Lecture 13

- Software Maintenance & Control

“How do we manage the changes?”

Topics

- Project Monitoring & Control
- Software Configuration Management
- Software Maintenance

1. Project Monitoring & Control

Definitions

- *Monitoring* is collecting, recording & reporting information concerning any aspect of project management
- *Controlling* uses the data supplied by monitoring to bring actual performance closer to planned performance
- *Evaluation* is the judgment about quality & effectiveness of project management

What Must Be Controlled and Monitored?

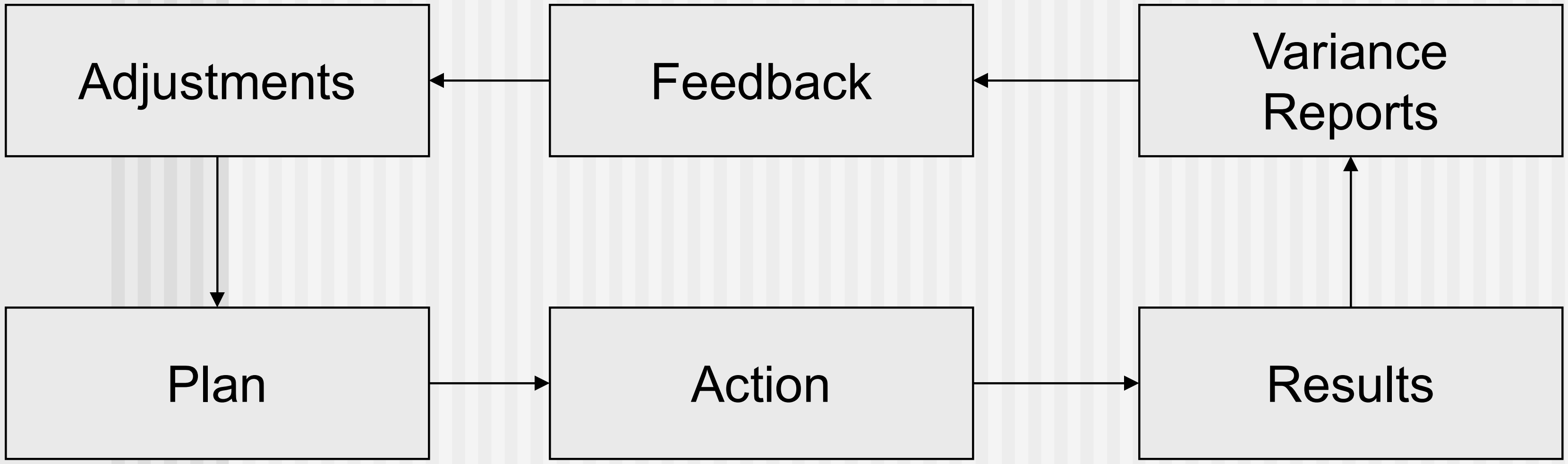
- Performance
- Quality
- Cost
- Time

It is a common practice to focus monitoring activities on data that are <u>easily gathered</u> rather than <u>importance</u>

Activities

- Performance criteria, standard & data collection procedure should be established
- Identify information that need to be collected
- Collect data
- Prepare report
- Update plans, charts & tables

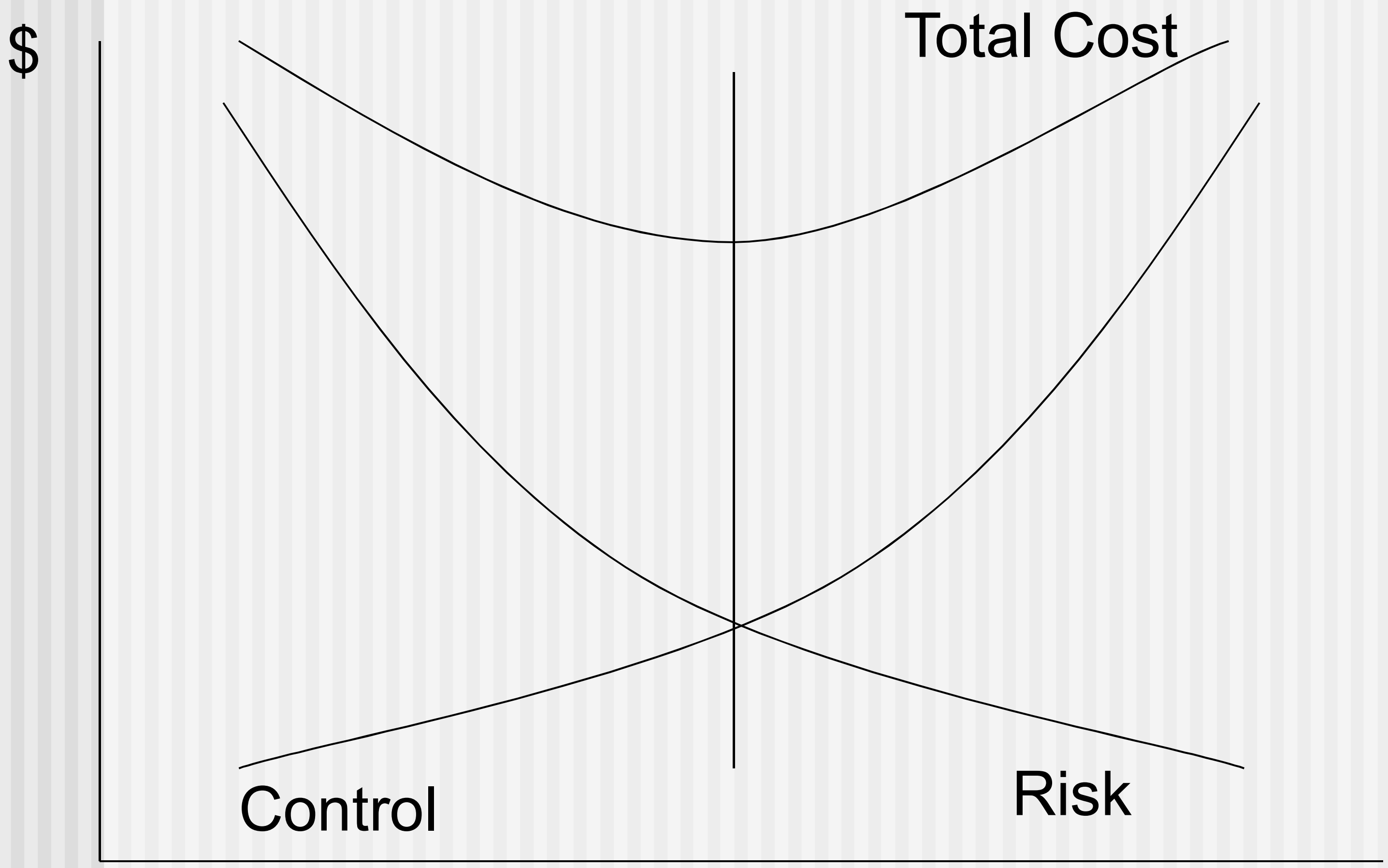
Feedback Loop



Features

- Monitoring can maintain high morale
- Variations should be highlighted
- Honesty & unbiased reporting is important

Control vs Risk



Control vs Risk Considerations

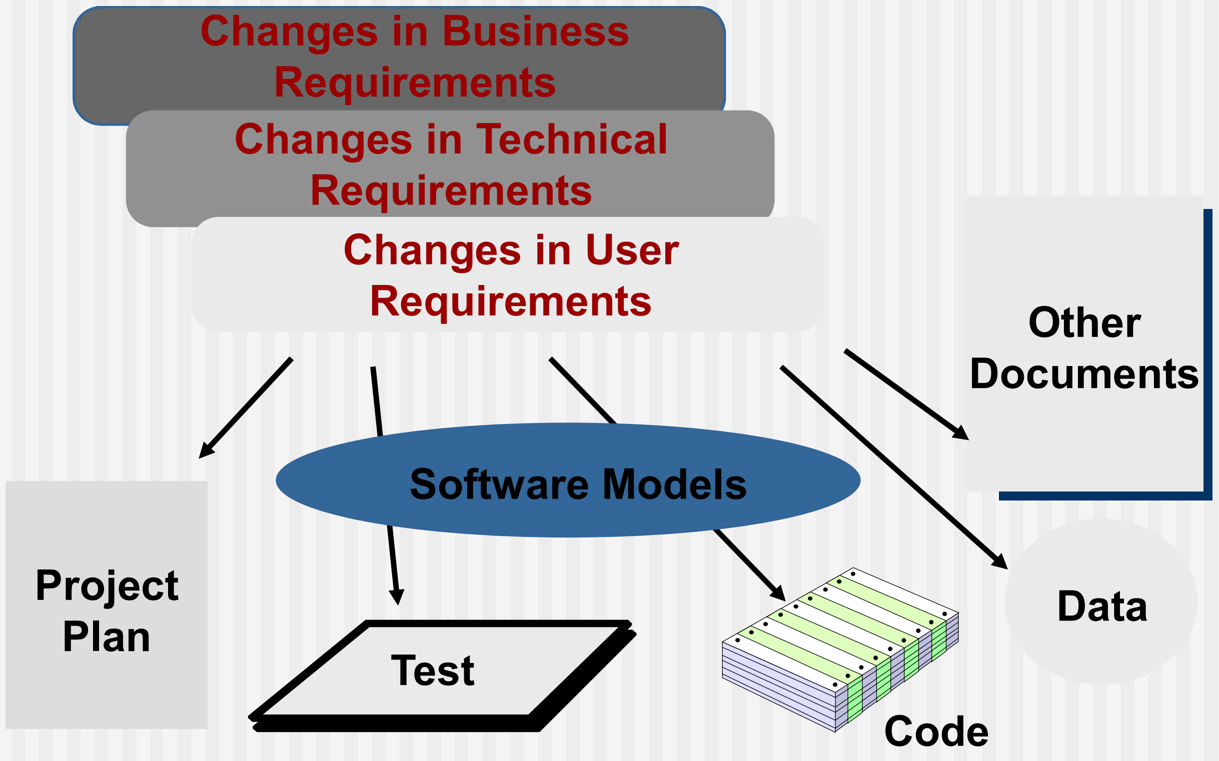
- High Control - Low Risk
  - Risk reduced when more controls
  - More controls will incur higher costs
- Low Control - High Risk
  - Low control will allow more problems to occur
- Achieving Balance
  - Project manager need to put adequate control and have a manageable level of risk

2. Software Configuration Management

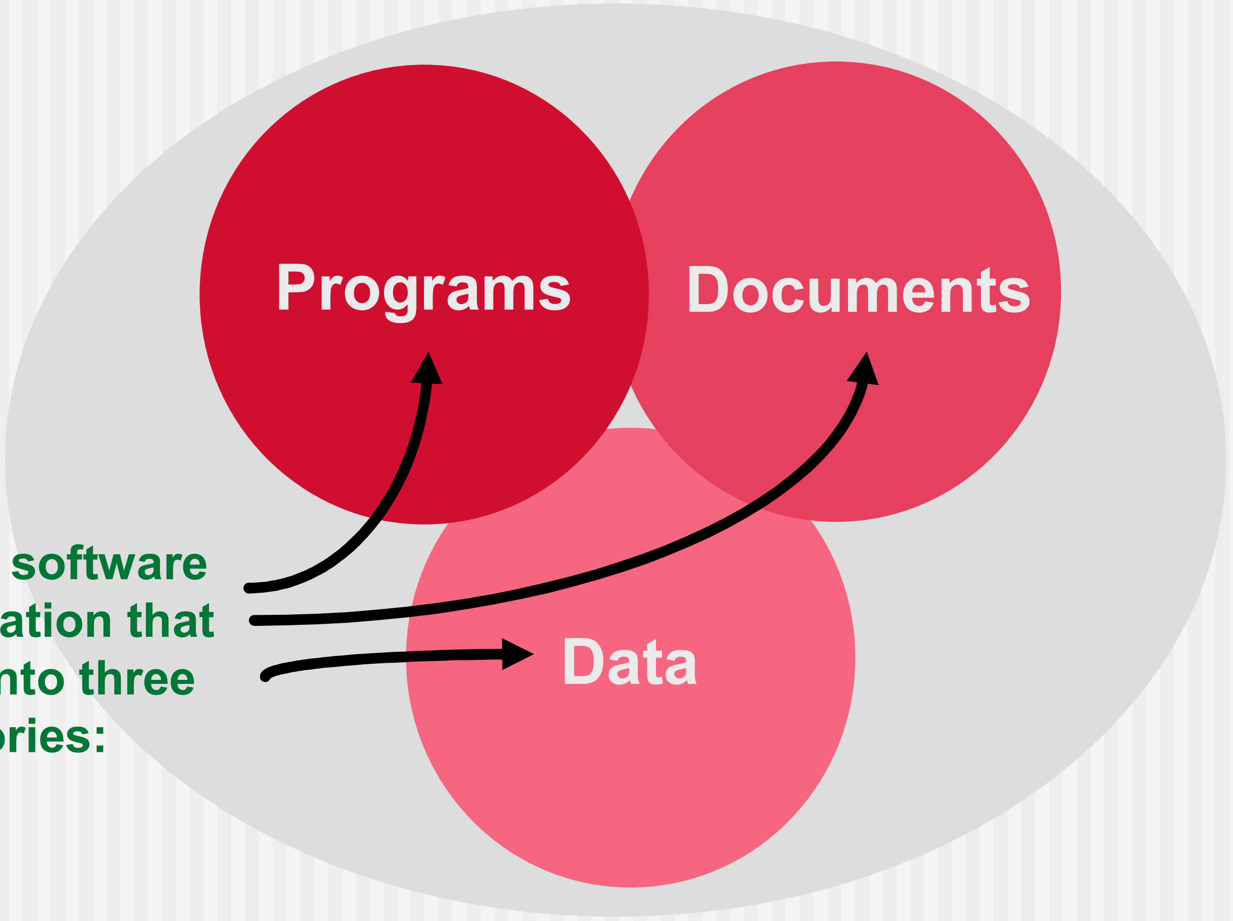
The “First Law”

No matter where you are in the system life cycle, the system will change, and the desire to change will persist throughout the life cycle.

What Are These Changes?


Changes in Business Requirements
Changes in Technical Requirements
Changes in User Requirements
Other Documents
Software Models
Project Plan
Test
Data
Code

The Software Configuration



The output of the software process is information that may be divided into three broad categories:

Software Configuration Management (SCM)

- SCM is the discipline for systematically controlling the changes that take place during development
- Umbrella activity that is applied throughout SE process
- SCM is easier at the start of development and gets more complex towards the end

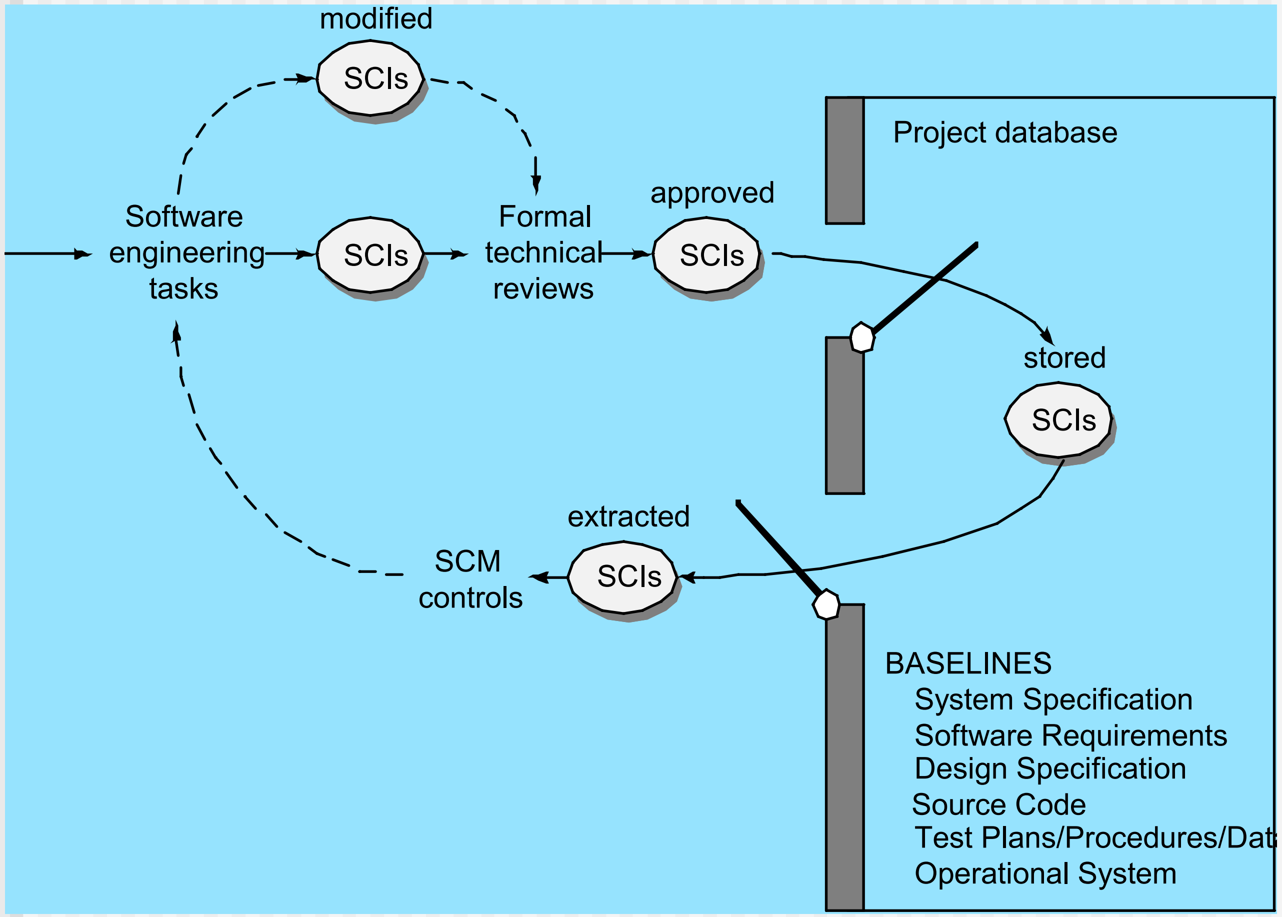
Main Tasks of SCM

- Identification
  - What are the configuration items?
- Control
  - How changes should be controlled?
- Status Accounting
  - What changes have been made?
- Auditing
  - Is the system being built to satisfy the needs?

Baselines

- The IEEE (IEEE Std. No. 610.12-1990) defines a baseline as a specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
- A baseline is **a milestone in the development of software** that is marked by the delivery of one or more software configuration items and the approval of these SCIs that is obtained through a formal technical review.

Baselines


modified
SCIs
Software engineering tasks
SCIs
Formal technical reviews
approved
SCIs
Project database
stored
SCIs
extracted
SCIs
SCM controls
BASELINES
System Specification
Software Requirements
Design Specification
Source Code
Test Plans/Procedures/Dat
Operational System

Examples of Baselines

- Functional baseline (requirements)
- Design baseline
- Product baseline (developed system)

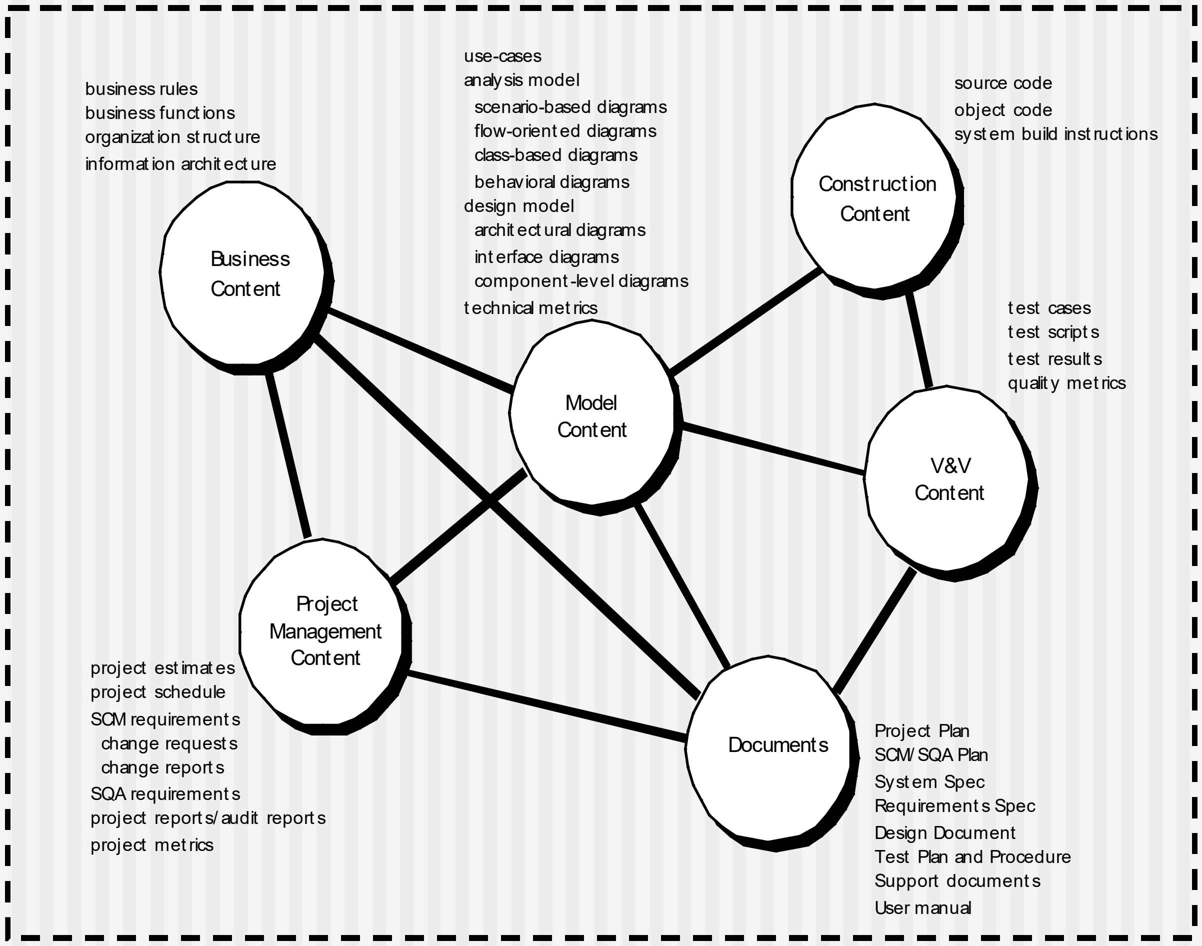
Elements of Configuration Management System

- **Component Elements**—A set of tools coupled within a file management system (e.g., a database) that enables access to and management of each software configuration item.
- **Process Elements**—A collection of procedures and tasks that define an effective approach to change management (and related activities) for all constituencies involved in the management, engineering and use of computer software.
- **Construction Elements**—A set of tools that automate the construction of software by ensuring that the proper set of validated components (i.e., the correct version) have been assembled.
- **Human Elements**—To implement effective SCM, the software team uses a set of tools and process features (encompassing other CM elements).

SCM Repository

- The SCM repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner.
- The repository performs or precipitates the following functions:
  - **Data Integrity**
  - **Information Sharing**
  - **Tool Integration**
  - **Data Integration**
  - **Methodology Enforcement**
  - **Document Standardization**

Repository Content



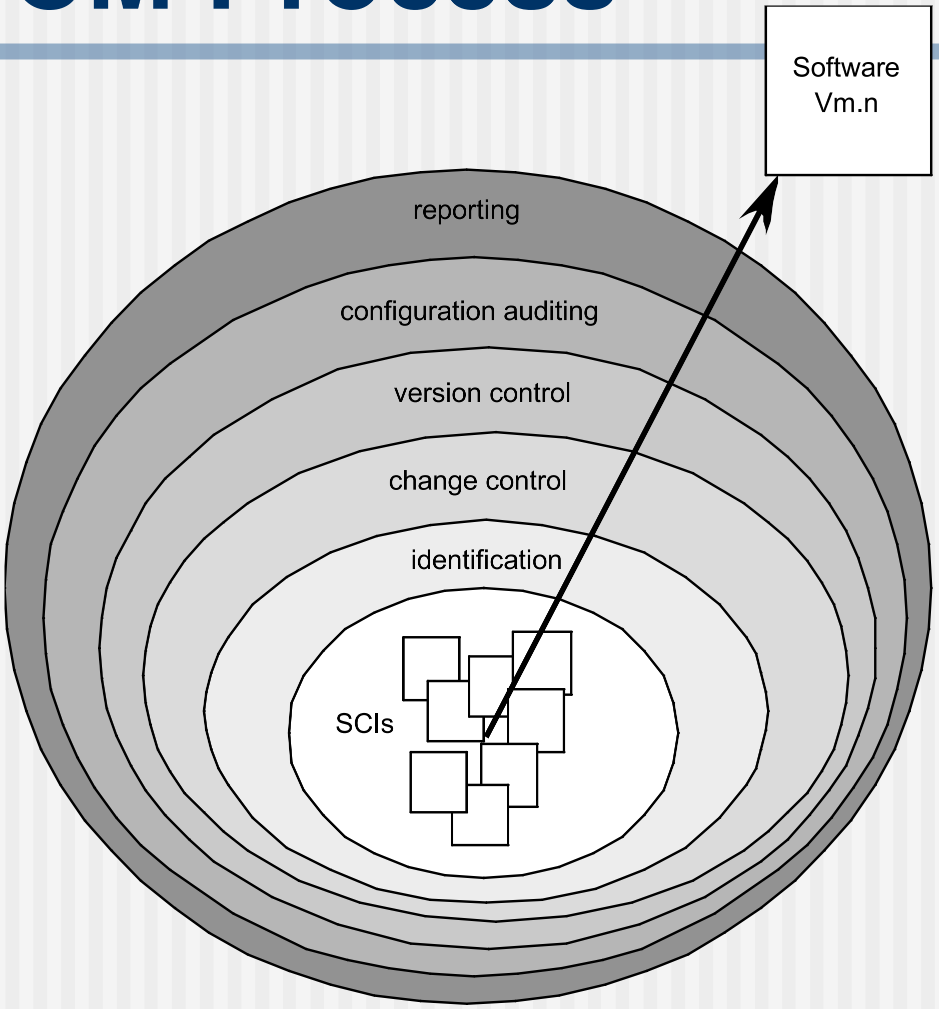
Repository Features

- **Versioning**
  - Saves all of these versions to enable effective management of product releases and to permit developers to go back to previous versions.
- **Dependency Tracking and Change Management.**
  - The repository manages a wide variety of relationships among the data elements stored in it.
- **Requirements Tracing**
  - Provides the ability to track all the design and construction components and deliverables that result from a specific requirement specification.

Repository Features

- **Configuration Management.**
  - Keeps track of a series of configurations representing specific project milestones or production releases. Version management provides the needed versions, and link management keeps track of interdependencies.
- **Audit Trails**
  - Establishes additional information about when, why, and by whom changes are made.

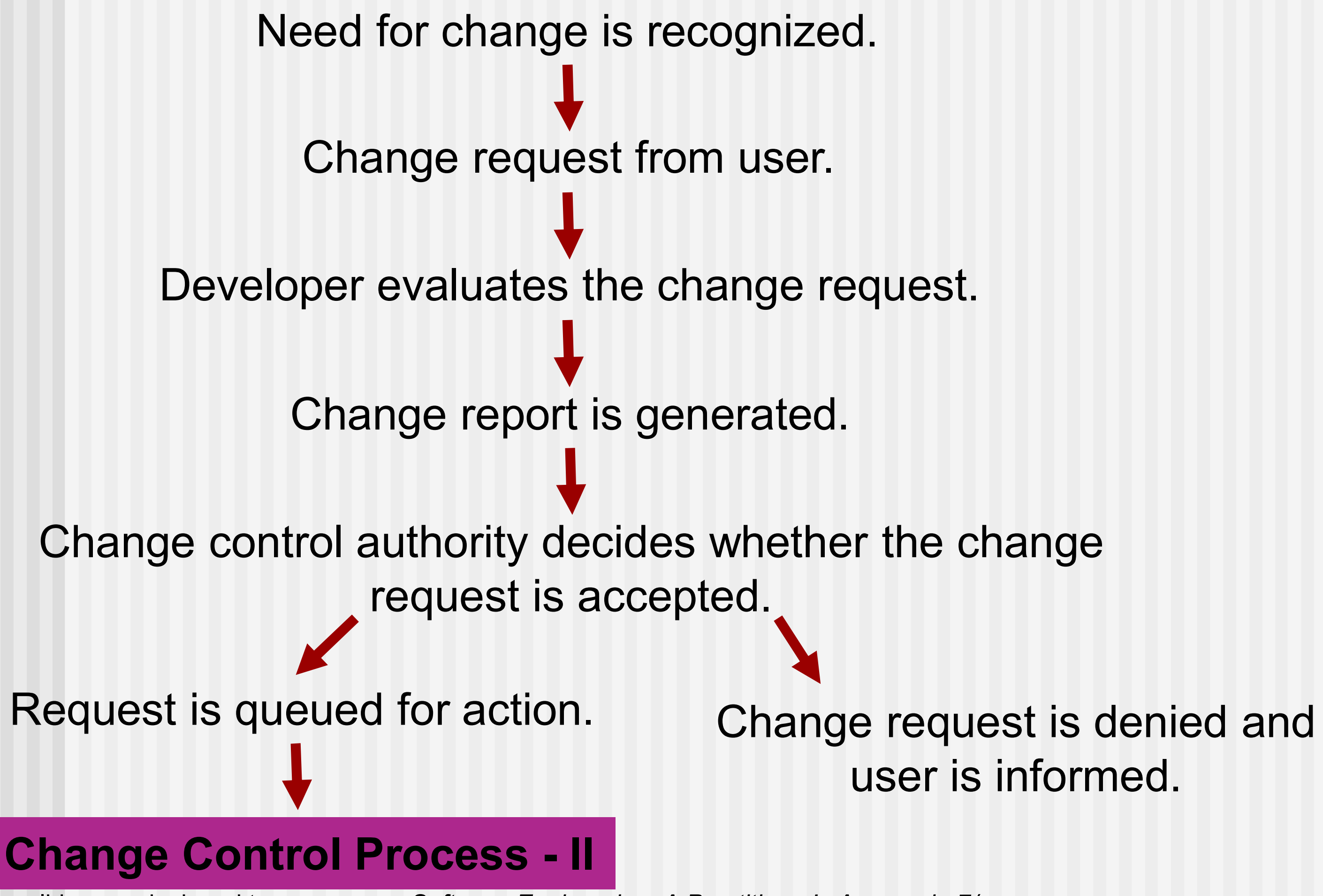
The SCM Process



Version Control

- Version control combines procedures and tools to manage different versions of configuration objects that are created during the software process.
- A version control system implements or is directly integrated with four major capabilities:
  - A **project database** (repository) that stores all relevant configuration objects
  - A **version management** capability that stores all versions of a configuration object (or enables any version to be constructed using differences from past versions);
  - A **make facility** that enables the software engineer to collect all relevant configuration objects and construct a specific version of the software.
  - An **issues tracking** (also called *bug tracking*) capability that enables the team to record and track the status of all outstanding issues associated with each configuration object.

Change Control Process - I


Need for change is recognized.
Change request from user.
Developer evaluates the change request.
Change report is generated.
Change control authority decides whether the change request is accepted.
Request is queued for action.
Change request is denied and user is informed.
Change Control Process - II

Change Control Process - II

↓

Assign people to SCIs.

↓

Check-out SCIs.

↓

Make the change.

↓

Review/audit the change.

↓

Establish a baseline for testing.

↓

Change Control Process - III

Change Control Process - III

↓

Perform SQA and testing activities.

↓

Check-in the changed SCIs.

↓

Promote SCI for inclusion in next release.

↓

Rebuild appropriate version.

↓

Review/audit the change.

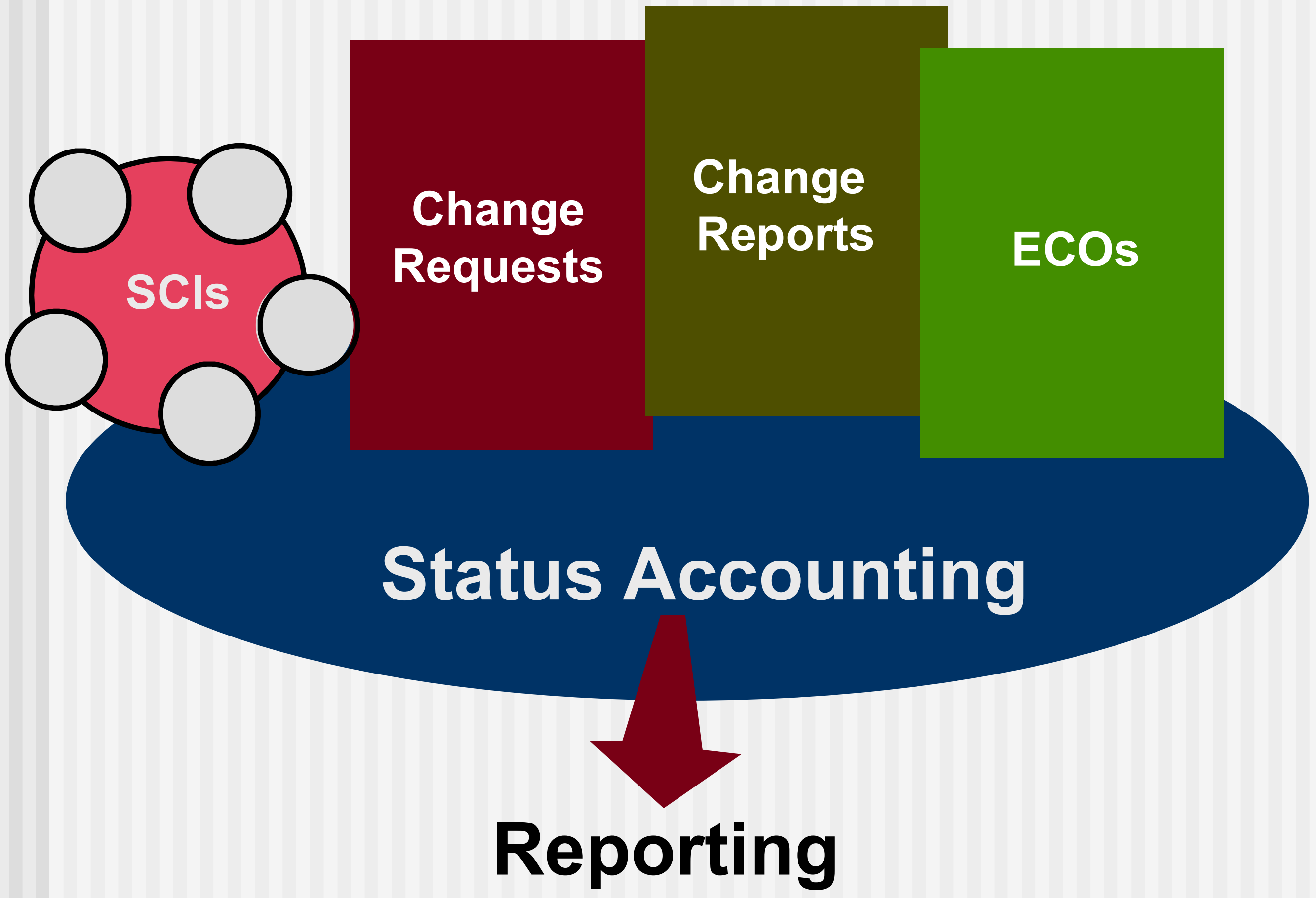
↓

Include all changes in release.

Status Accounting / Reporting

- Keeps record of how the system evolves and what is its current status (administrative nature)
- Can be complex due to the existence of executable and non-executable forms
- Main tasks:
  - Record baseline establishment time
  - Record when SCI came into being
  - Information about each SCI
  - Engineering change proposal status
  - Status of the approved changes

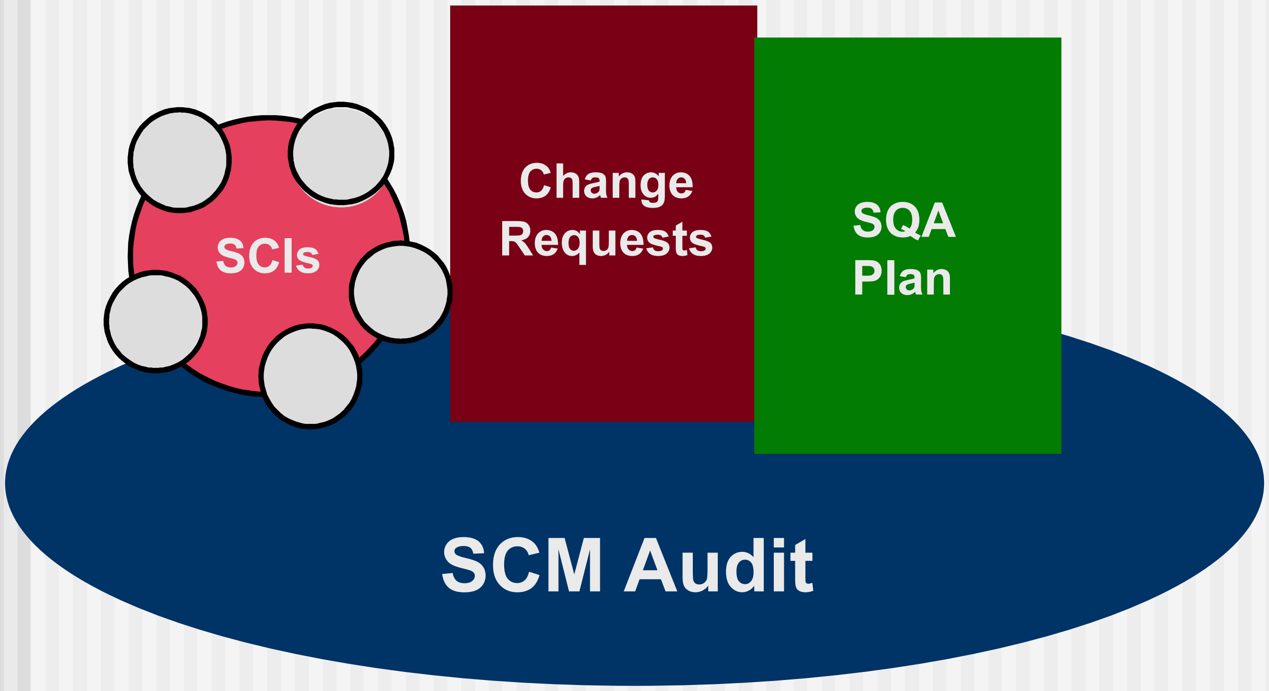
Reporting


SCIs
Change Requests
Change Reports
ECOs
Status Accounting
Reporting

Configuration Audit

- Concerned with determining how accurately the current software system implements the system defined in the baseline & requirements document
- Also concerned with increasing the visibility and traceability of the software
- How do we know a change is properly implemented?

Auditing



3. Software Maintenance

Software Maintenance

- Process of changing a system after it has been delivered and is in use
- Last phase in software engineering process
- *Evolution* of software - changing it to maintain its ability to survive
- Goal of software engineering - to ease the maintenance process & reduce costs

Maintenance Activities

- *Corrective Maintenance* - diagnosis and correction of reported errors
- *Adaptive Maintenance* - modifications to properly interface with changing environments $\rightarrow$ new hardware, OS, devices
- *Perfective Maintenance* - implementing new system requirements after system is successful

Maintainable Software

- Maintainable software exhibits effective modularity
- It makes use of design patterns that allow ease of understanding.
- It has been constructed using well-defined coding standards and conventions, leading to source code that is self-documenting and understandable.
- It has undergone a variety of quality assurance techniques that have uncovered potential maintenance problems before the software is released.
- It has been created by software engineers who recognize that they may not be around when changes must be made.
    - *Therefore, the design and implementation of the software must "assist" the person who is making the change*

Classic Maintenance Problems

- Difficult or impossible to trace evolution of software through many versions/releases
- Difficult or impossible to trace the process through which software was created
- Difficult to understand “someone else’s” program
- “Someone else” is often not around to explain
- Documentation doesn’t exist or is awful
- Most software is not designed for change

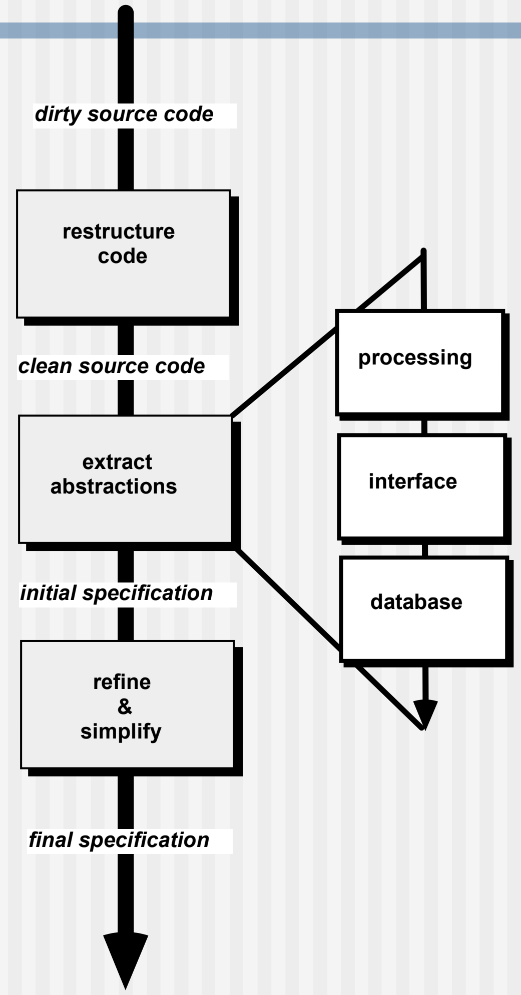
Preventive Maintenance

- Software is changed to improve future maintainability or reliability
- More common term in maintenance of hardware and other physical systems
- Characterised by
  - reverse engineering
  - re-engineering

Software Reverse Engineering

- Originally a process in hardware - disassemble a hardware product to understand the design and specifications
- In software, it’s the process of analysing a program to create a representation at a higher level of abstraction
- Usually from the source code or executable code, create the design and specifications of a software

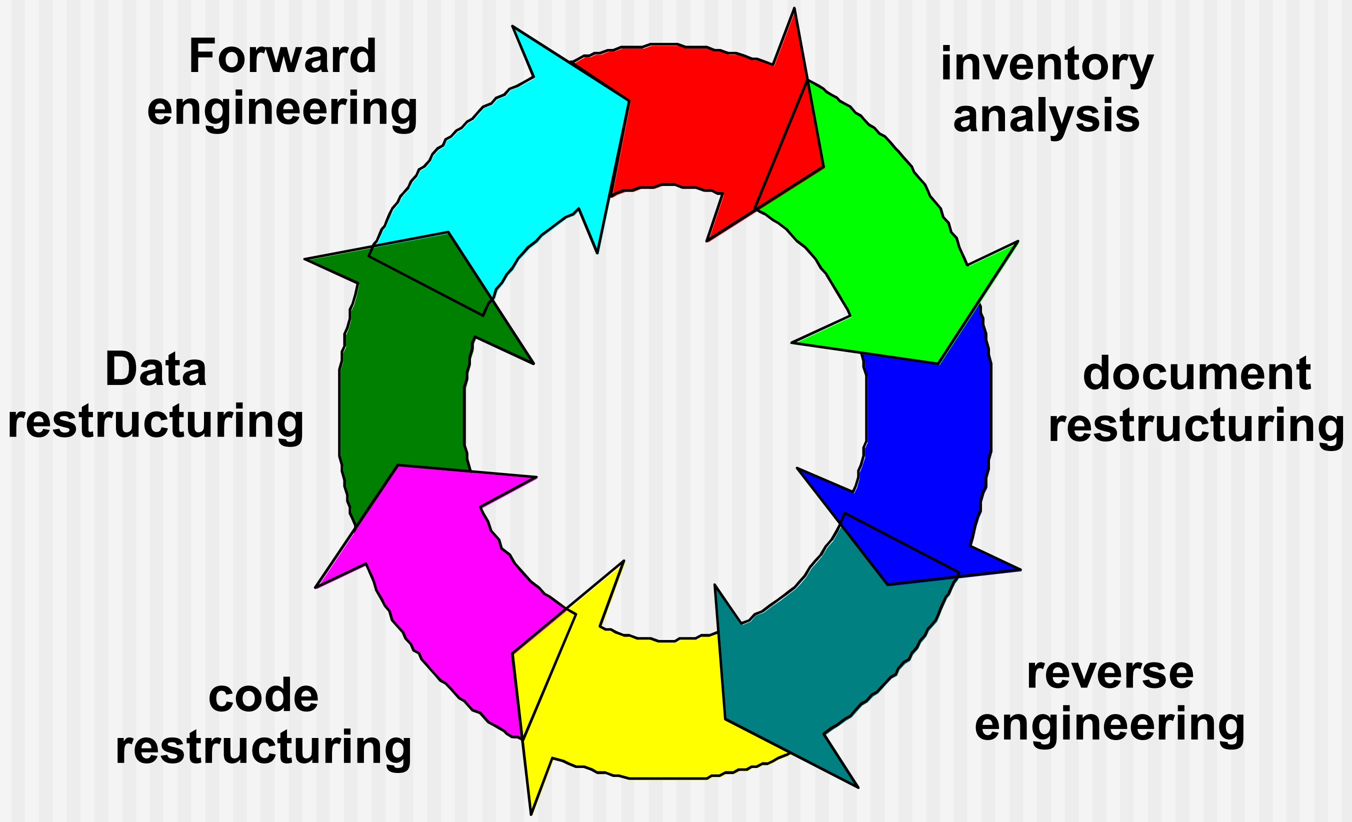
Reverse Engineering



Software Re-engineering

- Takes information of an existing system and restructures the system to achieve higher quality
- Usually takes the result of reverse engineering as starting point
- Makes the system more maintainable - improve the structure, create new documentation, and easier to understand

Software Reengineering


inventory analysis
document restructuring
reverse engineering
code restructuring
Data restructuring
Forward engineering