SCM and the CR Workflow

Idaho State University | Computer Science

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Idaho State University
Computer Science

Outcomes

After today's lecture you will:

- Know the benefits of SCM in Software Projects
- Be aware of the functions of SCM systems
- Understand how Version Control fits in with SCM
- Understand the basic ideas of the CR Workflow

ROAR

Software Configuration Management

CS 4423/5523

ROAR

Configuration Management

- The concept of configuration management (CM) was developed to manage changes in large systems.
- It handles the control of all product items and changes to those items
- Software Configuration Management (SCM) is applied to software products
- The product items include document, executable software, source code, hardware, and disks
- SCM accrues two kinds of benefits to an organization
  - SCM ensure that development processes are traceable and systematic so that all changes are precisely managed
  - SCM enhances the quality of the delivered system and the productivity of the maintainers

SCM Objectives

- The objectives of SCM are to:
  - Uniquely identified every version of every software at various points in time
  - Retain past versions of documentations and software
  - Provide a trail of audit for all modifications performed
  - Throughout the software life-cycle, maintain the traceability and integrity of the system changes

Idaho State University
Computer Science

SCM Project Benefits

1. Confusion is reduced and order is established
2. To maintain product integrity, the necessary activities are organized
3. Correct product configurations are ensured
4. Quality is ensured – and better quality software consumes less maintenance efforts
5. Productivity is improved, because analysts and programmers know exactly where to find any piece of the software
6. Liability is reduced by documenting the trail of actions
7. Life-cycle cost is reduced
8. Conformance with requirements is enabled
9. A reliable working environment is provided
10. Compliance with standards is enhanced
11. Accounting of status is enhanced

ROAR

Brief History

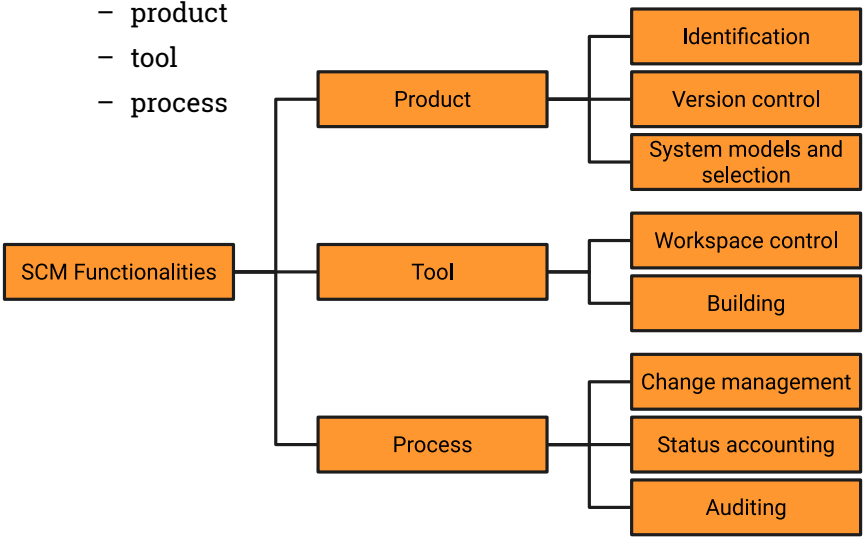
- A need for configuration management was originally felt in the aerospace industry in the 1950s
- In the 1970s, large scale computer software began to pose many of those same change management problems
- Software maintenance engineers borrowed configuration management techniques from the aerospace industry to manage software modifications
- In the beginning, punch cards with different colors were used to indicate changes
- In the late 1960s, to indicate changes to the UNIVAC-1100 EXEC-9 operating system, maintenance personnel used “corrective cards.”
- At that time, development of operating systems benefited from SCM

Brief History

- In the **1970s** and the **1980s**, SCM emerged as a distinct discipline
- The Unix-based software tool **Make** accepts descriptions of system configuration, and can automatically construct the system from its descriptions
- In the 1980s, **delta** algorithms based on text matching were developed to enable SCM tools to store just the differences among versions
- Consequently, novel algorithms were developed for efficient storage and retrieval of non-textual objects
- These days SCM systems support the management of evolution of a broad range of software systems that are being modified by a large number of maintenance personnel working in different countries and utilizing a variety of machines

SCM Spectrum of Functionality

- Estublier et al. classified the functionalities into three broad areas:
  - product
  - tool
  - process



Identification

- The items whose configuration need to be managed are identified in this function, including:
  - specification
  - design
  - documents
  - data
  - drawings
  - source code
  - test plan
  - test script
  - hardware components
  - components of the dev environment
    - compilers
    - debuggers
    - emulators
- Project plan and customer requirements should also be included
- To accurately identify products, including their configuration and version levels, a schema of names and numbers is designed
- Finally, for all configuration items and systems, a baseline configuration is established

Version Control

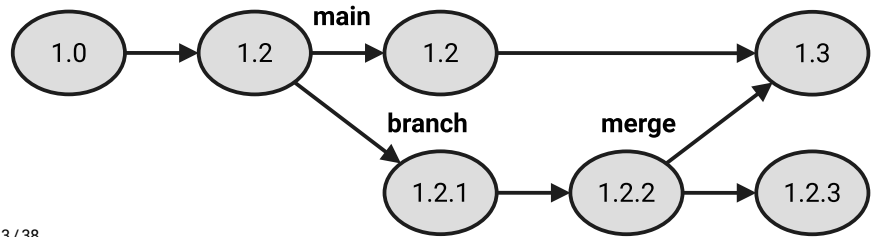
- To avoid confusion during the process of artifact evolution, a new identifier is assigned to the artifact every time the artifact is modified
- One may be interested in recording a fact that a given artifact fixes a subset of defects found in an earlier release
- The aforementioned kind of relation can be recorded by means of the version control (VC) functionality of SCM by:
  1. interpreting software artifacts as configuration items
  2. identifying the relations, if there is any, among the configuration items

Version Control

- The basic version control idea is to have two separate files: master copies and working copies
- The former is stored in a centralized repository
- Software developers check out working copies from the repository, modify the working copies, and, finally, check in the working copies into the repository
- Checking in a file means committing to the changes made to the working copies
- Conflicts can arise if many software developers want to use the same version of a file
- Conflicts can be resolved by means of two techniques: lock-modify-unlock and copy-modify-merge

Version Control

- Version control must support parallel development by allowing branching of versions
- Consider the scenario:
  1. an organization is currently developing the next version of their already released application, and
  2. a report about a major defect is received from the end users
- Now the development group has the option to retrieve the released version and create a branch, as illustrated in the Figure, to fix the defect



System Models and Selections

- It is neither efficient nor effective to manage a project file-by-file
- There is a need to support aggregate artifacts so that maintenance personnel can enforce consistency in large projects by means of relationships among artifacts and attributes
- Relationships among artifacts and attributes are captured by developing models which support the idea of software configurations
- Intuitively, a configuration means an aggregate of versionable items
- The general idea of configuration raises a need for enabling users to have selective access to parts and versions of such aggregated artifacts
- By default, SCCS and RCS keep in the the workspace the most recent version of the principal variant

Workspace Control

- An environment that enables the maintainer to make and test the changes in an isolated manner is called workspace.
- In an SCM system, software versions are stored in a repository that cannot be directly modified. Rather, when a need to modify some files arises, the files are copied into a workspace.
- One can realize a workspace in two ways:
  1. it can be as simple as the home directory of the programmer who wants to modify the files
  2. it can be a complex mechanism such as an IDE and a database

Workspace Control

- In general, three basic functions are performed in a workspace:
  - **Sandbox**: Checked out files are put in a workspace to be freely edited. In addition, it is not necessary to lock the original files in the repository.
  - **Building**: An SCM system general stores the differences between successive versions to save space. Therefore, the workspace expands the deltas into full-fledged source files. In addition, the workspace stores the derived binaries.
  - **Isolation**: Every developer maintains at least one workspace. Therefore, the developer makes modifications to the source code, compiles the files, performs tests, and debugs code without impacting the works of other developers.

Building

- Efficiency is a key requirement of SCM systems so that developers can quickly build an executable file from the versioned source files.
- Second requirement of SCM systems is that it must enable the building of old versions of the system for recovery, testing, maintenance, or additional release purpose.
- Finally, SCM systems must support building of software
- The build process and their products are assessed for quality assurance
- Outputs of the build process become quality assurance records, and the records may be needed for future reference
- The **make** application on the Unix OS, originally developed by researchers at Bell Labs, is a classical example of a build process
- Commercial SCM systems such as ClearCase continue to rely on variants of **make**.

Change Management

- SCM systems must:
  1. enable users to understand the impact of modifications
  2. enable users to identify the products to which a specific modification applies
  3. provide maintenance personnel with tools for change management so that all activities from specifying requirements to coding can be traced
- In the beginning, CRs were managed in paper form
- However, these days CRs are saved in the SCM repository and are linked with the actual modifications, in addition to being automated.

Accounting

- The primary purpose of status accounting is to:
  1. keep formal records of already existing configurations
  2. produce periodic reports about the status of the configurations
- These records:
  - describe product correctly
  - are used to verify the configuration of the system for testing and delivery
  - maintain a history of change requests, including both the approved ones and the rejected ones

Accounting

- A history of change request includes the answers to the following questions:
  - Why are changes made?
  - When are the changes made?
  - Who makes the changes?
  - What changes are made?
- Status accounting is useful in communicating important details of the project and configuration items to the stakeholders of the project.

Auditing

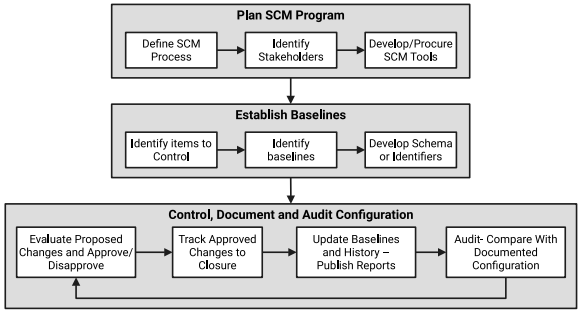
- SCM systems need to provide the following features:
  1. roll back to earlier stable points
  2. identify which modifications were performed, why those modifications were performed, and who performed those modifications
- By means of auditing, the organization maintains the integrity of the baselines and release configurations for all products
- Two kinds of audits are performed before a software product is released:
  - **audit for functional configuration:** determines whether or not the software satisfies the user requirement specification and the system requirement specification
  - **audit for physical configuration:** it verifies if the reference and design documents accurately represent the software

Auditing

- Overall, a **configuration audit** tries to find answers to the following:
  - To what extent are the requirements satisfied by the modified system?
  - Does the software release under consideration reflect the modification requests?
- The activities to perform a **configuration audit** are as follows:
  - Procedures and an audit schedule are defined
  - The personnel to perform the audits are identified
  - Establishes baselines are audited
  - Audit reports are generated

SCM Process

- The three major SCM implementation activities are:
  - planning
  - baseline development
  - configuration control



Planning

- Planning is begun with two activities:
  1. define the SCM process
  2. establish procedures
- A key step during planning is the identification of the stakeholders
- The stakeholders in a configuration management are:
  - maintainers
  - development engineers
  - sustaining test engineers
  - quality assurance auditors
  - users
  - the management

Configuration Control Board

- The stakeholders are also known as configuration control board (CCB) members
- Not all changes are reviewed by the board. Rather, small groups review and approve most of the changes
- Therefore, those groups need to be identified in the planning phase.
- Various SCM tools are used to maintain configuration history and facilitate the SCM process flow
- Examples of such tools are CVS (concurrent version system) and ClearCase

Establishing Baseline

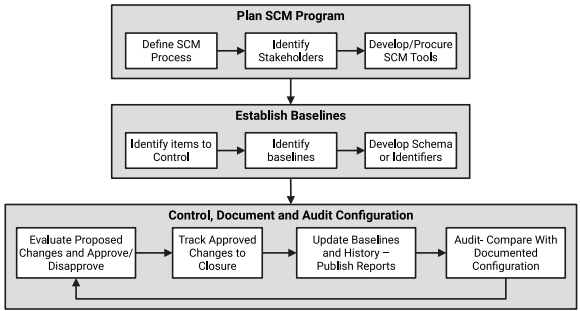
- Once an SCM program plan is in place, the next step is identification of items (code, data, and documents) that are the subject of configuration control
- After the configuration items identified, a software baseline library is established to make the set of configurable items publicly available
- The library, called repository, is the heart of the SCM system
- The repository has information about all the baselined items

Baselining Process

1. Create a snapshot of the current version of the product and it s configuration items, and allocate a configuration identifier to the entire configuration
2. Allocate version numbers to the configuration items and check in the configuration
3. Store the approved authority information as part of meta data in the repository
4. Broadcast all of the above information to the stakeholders
5. To accurately identify the configuration version, design a schema of words, numbers, or letters for common types of configuration items. In addition, project requirements may dictate specific nomenclature

Controlling, Documenting and Auditing

- After establishing a baseline, it is important to:
  1. keep the actual and the documented configuration identical
  2. ensure that the baseline complies with a project's configuration described in the requirements document
- The aforementioned requirements of a baseline are realized by means of a four-step iterative process as illustrated



Controlling, Documenting and Auditing

- The stakeholders specified in the SCM plan review and evaluate all changes to the configuration
- After their evaluations, both approvals and disapprovals are documented
- Approved changes are tracked until they are verified
- Next, the appropriate baseline is revised in conjunction with all relevant documents, and reports are generated.
- At regular intervals, records and products are audited to verify that:
  - There is acceptable matching between the documented configuration and the actual configuration
  - The configuration conforms with the requirements of the project
  - Documentations of all change activities are complete and up-to-date
- The three steps in the cycle, namely, controlling, documenting, and auditing, are repeatedly executed throughout the lifetime of the project

Change Request Workflow

CS 4423/5523

ROAR

Change Request Workflow

- A change request (CR), also called a modification request (MR), is a vehicle for recording information about a system defect, requested enhancement, or quality improvement
- Change requests are placed under the control of a change management system
- Change management systems control changes by an automated system in the form of work-flow.
- The basic objective of change management is to uniquely identify, describe, and track the status of each requested change

Change Management Objectives

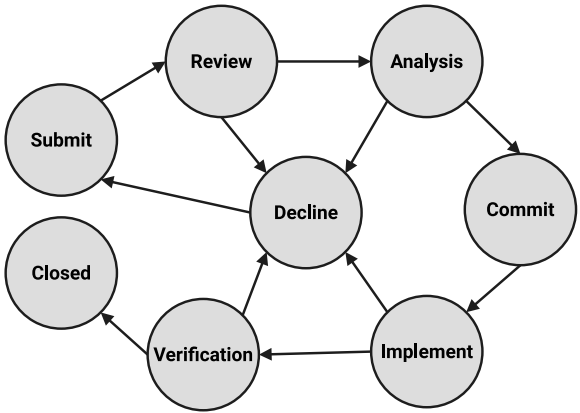
- The objectives of change management system are as follows:
  - Provide a common method for communication among stakeholders
  - Uniquely identify and track the status of each CR. This feature simplifies progress reporting and provides better control over changes
  - Maintain a database about all changes to the system. This information can be used for monitoring and measuring metrics

Change Request Workflow

- A change request describes the desires and needs of users which the system is expected to implement
- While describing a CR, two factors need to be taken into account:
  - Correctness of CRs
  - Clear communication of CRs to the stakeholders
- The results of interpreting a CR in different ways are as follows:
  - The team carrying out actual changes to the system and the team performing tests may develop contradicting views about the new system's quality
  - The changed system may not meet the needs and desires of the end users
- CRs need to be represented in an unambiguous manner, and made available in a centralized repository
- Wide availability of CRs to all the stakeholders is likely to reveal differences in interpretations by different groups

Change Request Workflow

- The life-cycle of a CR is shown below, by means of a state-transition diagram
- Each state represents a distinct stage in the life-cycle of a CR



Idaho State University
Computer Science

Change Request Schema

Field Name	Description
change_request_id	A unique identifier of the CR
title	A concise summary of the CR
description	A short description of the CR
maintenance_type	Classification fo the maintenance type in terms of a member of {Corrective, Adaptive, Perfective, Preventive}
product	Product name
component	Component where the change is needed, or where the problem occurred
state	Present state of the CR in terms of a member of {Submit, Review, Analysis, Commit, Implementation, Verification, Closed, Declined}
customer	Name of the customer making the change request
problem_origin	The origin of the problem
impacts	Components that are affected by a change and its ripple effect
resolution	Documentation of what was changed, how, and why
note	Additional information provided by the submitter for subsequent decision making
software_release	The version number of the product release in which the CR is likely to be effective
committed_release	The version number of the product release in which the CR will be effective
priority	Priority of CR, which is an element of a set, namely, {normal, high}
severity	Severity of CR, which is an element of a set, namely, {normal, critical}
marketing_justification	The business justification for the CR to exist
time_to_implement	The time, in person-week, required to effect the change
eng_assigned	The engineering personnel assigned to analyze the CR

ROAR

Idaho State University
Computer Science

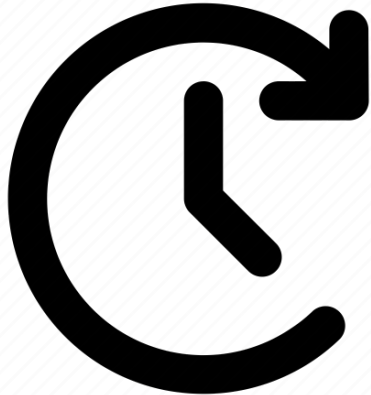
Change Request Schema

Field Name	Description
functional_spec_title	Title of the specification for functional requirements
functional_spec_name	Name of the file describing the functional requirements
functional_spec_version	This is the most recent version number of the specification of functional requirements
decline_note	The reason for decline the CR
ec_note	The identifier of the engineering change (EC) document
attachment	Attachment to further describe the CR (if any)
tc_id	Identifiers of test cases used in effecting the CR
tc_results	The result of testing: {Untested, Passed, Failed, Blocked, Invalid}
verification_method	Record the methods of verification of the CR: analysis, testing, inspection, and/or demonstration
verification_status	The verification state of the CR: Passed, Failed, or Incomplete
compliance	The level of compliance: {Non-compliance, Partial Compliance, or Compliance}
testing_note	Reports from the test personnel, possibly describing the demonstration given to the customers, analysis performed on the change, or inspection of the code performed by test personnel
defect_id	Defect identifier. If "Failed" assigned to the tc_results field, the defect identifier is associated with the failed test to indicate the defect causing the failure. The defect identifier is obtained from the test database.

ROAR

For Next Time

- Review EVO Chapter 3.8 - 3.10
- Read EVO Chapter 4.1 - 4.4.1
- Watch Lecture 07
- **4423: Course Project Part 1: Team Selection is due Friday at 11:00 pm**
- **4423: Continue working on Homework 01**
- **4423: Weekly Quiz due by Sunday at 11:00 pm**



Are there any questions?