Software configuration and maintenance summary

COMMON SCM MYTHS

* **SCM Means More Work and procedures:** SCM will add more work, make work more difficult, and force unnecessary procedures.

**Solution:** the management and the implementation teams do their job properly, ensuring that employees are told what to expect and given proper training, then the transition can be smooth.

* **SCM Will Change Current practices and it Will Create product Failures:** Some of the business practices of the organizations where the SCM packages are implemented will be different from the best practices in the SCM system.

**Solution:** The better option is to change the organization’s business practices and adopt the best practices from the SCM package.

* **SCM is a difficult, Monotonous, and Time-Consuming activity:** During the early days of SCM, where all the CM tasks had to be done manually, CM was a difficult, monotonous, and time-consuming activity.

**Solution:** Today’s SCM tools automate many of the repetitive, monotonous, and tedious SCM activities, significantly simplifying the practice of SCM.

* **SCM is Just for developers:** Developers benefit the most from a properly implemented SCM system. Problems such as missing source code, the inability to find the latest version of a file, corrected mistakes reappearing, missing requirements and etc. can be avoided if a good SCM system is in place.

**Solution:** Many others stand to profit, including testers, QA personnel, maintenance teams, and support teams.

* **SCM Will Make Many Employees redundant and Jobless:** Although properly implemented SCM systems will automate many tasks, they will not necessarily make people redundant. Yes, there will be changes in job descriptions and in the activities, people used to do.

**Solution:** SCM systems create new job opportunities, and the very same people whose jobs have been automated can fill these new positions after receiving proper training.

**Software Configuration Management (SCM)** is a software-engineering discipline comprising the tools and techniques (processes or methodology) that a company uses to manage change to its software assets.

The software development process is a set of actions required for efficiently transforming a user’s need into an effective software solution.

The software development process defines the activities required for building the software systems and incorporating the methods and practices to be adopted.

Scientific software development—also known as software engineering—uses scientific and management techniques and productivity improvement tools for developing the software.

Software can be of two types:

* **generic products** (products that are produced by a software development organization and sold in the open market) and
* **customized products** (products that are developed to meet the specific needs of a customer).

The series of steps through which the software product goes through (from conceptualization until retirement) is called the software development life cycle (SDLC).

The most prominent process models include: the waterfall model, the spiral model, the win-win spiral model, the incremental model, the operational model, the joint application development (JAD) model, the evolutionary development model, the component assembly model, the cleanroom software engineering model, the concurrent development model.

The various phases or steps in the SDLC are listed as follows: Project start-up, Requirements analysis and requirements specification, Systems analysis, Systems design (high-level and low-level or detailed design), Development/coding and unit testing, System/integration testing, Acceptance testing, Implementation, Project windup, Maintenance, Retirement.

**Project Start-up**

The project team is formed and the project leader identified and the project is organized (i.e., modules are identified, key members are enlisted, and the people who will carry out the support functions such as internal quality assurance and CM are identified).

**Requirements Analysis and Requirements Specification**

user requirements are captured and documented, and a detailed plan for the phase is prepared.

**Systems Analysis**

In the systems analysis phase, the proposed system is defined after analysing various alternatives.

**HIGH-LEVEL DESIGN**: In this phase, the system design objectives are defined based on the deliverables from the previous step (System Analysis). Sometimes a prototype is developed to demonstrate the user interface design, screens, navigation, and other features of the system.

**LOW-LEVEL (LLD) OR DETAILED DESIGN (DD**): In this phase, the copy libraries, common routines, and program skeletons to be used are finalized.

**CODING AND UNIT TESTING**

During this phase the programs, copy libraries, functions, and other program elements are coded (or generated) and tested (unit testing). The main people involved in this phase are developers and programmers, analysts, the QA team, and testers.

**SYSTEM TESTING**

This is the phase in the SDLC where system testing or integration testing is carried out. System testing is done using STP, STS, and system test data.

**ACCEPTANCE TESTING**

Acceptance testing is the formal testing that is conducted (usually by the user, client, or an authorized entity) to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. This phase is carried out only if the system is developed for a particular client or customer.

**IMPLEMENTATION**

Once the integration, system and acceptance (in some cases, alpha and beta testing) testing are over. the software product or the system is turned over to the customers or clients or installed at the client site.

**PROJECT WINDUP**

In this phase, the project windup activities are completed. ✓ all the resources acquired for the project are released.

**PROJECT MAINTENANCE**

Once the system has been developed and tested, it is released to the users. From this point onward, the maintenance phase starts.

**RETIREMENT**

After many years of service, software products or systems reach a stage when any further maintenance would not be cost-effective.

SCM helps prevent the communications breakdown by controlling and managing change. SCM ensures that if something is changed, then all the people who need to know about the change are made aware of it. Configuration status accounting (CSA) is an SCM function that captures all the activities in a project and keeps accurate and traceable records of these activities.

The shared data problem is a very common source of trouble in any environment where two or more programmers or programs share a common resource

WHY SCM IS NEEDED?

SCM helps reduce these problems by coordinating the work and effort of many different people working on a common project. SCM plays an important role in the software development process—analysis, design, development, testing, and maintenance—by ensuring (through configuration audits) that what was designed (as specified in the characteristics document) is what is built

REASONS FOR IMPLEMENTING SCM

Some of the reasons for implementing SCM:

* The nature of software products, projects, and development teams
* The increased complexity of the software systems
* The increased demand for software
* The changing nature of software and the need for change management

BENEFITS OF SCM

•**Improved organizational competitiveness:** In this race to survive, only the organizations that continuously improve their processes and implement and use scientific tools, techniques, and business practices will succeed.

•**Better customer service and improved customer goodwill:** Since the SCM system keeps a record of all the changes made to each and every CI, identifying the cause of a problem is easy.

•**Better return on investment:** A good SCM system automates the process of changing and deploying software applications.

•**Improved management control over software development activities:** SCM systems improve communication in software development projects.

•**Improved software development productivity:** With an SCM system in place, developers have more time to develop software instead of wasting time looking for missing items, fixing bugs that have already been fixed, and solving problems caused by using different versions of the same code.

•**Easier handling of software complexity:** SCM identifies all the components or artifacts of a software project, captures their description and physical and functional characteristics, and scientifically and systematically classifies, categorizes, and names them.

•**Improved security:** SCM prevents unauthorized changes to the different components of the project.

•**Higher software reuse:** SCM system maintains a record of the past (project history or change history), the present (project tracking and change logs), and the future (information about planned versions and variants)

•**Lower software maintenance costs:** If the software was developed in a systematic manner, if the documentation is perfect, the changes made to the programs are recorded, and the program dependencies are defined, then the task of the maintenance team is easy

•**Better quality assurance:** SCM has formal mechanism for problem reporting or defect logging and tracking

•**Reduction of defects and bugs:** Once you have a defect logging and tracking system in place, once the QA teams start looking into the causes of the problems and correcting them

•**Faster problem identification and bug fixes:** if there is a mechanism for logging bug and problem reports, categorizing them, analysing the causes, and recording how the problem was solved, then much time and effort can be saved the next time a similar bug or problem occurs

•**Process-dependent development rather than person-dependent development:**

•**Assurance that the correct system was built:** SCM has a mechanism for checking or auditing the software system or product that is being delivered to the customer and certifying that the product satisfies the requirements.

SCM Plan

A SCMP clearly describes how SCM is accomplished and how consistency between a system’s configuration and the configuration records is achieved and maintained.

The SCMP has a section on SCM resources in which details are given about the software tools, techniques, equipment, personnel, and training necessary for the implementation of the specified SCM activities.

Good SCMPs take time; thorough knowledge of SCM functions, the peculiarities of the project, and the organization; and knowledge of other procedures such as auditing and testing.

Below are some practices and tips that will help in the creation of good SCMPs:

* Decision about the knowledge and capability of the person or team that writes the plan.
* Selection of the standard on which the SCMP is going to be based. - it is good practice to see what other standards have to offer.
* Next is to identify the procedures that should be followed in the practice of SCM. - adopted standards should be brought under document control for reference.
* The next step is to write the plan using any of the existing templates that are available. - Many SCMP templates are available on the Web
* Request for feedback. - Once the feedback from the various groups is received, the SCMP’s authors can review it, accept valid comments, and incorporate them into the plan.
* External review and audit of the final draft.

CONTENTS OF A TYPICAL SCMP

The SCMP can be written in any format as long as it contains all

necessary information.

A sample outline for an SCMP is as follow:

* **Cover Page:** This page should have the title “SCMP” and details on the project such as the organization, the authorities, the version number, and the release date.
* **Copyright Page:** This page should list the copyright information of the SCMP.
* **Distribution List:** This page should include the name and number of copies distributed and a description of how the documentation control activities will apply to this document.
* **About the Document:** A short description of the document and its sections.
* **INTRODUCTION:** An overview of the plan, the SCM activities, the audience for the plan, and how to use the plan, so that the user will have a clearer understanding of the plan. The introduction should contain at least the following four topics: purpose, scope, definitions, and references.
* **SCM MANAGEMENT:** Gives information on the organization of the SCM team and the allocation of responsibilities to teams and individuals, among other management topics.

‣ SCM Organization, SCM Responsibilities, Relationship of SCM to the Software Process Life Cycle, Interfaces to Other Organizations on the Project, SCM Responsibilities of the Organizations.

* **SCM ACTIVITIES:** Identifies the tasks and functions that are required to manage the configuration of the system as specified in the scope of the plan.
* **SCM SCHEDULES:** Describes the sequence of the SCM activities, their interdependencies and relationship to the project life cycle, and project milestones.
* **SCM RESOURCES:** identifies the software tools, techniques, equipment, personnel, budget, and training necessary for the implementation of the specified SCM activities.
* **SCMP MAINTENANCE:** Describes the activities that are required to keep the plan current during the life cycle of the project. The plan should be monitored and synchronized with the activities of the project

Sample SCMPs can be obtained from a host of sources.

* **The Internet.** Thousands of SCMPs covering a spectrum of projects—e.g., military, government, research, and commercial—of various sizes and complexity are hosted on the Internet.
* **ANSI/IEEE Std-1042–1987.** This is the IEEE Guide to SCM. There are four appendixes for this document, which are sample SCMPs for different types of projects.