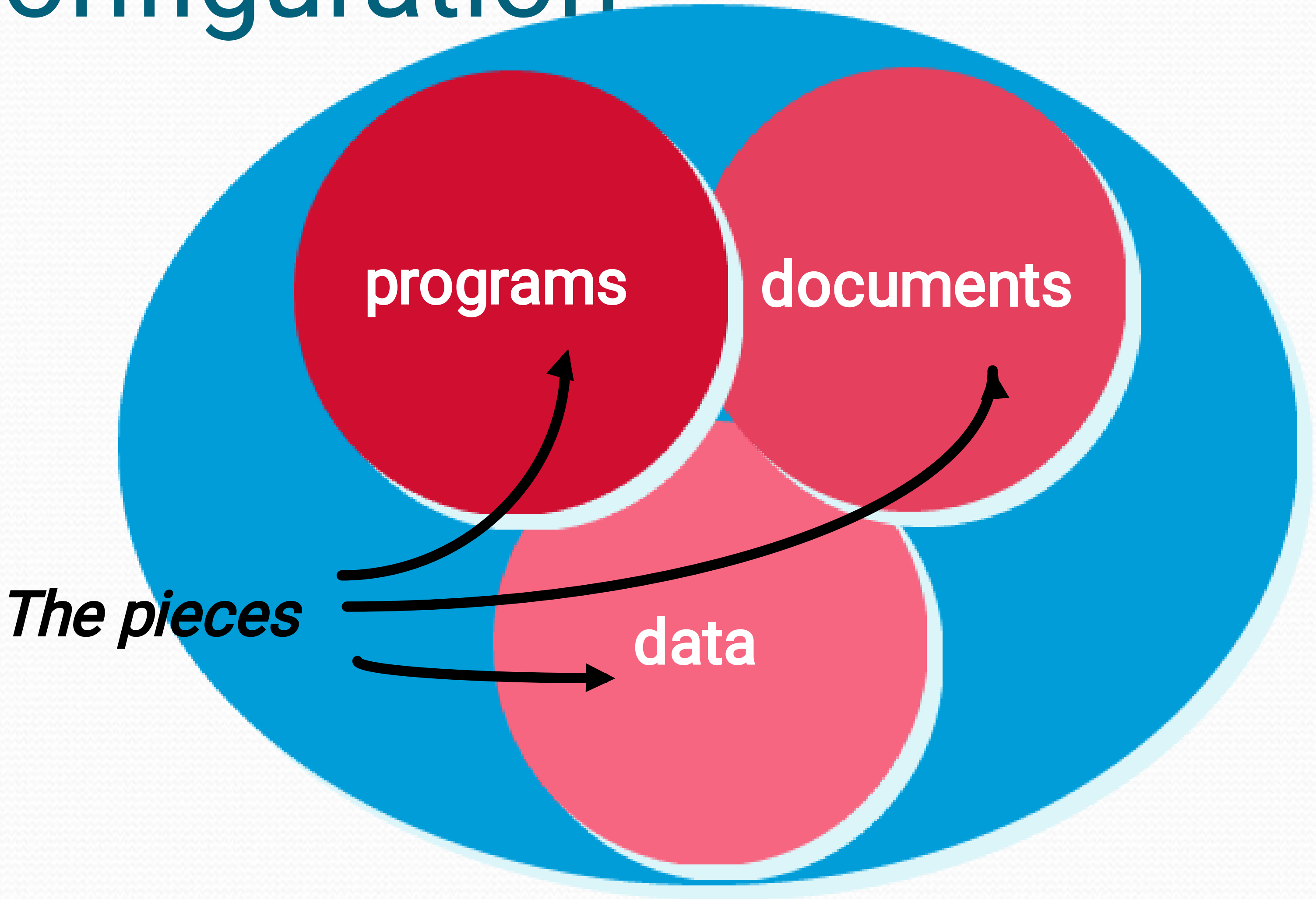
Software Configuration Management

The Software Configuration



What is Software Configuration Management ?

- Definition:
  - A set of management disciplines within the software engineering process to develop a baseline.
- Description:
  - Software Configuration Management encompasses the disciplines and techniques of initiating, evaluating and controlling change to software products during and after the software engineering process.
- Standards (approved by ANSI)
  - IEEE 828: Software Configuration Management Plans
  - IEEE 1042: Guide to Software Configuration Management

Why Software Configuration Management ?

- The problem:
  - Multiple people have to work on software that is changing
  - More than one version of the software has to be supported:
    - Released systems
    - Custom configured systems (different functionality)
    - System(s) under development
  - Software must run on different machines and operating systems

í *Need for coordination*

- Software Configuration Management
  - manages evolving software systems
  - controls the costs involved in making changes to a system

Basic concepts of Configuration Management :

Configuration Item:

A piece of software or work product which is subject to change is a configuration item

Change Request:

It is a formal report that contains the request for modification in a configuration item.

Workspace:

It is a library of promotions.

Baseline :

A collection of item versions that have been formally reviewed & agreed on, a version of configuration.

Basic concepts of Configuration Management : (contd.)

Versions and configurations:

A version identifies the state of a particular configuration item or a configuration at a well defined point in time

Promotion:

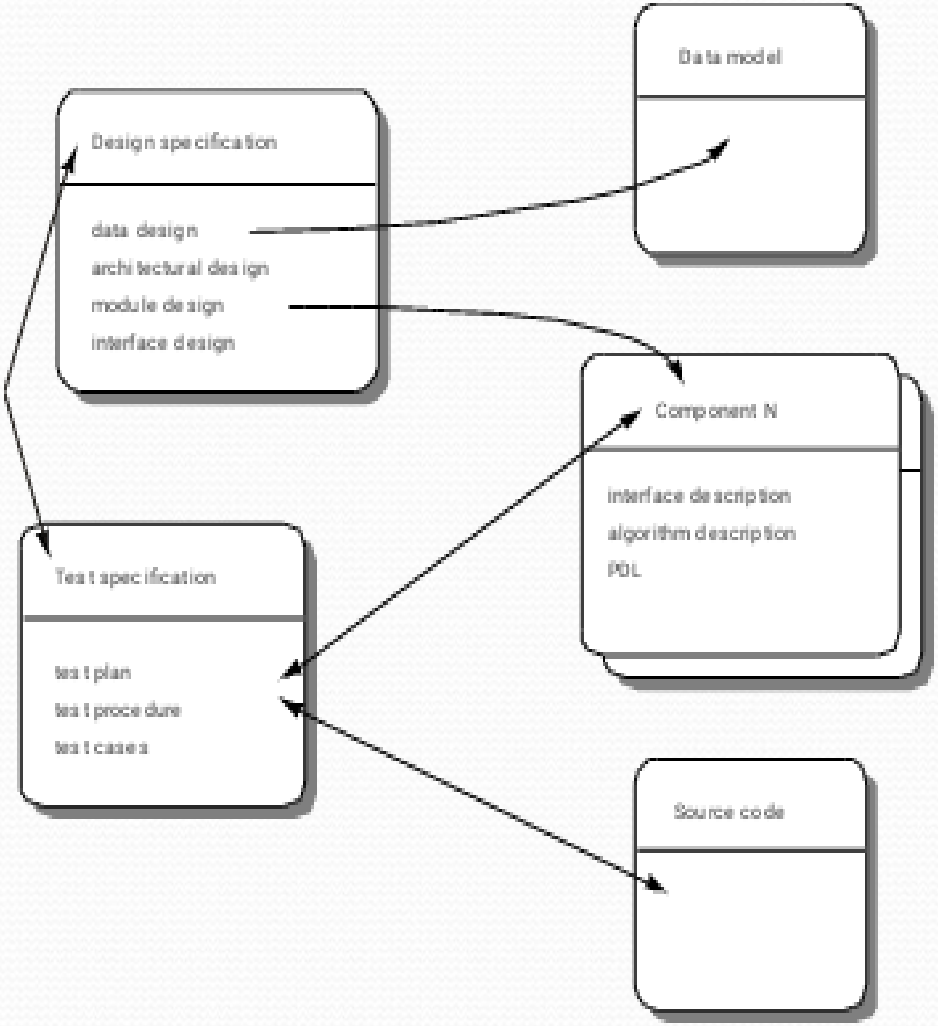
A promotion is a version of a configuration item/CM aggregate that has been available to other developers in a project.

Release:

A release is a version that has been available to the user or the client.

Repository: It stores the various releases of a CM item/ aggregate.

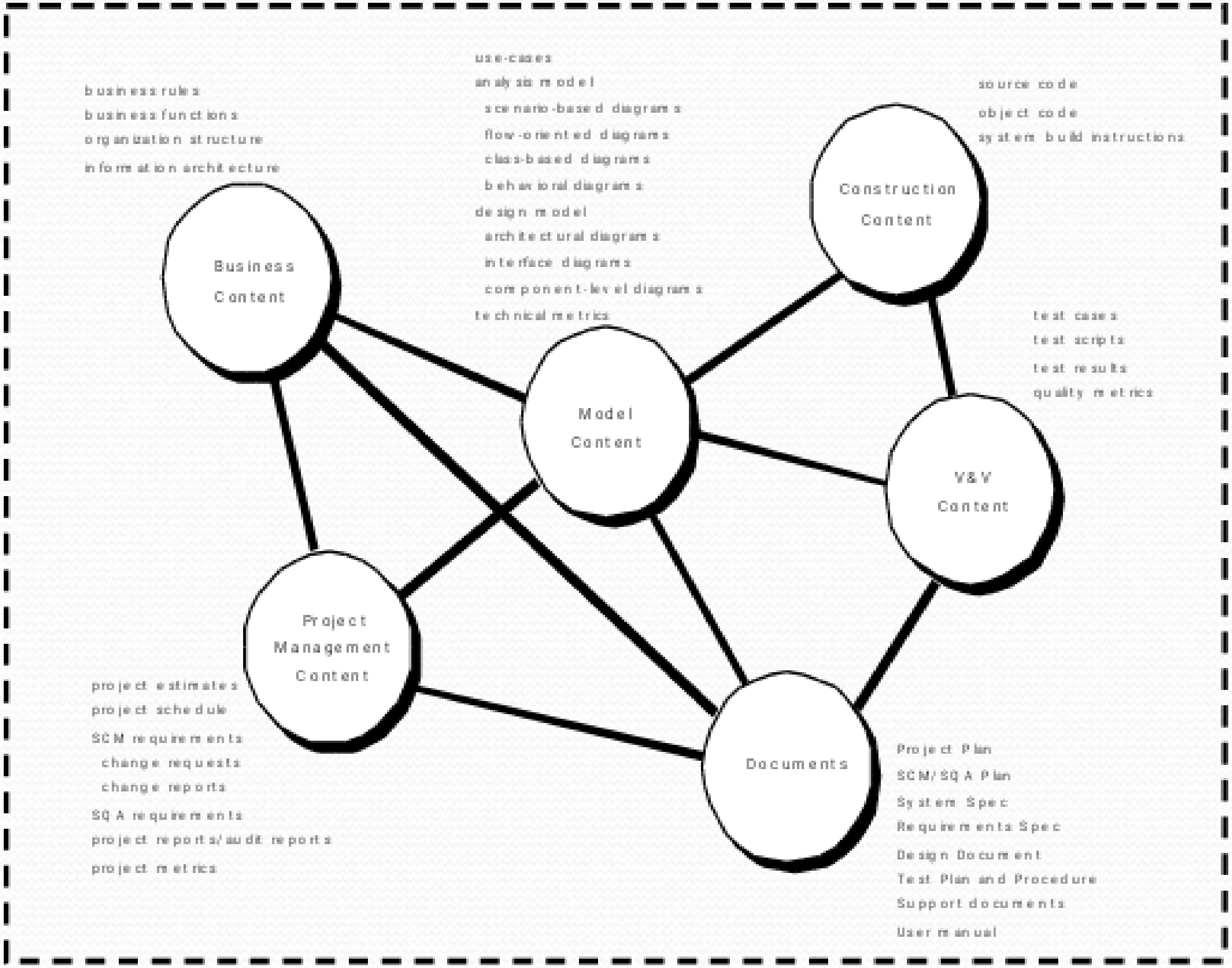
Software Configuration Objects



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

Repository Content



business rules
business functions
organization structure
information architecture

use-cases
analysis model
  scenario-based diagrams
  flow-oriented diagrams
  class-based diagrams
  behavioral diagrams
design model
  architectural diagrams
  interface diagrams
  component-level diagrams
technical metrics

source code
object code
system build instructions

test cases
test scripts
test results
quality metrics

project estimates
project schedule
SCM requirements
  change requests
  change reports
SQA requirements
project reports/audit reports
project metrics

Project Plan
SCM/SQA Plan
System Spec
Requirements Spec
Design Document
Test Plan and Procedure
Support documents
User manual

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

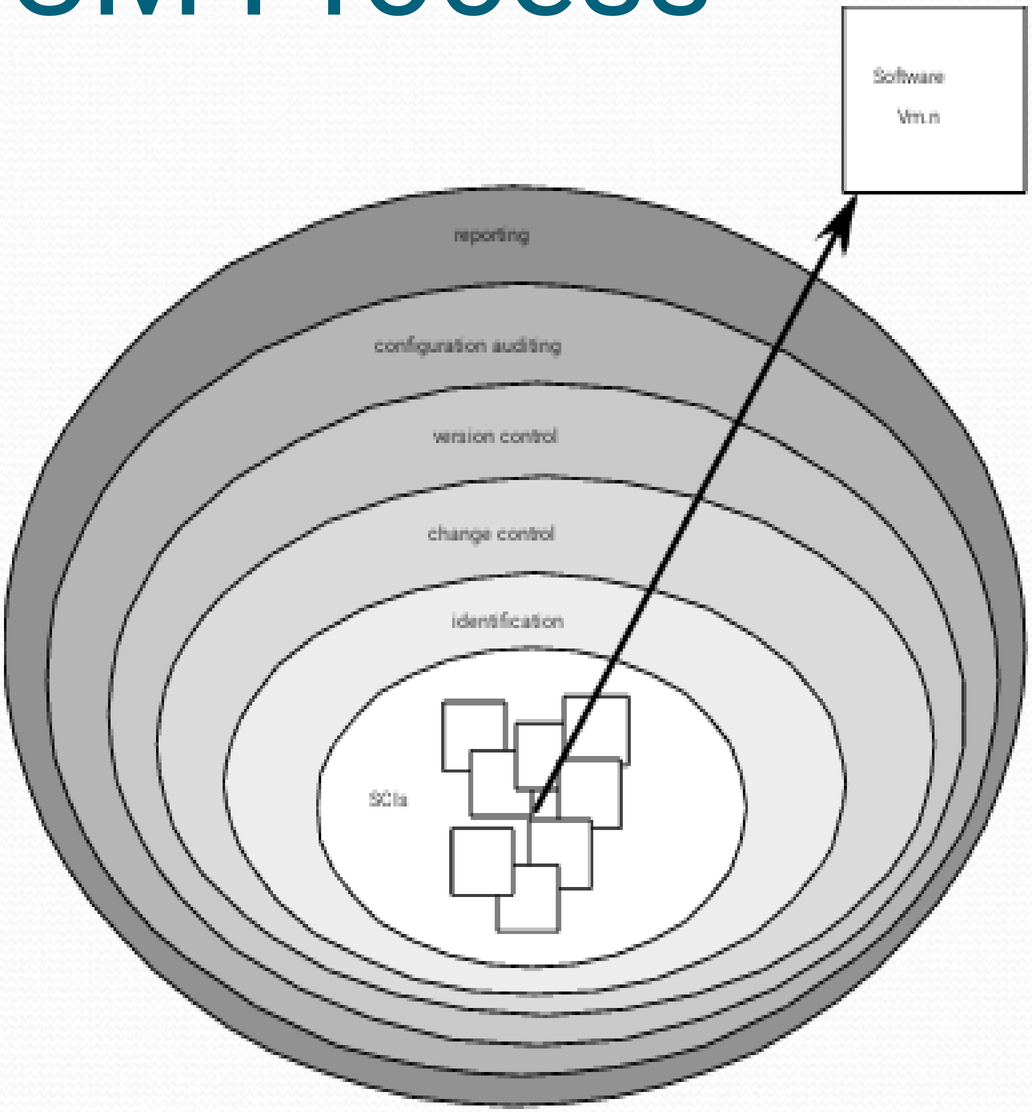
- ***Component elements*** —a set of tools coupled within a file management system (e.g., a database) that enables access to and management of each software configuration item.
- ***Process elements*** —a collection of procedures and tasks that define an effective approach to change management (and related 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

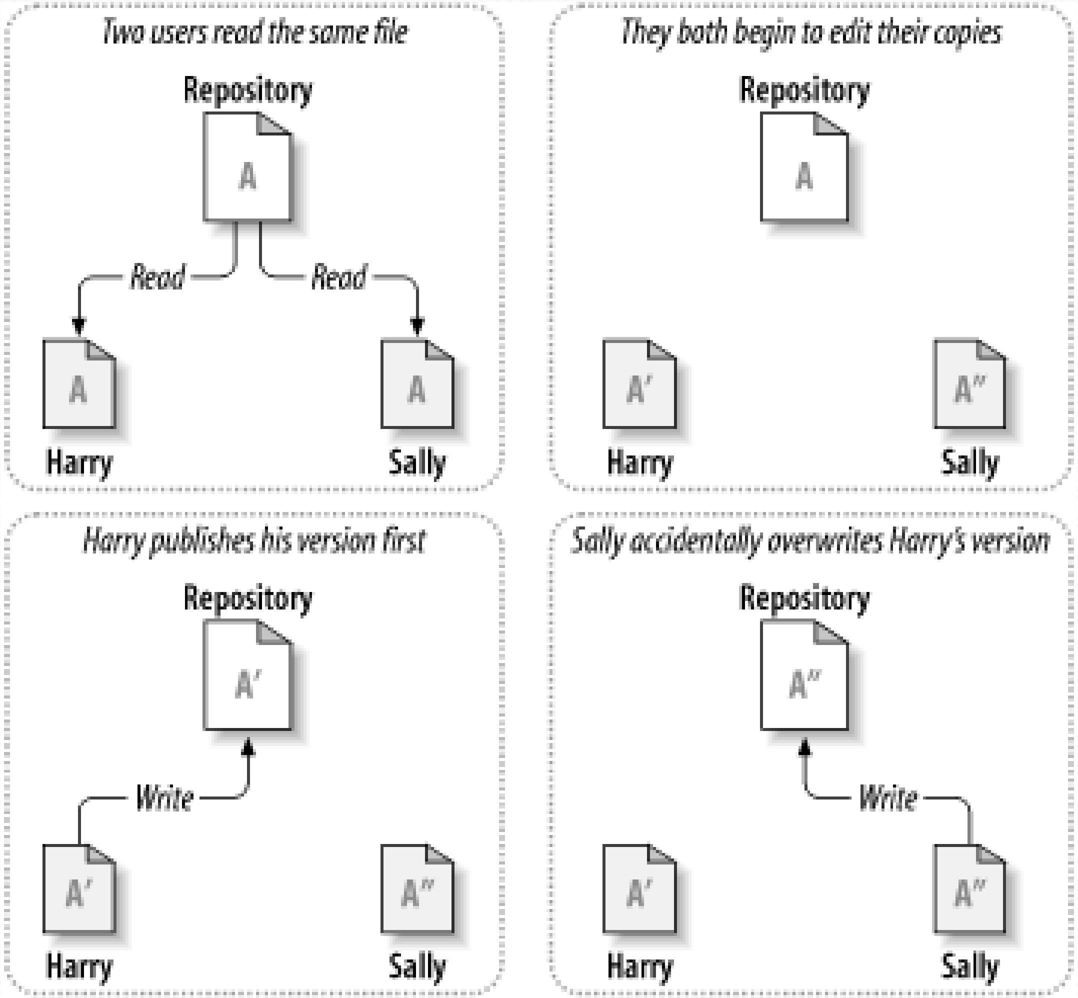


Version Control

- 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.

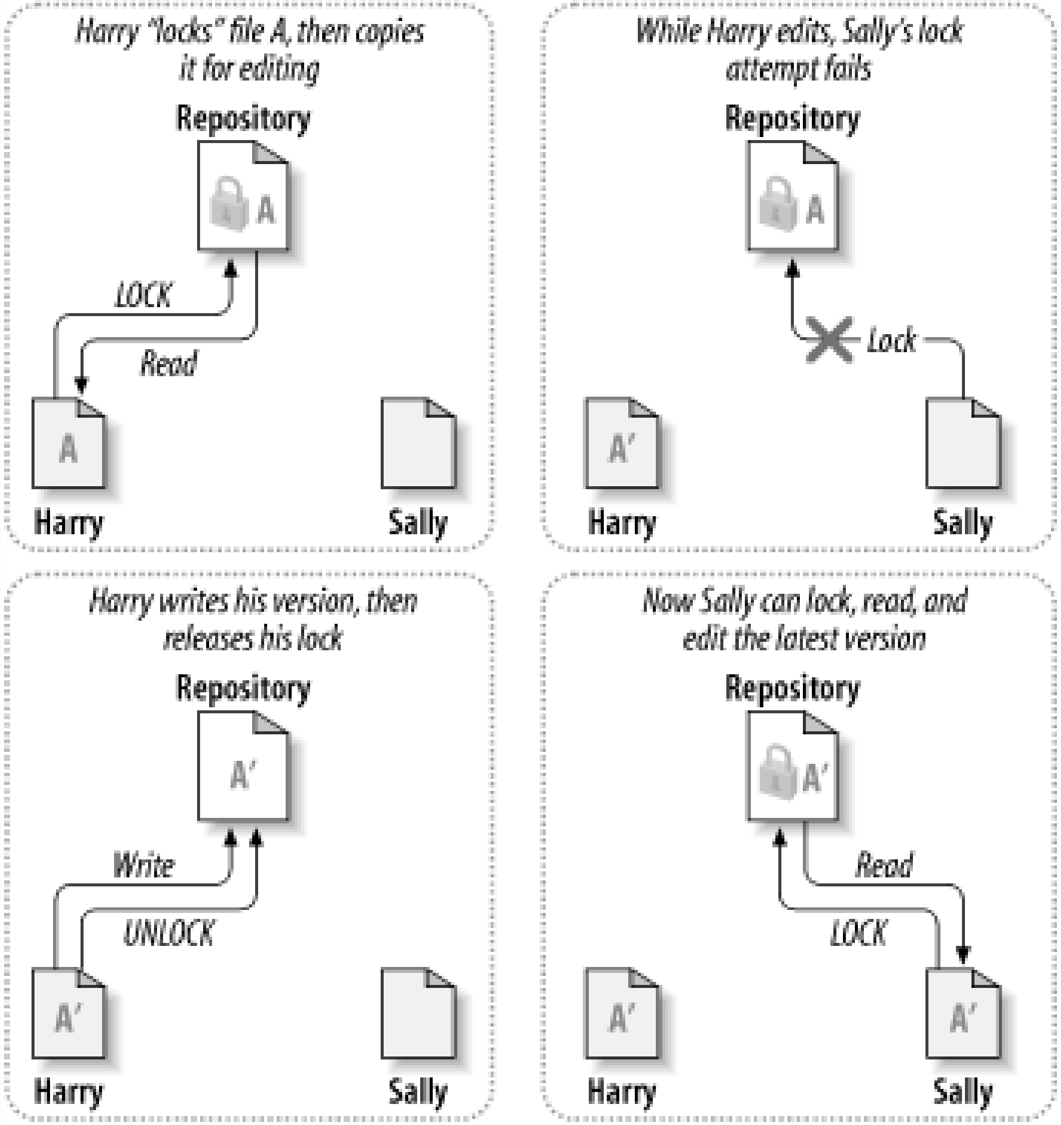
Version Control Models (1/3)

- Basic problem of collaborative work


Two users read the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Harry publishes his version first
Repository
A′
Write
A′
A″
Harry
Sally
Sally accidentally overwrites Harry's version
Repository
A″
Write
A′
A″
Harry
Sally

Version Control Models (2/3)

Model 1-Pessimistic: lock-modify-unlock

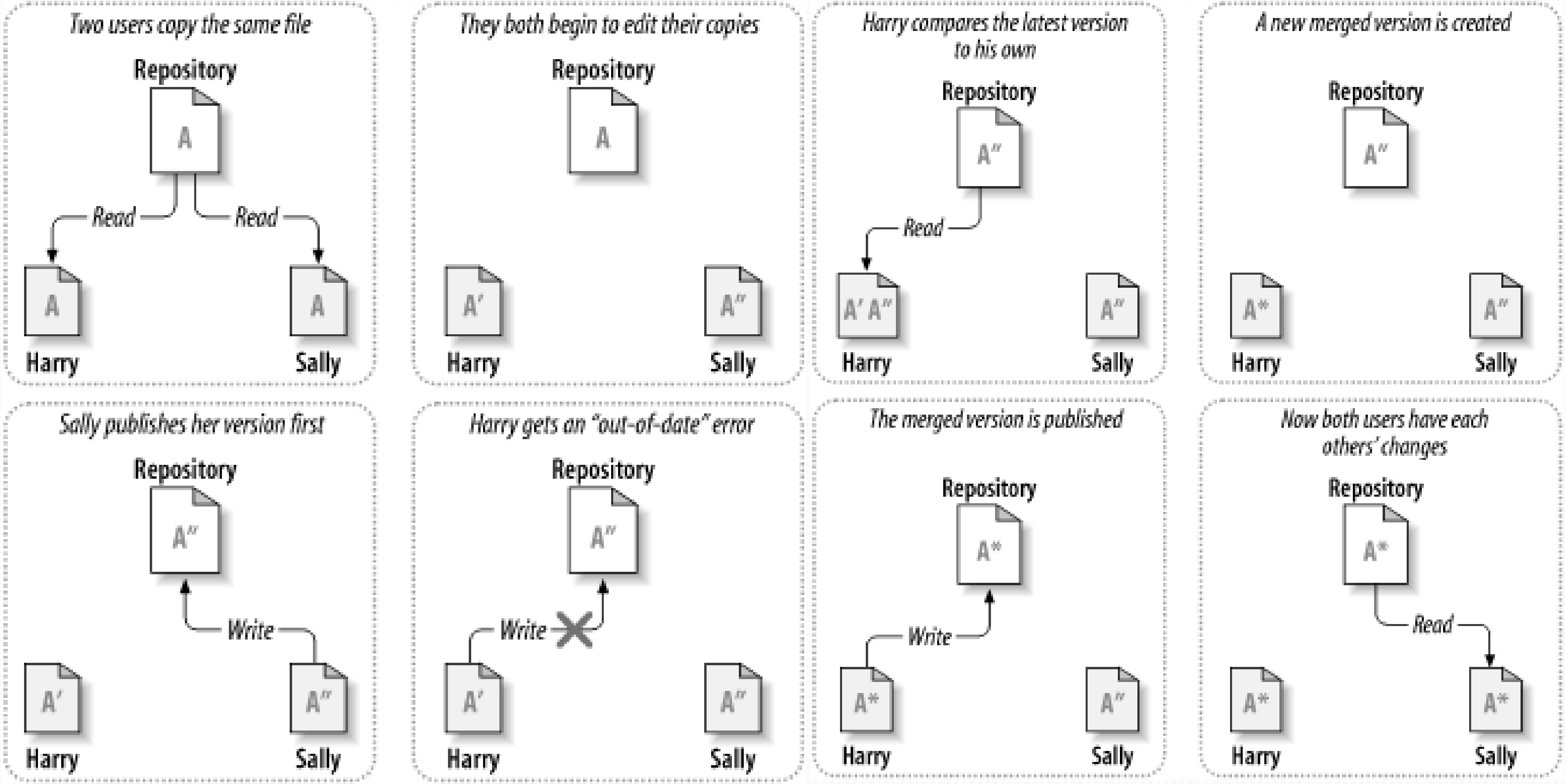

Harry "locks" file A, then copies it for editing
Repository
A
LOCK
Read
A
Harry
Sally
While Harry edits, Sally's lock attempt fails
Repository
A
Lock
A′
Harry
Sally
Harry writes his version, then releases his lock
Repository
A′
Write
UNLOCK
A′
Harry
Sally
Now Sally can lock, read, and edit the latest version
Repository
A′
Read
LOCK
A′
Harry
A′
Sally

Problems:

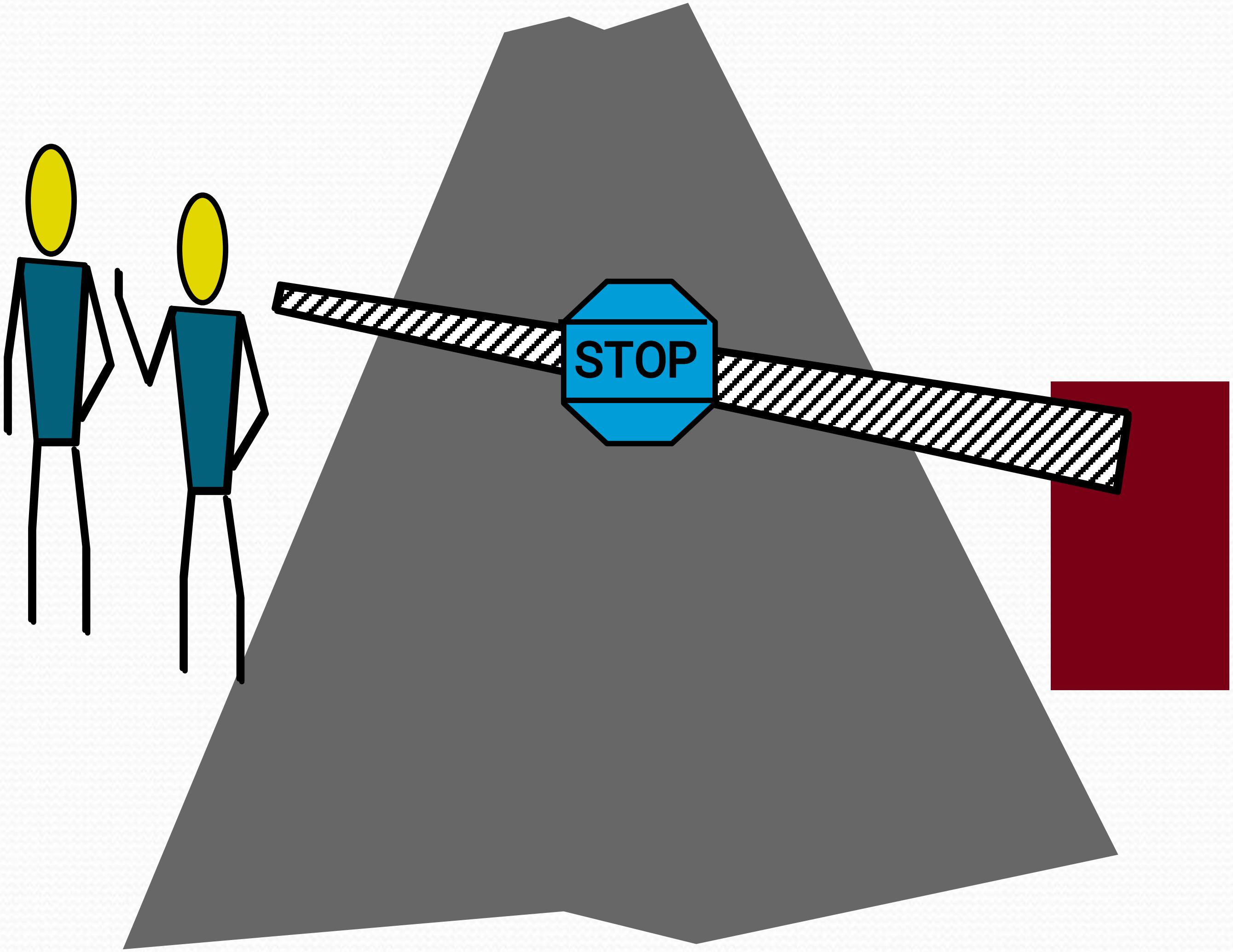
- Forget to unlock
- Parallel work not possible
- Deadlock

Version Control Models (3/3)

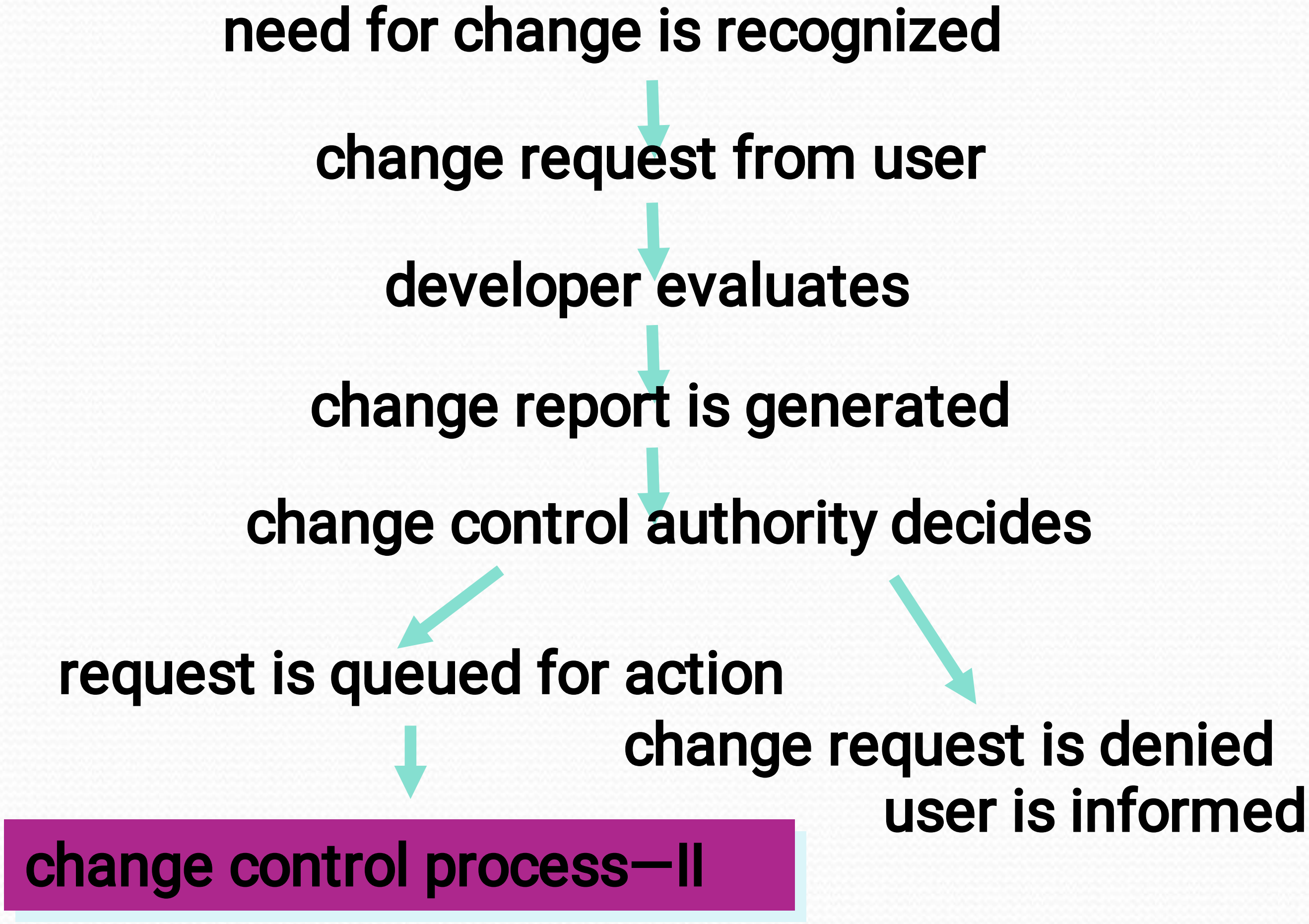
Model 2-Optimistic: copy-modify-merge


Two users copy the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Harry compares the latest version to his own
Repository
A″
Read
A′ A″
A″
Harry
Sally
A new merged version is created
Repository
A″
A*
A″
Harry
Sally
Sally publishes her version first
Repository
A″
Write
A′
A″
Harry
Sally
Harry gets an "out-of-date" error
Repository
A″
Write
A′
A″
Harry
Sally
The merged version is published
Repository
A*
Write
A*
A″
Harry
Sally
Now both users have each others' changes
Repository
A*
Read
A*
A*
Harry
Sally

Change Control



Change Control Process—I



need for change is recognized

change request from user

developer evaluates

change report is generated

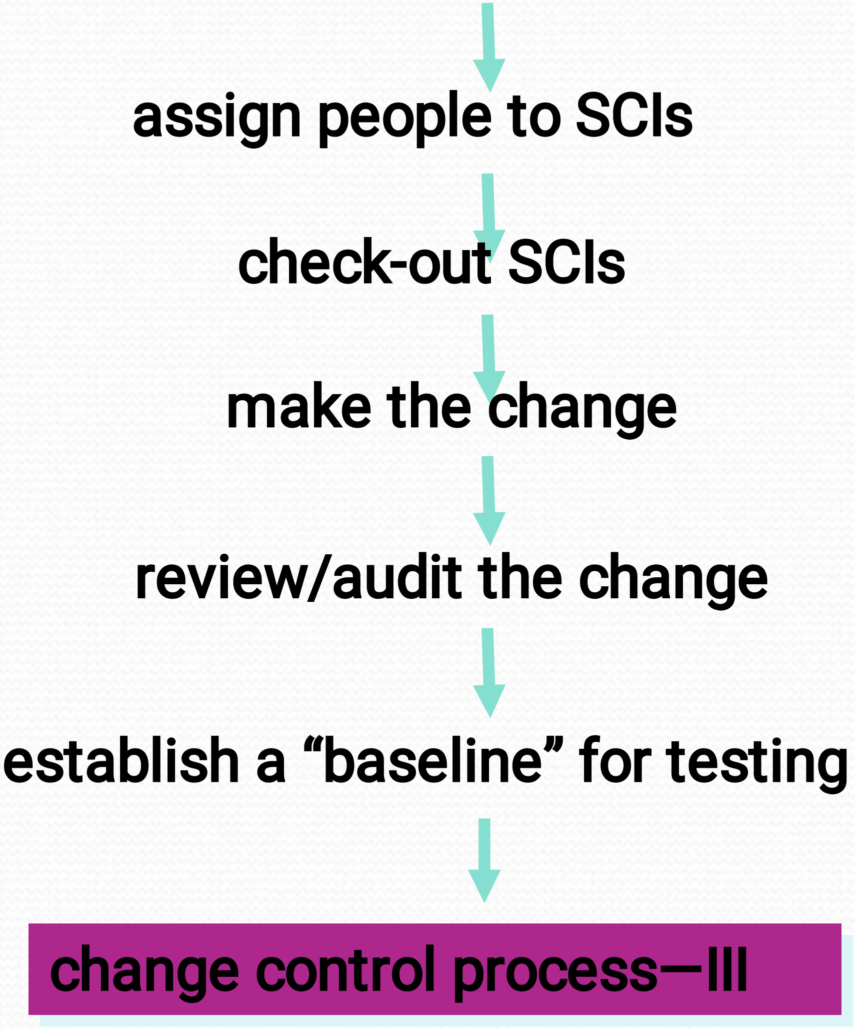
change control authority decides

request is queued for action

change request is denied
user is informed

change control process—II

Change Control Process-II



assign people to SCIs

check-out SCIs

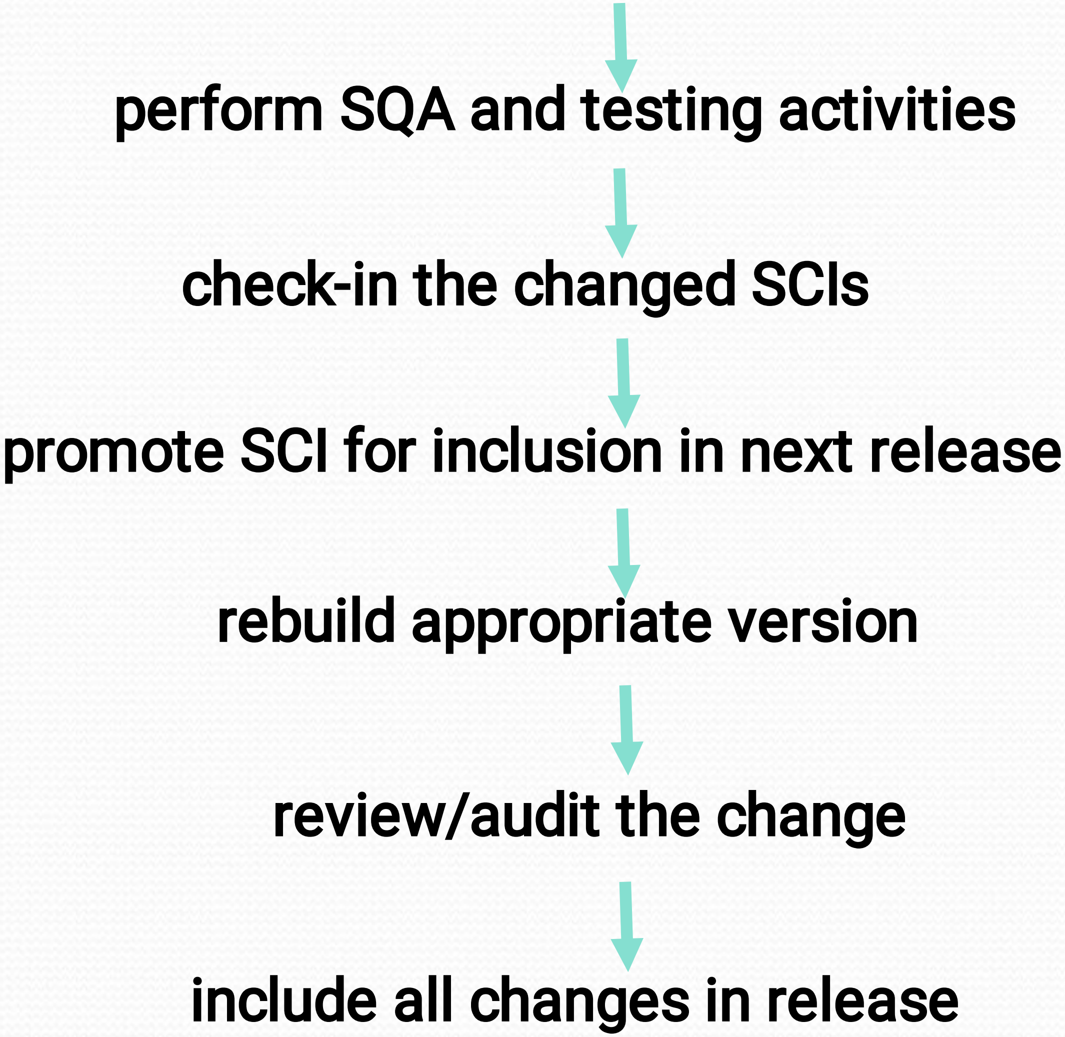
make the change

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

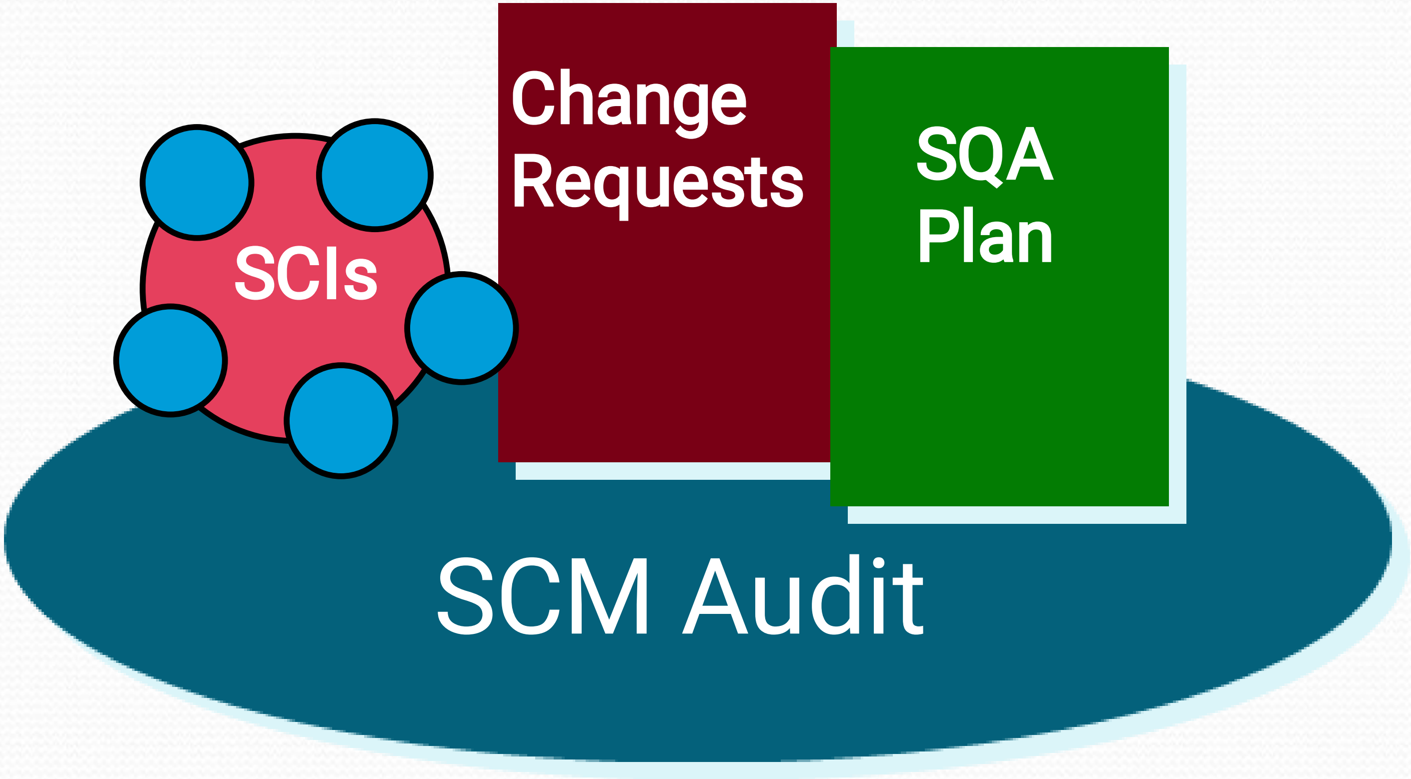
promote SCI for inclusion in next release

rebuild appropriate version

review/audit the change

include all changes in release

Auditing

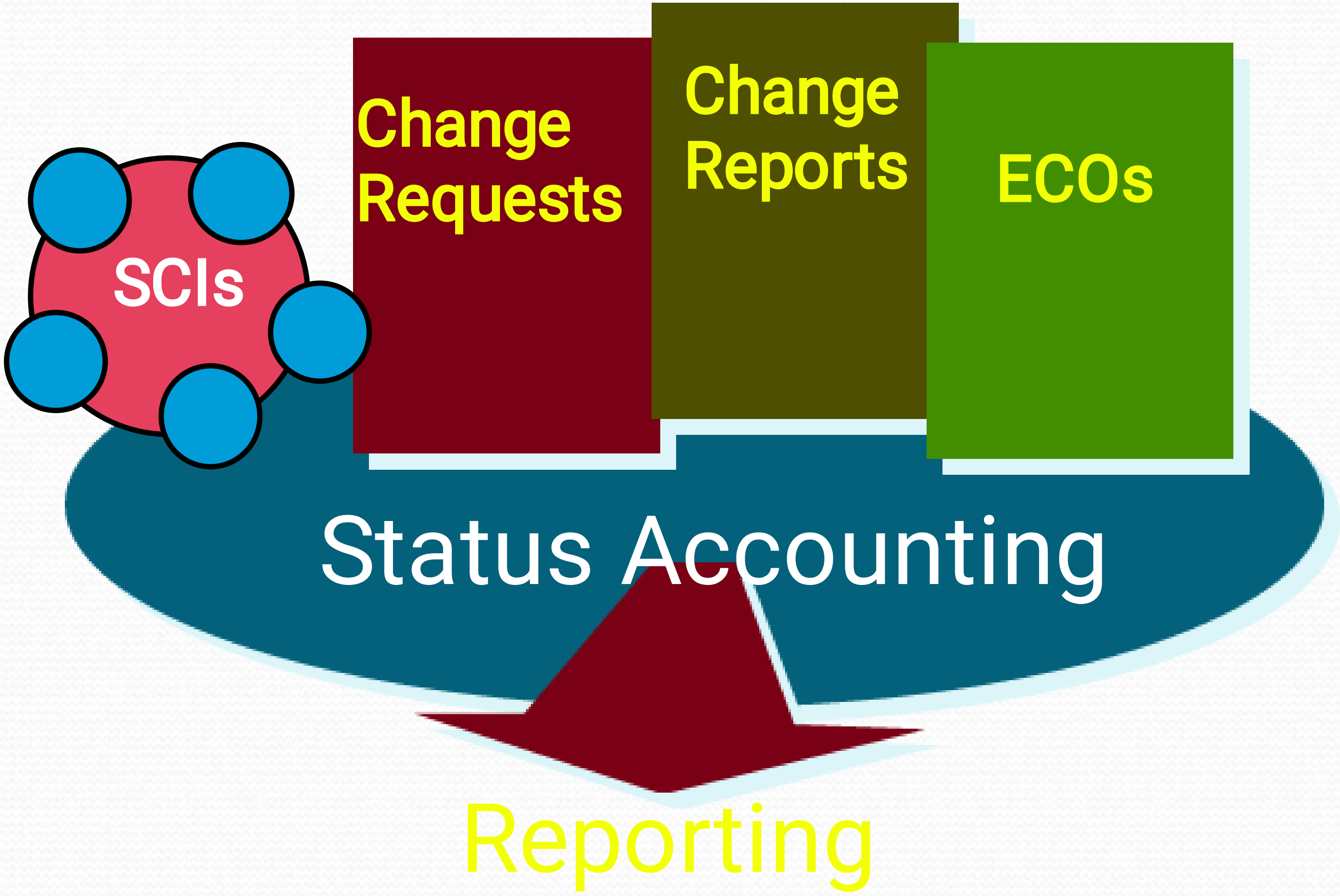


Configuration Audit

- Independent review or examination to assess if a product or process is in compliance with specification, standards, contractual agreement, or other criteria

- Examples
  - Verifies that CIs are tested to satisfy functional requirements
  - Verifies that baseline contains necessary and correct CI versions
  - Ensures that changes made to a baseline comply with the configuration status reports

Status Accounting



Status Accounting

- Administrative tracking and reporting of CIs in CM system

- Examples
  - Status of proposed changes
  - Status of approved changes
  - Progress of current version, on or behind schedule
  - Estimate of resources to finish one task
  - bugs identified by configuration audit

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?