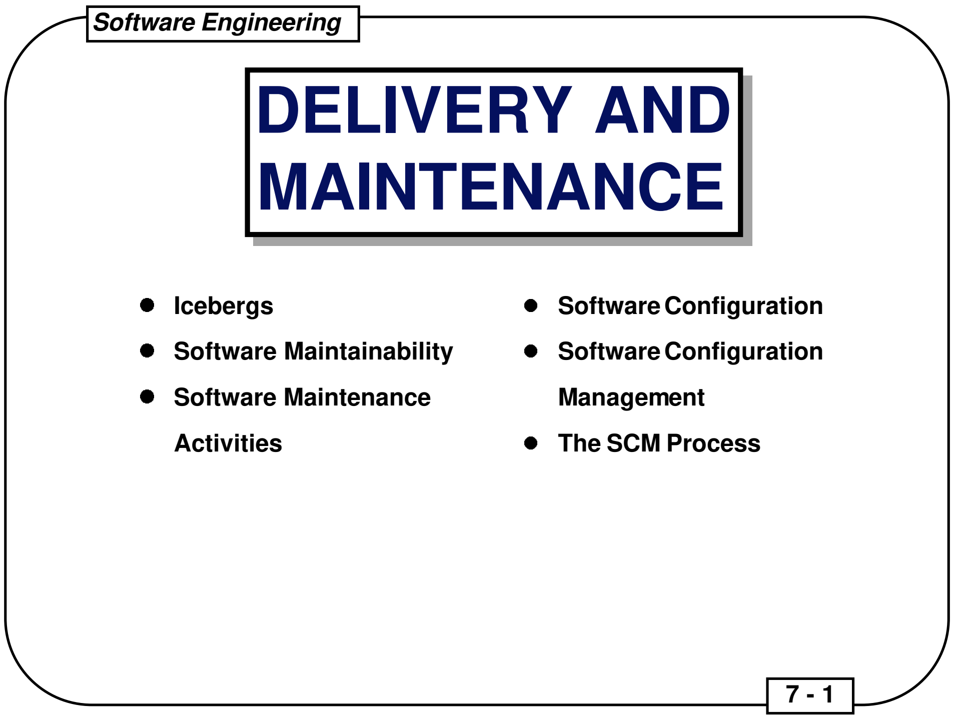


Software Engineering

DELIVERY AND MAINTENANCE

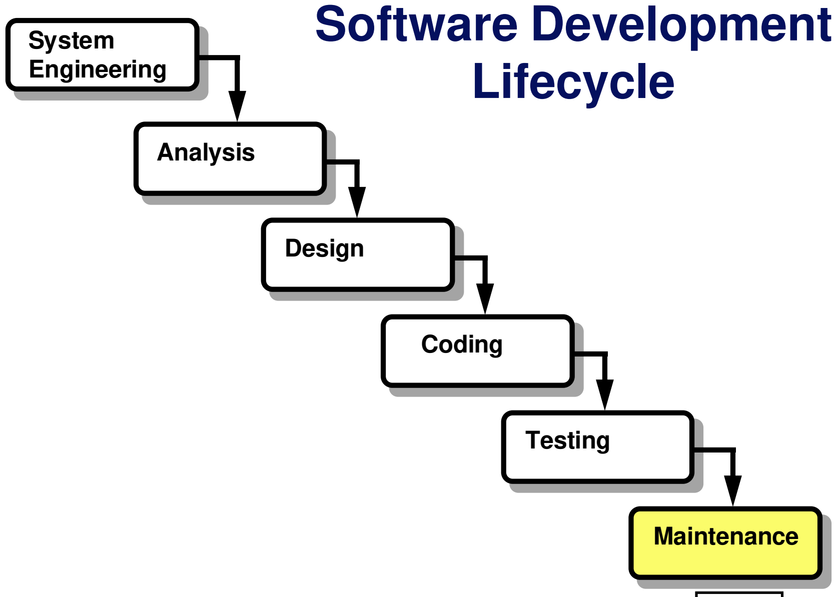
- Icebergs
- Software Maintainability
- Software Maintenance Activities
- Software Configuration
- Software Configuration Management
- The SCM Process

7 - 1

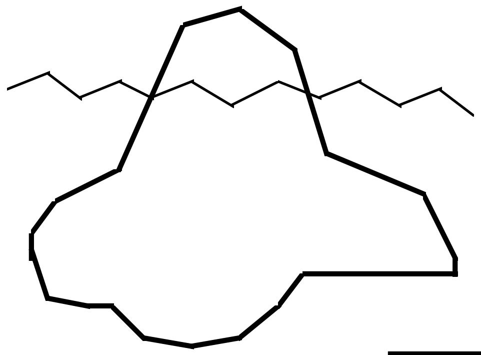
Objectives of Module 7

- Present and discuss the software maintenance process and its associated activities
- Present and discuss the concept of Software Configuration Management (SCM) in some detail

Software Development Lifecycle

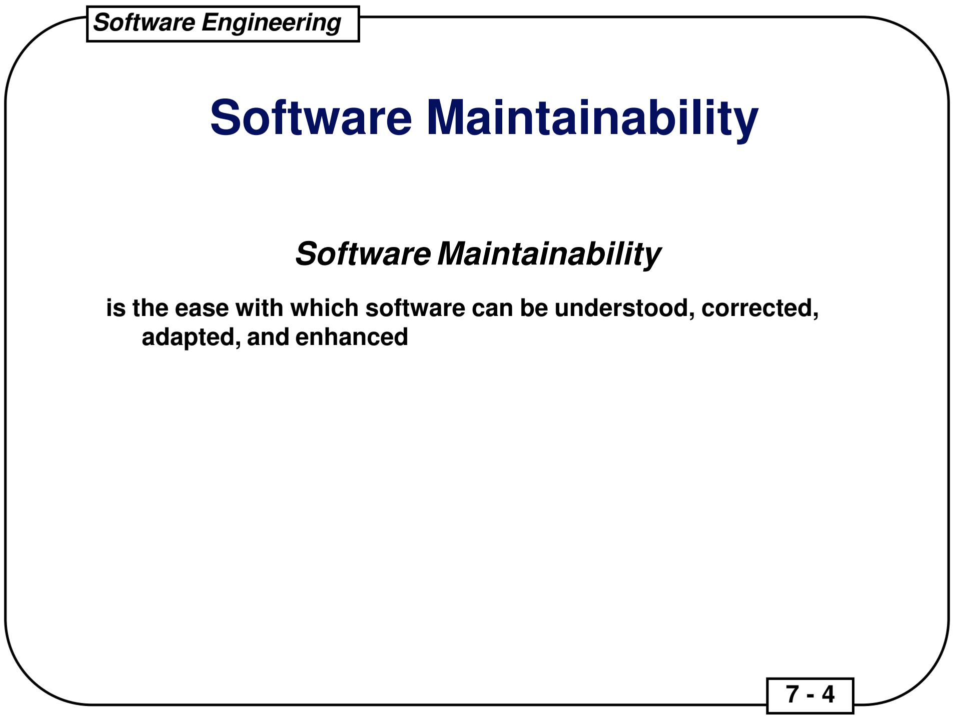


Icebergs



Software maintenance has been characterized as an "Iceberg," hoping that what is immediately visible is all there is to it.

- Maintenance of existing software can account for over 70% of all effort expended by a software organization.
- The possibility exists that in the near future "maintenance-bound" software organizations can no longer produce new software since all effort is expended in maintaining old software.
- Much software in use today is over 10 years old and produced at a time when software engineering techniques were not applied. The result is poor software structure, coding, logic, and documentation.

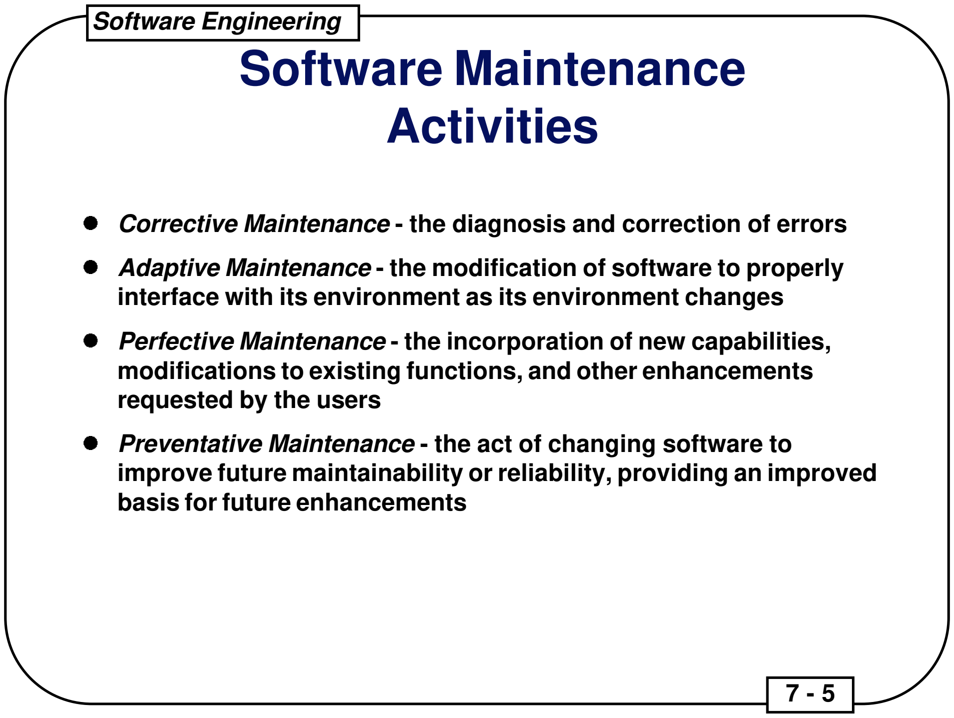


Software Maintainability

Software Maintainability

is the ease with which software can be understood, corrected, adapted, and enhanced

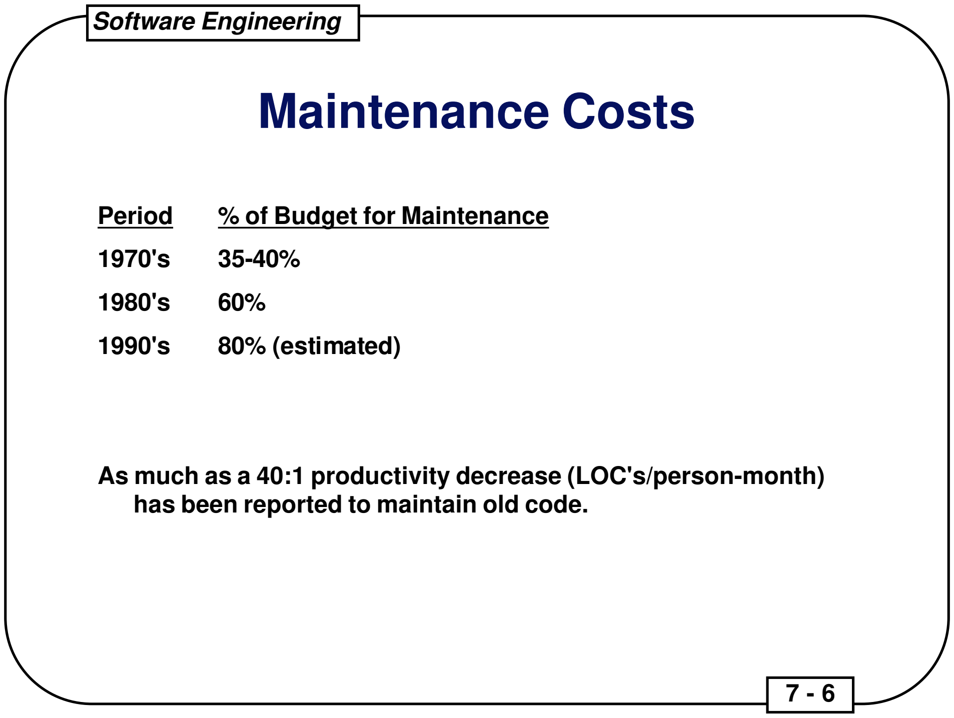
A primary goal of software engineering is to improve the ease which changes to software products can be made and to reduce the amount of effort expended in its maintenance.



Software Maintenance Activities

- ***Corrective Maintenance* - the diagnosis and correction of errors**
- ***Adaptive Maintenance* - the modification of software to properly interface with its environment as its environment changes**
- ***Perfective Maintenance* - the incorporation of new capabilities, modifications to existing functions, and other enhancements requested by the users**
- ***Preventative Maintenance* - the act of changing software to improve future maintainability or reliability, providing an improved basis for future enhancements**

- Adaptive and perfective maintenance involve the same tasks as new software development. Thus, the same software engineering processes are used in the same way.
- Unlike hardware, software breaks only when it is changed.



Software Engineering

Maintenance Costs

Period	% of Budget for Maintenance
1970's	35-40%
1980's	60%
1990's	80% (estimated)

As much as a 40:1 productivity decrease (LOC's/person-month) has been reported to maintain old code.

Besides the direct cost to maintain code, the indirect costs include:

- Customer dissatisfaction to the delayed resolution of a problem.
- Reduction of software quality due to bugs introduced during the maintenance activity.
- Loss of new software development because of the transfer of software engineers to the maintenance function.

Phases of Maintenance

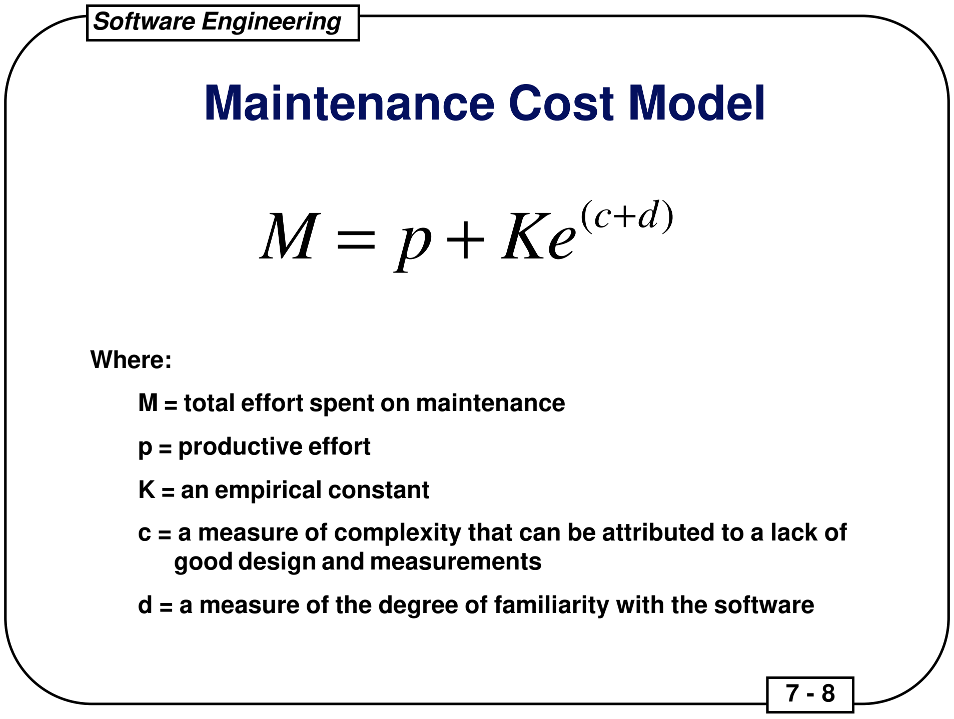
There are two phases during maintenance efforts:

1. "Wheel spinning"

   Understanding function and structure of code, data structures, interfaces, and performance requirements

2. Productive periods

   Analysis and evaluation, design modifications, coding, and documentation updating



Maintenance Cost Model

$$M = p + Ke^{(c+d)}$$

Where:

M = total effort spent on maintenance

p = productive effort

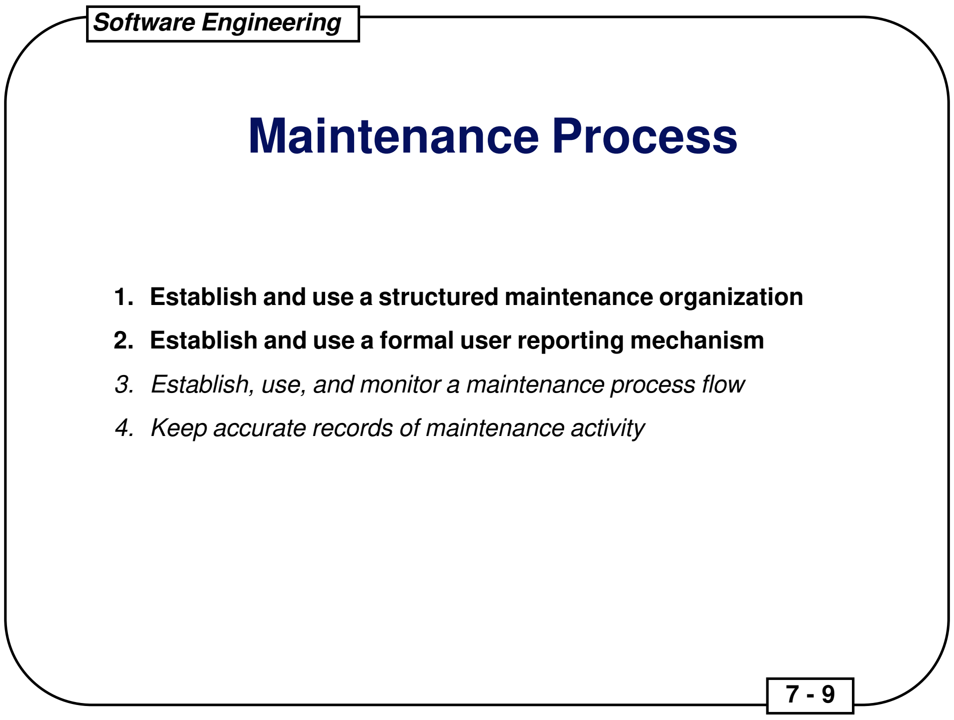
K = an empirical constant

c = a measure of complexity that can be attributed to a lack of good design and measurements

d = a measure of the degree of familiarity with the software

This model reflects the fact that maintenance costs increase exponentially if:

- Poor software engineering practices were used in the initial development
- The people who initially developed the software are no longer available


Software Engineering

Maintenance Process

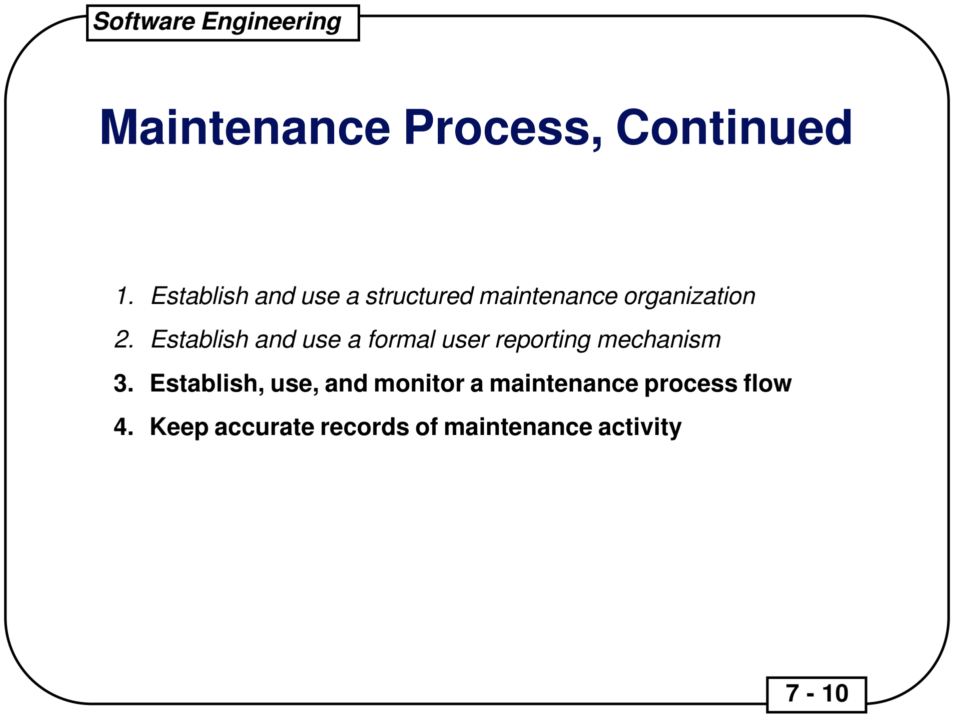
1. **Establish and use a structured maintenance organization**
2. **Establish and use a formal user reporting mechanism**
3. *Establish, use, and monitor a maintenance process flow*
4. *Keep accurate records of maintenance activity*

1. Maintenance organization:

- Can be informal except for very large organizations
- Use a maintenance controller to track maintenance needs
- Use a system supervisor to supervise maintenance tasks
- Use a change control board to decide on courses of action

2. User reporting:

- Use a Software Problem Report (SPR) that a user can fill out and send to software maintenance organization
- From the SPR, the internal software organization develops an Engineering Change Proposal (ECP) to identify:
  - Magnitude of effort to satisfy the SPR
  - Kind of software modifications needed
  - Priority of the request
  - Other data of significance about the request
- The ECP is submitted to the Change Control Board (CCB) for evaluation. Maintenance efforts begin after the go-ahead by the CCB.



Maintenance Process, Continued

1. *Establish and use a structured maintenance organization*
2. *Establish and use a formal user reporting mechanism*
3. **Establish, use, and monitor a maintenance process flow**
4. **Keep accurate records of maintenance activity**

3. Maintenance process flow:

- Determine if a maintenance request is a software error or a feature deficiency.
  - If an error, handle it as a crisis if warranted; otherwise, place the request on the work queue.
  - If a deficiency, consider whether adaptation or enhancement to the software is needed. Then place the request on the work queue.
- Remove the next request from the work queue and process it as per the normal software engineering process.

4. Record keeping and evaluation:

- Keep the necessary data on hand to support prediction of the software maintenance effort.
- Use measures of maintenance performance to assist in predicting future approaches.

Maintenance Effort Data and Prediction

At a minimum, collect the following data during maintenance:

- **Program ID**
- **Number of LOCs**
- **# of machine code instructions**
- **Programming language used**
- **# of program runs since installed**
- **# of failures over program runs**
- **Program change level and ID**
- **Program installation date**
- **Number of LOCs added**
- **Number of LOCs deleted**
- **Number of person-hours/change**
- **Program change date**
- **Software engineer ID**
- **SPR identification**
- **Maintenance type**
- **Maintenance start/end dates**
- **Cummulative maintenance pers-hrs**
- **Benefits of maintenance**

Keep a database of this information to be used to form quantitative assessments of the maintenance efforts.

Annual Maintenance Effort

COCOMO Model:

$$\text{Maint\_Effort} = 2.4 \frac{KLOCi}{CI} KLOCf^{\ 1.05}$$

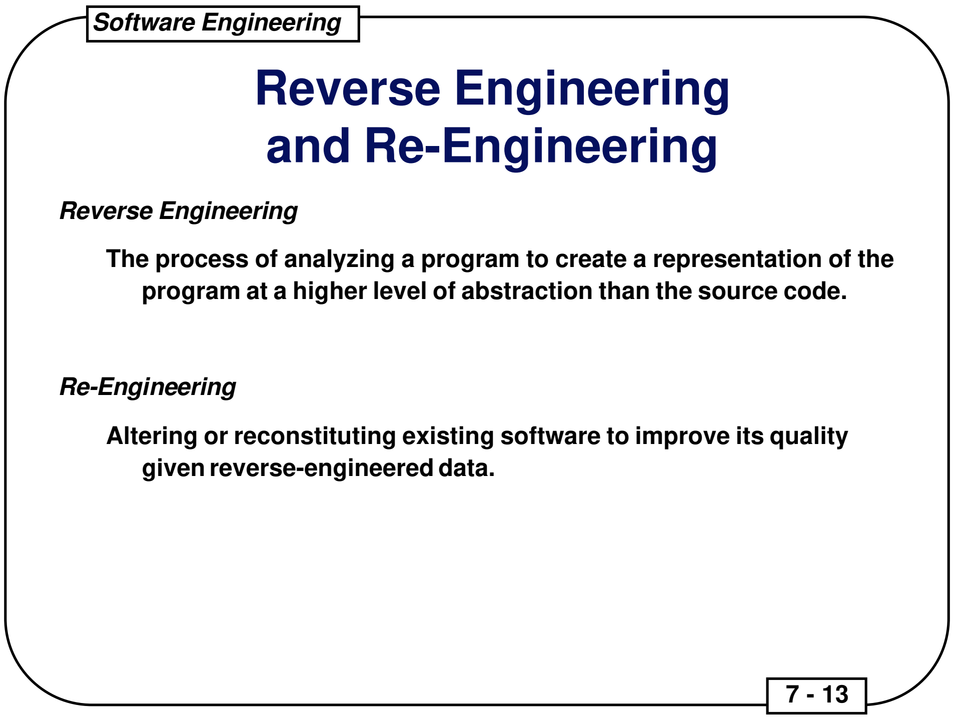
person-hrs/year

Where:

KLOCi = 1000*LOC for original system to be modified

CI = LOC added or modified

KLOC = 1000*LOC in final updated system



Reverse Engineering and Re-Engineering

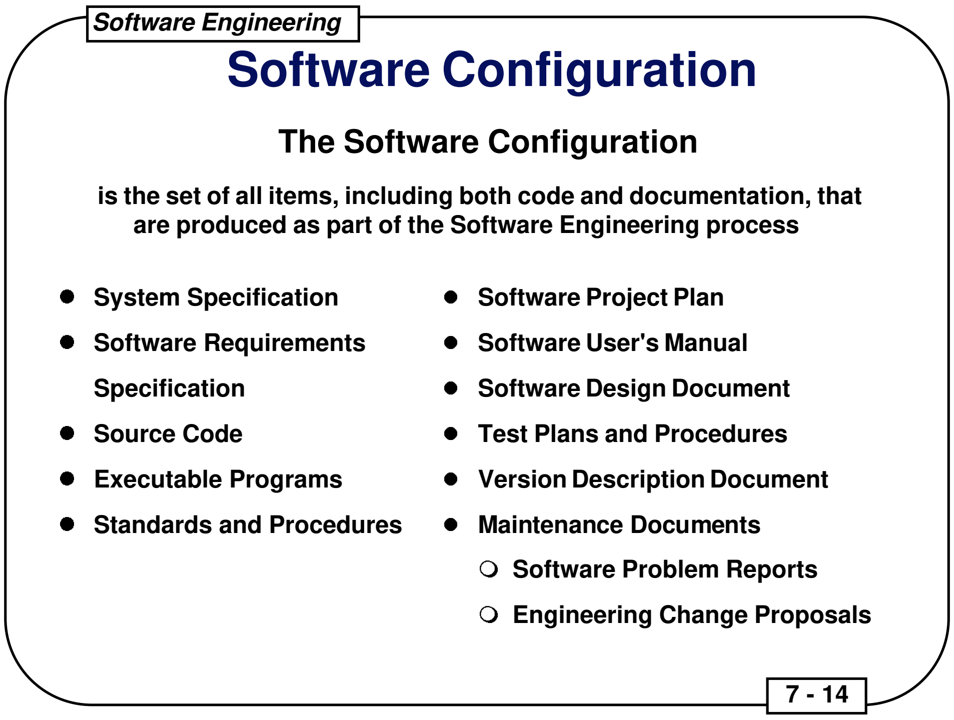
Reverse Engineering

The process of analyzing a program to create a representation of the program at a higher level of abstraction than the source code.

Re-Engineering

Altering or reconstituting existing software to improve its quality given reverse-engineered data.

- Reverse engineering is normally done by a company to maintain its own old code.
- Re-engineering is sometimes called *software renovation* or *reclamation*.
- Few tools are available to support reverse engineering, and these tools are relatively primitive.



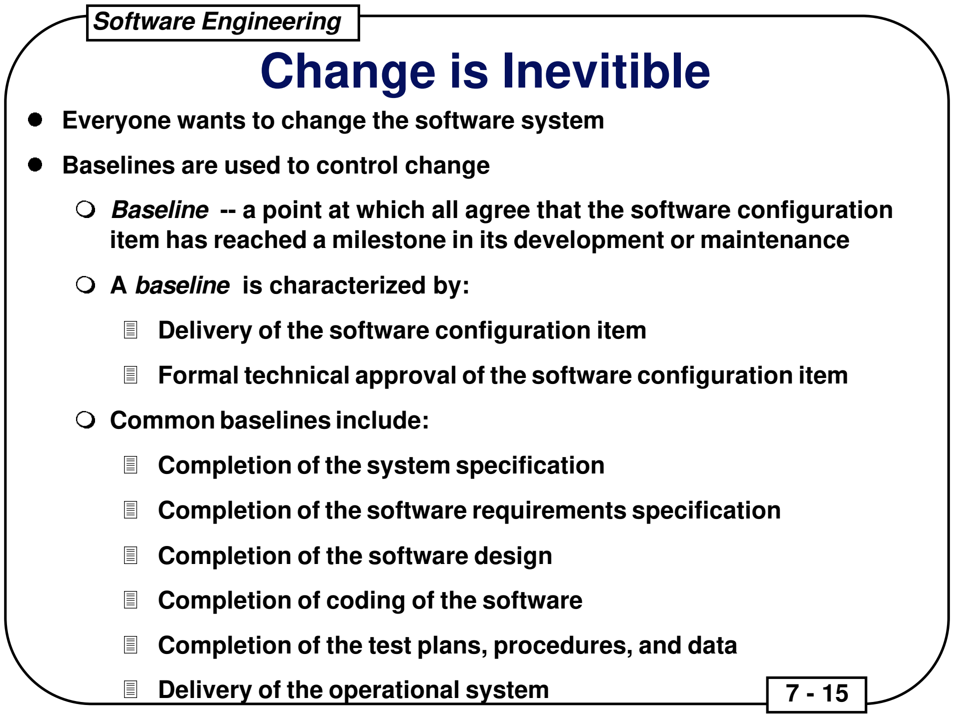
Software Configuration

The Software Configuration

is the set of all items, including both code and documentation, that are produced as part of the Software Engineering process

- **System Specification**
- **Software Requirements Specification**
- **Source Code**
- **Executable Programs**
- **Standards and Procedures**
- **Software Project Plan**
- **Software User's Manual**
- **Software Design Document**
- **Test Plans and Procedures**
- **Version Description Document**
- **Maintenance Documents**
  - **Software Problem Reports**
  - **Engineering Change Proposals**

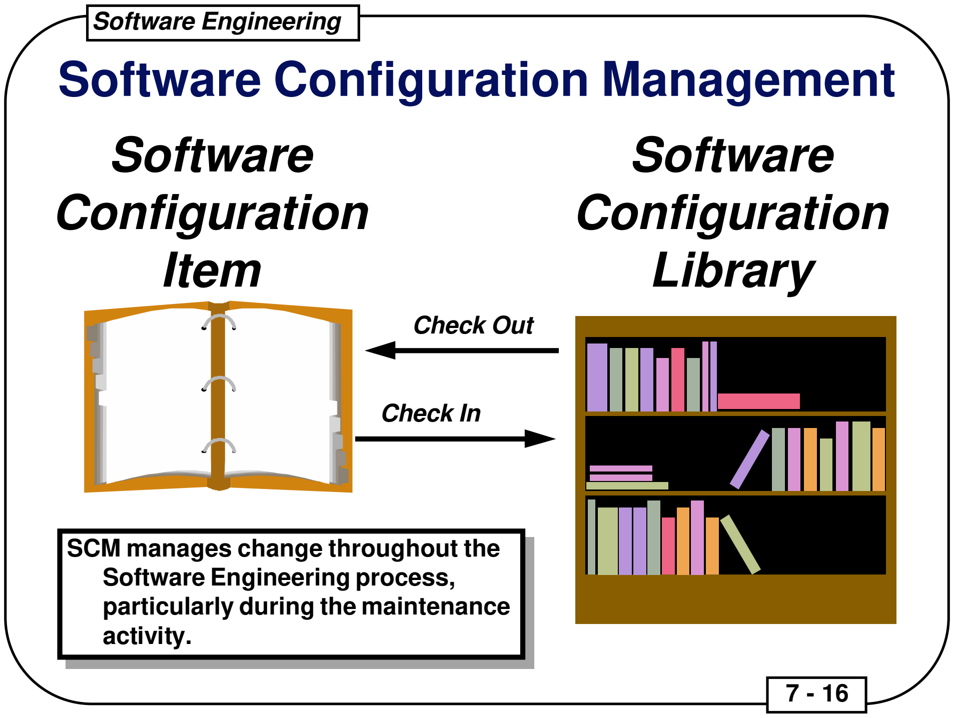
Configuration items for large systems are usually sections of an entire document, program source code, or data items (like test files), rather than the entire document, all the code, or all the data. For smaller systems, configuration items are usually entire documents, bodies of code, and data.



Change is Inevitible

- Everyone wants to change the software system
- Baselines are used to control change
  - *Baseline* -- a point at which all agree that the software configuration item has reached a milestone in its development or maintenance
  - A *baseline* is characterized by:
    - Delivery of the software configuration item
    - Formal technical approval of the software configuration item
  - Common baselines include:
    - Completion of the system specification
    - Completion of the software requirements specification
    - Completion of the software design
    - Completion of coding of the software
    - Completion of the test plans, procedures, and data
    - Delivery of the operational system

- Change occurs because we learn more about a system as it is being developed and used.
- A baseline is established before work progresses toward the next baseline.



Software Configuration Management

Software Configuration Item

Software Configuration Library

Check Out

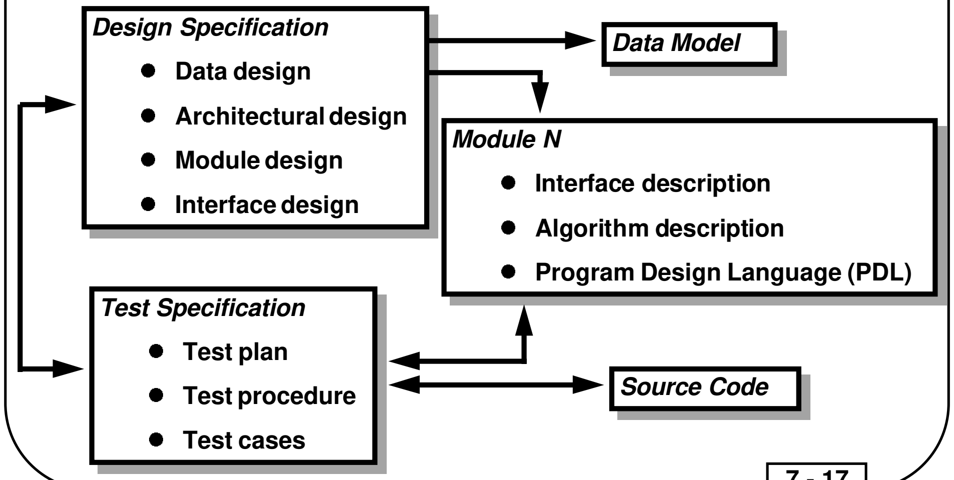
Check In

SCM manages change throughout the Software Engineering process, particularly during the maintenance activity.

- A software librarian controls the check-in and check-out permissions and the process. Automated tools, like SCCS or RCS, are employed once the permissions are established.
- An accounting file reflects the time, the Software Configuration Items, the designer, and the data for each check-in and check-out transaction.
- If no updates are permitted to the library while the Software Configuration Items are checkout-out for updating, the library is locked.

Configuration Objects

Software Configuration Items **(SCI's) can be grouped together in the library to form *configuration objects* referred to by a single name.**



- The Software Configuration Items are organized as entity-relationships in the library.
- For example, if "source code" is modified, so must "Module N" and "test specification" since the directed arcs in the figure form a spanning tree rooted at "source code."

The Software Configuration Management Process

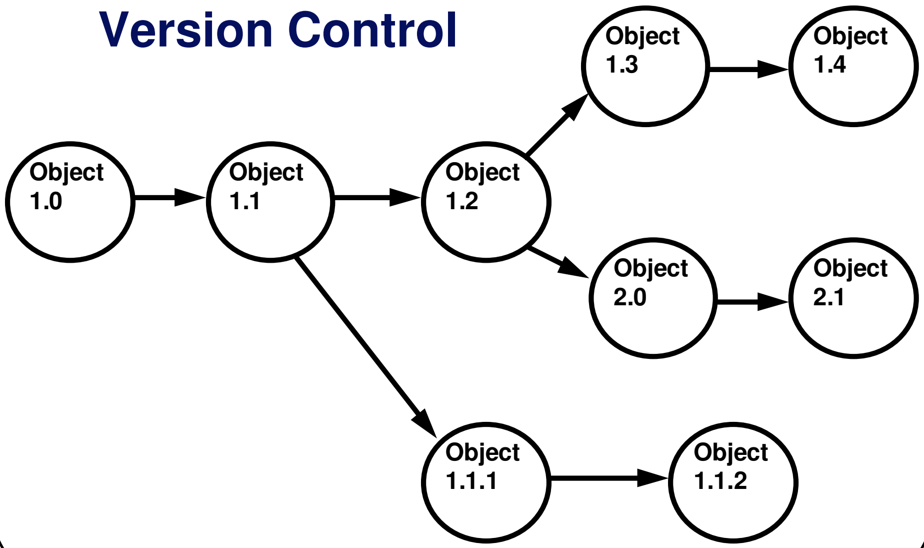
Activities --

- **Configuration Item Identification**
- **Version Control**
- **Change Control**
- **Configuration Audits**
- **Reporting**

Configuration Item Identification

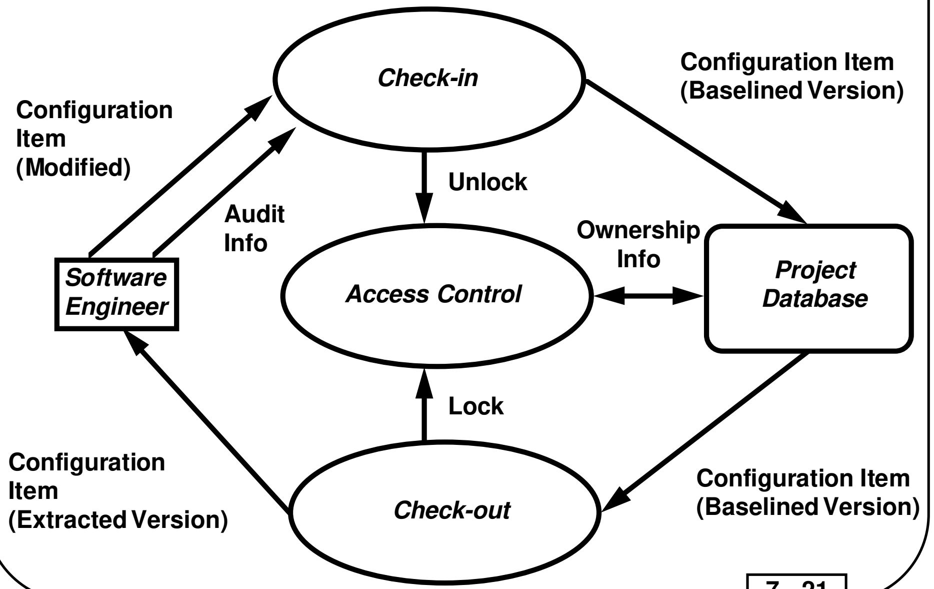
- **Each item in the library is either a:**
  - **basic object -- no references to other objects in the library**
  - **composite object -- one or more references to other objects**
- **Each Software Configuration Item has a name, type, project ID, and version identification**
- **Organization of SCI's in the library can be described by a "data model" similar to a database (an SCI library is a database) schema**
- **Objects can be collected to form working sets or delivery configurations, and these are related by the object identification fields**
- **Automated tools exist to support these tasks:**
  - **SCCS - Software Configuration Control System, 1975**
  - **RCS - Revision Control System, 1982**

Version Control



- *Evolution Graph* -- describes the change history of an object
- *Configuration Management (CM)* allows the user to specify alternative configurations of the software system.

Change Control


Check-in
Configuration Item (Baselined Version)
Configuration Item (Modified)
Unlock
Audit Info
Ownership Info
Software Engineer
Access Control
Project Database
Lock
Configuration Item (Extracted Version)
Check-out
Configuration Item (Baselined Version)

Change control involves --

- *Access control* -- governs which software engineers have access to modify a particular configuration item
- *Synchronization control* -- ensures parallel changes by two different people do not overwrite each other

Level of Change Control

- ***Informal change control*** **-- before the Software Configuration Item is baselined**
- ***Project-level change control*** **-- after the baseline but before delivery**
- ***Formal change control*** **-- after the Software Configuration Item is delivered**

Configuration Audit and Status Reporting

- To ensure change has been properly implemented requires:
  - formal technical reviews
  - software configuration audit
- Software configuration audits are usually conducted by a separate quality assurance group
- *Configuration Status Reporting* (CSR) -- Software Configuration Item data is dept online for review; this data includes:
  - What happened?
  - Who did it?
  - When did it happen?
  - What else will be affected?

Software Configuration Management Standards

- Military
  - DOD-STD-480A
  - DOD-STD-2167A
  - MIL-STD-1521A
- Non-Military
  - ANSI/IEEE 828-1983
  - ANSI/IEEE 1042-1987
  - ANSI/IEEE 1028-1988