**What is Software Configuration Management (SCM)?**

* SCM is a set of activities designed to control change in the context of building computer software.
* It aims to effectively manage changes by identifying the work products that are likely to change during the software development process.
* SCM establishes relationships among these work products, defines mechanisms for managing different versions of these work products, controls imposed changes, and provides auditing and reporting on the changes made.

**Who is Involved in SCM?**

* Everyone involved in the software engineering process is to some extent involved with SCM.
* In some cases, specialized support positions may be created to manage the SCM process.

**Why is SCM Important?**

* If change is not controlled, it can quickly lead to chaos in a software project.
* Uncontrolled changes can disrupt a well-run software project.
* SCM is essential for effective project management and maintaining software engineering practices.

**Steps in SCM:**

1. **Work Product Identification:** Each work product produced during software development is uniquely identified. This is crucial for effective SCM.
2. **Version and Change Control:** Mechanisms for controlling and managing different versions of work products are established.
3. **Audit:** The process is audited to ensure that quality is maintained as changes are made.
4. **Reporting:** Reporting is conducted to inform those who need to know about changes that have occurred.

**The Work Product in SCM:**

* The Software Configuration Management Plan defines the project strategy for SCM.
* When formal SCM is invoked, the change control process produces software change requests and reports, as well as engineering change orders.

In summary, this passage introduces the significance of Software Configuration Management (SCM) in the context of software development. SCM is described as a crucial set of activities for controlling and managing changes in software projects, which helps maintain order and quality in the development process. The steps and work products involved in SCM are outlined to provide an overview of its key components.

**9.1 Software Configuration Management (SCM)**

**Definition of Software Configuration:**

* The output of the software development process consists of three main categories: computer programs (both source and executable), documents describing the programs, and data (internal or external).
* These collectively form a software configuration, which encompasses all information produced during software development.

**Sources of Change:**

* The software development process is subject to change, which can occur for various reasons.
* Four fundamental sources of change include changes in business or market conditions, customer needs, organizational reorganization, and budgetary or scheduling constraints.

**Purpose of SCM:**

* Software Configuration Management (SCM) is a set of activities developed to manage change throughout the software development life cycle.
* SCM acts as a software quality assurance activity applied throughout the software process.

**9.1.1 Baselines:**

* A baseline is a key concept in SCM that helps control change while allowing justifiable changes to occur.
* It is defined as a specification or product that has been formally reviewed and approved and serves as the basis for further development.
* Once an item becomes a baseline, changes can still be made but require a formal procedure and evaluation.

**Justification for Changes:**

* Most changes in software development are justified as stakeholders acquire additional knowledge and insight over time.
* This additional knowledge is the driving force behind most changes, making them justifiable.

**The Baseline Analogy:**

* Baselines are like doors in a restaurant marked IN and OUT. Informal changes can be made before passing through this one-way door, but formal procedures are required after.

**Progression to a Baseline:**

* A baseline is achieved through the formal review and approval of one or more Software Configuration Items (SCIs).
* After approval, SCIs are placed in a project database, and changes can only be made through formal SCM controls.

**9.1.2 Software Configuration Items (SCIs):**

* SCIs are defined as pieces of information created as part of the software engineering process.
* They can encompass a wide range of items, including documents, test suites, and program components.
* In addition to software work products, some organizations also place software tools under configuration control.
* SCIs are organized to form configuration objects, which are cataloged in the project database and connected through relationships.
* Configuration objects can help identify how changes to one item might affect others.

This section provides an in-depth understanding of SCM, emphasizing the importance of baselines and how they help control change while acknowledging that most changes in software development are justified due to evolving knowledge and requirements. It also introduces the concept of configuration objects to manage SCIs effectively.

Software configuration management is an important element of software quality

assurance. Its primary responsibility is the control of change. However, SCM is also

responsible for the identification of individual SCIs and various versions of the soft

ware, the auditing of the software configuration to ensure that it has been properly

developed, and the reporting of all changes applied to the configuration.

Any discussion of SCM introduces a set of complex questions:

• How does an organization identify and manage the many existing versions of

a program (and its documentation) in a manner that will enable change to be

accommodated efficiently?

• How does an organization control changes before and after software is

released to a customer?

• Who has responsibility for approving and ranking changes?

• How can we ensure that changes have been made properly?

• What mechanism is used to appraise others of changes that are made?

These questions lead us to the definition of five SCM tasks: identification, version con

trol, change control, configuration auditing, and reporting

**9.3 Identification of Objects in the Software Configuration**

**Object Types:**

* To effectively control and manage software configuration items (SCIs), each item must be uniquely named and organized using an object-oriented approach.
* Two types of objects are identified: basic objects and aggregate objects.
* Basic objects represent "units of text" created by software engineers during various phases of the software development process, such as sections of requirements specifications, source listings, or test suites.
* Aggregate objects, such as the Design Specification, are collections of basic objects and other aggregate objects.
* Each object has distinct features that identify it uniquely, including a name, description, list of resources, and realization.

**Object Interrelationships:**

* Objects are connected through interrelationships. A basic object can be <part-of> an aggregate object, defining a hierarchy.
* Objects can also be interrelated, even across different branches of the object hierarchy, to represent relationships between various elements.
* Interrelationships can be represented using a module interconnection language (MIL) to describe dependencies among objects.
* Configuration objects can help assess the impact of changes on related objects.

**Evolution and Version Control:**

* Objects evolve throughout the software process. Changes occur before an object is baselined and even after baseline establishment.
* An evolution graph can describe the change history of an object, illustrating its evolution from one version to another.
* It's essential to have effective identification schemes in place to manage changes and ensure that all components and documents for different versions are properly referenced.
* Automated Software Configuration Management (SCM) tools play a crucial role in managing identification tasks and object versions.

**9.4 Version Control**

**Version Control Concept:**

* Version control combines procedures and tools to manage different versions of configuration objects created during the software development process.
* It allows users to specify alternative configurations by selecting appropriate versions of objects and associating attributes with each software version.

**Attributes for Version Control:**

* Attributes can be as simple as version numbers attached to each object or as complex as Boolean variables indicating specific types of functional changes.
* Attributes help define which entities should be included when constructing a specific variant or version of software.

**Entity, Variant, and Version Representation:**

* Entities, variants, and versions can be represented as objects in a three-dimensional space, showing the relationship between them.
* An entity is a collection of objects at the same revision level, while a variant is a different collection of objects at the same revision level.
* A new version is defined when significant changes are made to one or more objects.

**Automated Approaches to Version Control:**

* Various automated approaches to version control have been proposed, differing in the sophistication of attributes used for constructing versions and variants and the mechanics of the process for construction.

This section provides a comprehensive understanding of how objects in software configuration are identified and how version control plays a vital role in managing different versions and variants of software. It highlights the importance of effective identification schemes and the role of automated SCM tools in this process.

**9.6 Configuration Audit**

**The Need for Configuration Audit:**

* While identification, version control, and change control are crucial for maintaining order in software development, they only track changes until an Engineering Change Order (ECO) is generated.
* To ensure that changes have been properly implemented and to assess other characteristics of configuration objects, a software configuration audit is essential.

**Formal Technical Review and Software Configuration Audit:**

* The formal technical review (discussed in Chapter 8) primarily focuses on the technical correctness of a modified configuration object. It is conducted to assess consistency with other objects, check for omissions, and identify potential side effects.
* A formal technical review should be conducted for most changes, except for the most trivial ones.

**Audit Questions in Configuration Audit:**

* A software configuration audit assesses a configuration object using questions that typically are not considered during a formal technical review.
* The following questions are asked and answered during a configuration audit:
  1. Has the change specified in the ECO been implemented, and are there any additional modifications?
  2. Has a formal technical review been conducted to assess technical correctness?
  3. Has the software development process been followed, and have software engineering standards been properly applied?
  4. Is the change highlighted in the configuration object, including the change date and author? Do the object attributes reflect the change?
  5. Have Software Configuration Management (SCM) procedures for noting, recording, and reporting the change been followed?
  6. Have all related configuration objects been correctly updated?

**Separate Conduct of Configuration Audit:**

* In some cases, the audit questions are asked as part of a formal technical review.
* However, when SCM is a formal activity, the SCM audit is conducted separately by the quality assurance group.

**9.7 Status Reporting**

**The Role of Configuration Status Reporting (CSR):**

* Configuration status reporting, or status accounting, is an SCM task that answers questions related to software changes. These questions include what happened, who did it, when it happened, and what else will be affected.
* CSR helps track and communicate changes in software projects and plays a critical role in the success of large development projects.

**Flow of Information in Configuration Status Reporting:**

* Each time an Software Configuration Item (SCI) is assigned new or updated identification, a CSR entry is made.
* When a change is approved by the Change Control Authority (CCA) through the issuance of an ECO, a CSR entry is recorded.
* The results of configuration audits are reported as part of CSR.
* CSR information may be placed in an online database for easy access by software developers and maintainers.
* Regular CSR reports are generated to keep management and practitioners informed about important changes.

**Importance of Communication in CSR:**

* CSR helps improve communication among all individuals involved in the software project.
* It prevents issues like conflicting modifications, development based on outdated information, and lack of awareness of critical changes.

This section emphasizes the importance of configuration audits and status reporting in ensuring that changes are correctly implemented and that information is effectively communicated within a software development project. It highlights the role of formal technical reviews and the separate conduct of SCM audits, as well as the significance of communication in managing software changes.