smartBuild

Author: Rajesh Kumar
Email: rajesh@scmGalaxy.com

What is SCM?

SCM is a "set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made.“

It maximizes productivity by minimizing mistakes.

Benefits the Organization in Four Major Ways

It facilitates the ability to communicate status of documents and code as well as changes that have been made to them. High-quality released software has been tested and used, making it a reusable asset and saving development costs.

SCM benefits an organization in four areas: control, management, cost savings, and quality. These four benefits are mapped to an organization's overall goals and objectives when the decisions are made to bring a SCM tool in-house. The features of a SCM tool further support these benefits.

Cont…

Control

Control in SCM provides the ability to review, approve, and incorporate changes into a configuration item. All project personnel use the tool. Inherent in the tool is a standardized, measurable process for change. Integrity maintenance of CIs is enforced throughout the product life cycle. The tool permits only controlled change to the baseline CIs, and all changes are tracked.

Management

Management in SCM is concerned with the automation of identifying and guiding configuration items through their life cycle to final assembly as part of product and delivery. **Identification of CIs** through a unique naming convention allows version, release, update, and full change tracking. **Baselining of CIs** with the ability to produce product deltas from the baseline satisfies requirement and schedule changes along with product family support**. Rapid reviews and audits** of CIs are accomplished through the analysis of historic information collected. **Project status reporting** is accomplished in a clear and consistent format based on SCM collected information on all CIs under configuration management.

Cost Savings

Cost savings are realized across the entire product development life cycle with SCM. Maintaining product integrity through defined, tracked, and audited CIs provides a managed bill of materials for the product released to customers. Cost savings scale with SCM use and application across applications.

Quality

Software development is a people-intensive activity, and quality must be considered at every person-to-tool interface. Ensuring a high-quality work environment must address the process of building software products in an automated fashion. Quality is an ongoing process. The lessons learned in one product must be transferred to new, related products and entire product families.

smartSCM contains following

- smart Change Management
- Smart Build Management
- Smart Build Verification Tests and Smoke Tests
- Smart Staging, Deployment, and Release

smart Change Management System who can help in finding following solution

- Dealing with Changing Project Requirements
- Increasing Software System Complexity
- Increasing Software Size and Architectural Complexity
- Inclusion of Third-Party Components and Software Reuse
- Increasing Project Environment Complexity
- Increasing Team Size and Parallel Development Support
- Geographically Distributed Development Teams
- Increasing Number and Frequency of Product Releases

Smart Build Management System which can help us in finding following solution

Pre-build :- Steps taken or tools run on code before the build is run to ensure zero build errors.

Post-build :-Includes scripts that are run to ensure that the proper build verification tests (BVTs) are run. This also includes security tests to make sure the correct code was built and nothing was fused into the build.

Clean build: - Deleting all obj files, resource files, precompiled headers, generated import libraries, or other byproducts of the build process. I like to call this cleaning up the "build turds." This is the first part of a clean build definition

Incremental build: - The secret to getting out a daily build to the test team, regardless of circumstances, is to perform incremental builds instead of daily clean builds. This is also the best way that you can maintain quality and a known state of a build

Continuous integration build : - This term is borrowed from the extreme programming (XP) practice. It means that software is built and tested several times per day as opposed to the more traditional daily builds. A typical setup is to perform a build every time a code check-in occurs.

Build break solution: - In the simplest definition, a build break is when a compiler, linker, or other software development tool (such as a help file generator) outputs an error caused by the source code it was run against.

Build defect solution: - This type of problem does not generate an error during the build process; however, something is checked into the source tree that breaks another component when the application is run. A build break is sometimes referred to or subclassed as a build defect.

Last known good (LKG) builds: - These terms are used as markers to indicate that the build has reached a certain quality assurance criterion and that it contains new high-priority fixes that are critical to the next baseline of the shipping code.

Smart Build Verification Tests and Smoke Tests

Smoke Test: In software, the term smoke test describes the process of validating code changes before checking them into the source tree. Smoke tests are focused tests that ensure that changes in the code function as expected and don't destabilize an entire build. If your smoke tests are good, they keep your source trees from catching on fire.

Verify Basic Functionality using Smoke test
Subject each build to a smoke test that verifies that the application has not broken in an obvious way.

Build Verification Tests
BVTs are automated suites of tests designed to validate the integrity of each new build and the basic functionality of the build before it is released for more in-depth testing. A BVT is not a comprehensive test; it is a type of test that is designed to do the following:

- Validate the integrity and testability of each new build.
- Ensure basic functionality for continued in-depth testing.
- Test critical functionality and the highest priority use cases or user scenarios.
- BVTs must be able to determine if the general quality of the build is sufficient for self-hosting, or if the build is unstable and should be used only in testing environments. Unit tests are sometimes used for BVTs, but only if they are critical to the execution of the program.

Smart Staging, Deployment, and Release

Two other very important processes with respect to SCM are staging, Deployment and release. **Staging** is the process of putting derived object files (executables, libraries, data files, generated header files, and so on) under version control. **Release** is the process of putting the runtime software into its final form and making it available to its intended users. These staged artifacts typically get **deployed** to various systems for further testing. They also get deployed to the production environment.

Reference

www.scmGalaxy.com