Chapter 26
Quality Management

Software Engineering: A Practitioner's Approach, 6th edition
by Roger S. Pressman

Quality

- The American Heritage Dictionary defines quality as
  - "a characteristic or attribute of something."
- For software, two kinds of quality may be encountered:
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.
  - user satisfaction = compliant product + good quality + delivery within budget and schedule

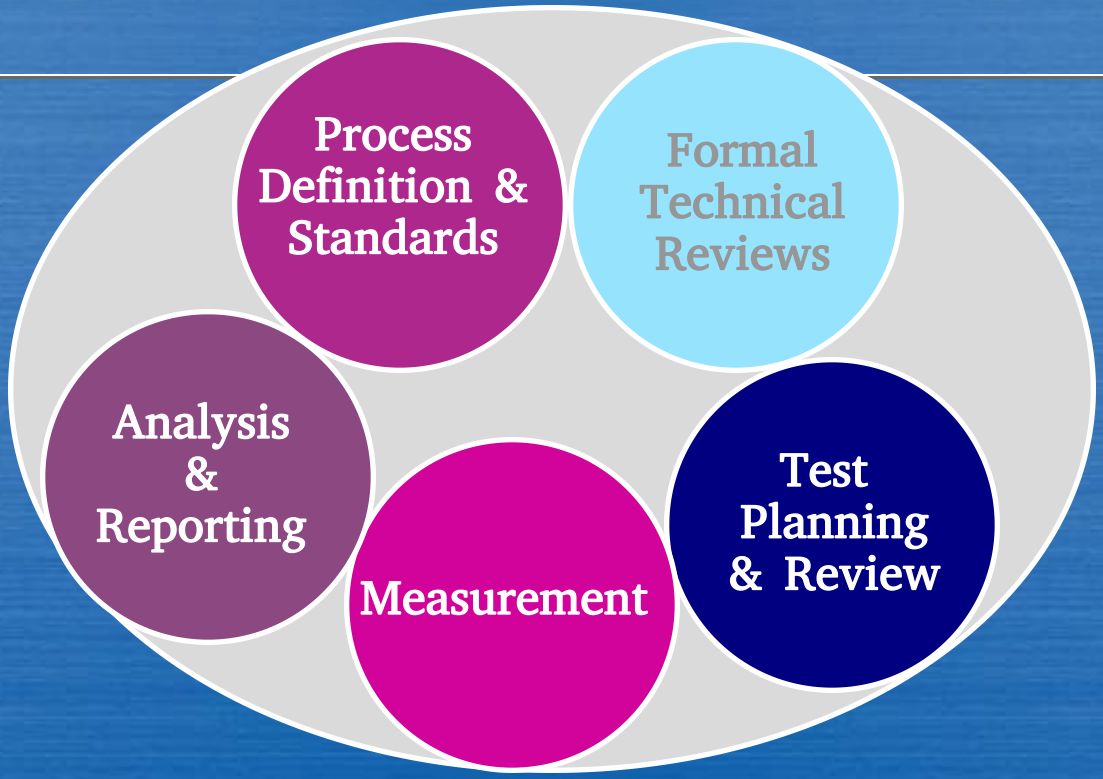
Software Quality

Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

Cost of Quality

- Prevention costs include
  - quality planning
  - formal technical reviews
  - test equipment
  - Training
- Internal failure costs include
  - rework
  - repair
  - failure mode analysis
- External failure costs are
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work

Software Quality Assurance



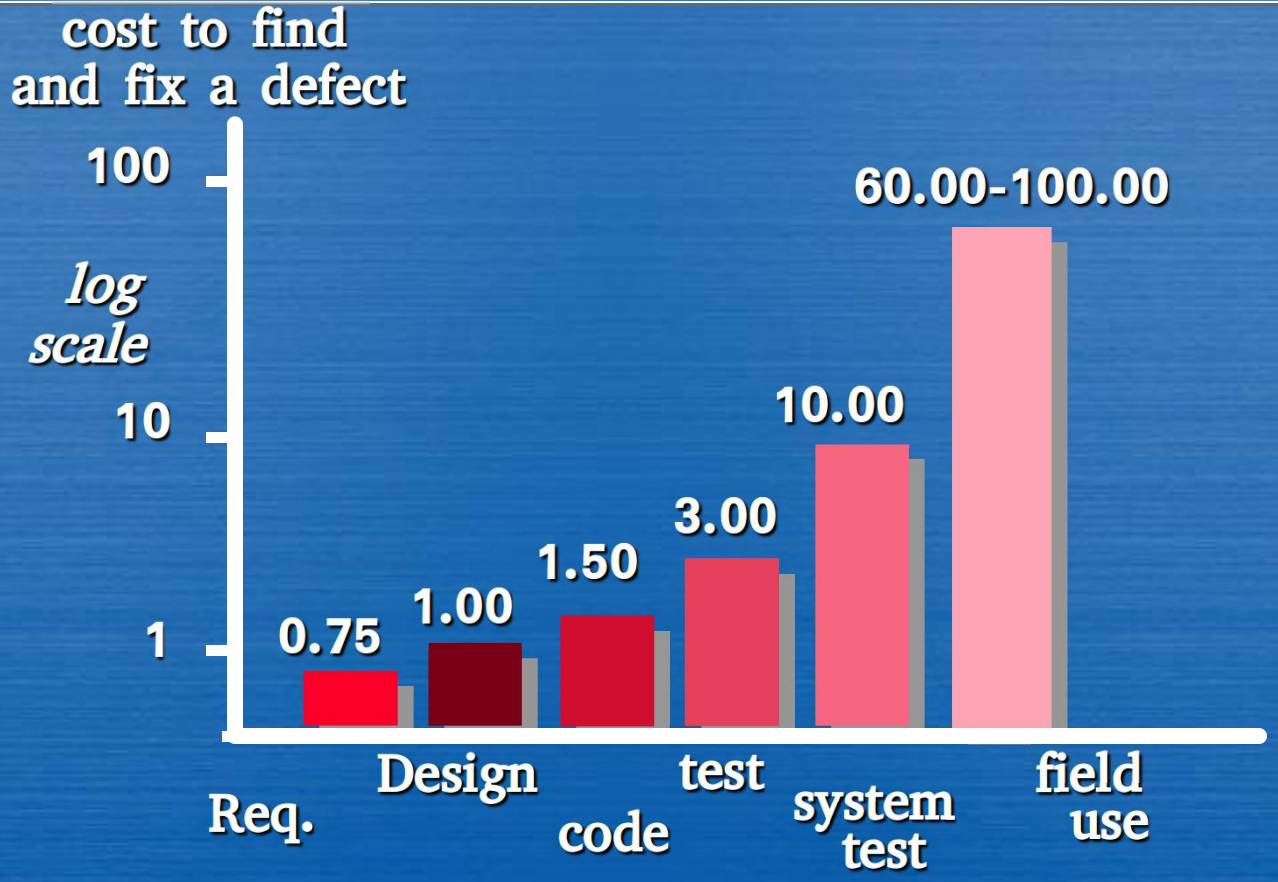
Role of the SQA Group-I

- Prepares an SQA plan for a project.
  - The plan identifies
    - evaluations to be performed
    - audits and reviews to be performed
    - standards that are applicable to the project
    - procedures for error reporting and tracking
    - documents to be produced by the SQA group
    - amount of feedback provided to the software project team
- Participates in the development of the project's software process description.
  - The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.

Role of the SQA Group-II

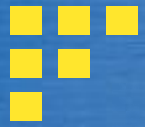
- Reviews software engineering activities to verify compliance with the defined software process.
  - identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
- Audits designated software work products to verify compliance with those defined as part of the software process.
  - reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
  - periodically reports the results of its work to the project manager.
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure.
- Records any noncompliance and reports to senior management.
  - Noncompliance items are tracked until they are resolved.

Why SQA Activities Pay Off?



Reviews & Inspections

> ... there is no particular reason why your friend and colleague cannot also be your sternest critic.
>
> *Jerry Weinberg*



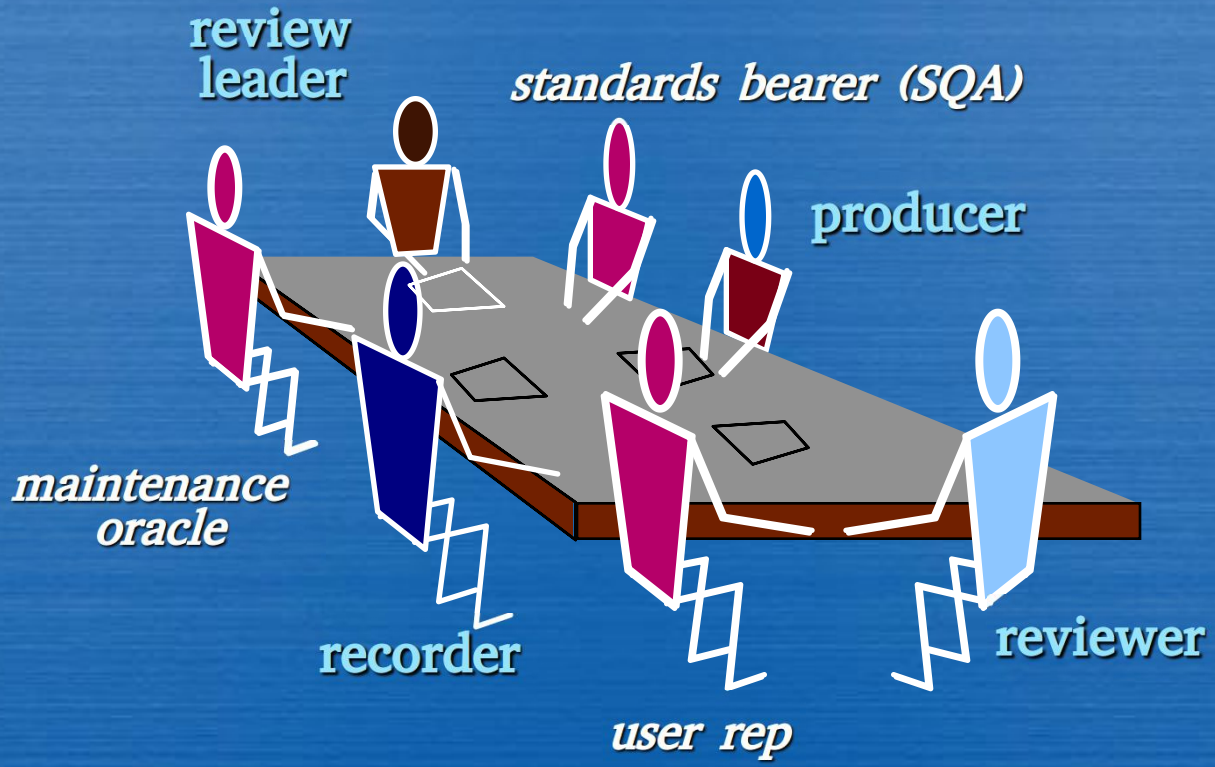
What Are Reviews?

- a meeting conducted by technical people for technical people
- a technical assessment of a work product created during the software engineering process
- a software quality assurance mechanism
- a training ground

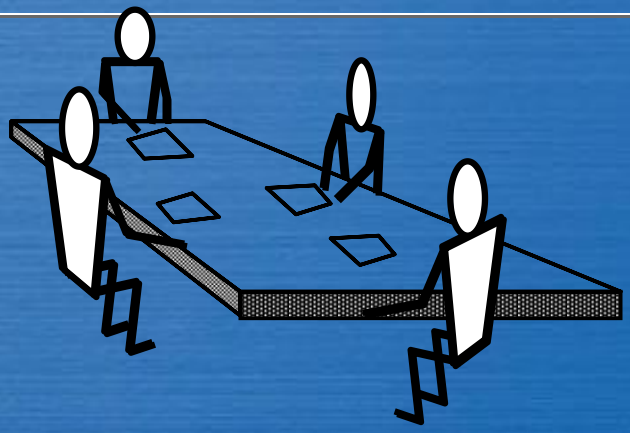
What Reviews Are Not

- A project summary or progress assessment
- A meeting intended solely to impart information
- A mechanism for political or personal reprisal!

The Players


review leader
standards bearer (SQA)
producer
maintenance oracle
recorder
user rep
reviewer

Conducting the Review



1. be prepared—evaluate product before the review
2. review the product, not the producer
3. keep your tone mild, ask questions instead of making accusations
4. stick to the review agenda
5. raise issues, don't resolve them
6. avoid discussions of style—stick to technical correctness
7. schedule reviews as project tasks
8. record and report all review results

Review Options Matrix

	IPR *	WT	IN	RRR
trained leader	no	yes	yes	yes
agenda established	maybe	yes	yes	yes
reviewers prepare in advance	maybe	yes	yes	yes
producer presents product	maybe	yes	no	no
"reader" presents product	no	no	yes	no
recorder takes notes	maybe	yes	yes	yes
checklists used to find errors	no	no	yes	no
errors categorized as found	no	no	yes	no
issues list created	no	yes	yes	yes
team must sign-off on result	no	yes	yes	maybe

*IPR—informal peer review WT—Walkthrough
IN—Inspection RRR—round robin review

Sample-Driven Reviews (SDRs)

- SDRs attempt to quantify those work products that are primary targets for full FTRs.

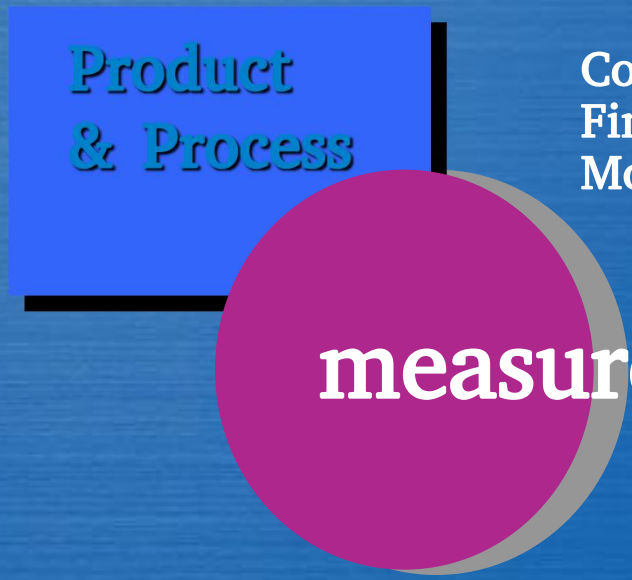
To accomplish this …

- Inspect a fraction a_i of each software work product, *i.* Record the number of faults, f_i found within a_i.
- Develop a gross estimate of the number of faults within work product *i* by multiplying f_i by $1/a_i$.
- Sort the work products in descending order according to the gross estimate of the number of faults in each.
- Focus available review resources on those work products that have the highest estimated number of faults.

Metrics Derived from Reviews

- inspection time per page of documentation
- inspection time per KLOC or FP
- inspection effort per KLOC or FP
- errors uncovered per reviewer hour
- errors uncovered per preparation hour
- errors uncovered per SE task (e.g., design)
- number of minor errors (e.g., typos)
- number of major errors (e.g., nonconformance to req.)
- number of errors found during preparation

Statistical SQA



Collect information on all defects
Find the causes of the defects
Move to provide fixes for the process

... an understanding of how to improve quality ...

Six-Sigma for Software Engineering

- The term "six sigma" is derived from six standard deviations—3.4 instances (defects) per million occurrences—implying an extremely high quality standard.
- The Six Sigma methodology defines three core steps:
  - Define customer requirements and deliverables and project goals via well-defined methods of customer communication
  - Measure the existing process and its output to determine current quality performance (collect defect metrics)
  - Analyze defect metrics and determine the vital few causes.
  - Improve the process by eliminating the root causes of defects.
  - Control the process to ensure that future work does not reintroduce the causes of defects.

Software Reliability

- A simple measure of reliability is *mean-time-between-failure* (MTBF), where

  MTBF = MTTF + MTTR

- The acronyms MTTF and MTTR are *mean-time-to-failure* and *mean-time-to-repair*, respectively.
- *Software availability* is the probability that a program is operating according to requirements at a given point in time and is defined as

  Availability = [MTTF/(MTTF + MTTR)] x 100%

Software Safety

- Software safety is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.
- If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards.

Mistake-Proofing

- Poka-yoke (mistake-proofing) devices—mechanisms that lead to
  - the prevention of a potential quality problem before it occurs or
  - the rapid detection of quality problems if they are introduced.
- An effective poka-yoke device exhibits a set of common characteristics:
  - It is simple and cheap. If a device is too complicated or expensive, it will not be cost effective.
  - It is part of the process. That is, the poka-yoke device is integrated into an engineering activity.
  - It is located near the process task where the mistakes occur. Thus, it provides rapid feedback and error correction.

ISO 9001:2000 Standard

- ISO 9001:2000 is the quality assurance standard that applies to software engineering.
- The standard contains 20 requirements that must be present for an effective quality assurance system.
- The requirements delineated by ISO 9001:2000 address topics such as
  - management responsibility, quality system, contract review, design control, document and data control, product identification and traceability, process control, inspection and testing, corrective and preventive action, control of quality records, internal quality audits, training, servicing, and statistical techniques.

Chapter 27
Change Management

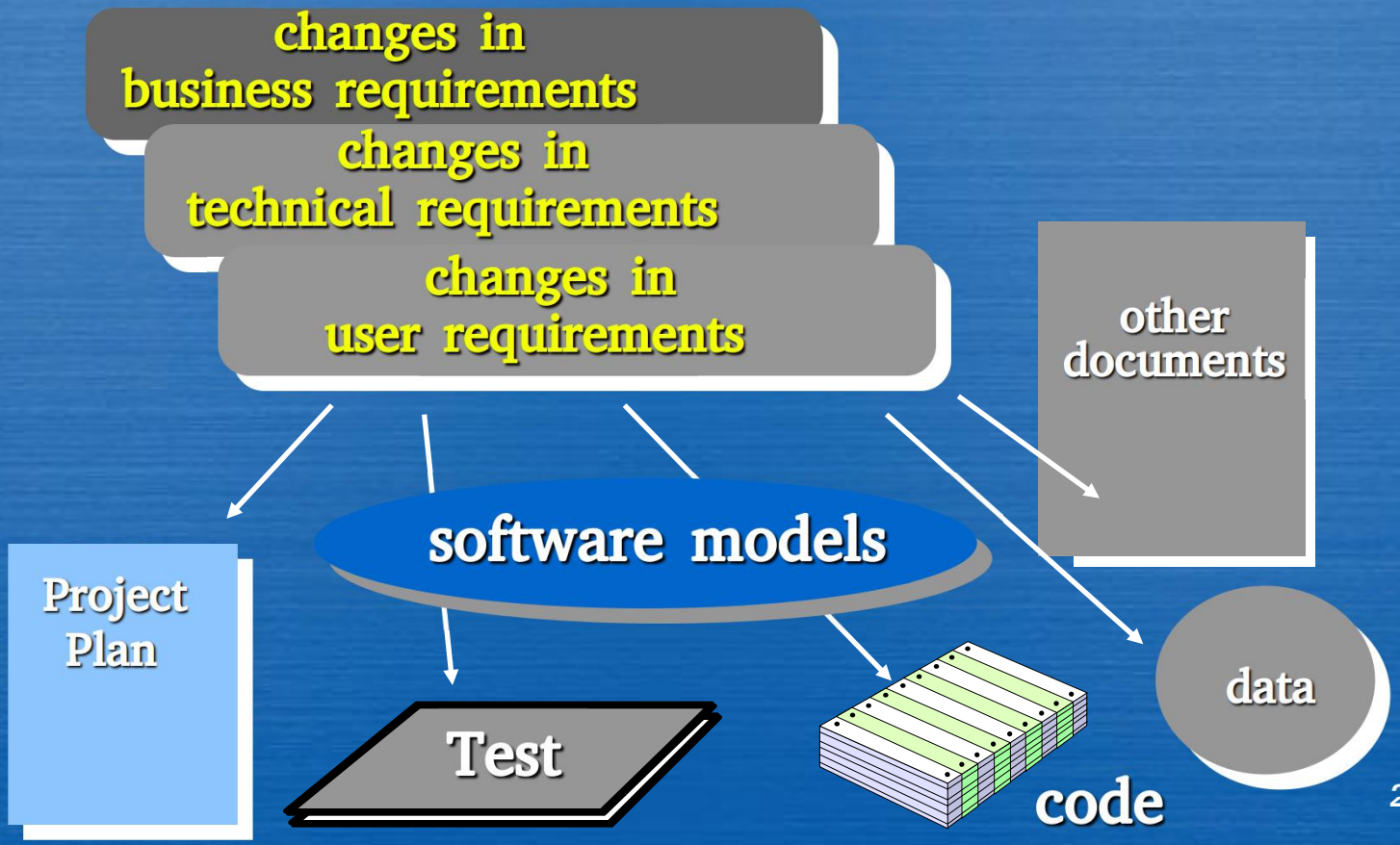
Software Engineering: A Practitioner's Approach, 6th edition
by Roger S. Pressman

The "First Law"

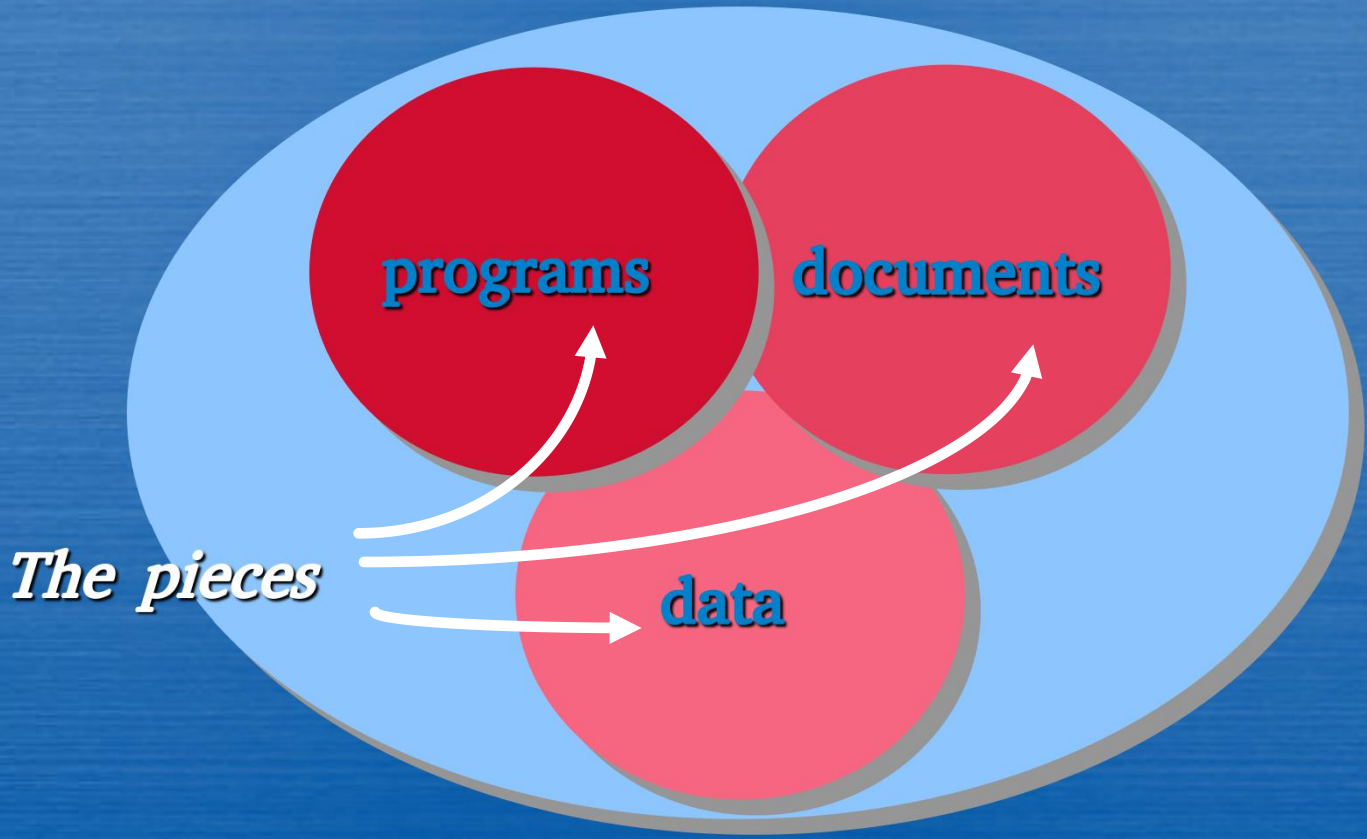
No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle.

Bersoff, et al, 1980

What Are These Changes?



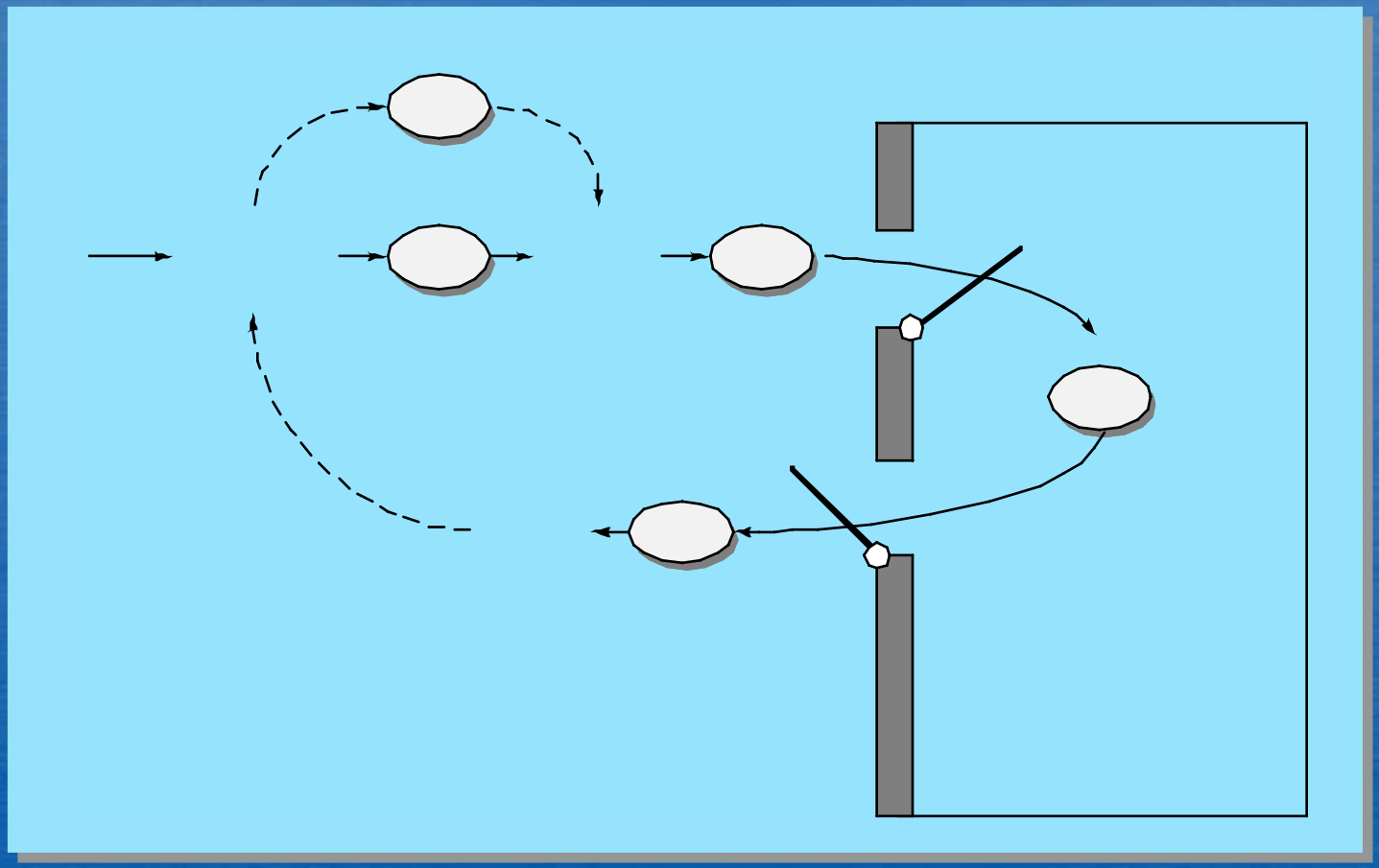
The Software Configuration



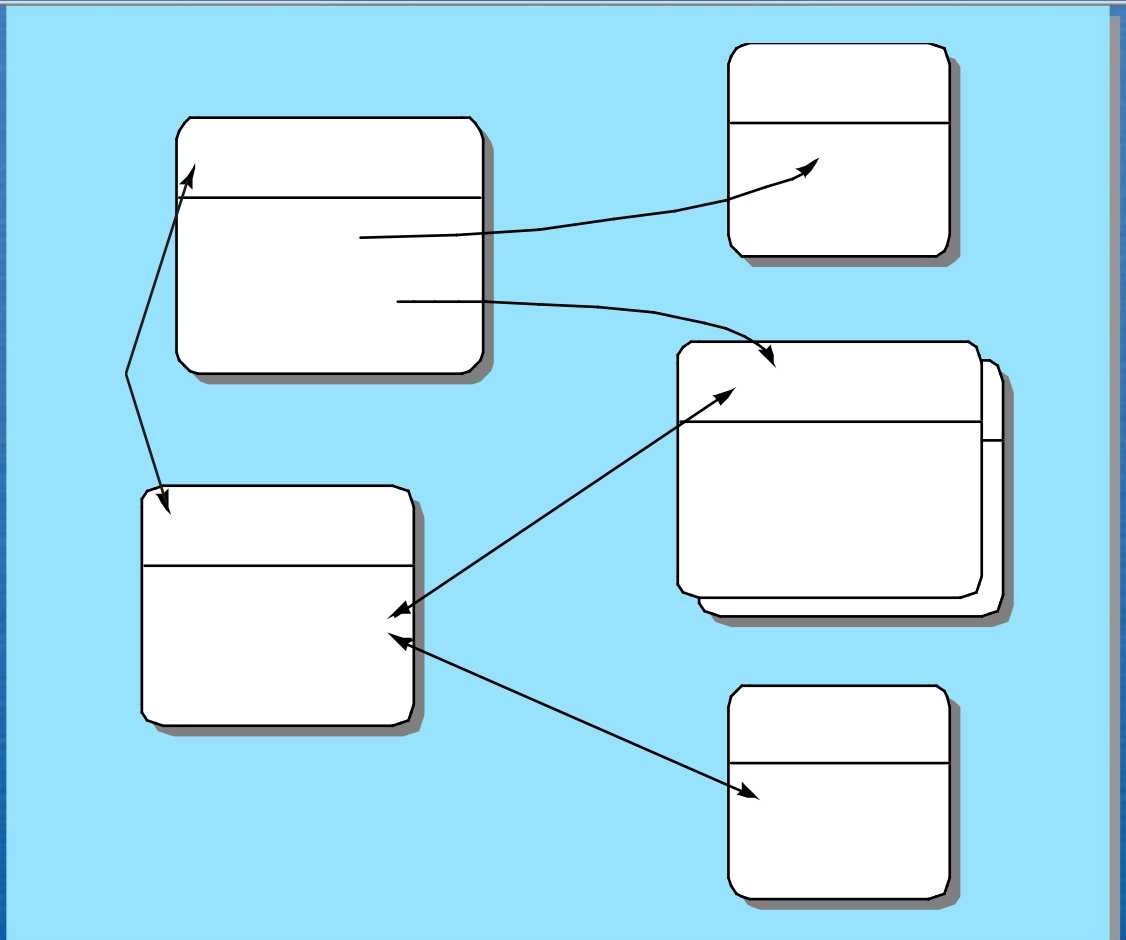
Baselines

- 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



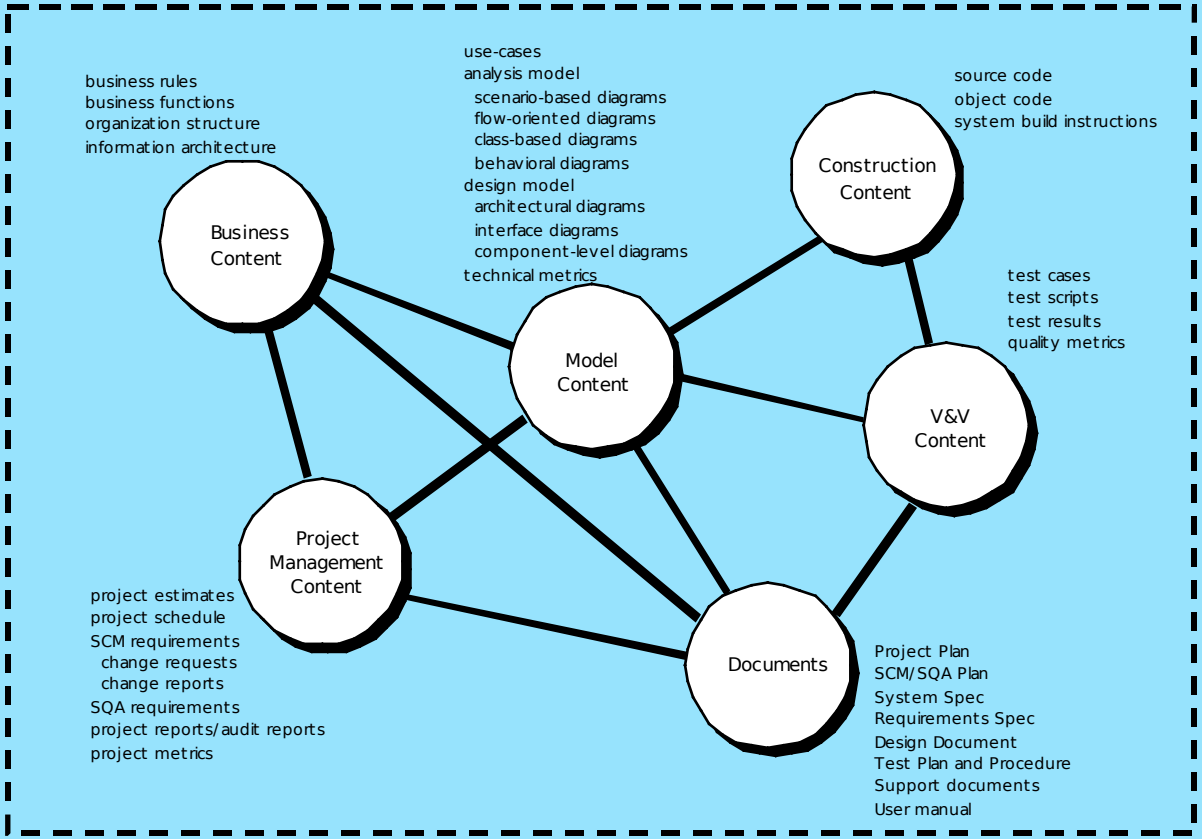
Software Configuration Objects



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 [FOR89]:
  - Data integrity
  - Information sharing
  - Tool integration
  - Data integration
  - Methodology enforcement
  - Document standardization

Repository Content

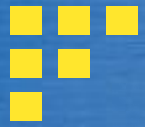


Repository Features

- 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
- 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.

SCM Elements

- 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)

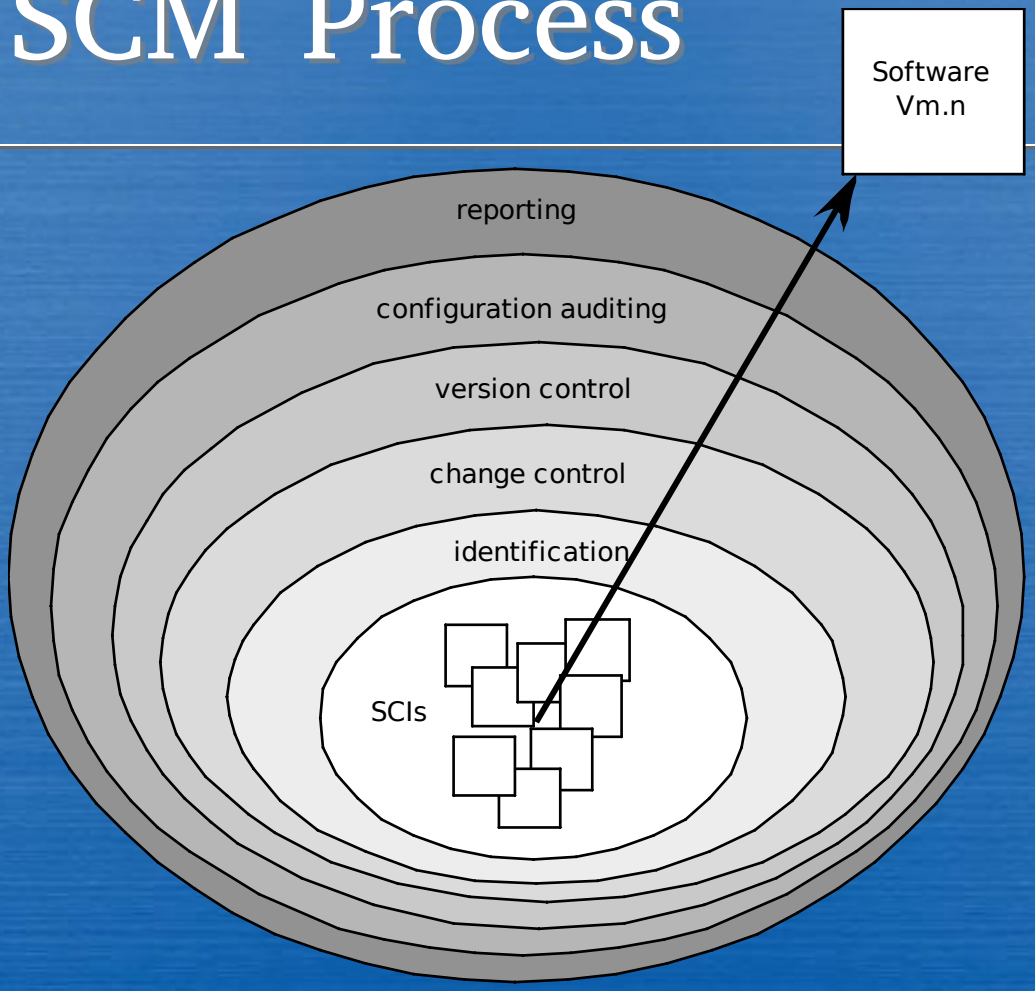


The SCM Process

Addresses the following questions ...

- How does a software team identify the discrete elements of a software configuration?
- How does an organization manage the many existing versions of a program (and its documentation) in a manner that will enable change to be accommodated efficiently?
- How does an organization control changes before and after software is released to a customer?
- Who has responsibility for approving and ranking changes?
- How can we ensure that changes have been made properly?
- What mechanism is used to appraise others of changes that are made?

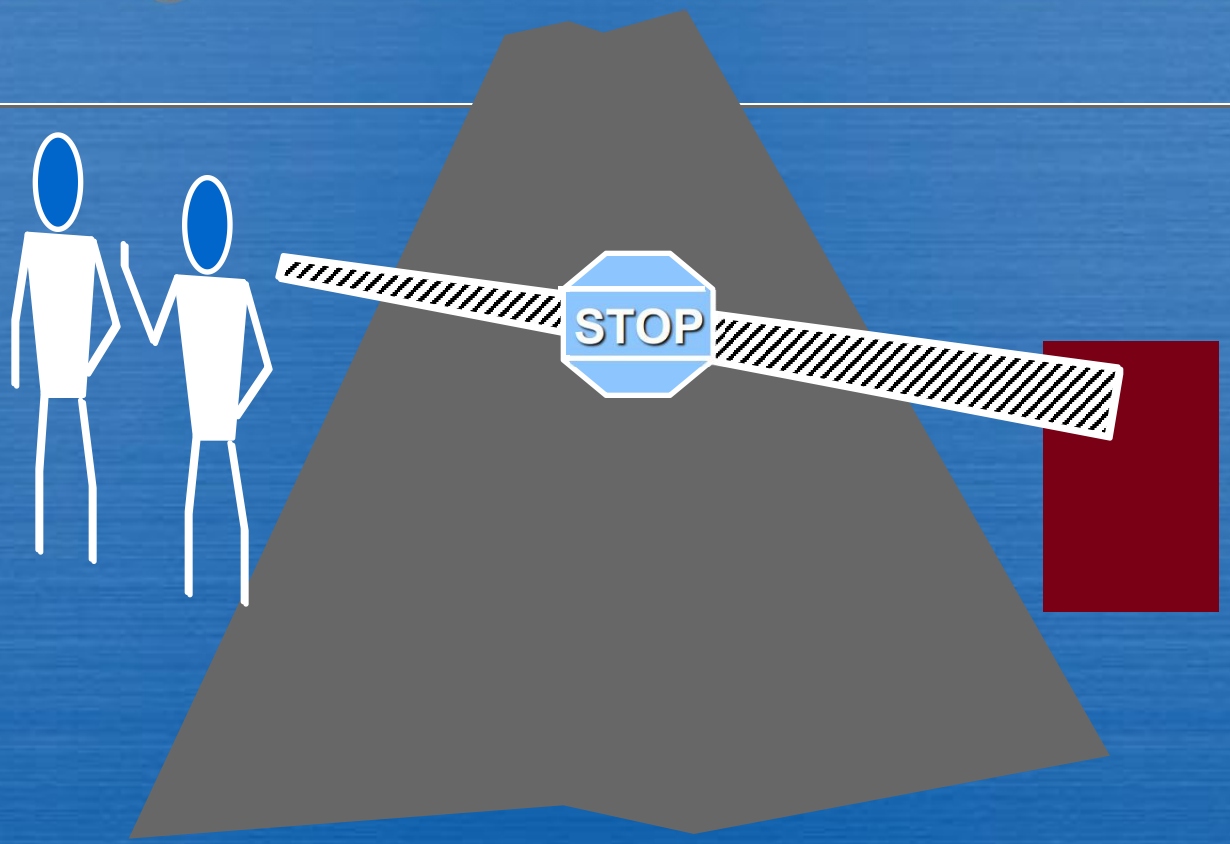
The SCM Process



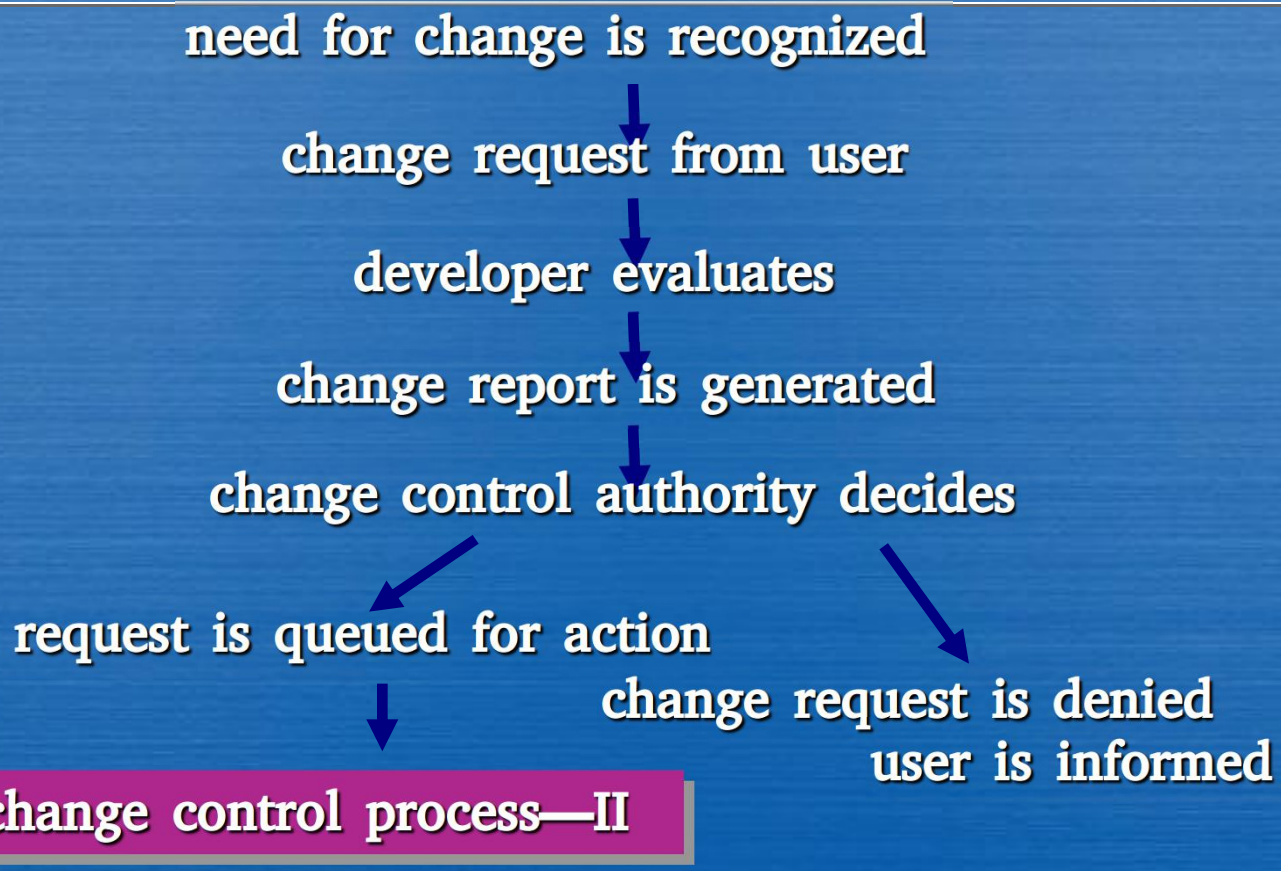
Version Control

- 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.

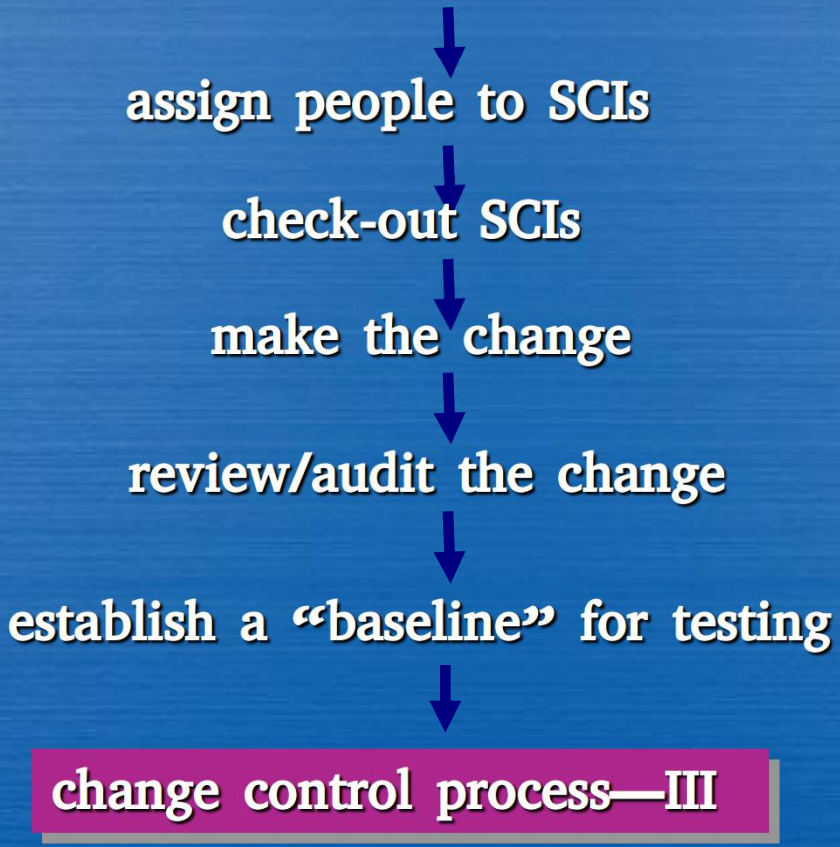
Change Control



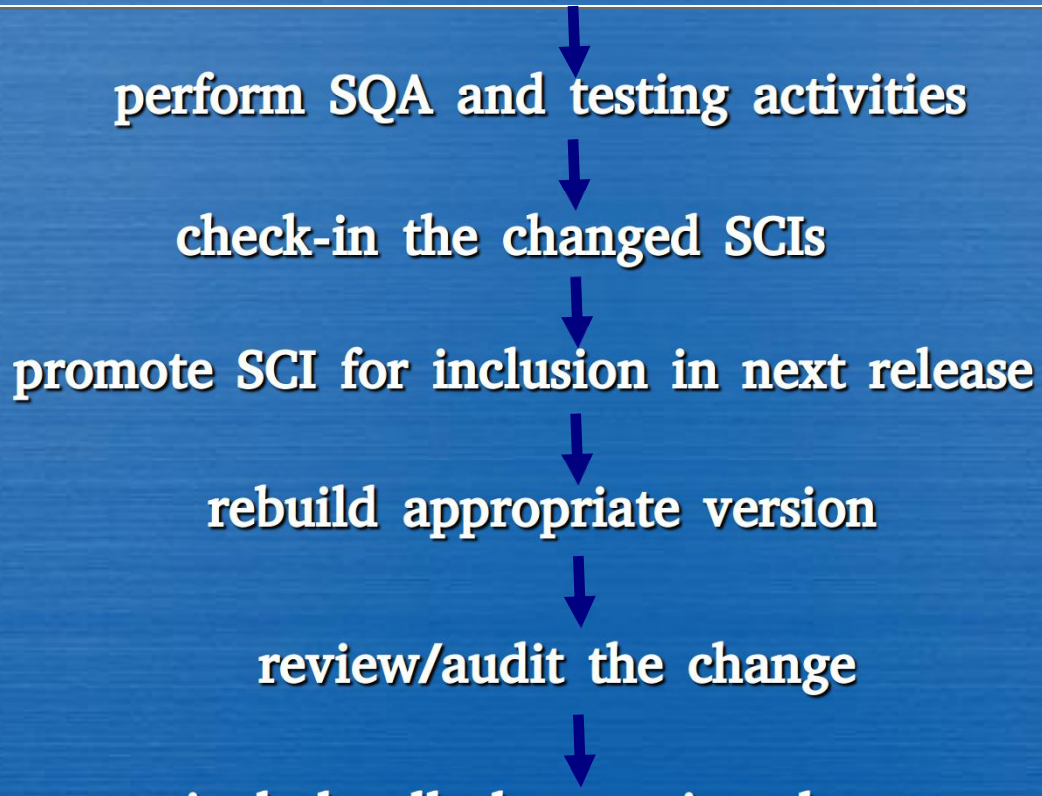
Change Control Process—I



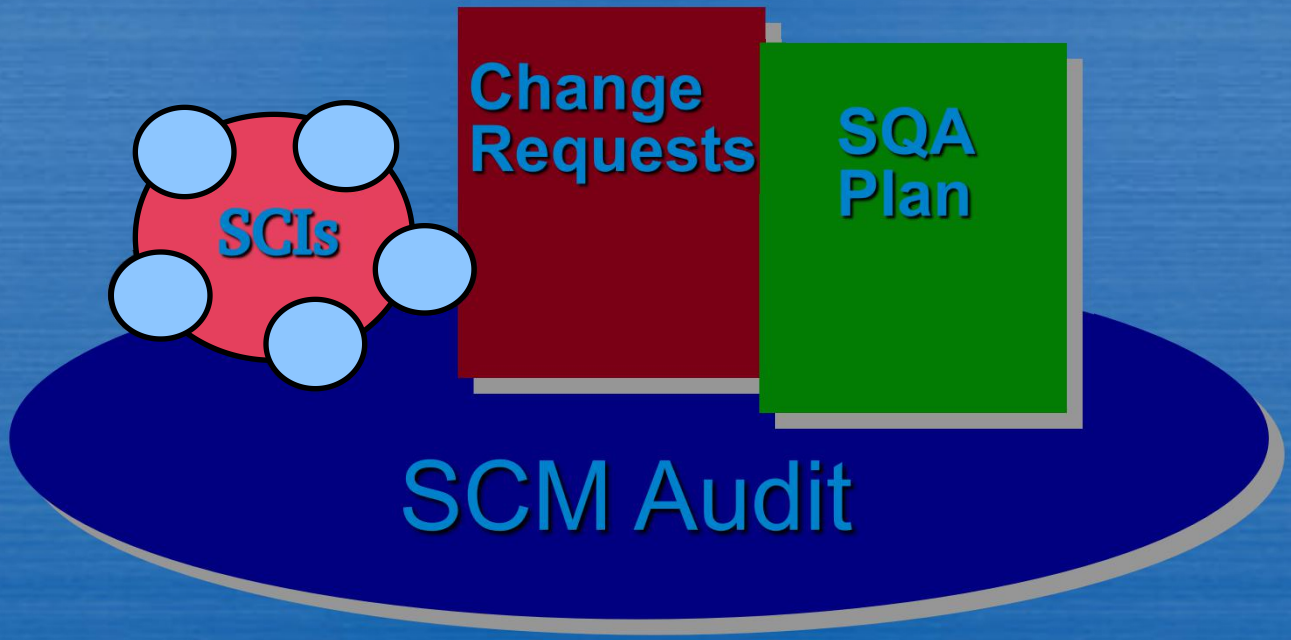
Change Control Process-II



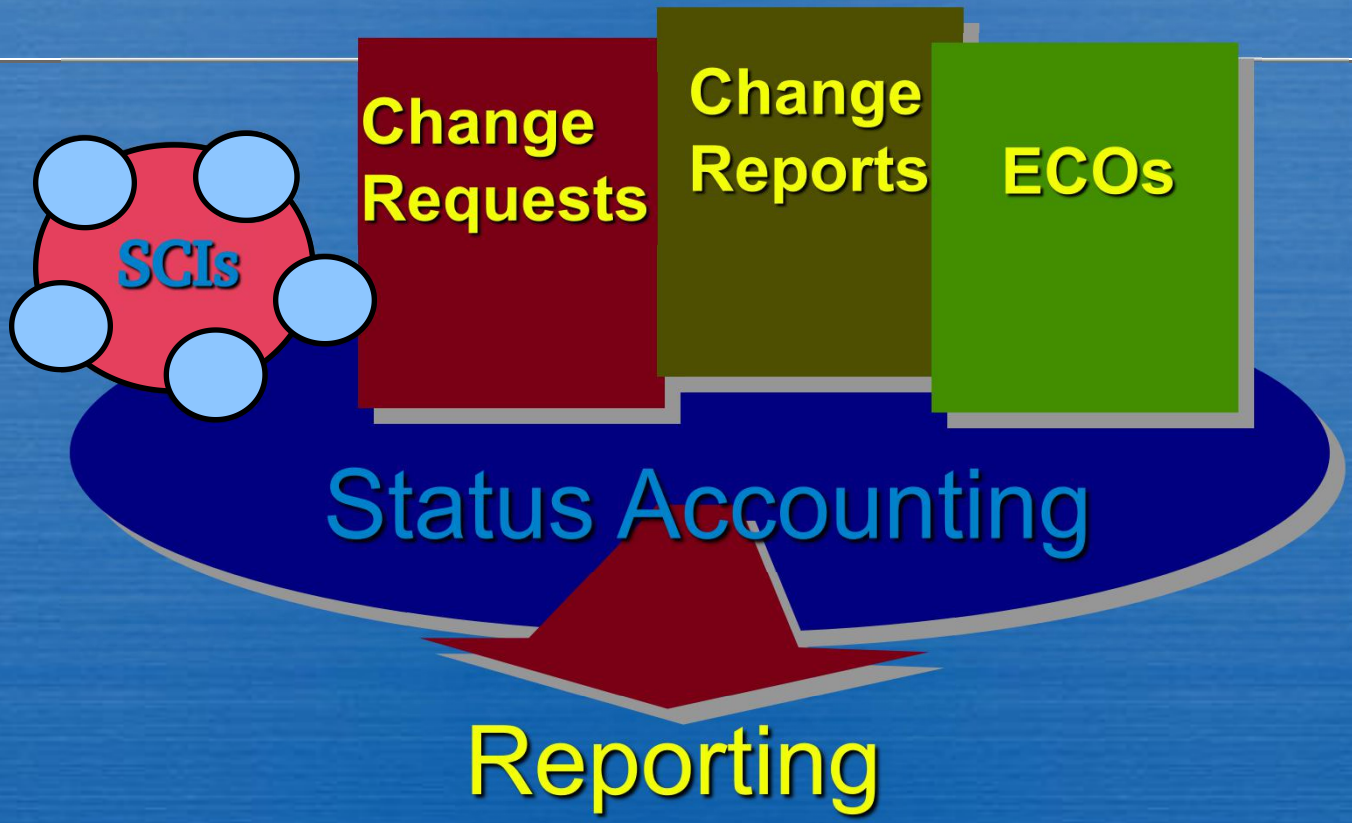
Change Control Process-III



Auditing



Status Accounting



SCM for Web Engineering-I

- **Content.**
  - A typical WebApp contains a vast array of content—text, graphics, applets, scripts, audio/video files, forms, active page elements, tables, streaming data, and many others.
  - The challenge is to organize this sea of content into a rational set of configuration objects (Section 27.1.4) and then establish appropriate configuration control mechanisms for these objects.
- **People.**
  - Because a significant percentage of WebApp development continues to be conducted in an ad hoc manner, any person involved in the WebApp can (and often does) create content.

SCM for Web Engineering-II

- **Scalability.**
  - As size and complexity grow, small changes can have far-reaching and unintended affects that can be problematic. Therefore, the rigor of configuration control mechanisms should be directly proportional to application scale.
- **Politics.**
  - Who 'owns' a WebApp?
  - Who assumes responsibility for the accuracy of the information on the Web site?
  - Who assures that quality control processes have been followed before information is published to the site?
  - Who is responsible for making changes?
  - Who assumes the cost of change?

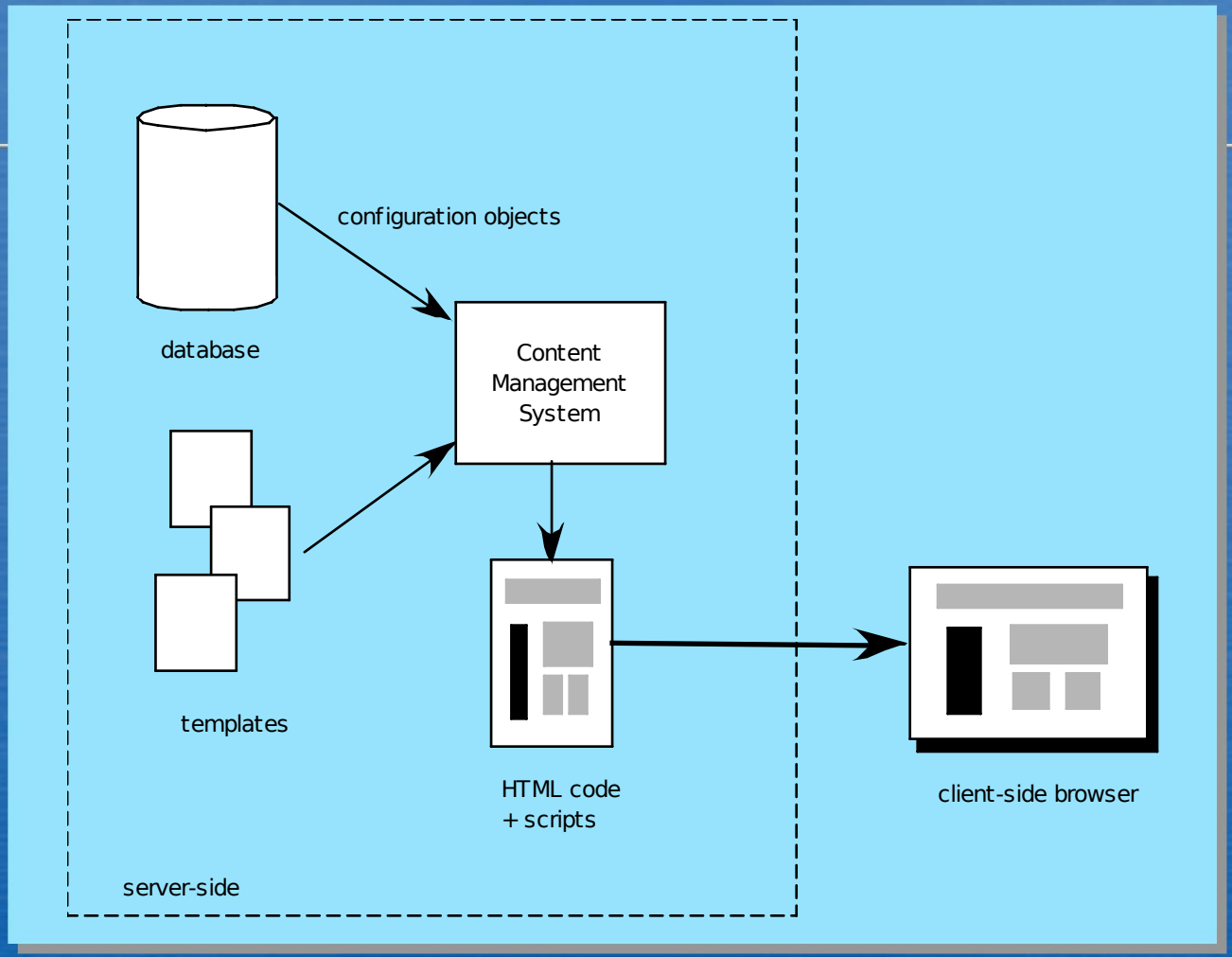
Content Management-I

- **The collection subsystem** encompasses all actions required to create and/or acquire content, and the technical functions that are necessary to
  - convert content into a form that can be represented by a mark-up language (e.g., HTML, XML
  - organize content into packets that can be displayed effectively on the client-side.
- **The management subsystem** implements a repository that encompasses the following elements:
  - *Content database*—the information structure that has been established to store all content objects
  - *Database capabilities*—functions that enable the CMS to search for specific content objects (or categories of objects), store and retrieve objects, and manage the file structure that has been established for the content
  - *Configuration management functions*—the functional elements and associated workflow that support content object identification, version control, change management, change auditing, and reporting.

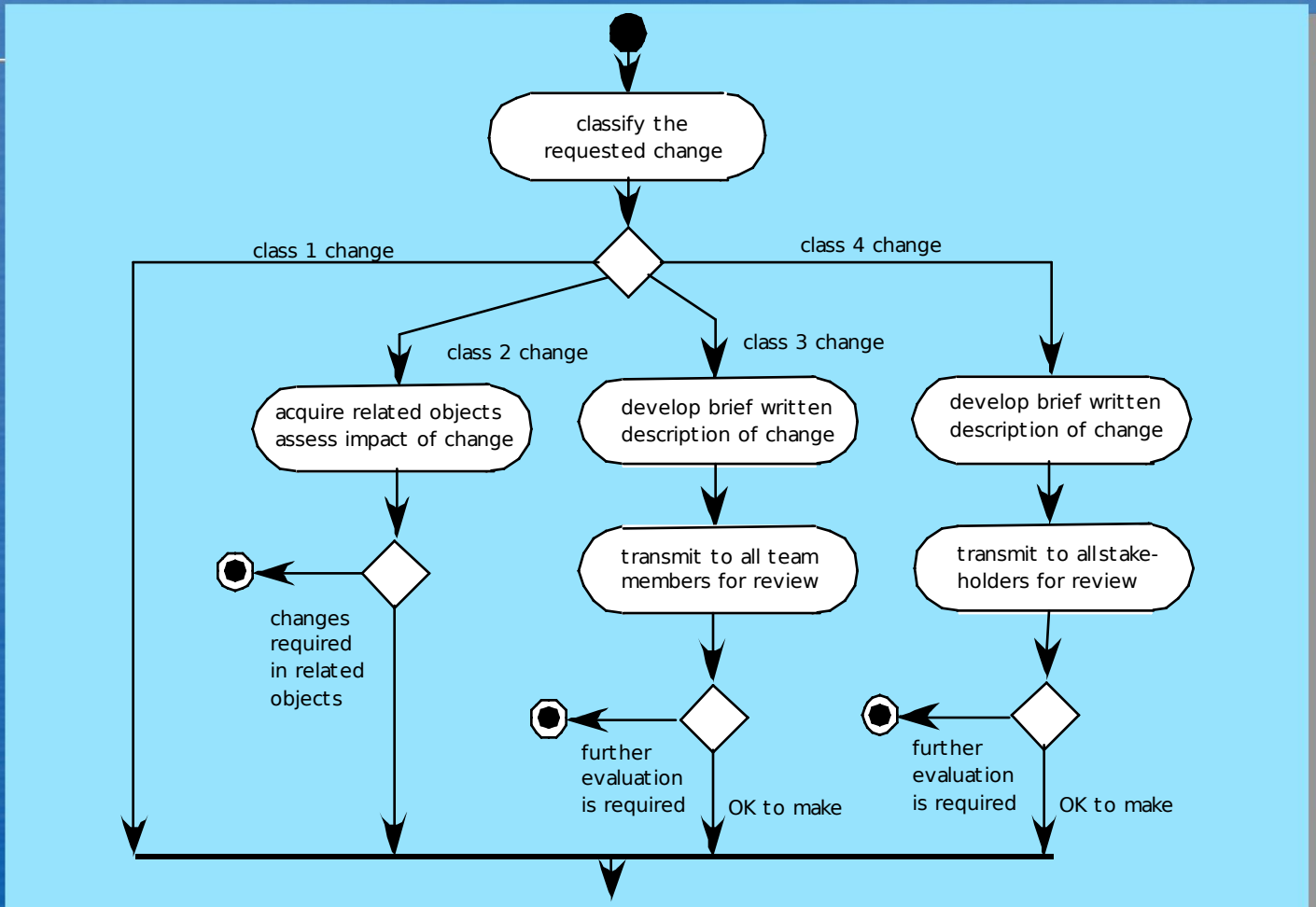
Content Management-II

- The **publishing subsystem** extracts from the repository, converts it to a form that is amenable to publication, and formats it so that it can be transmitted to client-side browsers. The publishing subsystem accomplishes these tasks using a series of templates.
- Each *template* is a function that builds a publication using one of three different components [BOI02]:
  - *Static elements*—text, graphics, media, and scripts that require no further processing are transmitted directly to the client-side
  - *Publication services*—function calls to specific retrieval and formatting services that personalize content (using predefined rules), perform data conversion, and build appropriate navigation links.
  - *External services*—provide access to external corporate information infrastructure such as enterprise data or "back-room" applications.

Content Management



Change Management for WebApps-I



Change Management for WebApps-II