UNIT 4:

Software Configuration Management

Baseline is milestone and reference point in software development that is marked by completion or delivery of one or more software configuration items and formal approval of set of predefined products is obtained through formal technical review. Baseline is shared project database. It is task of Software Configuration Management (SCM) that is used to maintain integrity of set of products.

Main aim of baseline is to reduce and control vulnerability i.e. Weakness of projects that can easily affect project and leads to changes that are uncontrollable. This can be achieved by fixing and changing configuration items (various key deliverables) in the development life cycle of product at some critical points. Each element that is associated with baseline needed to be kept under formal change control.

Process :

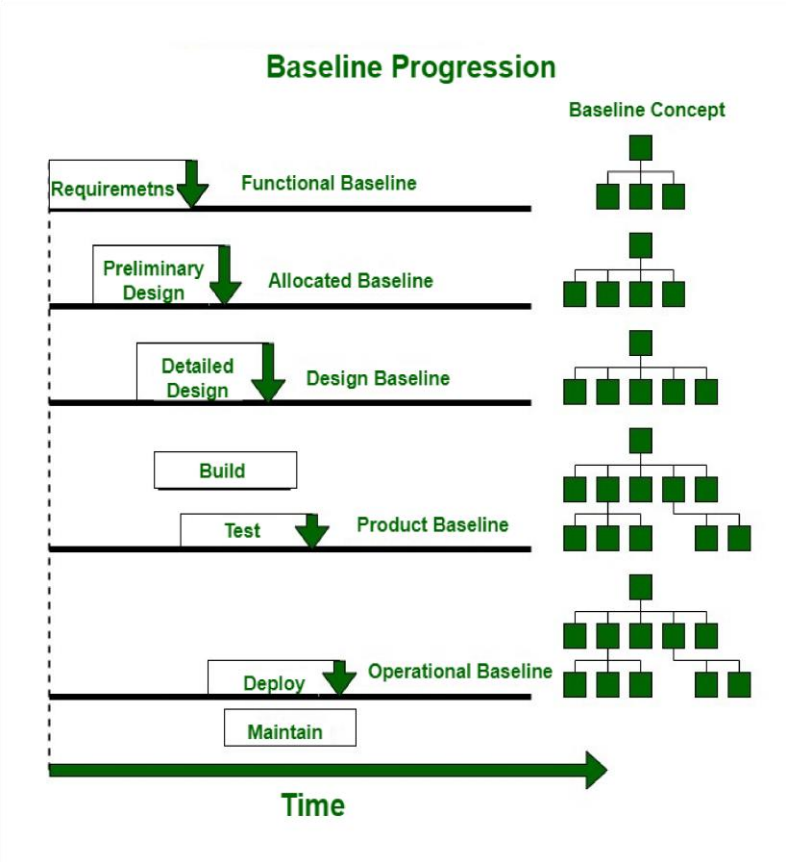
1. Elements need to be documented in proper way and reviewed to find if there is an issue of design model. If any error or defect is found, then these errors and defects are corrected and fixed.
2. All parts of model are being reviewed properly and all problems found are being fixed and approved.
3. Design base model is now Baseline.
4. Any further changes in the program architecture that is actually documented in the design model can be allowed to be done only after each has been evaluated and approved.

Baseline Components :

A typical baseline includes following components :

1. **Functional Baseline –**
   Operation Document, System requirements.
2. **Allocated Baseline –**
   High-level document, Preliminary Design, Interface control documents.
3. **Design Baseline –**
   Detailed design documents.
4. **Product Baseline –**
   Source and executable code units, final system specifications, user and maintenance manuals, Hardware and software specifications,
5. **Operational Baseline –**

Source and executable code units, final system specifications, user and maintenance



manuals, acceptance test plans, test procedures, site integration test cases and data sets and test reports

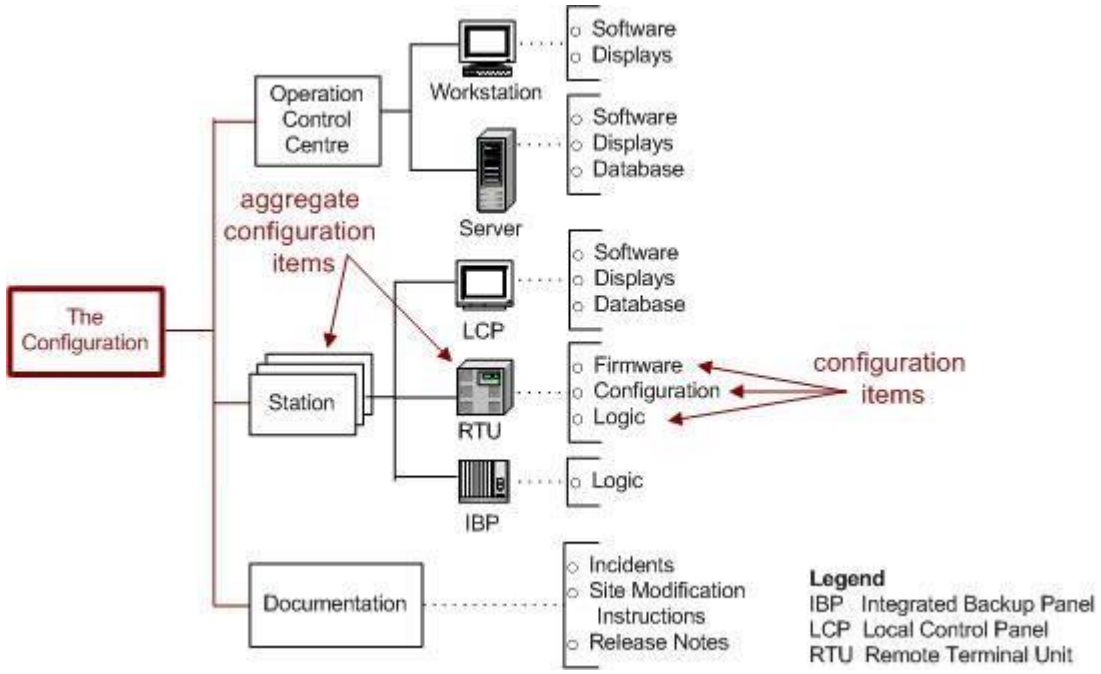
6. **Acceptance Test –**
   Source and executable code units, integration test plans, test procedures, test cases, and data sets and test reports
7. **Integration Test –**
   Source and executable code units, unit test plans, test procedures, test cases, and data sets and test reports
8. **Unit Test –**
   Source and executable code modules **Example**

:

Software Configuration Item:-

A component of a system that is treated as a self contained unit for the purposes of identification and change control. All configuration items (CIs) are uniquely identified by CI registration codes and version numbers. A CI may be a primitive system building block (e.g. code module) or an aggregate of other CIs (e.g. a sub-system is an aggregate of software units).

Examples of Configuration Items:-



The figure depicts the configuration of an underground railway environmental control system. The system has a central operation control centre and many railway stations. Each station has:

A local control processor (LCP) that allows operators to display and manipulate the air conditioning system and smoke extraction fans.

A remote terminal unit (RTU) that performs the control function and

An integrated backup panel (IBP) that allows operators to take over manual control of the system.

For the purposes of example let's look at the configuration items identified for the remote terminal unit control software.

The RTU has three components that are managed as self-contained units for the purposes of identification and change control:

The firmware: - the operating system that runs the remote terminal unit.

The configuration :- this is a data file that determines the operation of the RTU.

The logic :- the application software that performs direct digital control functions.

These three components were identified as primitive level configuration items as they cannot be further decomposed and are managed as self-contained units. For example each logic file that is loaded into the RTU has a unique identifier and version number.

The RTU is an example of an aggregate configuration item. An RTU CI release is an aggregate of firmware, configuration and logic CIs. The station CI is another aggregate configuration item. A station CI release is an aggregate of the individual LCP, RTU and IBP CI releases.

Designating Configuration Items:-

The choice of aggregate and primitive level configuration items is driven by the way the development, operation and maintenance of a system is managed. For example the station CI was selected as a configuration control item because it is useful to have one revision identifier that indicates the configuration status of all control system components at that location. The revision level assigned to the station CI denotes the level of functionality available and the possible modes of communication with the operation control centre.

General guidelines for identifying configuration items are:

1)CIs are released as a unit to the customer

Example: Mission computer, accounting package

2) CIs may be replaced as a unit in the field

Example: Modem, Remote Terminal Unit (RTU)

3) CIs perform a meaningful stand-alone function

Example: RTU

Software Configuration Management Process: 5 Steps

As you're no doubt aware, in software engineering, software configuration management (SCM) refers to a process for maintaining computer systems, servers, and software in a desired, consistent state.

Sometimes called software change management or IT automation, the idea is to have a system in place to track and compare changes made across a system throughout the development process and also identify who made these revisions.

The purpose is to make project management easier, minimize errors, increase traceability, and improve overall software quality.
The benefits of a SCM process are:

- **Multi-User Updates:** Often many people work on software development, meaning there are constant updates and changes. SCM allows you to track all code and configurations deployed into production (aka a configuration audit) and also introduces traceability by determining which contributor made each one.

- **Productivity:** SCM enhances the productivity of the software as it ensures minimal errors. For example, you can always make sure that your test and production environments match.
- **Communication and Collaboration:** Communication between team members is easy with SCM, making it simple for stakeholders involved in the project to work together and enhance the quality of the product.
- **Platform and OS Testing:** Ensure that the software runs effectively across multiple platforms and operating systems.
- **Change Accommodation:** SCM makes it easier to accommodate changes in schedule, policy, and users.
- **Cost Control:** By tracking team members and project workflow, SCM helps to control costs and increased efficiency.

In this article, I'll take you through the different steps of a SCM process, who needs to be involved, and what tools are available to help.

The 5 steps of a SCM plan

The software configuration management process is a series of steps designed to track and manage all the defects, resources, codes, documents, hardware and budgets throughout a project.

SCM is an interdisciplinary process involving people at every level, including DevOps, developers, project managers/owners, SysAdmin and testers.

1. Planning and Identification

The first step in the process is planning and identification. In this step, the goal is to plan for the development of the software project and identify the items within the scope. This is accomplished by having meetings and brainstorming sessions with your team to figure out the basic criteria for the rest of the project.

Part of this process involves figuring out how the project will proceed and identifying the exit criteria. This way, your team will know how to recognize when all of the goals of the project have been met.

Specific activities during this step include:

- Identifying items like test cases, specification requirements, and code modules
- Identifying each computer software configuration item in the process
- Group basic details of why, when, and what changes will be made and who will be in charge of making them
- Create a list of necessary resources, like tools, files, documents, etc.

2. Version Control and Baseline

The version control and baseline step ensures the continuous integrity of the product by identifying an accepted version of the software. This baseline is designated at a specific time in the SCM process and can only be altered through a formal procedure.

The point of this step is to control the changes being made to the product. As the project develops, new baselines are established, resulting in several versions of the software.

This step involves the following activities:

- Identifying and classifying the components that are covered by the project
- Developing a way to track the hierarchy of different versions of the software
- Identifying the essential relationships between various components
- Establishing various baselines for the product, including developmental, functional, and product baselines
- Developing a standardized label scheme for all products, revisions, and files so that everyone is on the same page.

Baselining a project attribute forces formal configuration change control processes to be enacted in the event that these attributes are changed.

3. Change Control

Change control is the method used to ensure that any changes that are made are consistent with the rest of the project. Having these controls in place helps with quality assurance, and the approval and release of new baseline(s). Change control is essential to the successful completion of the project.

In this step, requests to change configurations are submitted to the team and approved or denied by the software configuration manager. The most common types of requests are to add or edit various configuration items or change user permissions.

This procedure includes:

- Controlling ad-hoc changes requested by the client
- Checking the merit of the change request by examining the overall impact they will have on the project
  Making approved changes or explaining why change requests were denied.

4. Configuration Status Accounting

The next step is to ensure the project is developing according to the plan by testing and verifying according to the predetermined baselines. It involves looking at release notes and related documents to ensure the software meets all functional requirements.

Configuration status accounting tracks each version released during the process, assessing what is new in each version and why the changes were necessary. Some of the activities in this step include:

- Recording and evaluating changes made from one baseline to the next
- Monitoring the status and resolution of all change requests
- Maintaining documentation of each change made as a result of change requests and to reach another baseline
- Checking previous versions for analysis and testing.

5. Audits and Reviews

The final step is a technical review of every stage in the software development life cycle. Audits and reviews look at the process, configurations, workflow, change requests, and everything that has gone into developing each baseline throughout the project's development.

The team performs multiple reviews of the application to verify its integrity and also put together essential accompanying documentation such as release notes, user manuals, and installation guides.

Activities in this step include:

- Making sure that the goals laid out in the planning and identification step are met
- Ensuring that the software complies with identified configuration control standards
- Making sure changes from baselines match the reports
- Validating that the project is consistent and complete according to the goals of the project.
- Software Engineering-Identification of Objects in Software Configuration

---

- To control and manage software configuration items, each must be separately named and then organized using an object-oriented approach. Two types of objects can be identified : basic objects and aggregate objects.A basic object is a "unit of text" that has been created by a software engineer during analysis, design, code, or test. For example, a basic object might be a section of a requirements specification, a source listing for a component, or a suite of test cases that are used to exercise the code. An aggregate object is a collection of basic objects and other aggregate objects. Design Specification is an aggregate object. Conceptually, it can be viewed as a named (identified) list of pointers that specify basic objects such as **data model** and **component N.**

  Each object has a set of distinct features that identify it uniquely: a name, a description, a list of resources, and a "realization." The object name is a character string that identifies the object unambiguously. The object description is a list of data items that identify • the SCI type (e.g., document, program, data) represented by the object a project identifier
  • change and/or version information

  Resources are "entities that are provided, processed, referenced or otherwise required by the object ." For example, data types, specific functions, or even variable names may be considered to be object resources. The realization is a pointer to the "unit of text" for a basic object and null for an aggregate object. Configuration object identification must also consider the relationships that exist between named objects. An object can be identified as <part-of> an aggregate object. The relationship <part-of> defines a hierarchy of objects. For example, using the simple notation

E-R diagram 1.4 <part-of> data model; data model <part-of> design specification; we create a hierarchy of SCIs.

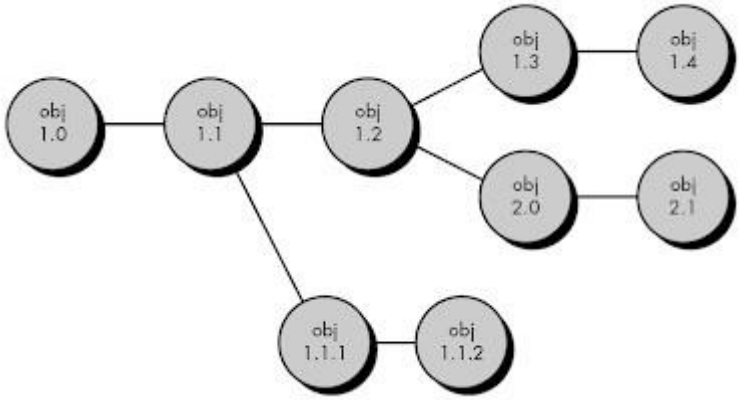
It is unrealistic to assume that the only relationships among objects in an object hierarchy are along direct paths of the hierarchical tree. In many cases, objects are interrelated across branches of the object hierarchy. For example, a data model is interrelated to data flow diagrams (assuming the use of structured analysis) and also interrelated to a set of test cases for a specific equivalence class. These cross structural relationships can be represented in the following manner:

data model <interrelated> data flow model; data model <interrelated> test case class m;

In the first case, the interrelationship is between a composite object, while the second relationship is between an aggregate object (data model) and a basic object (test case class m).

The interrelationships between configuration objects can be represented with a module interconnection language. A MIL describes the interdependencies among configuration objects and enables any version of a system to be constructed automatically.

The identification scheme for software objects must recognize that objects evolve throughout the software process. Before an object is baselined, it may change many times, and even after a baseline has been established, changes may be quite frequent. It is possible to create an evolution graph for any object. The evolution graph describes the change history of an object. Configuration object 1.0 undergoes revision and becomes object 1.1. Minor corrections and changes result in versions 1.1.1 and 1.1.2, which is followed by a major update that is object 1.2. The evolution of object 1.0 continues through 1.3 and 1.4, but at the same time, a major modification to the object results in a new evolutionary path, version 2.0.Both versions are currently supported.



-
- Changes may be made to any version, but not necessarily to all versions. How does the developer reference all components, documents, and test cases for version 1.4? How does the marketing department know what customers currently have version 2.1? How can we be sure that changes to the version 2.1 source code are properly reflected in the corresponding design documentation? A key element in the answer to all these questions is identification.

A variety of automated SCM tools has been developed to aid in identification (and other SCM) tasks. In some cases, a tool is designed to maintain full copies of only the most recent version. To achieve earlier versions (of documents or programs) changes (cataloged by the tool) are "subtracted" from the most recent version . This scheme makes the current configuration immediately available and allows other versions to be derived easily.

What is a “version control system”?

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

Why Version Control system is so Important?

As we know that a software product is developed in collaboration by a group of developers they might be located at different locations and each one of them contributes to some specific kind of functionality/features. So in order to contribute to the product, they made modifications to the source code(either by adding or removing). A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what changes have been made. A separate branch is created for every contributor who made the changes and the changes aren’t merged into the original source code unless all are analyzed as soon as the changes are green signaled they merged to the main source code. It not only keeps source code organized but also improves productivity by making the development process smooth.

Basically Version control system keeps track on changes made on a particular software and take a snapshot of every modification. Let’s suppose if a team of developer add some new functionalities in an application and the updated version is not working properly so as the

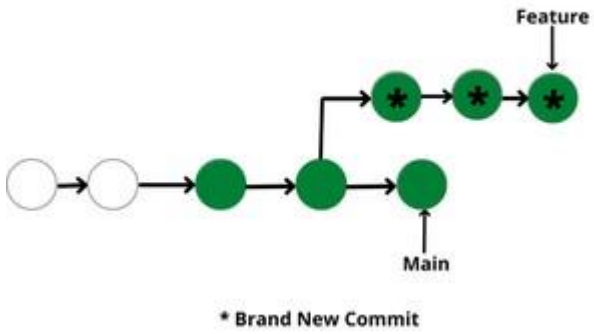
version control system keeps track of our work so with the help of version control system we can omit the new changes and continue with the previous version.

Benefits of the version control system:

- Enhances the project development speed by providing efficient collaboration,
- Leverages the productivity, expedites product delivery, and skills of the employees through better communication and assistance,
- Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
- Employees or contributors of the project can contribute from anywhere irrespective of the different geographical locations through this **VCS,**
- For each different contributor to the project, a different working copy is maintained and not merged to the main file unless the working copy is validated. The most popular example is **Git, Helix core, Microsoft TFS,**
- Helps in recovery in case of any disaster or contingent situation, Informs us about Who, What, When, Why changes have been made.

Use of Version Control System:

- **A repository:** It can be thought of as a database of changes. It contains all the edits and historical versions (snapshots) of the project.
- **Copy of Work (sometimes called as checkout):** It is the personal copy of all the files in a project. You can edit to this copy, without affecting the work of others and you can finally commit your changes to a repository when you are done making your changes.
- **Working in a group:** Consider yourself working in a company where you are asked to work on some live project. You can't change the main code as it is in production, and any change may cause inconvenience to the user, also you are working in a team so you need to collaborate with your team to and adapt their changes. Version control helps you with the, merging different requests to main repository without making any undesirable changes. You may test the functionalities without putting it live, and you don't need to download and set up each time, just pull the changes and do the changes, test it and merge it back. It may be visualized as.

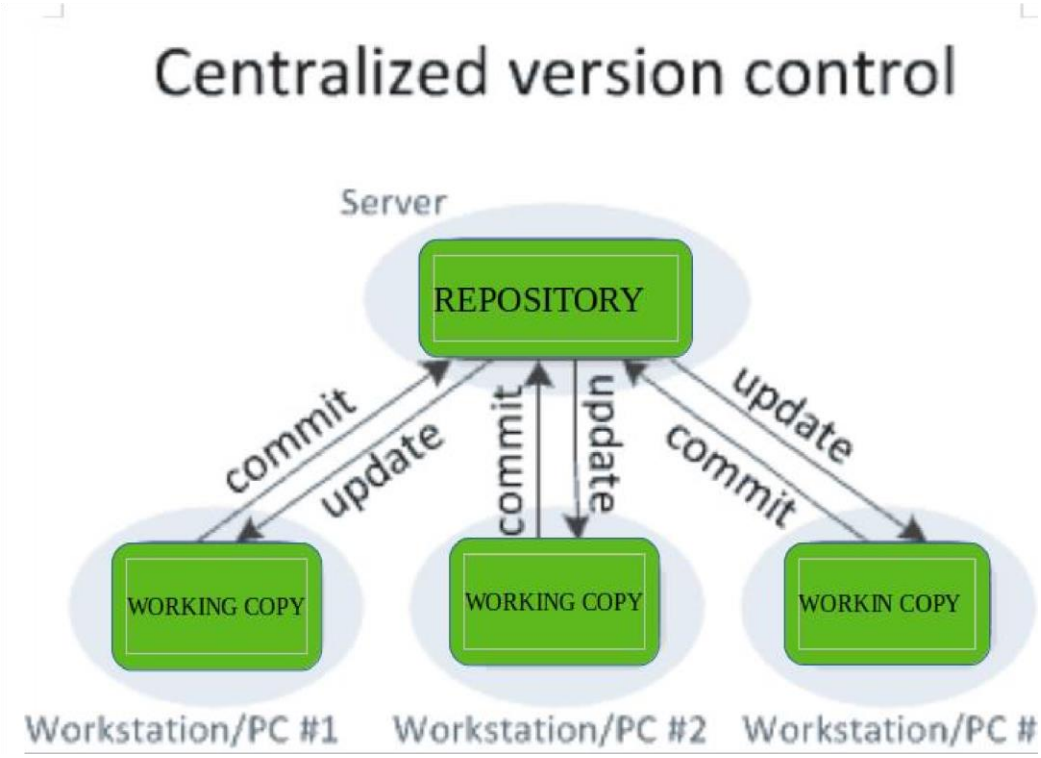


Types of Version Control Systems:

- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems

Local Version Control Systems: It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.
Centralized Version Control Systems: Centralized version control systems contain just one repository globally and every user need to commit for reflecting one's changes in the repository. It is possible for others to see your changes by updating. Two things are required to make your changes visible to others which are:

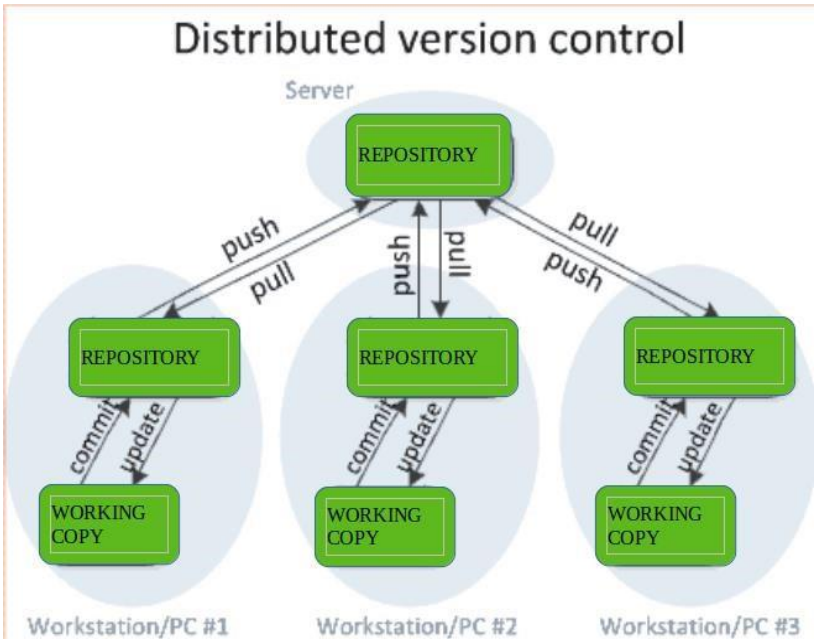
- You commit
- They update



The **benefit** of CVCS (Centralized Version Control Systems) makes collaboration amongst developers along with providing an insight to a certain extent on what everyone else is doing on the project. It allows administrators to fine-grained control over who can do what. It has some **downsides** as well which led to the development of DVS. The most obvious is the single point of failure that the centralized repository represents if it goes down during that period collaboration and saving versioned changes is not possible. What if the hard disk of the central database becomes corrupted, and proper backups haven't been kept? You lose absolutely everything.
Distributed Version Control Systems: Distributed version control systems contain multiple repositories. Each user has their own repository and working copy. Just committing your changes will not give others access to your changes. This is because commit will reflect those changes in your local repository and you need to push them in order to make them visible on the central repository. Similarly, When you update, you do not get others' changes unless you have first pulled those changes into your repository.
To make your changes visible to others, 4 things are required:

- You commit
- You push
- They pull
- They update

The most popular distributed version control systems are Git, and Mercurial. They help us overcome the problem of single point of failure.



Purpose of Version Control:

- Multiple people can work simultaneously on a single project. Everyone works on and edits their own copy of the files and it is up to them when they wish to share the changes made by them with the rest of the team.
- It also enables one person to use multiple computers to work on a project, so it is valuable even if you are working by yourself.
- It integrates the work that is done simultaneously by different members of the team. In some rare cases, when conflicting edits are made by two people to the same line of a file, then human assistance is requested by the version control system in deciding what should be done.
- Version control provides access to the historical versions of a project. This is insurance against computer crashes or data loss. If any mistake is made, you can easily roll back to a previous version. It is also possible to undo specific edits that too without losing the work done in the meanwhile. It can be easily known when, why, and by whom any part of a file was edited.

Change Control

Change control is the process used to manage all these variables. If a change happens (which it always does) then it's crucial that you have a mechanism in place to control that process. But what is change control in project management, and what are the steps necessary to implement it?

What Is Change Control?

Change control is a methodology used to manage any change requests that impact the baseline of your project. It's a way to capture that change from the point where it's been identified through every step of the project cycle. That includes evaluating the request and then approving, rejecting or deferring it.

The purpose of this process is to make sure that you're not changing things in the project that don't need to be changed. The last thing you want to do is disrupt the project for no good reason, wasting valuable time and resources. Any changes that are approved are then documented. The change control process is part of the larger change management plan.

Key Elements of Change Control

There are some key elements that build a change control framework for project management. Here's a brief description of them.

- **Change Control Board:** A change control board is a group of representatives from the project team that regularly meet to approve or disapprove change requests. If they approve a change request, it can turn into a change order.

- **Change Requests:** A change request is a formal petition for change in a project. It's a document that explains what are the changes to be made and the main reasons why they should be implemented.

  Change requests can either be submitted by internal or external project stakeholders. Our free change request template can help you streamline this process.

- **Change Orders:** Once the change control board has approven a change request, a change order is signed by the board and the clients or stakeholders. This is an agreement from both parties to change the conditions that were first drafted in the original contract. Our free change order template is a great tool to create your change orders.

- **Change Log:** A change log is a change management tool that's used to document all the changes made to a project plan or any contracts. It's a musthave tool for any project manager.

The Role of the Change Request

A change request is usually the trigger that starts the process of change control. The change request can originate from stakeholders asking for new features, the need to repair something that proves faulty during the execution phase, upgrades or any number of other causes. Whatever or wherever the change comes from, change control determines its value and how to feasible implement it.

Change management procedures may vary across industries. For example, change order forms are used by construction companies to make changes to the scope of a construction project.

Change control comes in many forms, but the change control process is most effective when executed with project management software. ProjectManager allows you to capture the change in your project when you discover it on a task list, kanban board, Gantt chart or spreadsheet. Then manage that change with real-time dashboards and instant reports. Get started for free.

CONFIGURATION AUDIT

The **configuration audit** is an activity that is conducted to determine that a system or item meets it functional requirements and has been built in accordance with its blueprints, source code, or other technical documents. In plain English, audits are done to prove that (1) the thing works the way it should and (2) the builder's factory production system has reliable quality control, especially as it pertains to technical documentation (the documents that depict, describe, and define the thing).

The nature of the particular system or item determines the extent to which its hardware and software are reviewed for correlation with test results and design documentation. Generally,

developed items are more thoroughly audited than “commercial off the shelf” (COTS) items. However, the bottom line of the audits is that they (1) validate that the system meets its functional requirements and (2) establish that the system’s technical documentation is accurate and well maintained. There are three basic categories of audits: functional, physical, and privately developed item.

Functional Configuration Audit (FCA):- The FCA is a review of a system's or item’s test, analysis, inspection, or demonstration records to validate that it meets its performance and interface/interoperability contract requirements. That is, the FCA verifies that the system's or item's required functional, physical, and interoperability characteristics meet the specified contractual requirements. A functional audit is a prerequisite to the performance of a physical audit.

Physical Configuration Audit (PCA):- A PCA involves the matching of a functionally successful "as built" system or item with its proposed product configuration identification (PCI), (the “PCI” is the set of documents that will eventually be used for production, acceptance, and modification management) Successful completion of a PCA will provide reasonable assurance that the PCI is (1) complete, (2) matches the system or item, and (3) is suitable for full-scale production, field operations, maintenance, and life-cycle modifications.

Privately Developed Item Audits:- Configuration audits on privately developed items, i.e., those not developed with Government funds, are usually either not audited or in some cases, a functional audit may be done to assure that its functional characteristics satisfy Government requirements. The audit is usually limited to an examination of test data provided by the contractor.

Why bother with these audits? The question is often asked why these audits should be done; i.e., what, if any, value do these audits bring to a system acquisition program? I have participated and been the lead for many audits and have found that they can be vitally important for the success of a system acquisition. These audits are often done before a system is actually shipped to the field for use. In some cases, these audits uncover safety hazards and other potentially serious problems that were heretofore unrecognized..

These audits are also extremely valuable in that they give a decision maker confidence that a program can proceed to full-scale production. That is, knowing that a system has been thoroughly audited and has been verified to perform as expected, and further knowing that all the vital technical documentation that will be used to build and sustain the system is accurate and is well maintained, provides the decision maker with confidence in granting permission to proceed with full-scale production and deployment of the system.

How to Prepare for an Audit. If an audit is a contract requirement, then you can usually follow these steps and be reasonably safe:

1) Check Section C of the contract to see which documents are binding on the CLINS to be audited. For example,

CLIN 0001, Radio PME: "Radio System Specification 0001C" 2)

Check the SOW to verify the audit requirements.

3) Use Microsoft Excel or other spreadsheet application to create a matrix which matches requirements with verification methods. For example: Audit Checklist.

4) Coordinate the audit details and logistics with the contractor (i.e., time, date, location, participants, scope, etc.)
5) Conduct the audit and document (For example Sample Audit Document & Audit Template) any deviations or problem areas.
6) Take follow-up action to resolve any deviations or problem areas.

What Is a Project Status Report?

A *project status report* is a document that describes the progress of a project within a specific time period and compares it against the project plan. Project managers use status reports to keep stakeholders informed of progress and monitor costs, risks, time and work. Project status reports allow project managers and stakeholders to visualize project data through charts and graphs.

Project status reports are taken repeatedly, throughout every phase of the project's execution, as a means to maintain your schedule and keep everyone on the same page. The status report for a project will generally include the following:

- The work that's been completed
- The plan for what will follow
- The summary of the project budget and schedule
- A list of action items
- Any issues and risks, and what's being done about them

The true value of a project status report lies beyond its use as a communication channel. It also provides a documented history of the project. This gives you historical data, so the next time you're planning a similar project, you can avoid any missteps or bottlenecks.

Because project status reports cover so many topics, historically, they were time-consuming to create. Fortunately, modern project management software like ProjectManager expedites the all-important reporting process.

Create a project status report with just a few clicks with Project Manager—.

How to Write a Project Status Report

Writing a project status report is an essential project management task. Whether you generate one weekly, monthly or quarterly, the steps are essentially the same. Here's *how to write a project status report*:

1. Determine the objective
2. Target your audience (Clients, team members, sponsors, etc)
3. Choose the format and type
4. Collect your data
5. Structure the report
6. Make sure it's clear
7. Edit draft

Because a project status report follows a basic outline, it can be helpful to use a project status report template. However, a project status report template is only a static document. Using project status reporting software integrates with all your project management tools for greater efficiencies.

Goal of SCM

software configuration management or SCM is a task that allows you to track any change in a software development process, like if a new patch or a new code has been implemented, SCM allows you to track who did the changes and when, and when this code is doing its function, you can figure out how to apply it to your system. the goals of SCM are: * IDing the configuration: to know the ID of the person who did a change in the software which will be a

great tool to track down every change in your software. * Tracking defects: as a result of IDing configuration, you can easily track any defects in your software and solve it very quickly.
*Teamwork: it helps organizing the interaction between teammates and task handling.
* Environment management: managing both hardware and software that hosts your software.