Unit-10 Configuration Management

When we develop software, the product (software) undergoes many changes in their maintenance phase; we need to handle these changes effectively. Several individuals (programs) works together to achieve these common goals. This individual produces several work product e.g., Intermediate version of modules or test data used during debugging, parts of the final product. The elements that comprise all information produced as a part of the software process are collectively called a software configuration.

Whenever software is build, there is always scope for improvement and those improvements brings changes in picture. Changes may be required to modify or update any existing solution or to create a new solution for a problem. Requirements keeps on changing on daily basis and so we need to keep on upgrading our systems based on the current requirements and needs to meet desired outputs. Changes should be analyzed before they are made to the existing system, recorded before they are implemented, reported to have details of before and after, and controlled in a manner that will improve quality and reduce error. This is where the need of System Configuration Management comes. **Software Configuration Management (SCM)** is an arrangement of exercises which controls change by recognizing the items for change, setting up connections between those things, making/characterizing instruments for overseeing diverse variants, controlling the changes being executed in the current framework, inspecting and revealing/reporting on the changes made. It is essential to control the changes in light of the fact that if the changes are not checked legitimately then they may wind up undermining a well-run programming. In this way, SCM is a fundamental piece of all project management activities.

The key objectives of SCM are:

- Control the evolution of software systems: SCM helps to ensure that changes to a software system are properly planned, tested, and integrated into the final product.
- Enable collaboration and coordination: SCM helps teams to collaborate and coordinate their work, ensuring that changes are properly integrated and that everyone is working from the same version of the software system.

- Provide version control: SCM provides version control for software systems, enabling teams to manage and track different versions of the system and to revert to earlier versions if necessary.
- Facilitate replication and distribution: SCM helps to ensure that software systems can be easily replicated and distributed to other environments, such as test, production, and customer sites.
- SCM is a critical component of software development, and effective SCM practices can help to improve the quality and reliability of software systems, as well as increase efficiency and reduce the risk of errors.

The configuration management of a s/w system product involves four closely related activities:

Change management:

This involves keeping track of request for changes to the s/w from customer and developers working out the costs and impact of making these changes, and deciding if and when the changes should be implemented. Change management is a systematic approach to dealing with the transition or transformation of an organization's goals, processes or technologies. The purpose of change management is to implement strategies for effecting change, controlling change and helping people to adapt to change.

To be effective, the change management strategy must take into consideration how an adjustment or replacement will impact processes, systems and employees within the organization. There must be a process for planning and testing change, communicating change, scheduling and implementing change, documenting change and evaluating its effects. Documentation is a critical component of change management -- not only to maintain an audit trail should a rollback become necessary, but also to ensure compliance with internal and external controls, including regulatory compliance.

Version management:

This involves 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.

Version management is the process of keeping track of different versions of software components and the systems in which these components are used. It also involves ensuring that changes made by different developers to these versions do not interfere with each other. In other words, version management is the process of managing codeines and baselines

System Building:

This is the process of assembling program components, data and libraries and then compiling and linking these to create an executable system.

Release Management:

This involves preparing s/w for external release and keeping track of the system version that have been released for customer use. Release management refers to the process of planning, designing, scheduling, testing, deploying, and controlling software releases. It ensures that release teams efficiently deliver the applications and upgrades required by the business while maintaining the integrity of the existing production environment.

In the competitive, dynamic, and fluid world of business and IT, half-baked releases are the last thing you need. The modern enterprise is a truly dynamic environment; and not all these changes are happening at the same pace. IT organizations need a way to orchestrate these myriad changes. That’s where release control and deployment automation come in to play. They help ease the transition to continuous delivery; and work through the Digital Transformation one release at a time. This is the new normal of IT.