Object-Oriented Software Engineering: An Agile Unified Methodology by David Kung

Chapter 21: Software Maintenance

Key Takeaway Points

- Software maintenance is modifying a software system or component after delivery to correct faults, improve performance, add new capabilities, or adapt to a changed environment. (IEEE Standard 610.12-1991)
- Software maintenance consumes 60%–80% of the total life-cycle costs; 75% or more of the costs are due to enhancements.

What Is Software Maintenance?

- Software maintenance is modifying a software system or component after delivery to correct faults, improve performance, add new capabilities, or adapt to a changed environment. (IEEE Standard 610.12-1991)

Factors Demanding Changes

- Bug fixes.
- Change in system’s operating environment.
- Change in government policies and regulations.
- Change in business procedures.
- Change to prevent future problems.

Lehman's Laws of System Evolution

1. Law of continuing change (1974).
2. Law of increasing entropy or complexity (1974).
3. Law of self-regulation (1974).
4. Law of conservation of organizational stability (1978).
5. Law of conservation of familiarity (1978)
6. Law of continuing growth (1991).
7. Law of declining quality (1996).
8. Law of feedback systems (1996).

Types of Software Maintenance

- Corrective maintenance: planned, reactive modification of the software product to correct errors.
- Adaptive maintenance: modification of the software product to enable it to operate in a changed operating environment.
- Perfective maintenance: modification of the software product to improve its quality or performance.
- Emergency maintenance: unplanned corrective maintenance to keep the system operational.

Software Maintenance Activities

- Program understanding.
- Change identification and analysis.
- Configuration change control.
- Change implementation, verification, and incorporation.

Program Understanding

- It is process that extract design and specification information from the code.
- It presents the extracted design and specification information in some mental models:
  - UML diagrams
  - control flow diagrams
  - data flow diagrams
- It is also called program comprehension.

Change Identification and Analysis

- Change to a class or component may impact a few, or many other classes or components.
- These other classes or components may need to change.
- Alternative changes may be identified and analyzed to assess their costs of implementation.
- One of the alternatives is selected.

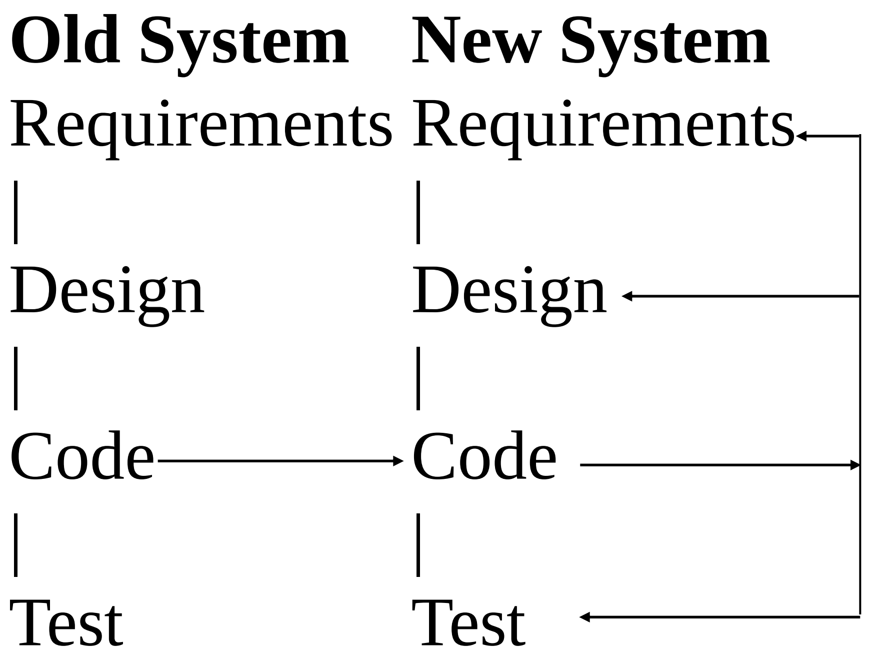
Configuration Change Control

- Change to a class or component may affect many other classes or components.
- Classes and components of a system are designed and implemented by different teams and team members.
- Changes must be coordinated, else inconsistent system configuration may occur.
- Configuration change control is a mechanism to coordinate change in a teamwork environment.

Software Maintenance Process Models

- Quick fix model
- Iterative enhancement model
- Full reuse model
- IEEE-1219 model
- ISO-12207 model

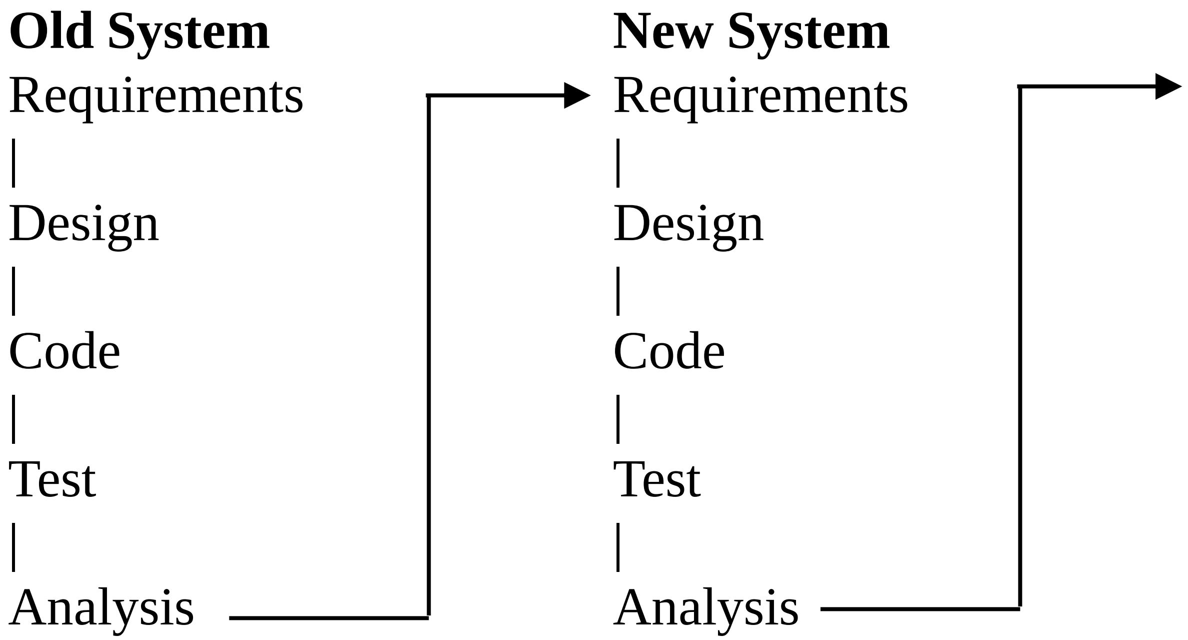
Quick Fix Model



(a) Quick fix model

Changes are made to the code, design and requirements are updated accordingly. The modified code is tested.

Iterative Enhancement Model



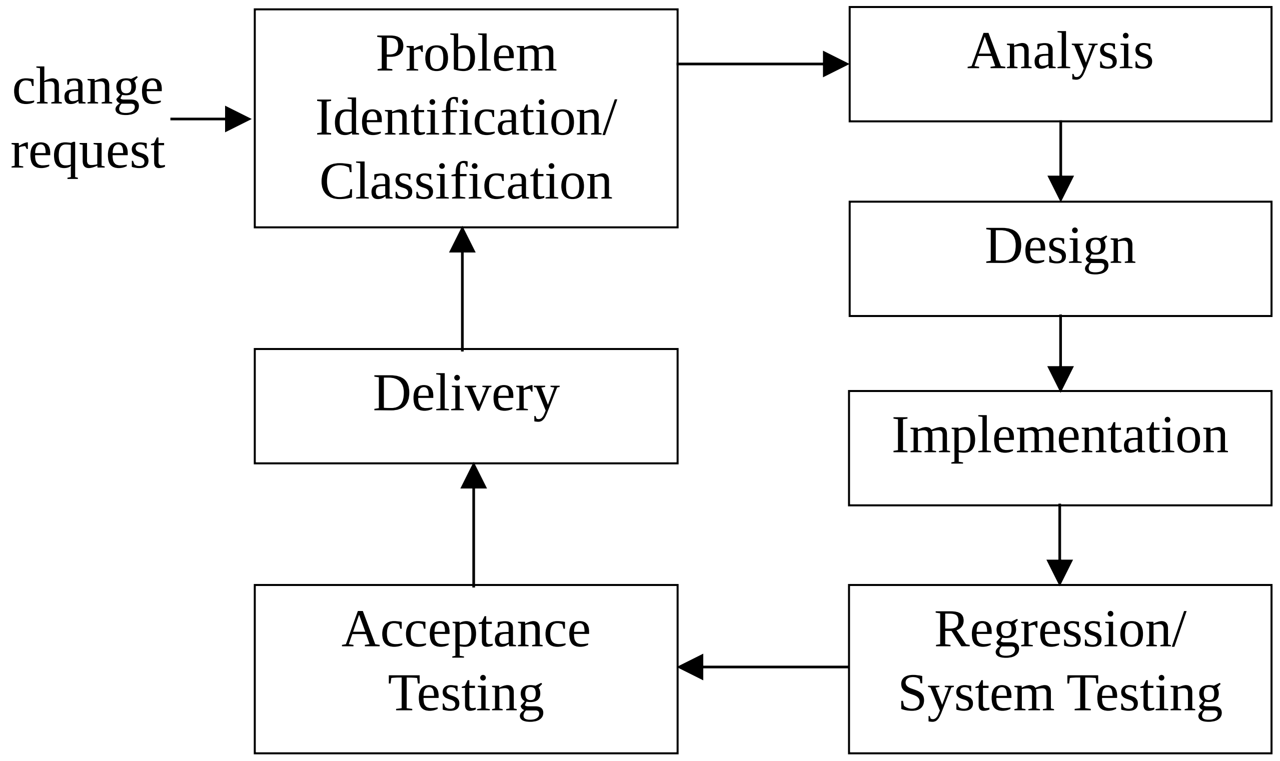
Analyzing the old system after delivery to identify update to requirements. Repeat the development life cycle to produce the updated system.

Full Reuse Model

$$\begin{tikzcd}
\textbf{Old System} & \textbf{Repository} & \textbf{New System} \\
\text{Requirements} \arrow[r] \arrow[d, dash] & (Ri) \arrow[r, leftrightarrow] & \text{Requirements} \arrow[d, dash] \\
\text{Design} \arrow[r] \arrow[d, dash] & (Di) \arrow[r, leftrightarrow] & \text{Design} \arrow[d, dash] \\
\text{Code} \arrow[r] \arrow[d, dash] & (Ci) \arrow[r, leftrightarrow] & \text{Code} \arrow[d, dash] \\
\text{Test} \arrow[r] & (Ti) \arrow[r, leftrightarrow] & \text{Test}
\end{tikzcd}$$

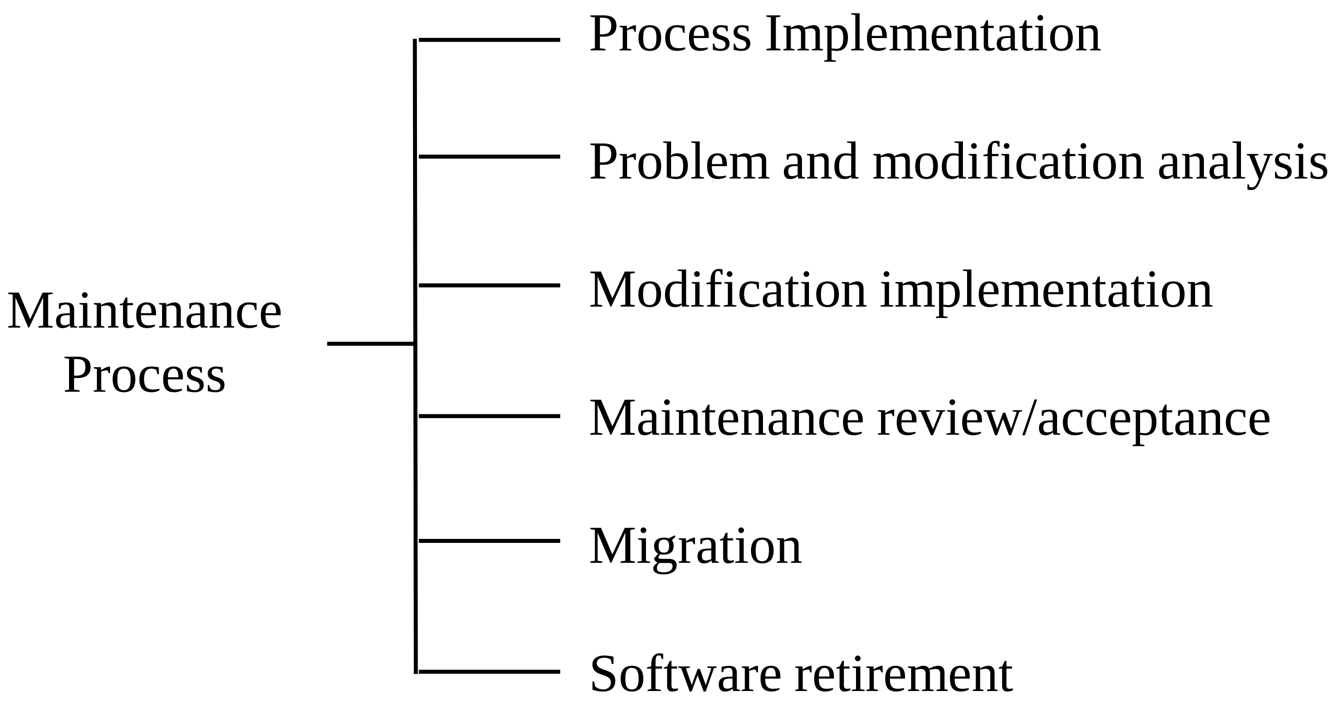
It emphasizes reusing components of the current system, or components from a reusable component repository. Components developed for the new system are made reusable and added to the repository.

IEEE-1219 Model



It is similar to the iterative enhancement model.

ISO-12207 Model

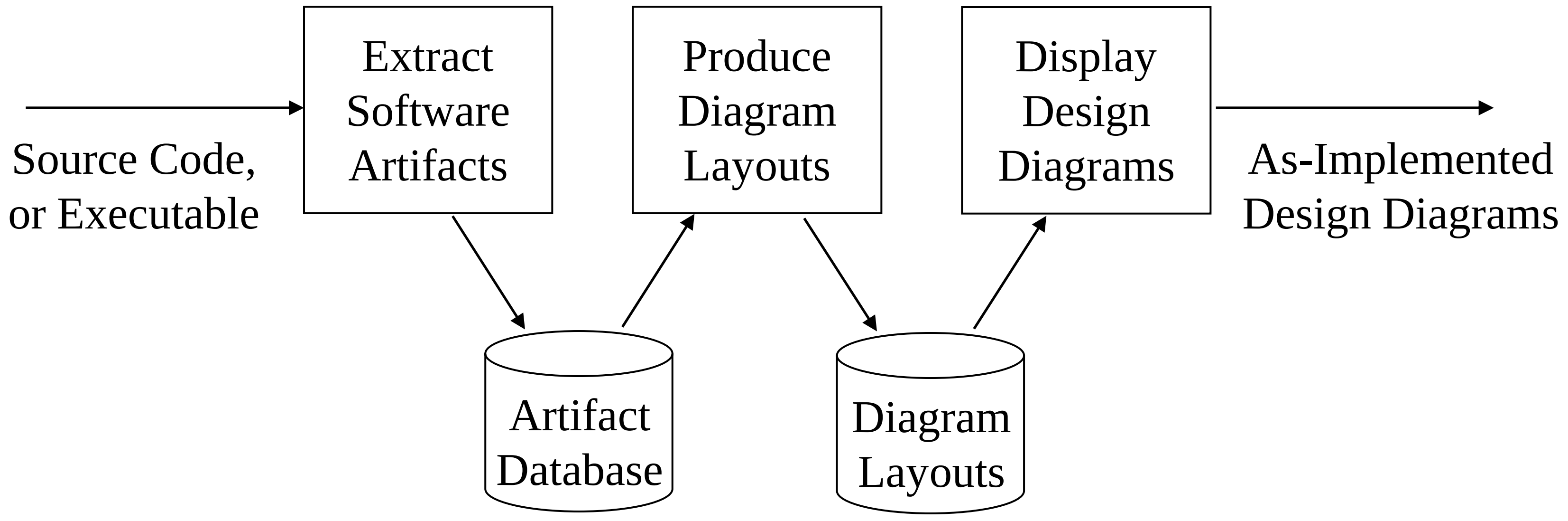

Maintenance Process
Process Implementation
Problem and modification analysis
Modification implementation
Maintenance review/acceptance
Migration
Software retirement

It is similar to the iterative enhancement model.

Reverse-Engineering

- A process of analyzing a subject system to identify system components and their interrelationships, and create representations of the system in another form or at a higher level of abstraction.

Reverse-Engineering Workflow


Extract Software Artifacts
Produce Diagram Layouts
Display Design Diagrams
Source Code, or Executable
As-Implemented Design Diagrams
Artifact Database
Diagram Layouts

Usefulness of Reverse-Engineering

- Program understanding.
- Formal analysis.
- Test generation.
- Software re-engineering.

Chapter 22: Software Configuration Management

Key Takeaway Points

- A baseline defines a significant state of progress of the system under development. It consists of a set of configuration items.
- Software configuration management (SCM) is baseline management, and configuration item management.
- SCM functions include configuration item identification, configuration change control, configuration auditing, and configuration accounting.

Software Configuration Management

- Software configuration management (SCM) is a discipline for
  - systematically identifying and labeling software configuration items (requirements, design, code module, test plan, etc.)
  - controlling changes to software configuration items
  - tracking implementation of the changes
  - reporting configuration status

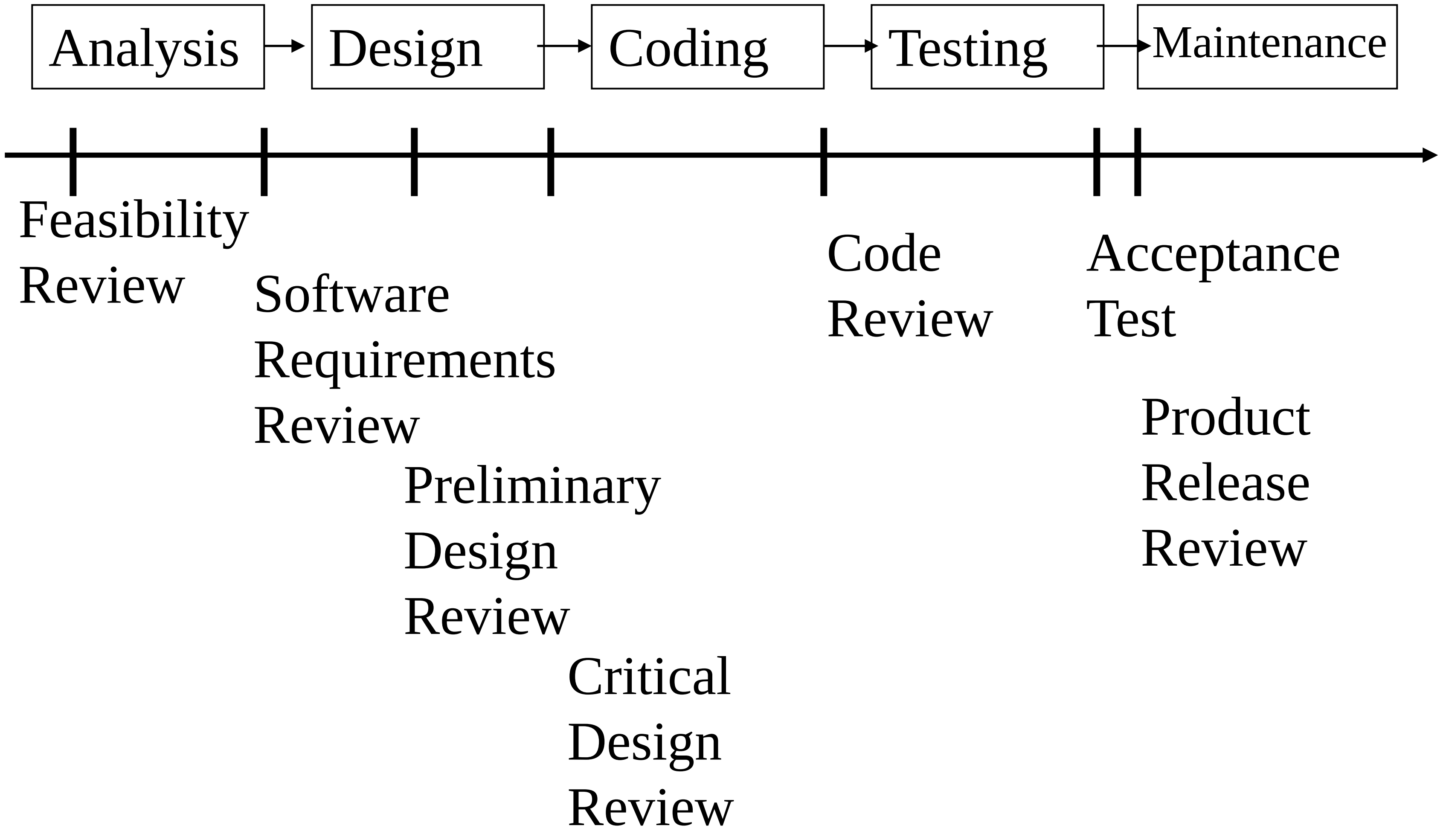
Software Configuration Management

- Software configuration management is
  - baseline management
  - software configuration item management
  - motivated by characteristics of software
- Requirements baseline, design baseline, implementation baseline, etc.
- Each baseline is a set of configuration items plus a set of changes

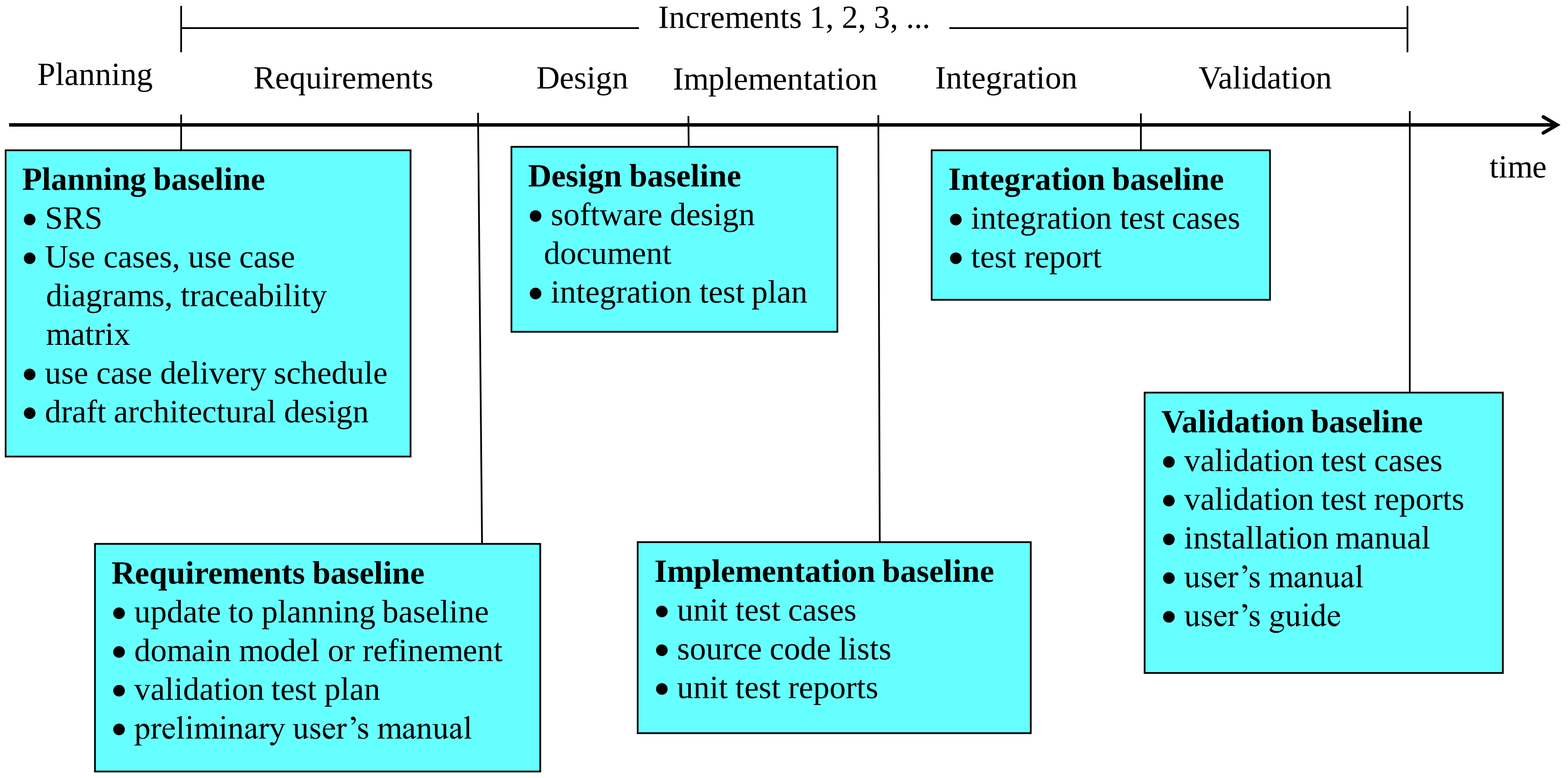
What Is a Baseline?

- A baseline is a set of documents, called software configuration items (SCM).
- It includes updates to a baseline.

Baselines of a Waterfall Process



Baselines of Agile Unified Methodology



Planning baseline
- SRS
- Use cases, use case diagrams, traceability matrix
- use case delivery schedule
- draft architectural design

Requirements baseline
- update to planning baseline
- domain model or refinement
- validation test plan
- preliminary user's manual

Design baseline
- software design document
- integration test plan

Implementation baseline
- unit test cases
- source code lists
- unit test reports

Integration baseline
- integration test cases
- test report

Validation baseline
- validation test cases
- validation test reports
- installation manual
- user's manual
- user's guide

SCM Functions

- SC identification
- SC control
- SC status accounting
- SC auditing

SC Identification

- Define the SC items
- Define a SCI naming scheme
- Define the relationships between the SCIs
- Determine the quality assurance (QA) personnel
- Determine who is responsible for delivering which SCI to SCM

Example SC Items

- System specification
- Project plan
- Software requirements spec
- Prototype (executable, or paper)
- Preliminary user's manual
- Design spec (preliminary, and detailed design specs)
- Source code listings

Example SC Items

- Test plan, procedure, cases, and results
- Installation manual
- Operation manual
- Executables
- As-built user's manual
- Maintenance documentation (problems, reports, requests, and ECP)
- Standards, and procedures

Requirements on Naming Scheme

- It can be used to uniquely identify the SCI.
- It should bear certain semantics.
- It should be such that related documents have related names.

Examples:

- RADC/UI/TOOLS/DE/ADI/F S
- RADC/UI/TOOLS/DE/ADI/CODE

Queries to SCM

- Question: What is a software system's configuration at a given baseline?
- Answer: It consists of SCI_1, SCI_2, SCI_3, ..., SCI_n.
- Class Exercise:
  - Identify SCIs for your team project.
  - Define a naming scheme.
  - Define the relationships between the SCIs.
  - Determine the quality assurance (QA) personnel.
  - Determine who is responsible for delivering which SCI to SCM.

SC Data Base

Queries to be answered:

- Which customer has a particular version of a system?
- Which particular version is currently used by which department?
- How many versions of a system have been created?

SC Data Base

- What were the creation dates?
- What versions might be affected if a particular component is changed?
- How many reported errors exist in a particular version?
- Which of the errors have been removed?

SC Change Control:

- Identifying needs for changes
- Preparing engineering change proposal (ECP)
- Approve or disapprove ECPs

Events Requiring Change

- Software deficiencies (inadequacy, incorrectness)
- Hardware changes (maybe due to H/W deficiency)
- New operational requirements Economic savings
- Schedule accommodation (compression, or extension)

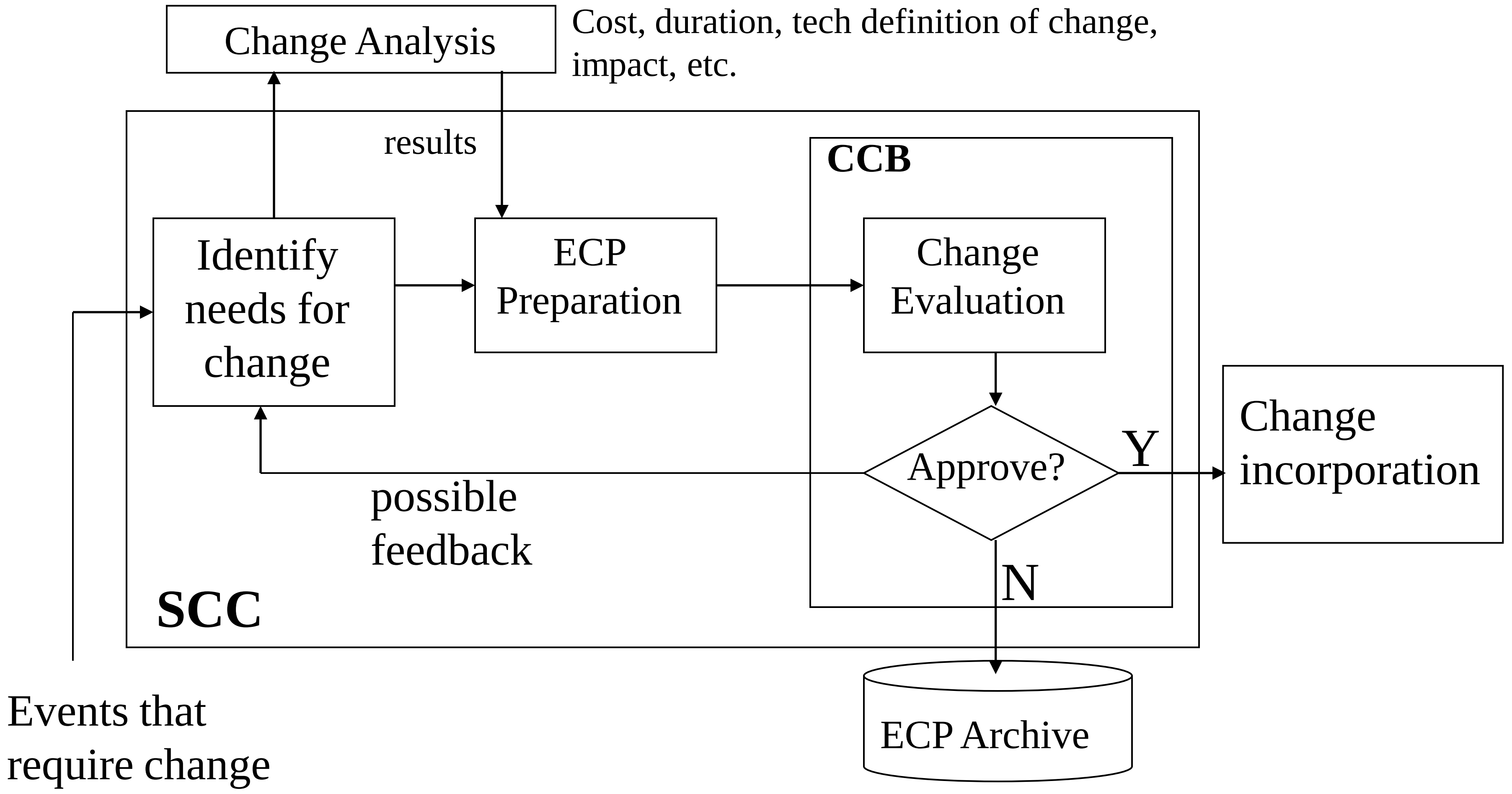
Basic Components in SC Control :

- Documentation that formally define the proposed changes to a software system (ECP)
  - Administrative forms
  - Supporting technical materials
  - Supporting administrative materials
- Configuration Control Board (CCB)
- A procedure for controlling the changes to a software system

Engineering Change Proposal

- Description of proposed changes
- Identification of originating organization
- Rationale for the changes
- Identification of the affected baselines and SCIs
- Costs
- Schedule impacts

Software Configuration Control Procedure



SC Auditing

- Determine the current status of the system with respect to current baseline and requirements
- Provide mechanisms for formally establishing a baseline
- Perform configuration verification and validation
- Ensure that changes are properly and timely implemented

Mechanisms to Accomplish SC Auditing

- Software quality assurance (formal technical reviews)
  - Defect list sheet
  - Review cover sheet
- Software configuration auditing
  - Version numbering
  - Software release
    - VDD
    - Directory listing comparison

SC Verification and Validation

- Configuration validation
  - ensure that the SCI solves the right problem
  - ensure that customer’s requirements are satisfied
  - You built the “right thing”
- Configuration verification
  - ensure that each software configuration item is produced according to baseline definition.
  - You built the “thing right”

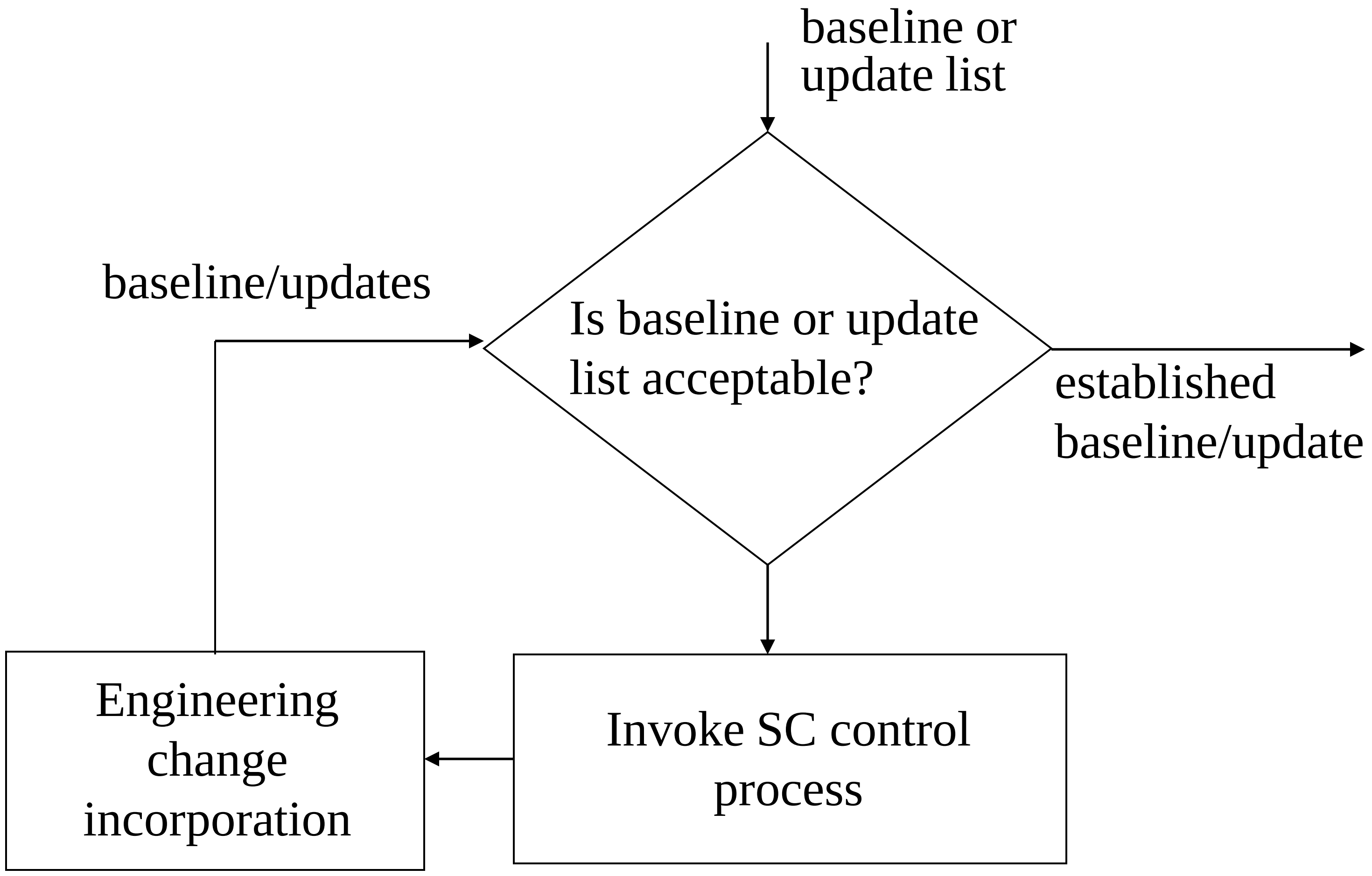
SC Auditing and Baselines

- System definition baseline focus on establishing clear trace between system requirements and the system concept
- Allocation baseline ensures that
  - software functions are defined
  - software functions are traceable to system functions
  - software functions are identified as, or assigned to SCIs

SC Auditing and Baselines

- Design baseline ensures that the design is unambiguous.
- Production baseline ensures that the executable SCIs perform adequately in the development environment.
- Operation baseline ensures that the product perform adequately in the target environment.

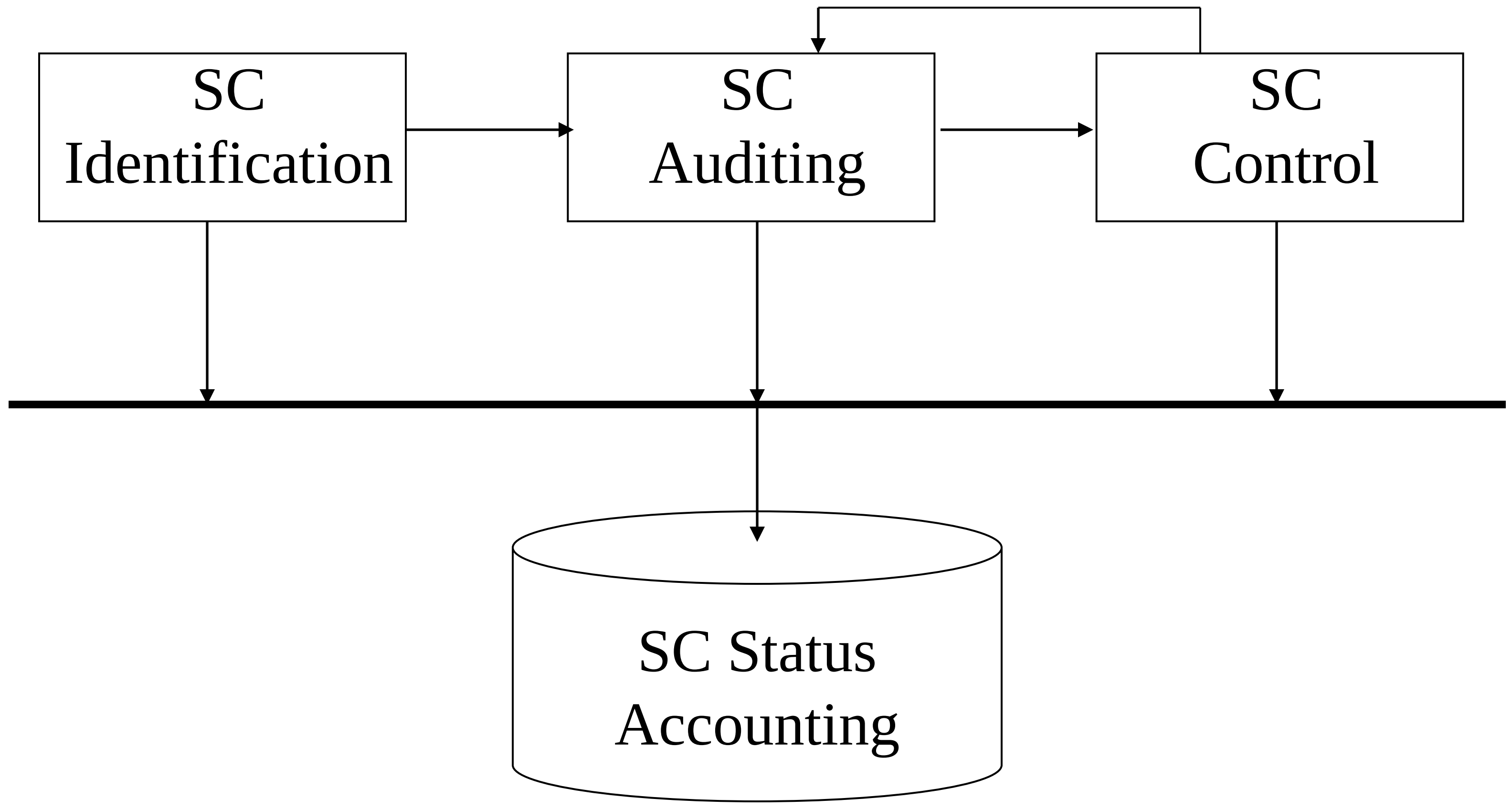
SC Auditing Process


baseline or update list
Is baseline or update list acceptable?
established baseline/update
baseline/updates
Invoke SC control process
Engineering change incorporation

SC Status Accounting

- Administrative tracking and reporting of SCIs formally identified and controlled
- DB support for the other three SCM tasks
- Problems to be addressed:
  - large amount of data
  - multiple representation of software
  - incomplete information
  - long transactions
  - inconsistency

Summary


SC Identification
SC Auditing
SC Control
SC Status Accounting