



Configuration Management

Outline of the Lecture

- ♦ Purpose of Software Configuration Management (SCM)
 - ♦ **Motivation: Why software configuration management?**
 - ♦ **Definition: What is software configuration management?**
 - ♦ **Activities and roles in software configuration management**
- ♦ Some Terminology
 - ♦ **Configuration Item, Baseline, SCM Directory, Version, Revision Release.**
- ♦ Software Configuration Management Activities
 - ♦ **Promotion Management, Release Management, Change Management**
- ♦ Outline of a Software Configuration Management Plans
 - ♦ **Standards (Example: IEEE 828-2005)**
 - ♦ **Basic elements of IEEE 828-2005**
- ♦ Configuration Management Tools

Why Software Configuration Management ?

- ♦ The problem:
 - ♦ Multiple people have to work on software that is changing
 - ♦ More than one version of the software has to be supported:
 - ♦ Released systems
 - ♦ Custom configured systems (different functionality)
 - ♦ System(s) under development
 - ♦ Software must run on different machines and operating systems
- ⇒ *Need for coordination*
- ♦ Software Configuration Management
 - ♦ manages evolving software systems
 - ♦ controls the costs involved in making changes to a system

What is Software Configuration Management?

- ♦ Definition:
 - ♦ A set of management disciplines within the software engineering process to develop a **baseline**.
- ♦ Description:
 - ♦ Software Configuration Management encompasses the disciplines and techniques of initiating, evaluating and controlling change to software products during and after the software engineering process.
- ♦ Standards (approved by ANSI)
 - ♦ IEEE 828-2005: Software Configuration Management Plans
 - ♦ IEEE 1042-1987: Guide to Software Configuration Management (*Archived*)

Managing Software Configurations

- ♦ Software Configuration Management is a **project function** (as defined in the software project management plan) with the goal to make technical and managerial activities more effective.
- ♦ Software Configuration Management can be staffed in several ways:
 - ♦ A single team performs all software configuration management activities for the whole organization
 - ♦ A separate configuration management team is set up for each project
 - ♦ All the software configuration management activities are performed by the developers themselves
 - ♦ Mixture of all of the above

Configuration Management Activities

- ♦ **Configuration item identification**
 - ♦ modeling of the system as a set of evolving components
- ♦ **Promotion management**
 - ♦ the creation of versions for other developers
- ♦ **Release management**
 - ♦ the creation of versions for the clients and users
- ♦ **Change management**
 - ♦ the handling, approval and tracking of change requests
- ♦ **Branch management**
 - ♦ the management of concurrent development efforts
- ♦ **Variant management**
 - ♦ the management of versions intended to coexist

- ◆ No fixed rules:
 - ◆ **SCM activities are usually performed in different ways (formally, informally)**
 - ◆ **depending on the project type and life-cycle phase (research, development, maintenance).**

Terminology

- ◆ We will define the following terms
 - ◆ **Configuration Item**
 - ◆ **Baseline**
 - ◆ **SCM Directories**
 - ◆ **Version**
 - ◆ **Revision**
 - ◆ **Release**
- ⇒ The definition of the terms follows the IEEE standard.
- ⇒ Different configuration management systems may use different terms.
 - ⇒ **Example: CVS configuration management system and ADAMS (that is used in our projects) use terms differing from the IEEE standard.**

Terminology: Configuration Item

“An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process.”

- ❖ Software configuration items are not only program code segments but all type of documents according to development, e.g.
 - ⇒ **all type of code files**
 - ⇒ **drivers for tests**
 - ⇒ **analysis or design documents**
 - ⇒ **user or developer manuals**
 - ⇒ **system configurations (e.g. version of compiler used)**
- ❖ In some systems, not only software but also hardware configuration items (CPUs, bus speed frequencies) exist!
- ❖ Even a commercial product used in the system can be a configuration item

Finding Configuration Items

- ◆ Large projects typically produce thousands of entities (files, documents, data ...) which must be uniquely identified.
- ◆ Any entity managed in the software engineering process can potentially be brought under configuration management control
- ◆ But not every entity needs to be under configuration management control all the time.
- ◆ Two Issues:
 - ◆ **What: Selection of Configuration Items**
 - ◆ What should be under configuration control?
 - ◆ **When: When do you start to place entities under configuration control?**
- ◆ Conflict for the Project Manager:
 - ◆ **Starting with CIs too early introduces too much bureaucracy**
 - ◆ **Starting with CIs too late introduces chaos**

Finding Configuration Items (continued)

- ◆ Some items must be maintained for the lifetime of the software. This includes also the phase, when the software is no longer developed but still in use;
- ◆ An entity naming scheme should be defined so that related documents have related names.

- ◆ Selecting the right configuration items is a skill that takes practice
 - ◆ **Very similar to object modeling**
 - ◆ **Use techniques similar to object modeling for finding CIs!**
 - ◆ Find the CIs
 - ◆ Find relationships between CIs

Which of these Entities should be Configuration Items?

- | | |
|---|---|
| ◆ Problem Statement | ◆ Source code |
| ◆ Software Project Management Plan (SPMP) | ◆ API Specification |
| ◆ Requirements Analysis Document (RAD) | ◆ Input data and data bases |
| ◆ System Design Document (SDD) | ◆ Test plan |
| ◆ Project Agreement | ◆ Test data |
| ◆ Object Design Document (ODD) | ◆ Support software that is part of the final system |
| ◆ Dynamic model | ◆ Support software that is not part of the product |
| ◆ Object model | ◆ User manual |
| ◆ Functional model | ◆ Administrator manual |
| ◆ Unit tests | |
| ◆ Integration test strategy | |

Possible Selection of Configuration Items

- ♦ Problem Statement
- ♦ Software Project Management Plan (SPMP)
- ✓ Requirements Analysis Document (RAD)
- ✓ System Design Document (SDD)
- ♦ Project Agreement
- ✓ Object Design Document (ODD)
- ♦ Dynamic Model
- ♦ Object model
- ♦ Functional Model
- ✓ Unit tests
- ♦ Integration test strategy
- ✓ Source code
- ♦ API Specification
- ✓ Input data and data bases
- ♦ Test plan
- ✓ Test data
- ✓ Support software (part of the product)
- ♦ Support software (not part of the product)
- ♦ User manual
- ♦ Administrator manual

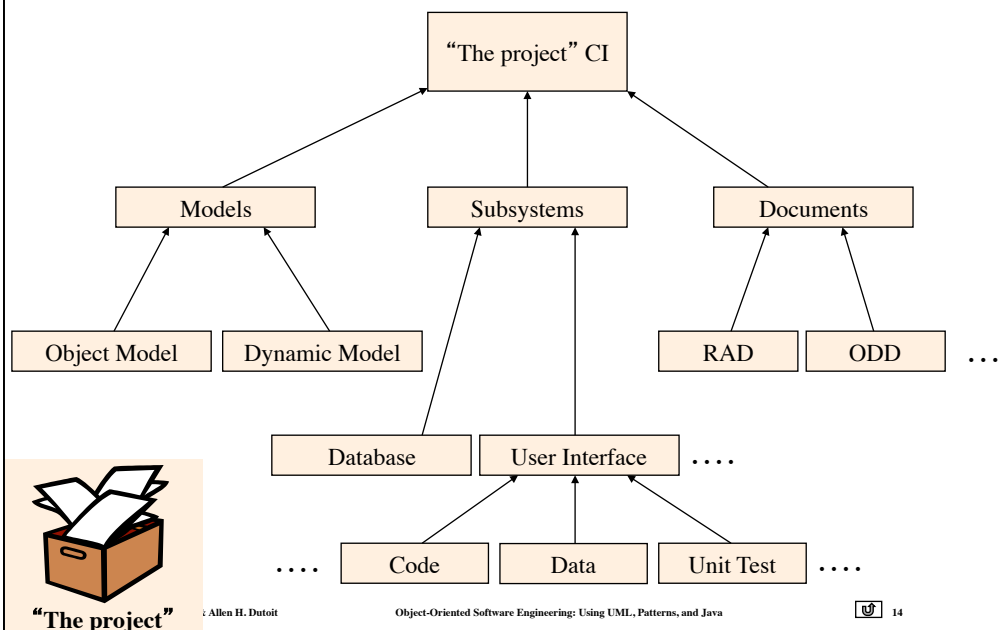
Once the Configuration Items are selected, they are usually organized in a tree

Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

13

Configuration Item Tree (Example)



Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

14

Terminology: Version

- ◆ The initial release or re-release of a configuration item associated with a complete compilation or recompilation of the item. Different versions have different functionality.

Terminology: Baseline



“A specification or product that has been formally reviewed and agreed to by responsible management, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures.”

Examples:

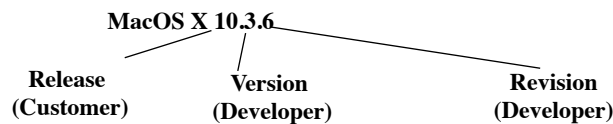
Baseline A: All the APIs have completely been defined; the bodies of the methods are empty.

Baseline B: All data access methods are implemented and tested.

Baseline C: The GUI is implemented.

More on Baselines

- ♦ As systems are developed, a series of baselines is developed, usually after a review (analysis review, design review, code review, system testing, client acceptance, ...)
- ♦ **Developmental baseline** (CIs: RAD, SDD, Integration Test, ...)
 - ♦ Goal: Coordinate engineering activities
- ♦ **Functional baseline** (CIs: first prototype, alpha release, beta release)
 - ♦ Goal: Get first customer experiences with functional system
- ♦ **Product baseline** (product)
 - ♦ Goal: Coordinate sales and customer support
- ♦ Many naming scheme for baselines exist (1.0, 3.14159, 6.01a,, ...)
- ♦ A 3 digit scheme is quite common:

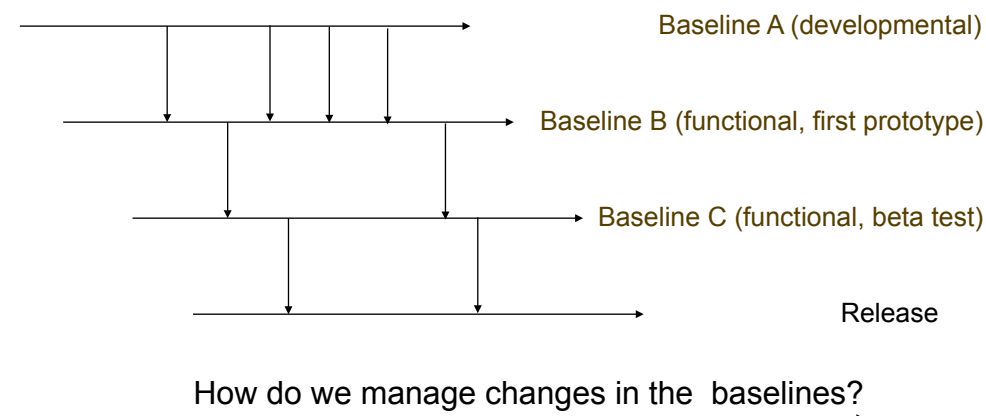


Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

17

Managing Baselines in SCM



Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

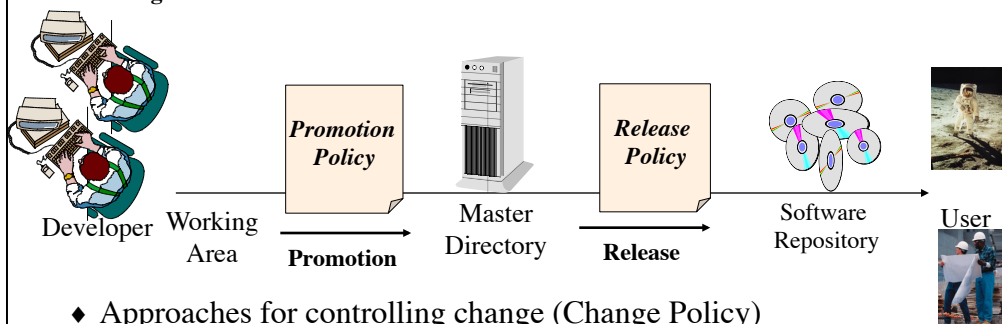
Time
18

Change management

- ♦ Change management is the handling of change requests
 - ♦ A change request leads to the creation of a new release
- ♦ General change process
 - ♦ The change is requested (this can be done by anyone including users and developers)
 - ♦ The change request is assessed against project goals
 - ♦ Following the assessment, the change is accepted or rejected
 - ♦ If it is accepted, the change is assigned to a developer and implemented
 - ♦ The implemented change is audited.
- ♦ The complexity of the change management process varies with the project. Small projects can perform change requests informally and fast while complex projects require detailed change request forms and the official approval by one more managers.

Controlling Changes

- ♦ Two types of controlling change:
 - ♦ **Promotion:** The internal development state of a software is changed.
 - ♦ **Release:** A changed software system is made visible outside the development organization.

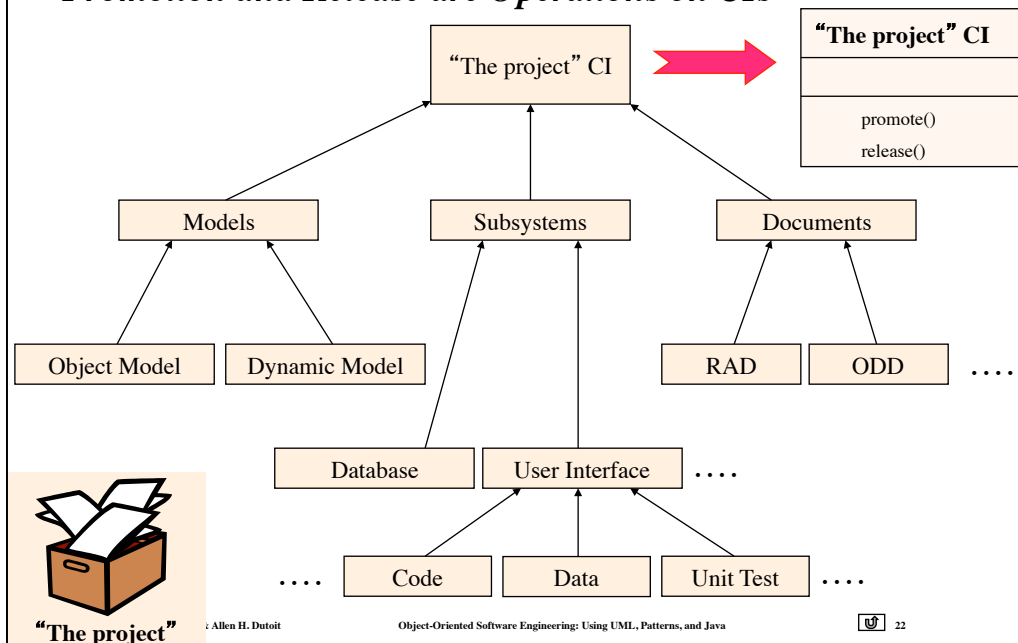


- ♦ Approaches for controlling change (Change Policy)
 - ♦ Informal (good for research type environments and promotions)
 - ♦ Formal approach (good for externally developed CIs and for releases)

Terminology: SCM Directories

- ♦ **Programmer's Directory** (IEEE: Dynamic Library)
 - ♦ Library for holding newly created or modified software entities
 - ♦ The programmer's workspace is controlled by the programmer only
- ♦ **Master Directory** (IEEE: Controlled Library)
 - ♦ Central directory for all promotions
 - ♦ Manages the current baseline(s) and for controlling changes made to them
 - ♦ Entry is controlled, usually after verification
 - ♦ Changes must be authorized
- ♦ **Software Repository** (IEEE: Static Library)
 - ♦ Archive for the various baselines (externally) released for general use
 - ♦ Copies of these baselines may be made available to requesting organizations

Promotion and Release are Operations on CIs



Let 's Create a Model for Configuration Management

We just learned that promotions are stored in the master directory and releases are stored in the repository

Problem: There can be many promotions and many releases

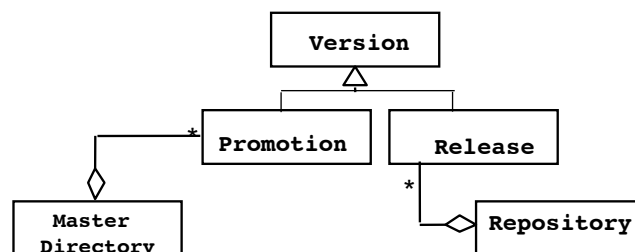
Solution: Use Multiplicity



Let 's Create a Model for Configuration Management

Insight: Promotions and Releases are both versions

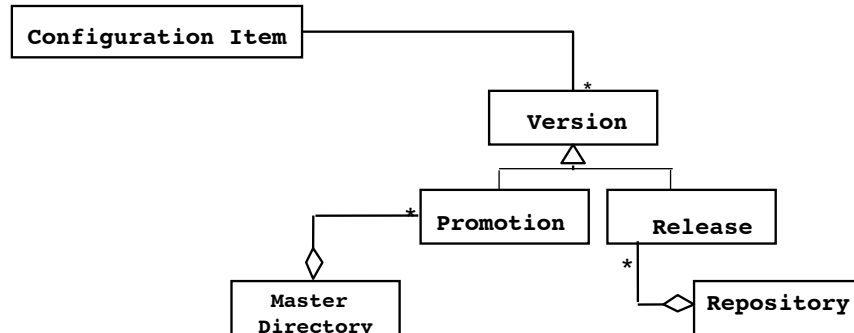
Solution: Use Inheritance



Let 's Create a Model for Configuration Management

Problem: A configuration item has many versions

Solution: Create a 1-many association between Configuration Item and Version



Bernd Bruegge & Allen H. Dutoit

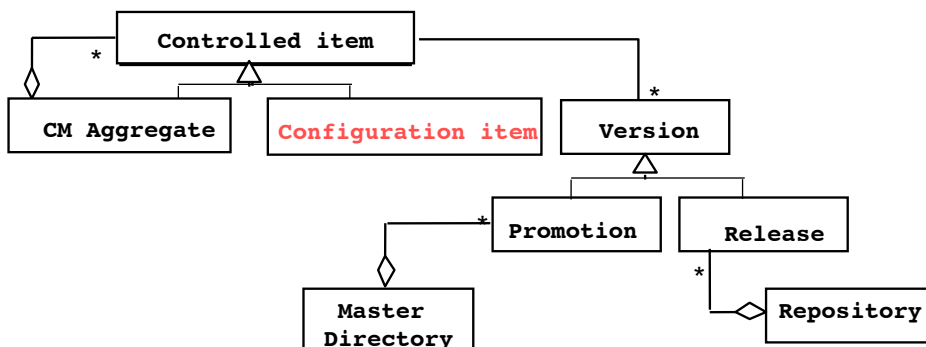
Object-Oriented Software Engineering: Using UML, Patterns, and Java

25

Let 's Create a Model for Configuration Management

Problem: Configuration items can themselves be grouped

Solution: Use the composite design pattern

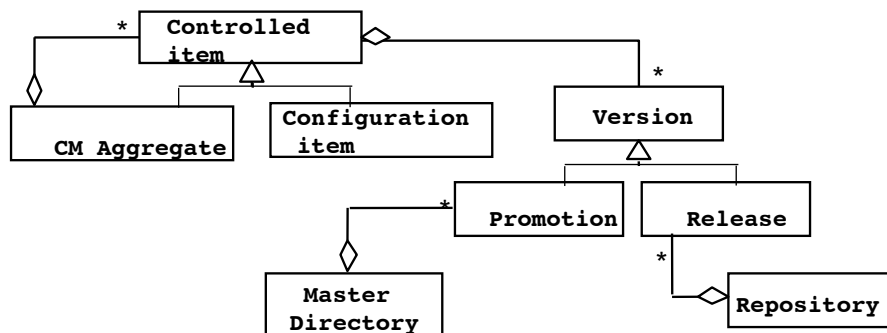


Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

26

Configuration Item Model (UML Class Diagram)



Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

27

Change Policies

- ◆ Whenever a promotion or a release is performed, one or more policies apply. The purpose of change policies is to guarantee that each **version**, **revision** or **release** conforms to commonly accepted criteria.



- ◆ Examples for change policies:

"No developer is allowed to promote source code which cannot be compiled without errors and warnings."

"No baseline can be released without having been beta-tested by at least 500 external persons."

Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

28

Terminology: Version vs Revision vs Release

Version:

- ♦ The state of a configuration item or configuration aggregate at a well-defined point in time.
- ♦ It is usually associated with a *complete compilation* or recompilation of the item.
- ♦ Different versions usually have different functionality.

Revision:

- ♦ *Change* to a version that corrects only errors in the design/code, but does not affect functionality.

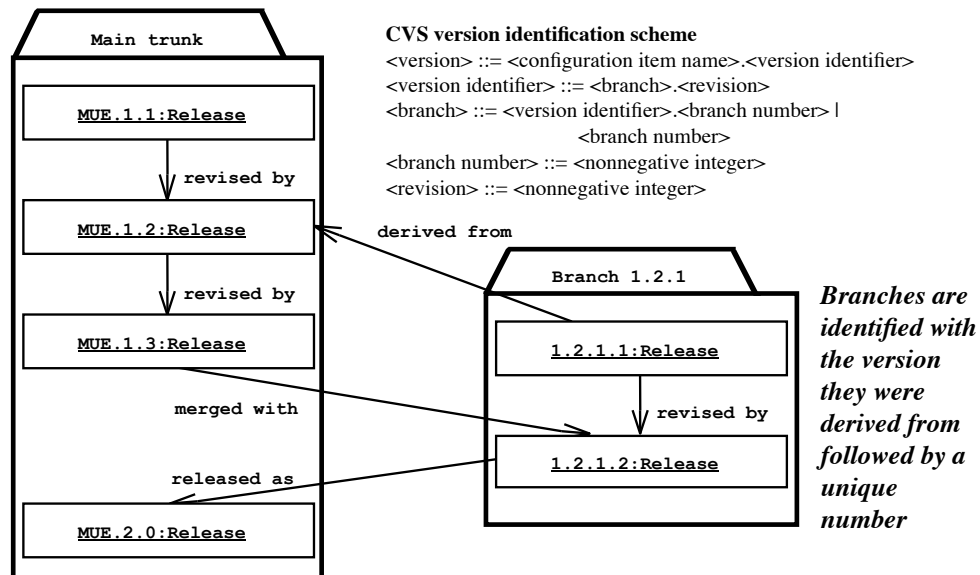
Release:

- ♦ A version that has been made available externally.
- ♦ The *formal distribution* of an approved version.

Branch Management

- ♦ While software is released in a sequential way, the development of different features can be done by different teams concurrently and later merged into a single version
- ♦ A **branch** identifies a concurrent development path requiring independent configuration management
- ♦ The sequence of versions created by each team is a branch, which is independent from the versions created by the other teams

Example: CVS version identification scheme



Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

31

Variant Management

- ◆ Variants are version that are intended to coexist
 - ◆ **Examples: variants for different platforms, variants released with multiple levels of functionality**
- ◆ Two approaches to deal with variants
- ◆ *Redundant teams.*
 - ◆ A team is assigned to each variant.
 - ◆ Each team is given the same requirements and is responsible for the complete design, implementation, and testing of the variants
 - ◆ A small number of configuration items are shared across variants
- ◆ *Single project.*
 - ◆ Design a subsystem decomposition that maximizes the amount of code shared across variants (core subsystems)
 - ◆ For multiple platforms, core subsystems are lower level subsystems
 - ◆ For multiple levels of functionality, confine increments of functionality in mostly independent subsystems

Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

32

Software Configuration Management Planning

- ◆ Software configuration management planning starts during the early phases of a project.
- ◆ The outcome of the SCM planning phase is the *Software Configuration Management Plan (SCMP)* which might be extended or revised during the rest of the project.
- ◆ The SCMP can either follow a public standard like the IEEE 828, or an internal (e.g. company specific) standard.

The Software Configuration Management Plan

- ◆ Defines the *types of documents* to be managed and a document naming scheme.
- ◆ Defines *who takes responsibility* for the CM procedures and creation of baselines.
- ◆ Defines *policies for change control* and version management.
- ◆ Describes the *tools* which should be used to assist the CM process and any limitations on their use.
- ◆ Defines the *configuration management database* used to record configuration information.

Outline of a Software Configuration Management Plan (SCMP, IEEE 828-2005)

1. Introduction
 - ♦ **Describes purpose, scope of application, key terms and references**
2. Management (WHO?)
 - ♦ **Identifies the responsibilities and authorities for accomplishing the planned configuration management activities**
3. Activities (WHAT?)
 - ♦ **Identifies the activities to be performed in applying to the project.**
4. Schedule (WHEN?)
 - ♦ **Establishes the sequence and coordination of the SCM activities with project mile stones.**
5. Resources (HOW?)
 - ♦ **Identifies tools and techniques required for the implementation of the SCMP**
6. Maintenance
 - ♦ **Identifies activities and responsibilities on how the SCMP will be kept current during the life-cycle of the project.**

SCMP Section 1: Introduction

- 1.1 Simplified overview of the configuration management activities.
- 1.2 Scope:
 - ♦ **Overview description of the project**
 - ♦ **Identification of the CI(s) to which software configuration management will be applied.**
- 1.3 Identification of other software to be included as part of the SCMP (support software and test software)
- 1.4 Relationship of SCM to hardware of system configuration management activities
- 1.5 Degree of formality and depth of control for applying SCM to project.
- 1.6 Limitations and time constraints for applying SCM to this project
- 1.7 Assumptions that might have an impact on the cost, schedule and ability to perform defined SCM activities.

SCMP Section 2: Management

2.1 Organization

- ♦ **Organizational context (technical and managerial) within which the SCM activities are implemented. Identifies**
 - ♦ **All organizational units (client, developers, managers) that participate in an SCM activity**
 - ♦ **Functional roles of these people within the project**
 - ♦ **Relationship between organizational units**

2.2. Responsibilities

- ♦ **For each SCM activity list the name or job title to perform this activity**
- ♦ **For each board performing SCM activities, list**
 - ♦ **purpose and objectives**
 - ♦ **membership and affiliations**
 - ♦ **period of effectivity, scope of authority**
 - ♦ **operational procedures**




3. Applicable Policies

- ♦ **External constraints placed on the SCMP**

Typical Configuration Management Roles

- ♦ **Configuration Manager**
 - ♦ **Responsible for identifying configuration items. The configuration manager can also be responsible for defining the procedures for creating promotions and releases**
- ♦ **Change control board member**
 - ♦ **Responsible for approving or rejecting change requests**
- ♦ **Developer**
 - ♦ **Creates promotions triggered by change requests or the normal activities of development. The developer checks in changes and resolves conflicts**
- ♦ **Auditor**
 - ♦ **Responsible for the selection and evaluation of promotions for release and for ensuring the consistency and completeness of this release**





SCMP Section 3: Activities

- 3.1 Configuration Identification
- 3.2 Configuration Control 
- 3.3 Configuration Status Accounting 
- 3.4 Configuration Audits and Reviews 
- 3.5 Interface Control
- 3.6 Subcontractor/Vendor control
- 3.7 Release management and delivery

3.2 Configuration Control



Defines the following steps

- 3.2.1 How to identify the need for a change (layout of change request form)** 
- 3.2.2 Analysis and evaluation of a change request** 
- 3.2.3 Approval or disapproval of a request** 
- 3.2.4 Verification, implementation and release of a change** 

3.2.1 Change Request

- ◆ Specifies the procedures for requesting a change to a baselined CI and the information to be documented:
 - ◆ **Name(s) and version(s) of the CI(s) where the problem appears**
 - ◆ **Originator's name and address**
 - ◆ **Date of request**
 - ◆ **Indication of urgency**
 - ◆ **The need for the change**
 - ◆ **Description of the requested change**

3.2.2 Evaluation of a Change

- ◆ Specifies the analysis required to determine the impact of proposed changes and the procedure for reviewing the results of the analysis.

3.2.3 Change Approval or Disapproval

- ◆ This section of the SCMP describes the organization of the configuration control board (CCB).
- ◆ Configuration Control Board (CCB)
 - ◆ **Can be an individual or a group.**
 - ◆ **Multiple levels of CCBs are also possible, depending on the complexity of the project**
- ◆ Multiple levels of CCBs may be specified.
 - ◆ **In small development efforts one CCB level is sufficient.**
- ◆ This section of the SCMP also indicates the level of authority of the CCB and its responsibility.
 - ◆ **In particular, the SCMP must specify when the CCB is invoked.**

3.2.4 Implementing Change



- ◆ This section of the SCMP specifies the activities for verifying and implementing an approved change.
- ◆ A completed change request must contain the following information:
 - ◆ **The original change request(s)**
 - ◆ **The names and versions of the affected configuration items**
 - ◆ **Verification date and responsible party**
 - ◆ **Identifier of the new version**
 - ◆ **Release or installation date and responsible party**
- ◆ This section must also specify activities for
 - ◆ **Archiving completed change requests**
 - ◆ **Planning and control of releases**
 - ◆ **How to coordinate multiple changes**
 - ◆ **How to add new CIs to the configuration**
 - ◆ **How to deliver a new baseline**

3.3 Configuration Status Accounting

- ◆ This section of the SCMP must contain the following sections
 - ◆ **What elements are to be tracked and reported for baselines and changes?**
 - ◆ **What types of status accounting reports are to be generated? What is their frequency?**
 - ◆ **How is information to be collected, stored and reported?**
 - ◆ **How is access to the configuration management status data controlled?**

3.4 Configuration Audits and Reviews

- ◆ This section of the SCMP identifies audits and reviews for the project.
 - ◆ **An audit determines for each Configuration Item if it has the required physical and functional characteristics.**
 - ◆ **A review is a management tool for establishing a baseline.**
- ◆ For each audit or review the plan has to define:
 - ◆ **Objective**
 - ◆ **The Configuration Items under review**
 - ◆ **The schedule for the review**
 - ◆ **Procedures for conducting the review**
 - ◆ **Participants by job title**
 - ◆ **Required documentation**
 - ◆ **Procedure for recording deficiencies and how to correct them**
 - ◆ **Approval criteria**

Tailoring the SCMP

- ◆ The IEEE standard allows quite a bit flexibility for preparing an SCMP.
- ◆ A SCMP may be
 - ◆ **tailored upward:**
 - ◆ to add information
 - ◆ to use a specific format
 - ◆ **tailored downward**
 - ◆ Some SCMP components might not apply to a particular project.
 - ◆ Instead of omitting the associated section, mention its applicability.
 - ◆ Information that has not been decided on at the time the SCMP is approved should be marked as “to be determined”.

Conformance to the IEEE Standard 828-2005

- ◆ Presentation format & Minimum information
 - ◆ A separate document or a section embedded in another document titled “Software Configuration Management Plan”.
 - ◆ 6 Sections: Introduction, Management, Activities, Schedules, Resources and Plan Maintenance
- ◆ Consistency Criteria:
 - ◆ All activities defined in the SCMP (Section 3.1 to 3.6) are assigned to an organizational unit or person and they are associated with resources to accomplish the activities.
 - ◆ All Configuration items identified in Section 2.1 have defined processes for baseline establishment and change control (Section 3.2) .
- ◆ If the above criteria are met, the SCMP can include the following sentence:

“This SCMP conforms with the requirements of IEEE Std 828-2005.”
- ◆ Note: The consistency criteria can also be used at a SCMP review meeting

Example SCM Plans (from the Guide IEEE 1042.1990)

Life-cycle Phase	Project Type	Size	SCM Tools	Life Span	Writing	Character of Project
A Development	Critical	Medium	Advanced	Short	Highly Structured	Complex system contracted to another company
B Concept	Prototype	Small	Basic	Short	Informal	Small software development project
C Maintenance	Support Software	Large	On-line	Full Life-Cycle	Structured	SCMP used by organization using contracted SW
D All	Commercial	Small	Integrated	Full Life-Cycle	Informal	Development of embedded applicataions

ARENA: Concept, Prototype, Small, On-line, Full Life-Cycle, Informal

Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

 49

Tools for Software Configuration Management

- ♦ Software configuration management is normally supported by tools with different functionality.
- ♦ Examples:
 - ♦ **RCS**
 - ♦ very old but still in use; only version control system
 - ♦ **CVS (Concurrent Version Control)**
 - ♦ based on RCS, allows concurrent working without locking
 - ♦ <http://www.cvshome.org/>
 - ♦ CVSWeb: Web Frontend to CVS
 - ♦ **Perforce**
 - ♦ Repository server; keeps track of developer's activities
 - ♦ <http://www.perforce.com>
 - ♦ **ClearCase**
 - ♦ Multiple servers, process modeling, policy check mechanisms
 - ♦ <http://www.rational.com/products/clearcase/>

Bernd Bruegge & Allen H. Dutoit

Object-Oriented Software Engineering: Using UML, Patterns, and Java

 50

Tasks for the Configuration Manager (Summary)

SCMP following the IEEE 828-2005 standard

Define configuration items

Define promote /release policy

Define activities and responsibilities

Set up configuration management system

References

- ◆ Readings used for this lecture
 - ◆ [Bruegge-Dutoit] Chapter 13 Configuration Management
 - ◆ [IEEE Std 828] Software Configuration Management
 - ◆ [IEEE Std 1042] Guide to Configuration Management Plan (SCMP)
- ◆ Additional References
 - ◆ CVS
 - ◆ Homepage: <http://www.cvshome.org/>
 - ◆ Online Documentation: <http://www.cvshome.org/docs/manual/cvs.html>
 - ◆ Jikes: Open Source Java Compiler maintained with CVS
 - ◆ Source tree (read only): <http://sourcery.org/jikes/anoncv.html>
 - ◆ Jikes project portal <http://sourcery.org/jikes>
 - ◆ CVSWEB example
 - ◆ <http://stud.fh-heilbronn.de/~zeller/cgi/cvsweb.cgi/>

Summary

- ◆ Software Configuration Management: Important part of project management to manage evolving software systems and coordinate changes to them.
- ◆ Software Configuration Management consists of several activities:
 - ◆ **Promotion and Release management**
 - ◆ **Branch, Variant and Change Management**
- ◆ Public standard for SCM plans: IEEE 828.
- ◆ The standard can be tailored to a particular project:
 - ◆ **Large projects need detailed plans to be successful**
 - ◆ **Small projects should not be burdened with the bureaucracy of detailed SCM plans**
- ◆ SCM should be supported by tools. These range from
 - ◆ **Simple version storage tools**
 - ◆ **Sophisticated systems with automated procedures for policy checks and support for the creation of SCM documents.**