource from the supplier. Ticketing systems are an obvious example of a resource allocation system, but a large number of apparently dissimilar programs are also actually resource allocation systems. Some examples of this class of system are:

- Timetabling systems that allocate classes to timetable slots. The resource being allocated here is a time period and there are usually a large number of constraints associated with each demand for the resource.
- Library systems that manage the lending and withdrawal of books or other items. In this case, the resources being allocated are the items that may be borrowed. In this type of system, the resources are not simply allocated but must sometimes be deallocated from the user of the resource.
- Air traffic management systems where the resource that is being allocated is a segment of airspace so that separation is maintained between the planes that are being managed by the system. Again, this involves dynamic allocation and reallocation of resource, but the resource is a virtual rather than a physical resource.

Software Risk Analysis and Management

- Very simply, a risk is a *potential* problem. It's an activity or event that may compromise the success of a software development project. Risk is the possibility of suffering loss, and total risk exposure to a specific project will account for both the *probability* and the *size* of the potential loss.
- Guesswork and crisis-management are never effective. Identifying and aggregating risks is the only predictive method for capturing the probability that a software development project will experience unplanned or inadmissible events. These include terminations, discontinuities, schedule delays, cost underestimation, and overrun of project resources

- Risk management includes the following tasks:

- ***Identify*** risks and their triggers
- ***Classify*** and prioritize all risks
- Craft a ***plan*** that links each risk to a mitigation
- ***Monitor*** for risk triggers during the project
- Implement the ***mitigating action*** if any risk materializes
- ***Communicate*** risk status throughout project