Chapter 29

- **Software Configuration Management**

Slide Set to accompany

Software Engineering: A Practitioner's Approach

by Roger S. Pressman

Configuration Management (CM)

- = Software Configuration Management (SCM)

- Configuration: [Merriam-Webster Dictionary]

  a) Relative arrangement of parts or elements:

  b) Something (as a figure, contour, pattern, or apparatus) that results from a particular arrangement of parts or components

  c) The stable structural makeup of a chemical compound especially with reference to the space relations of the constituent atoms"

- Software Configuration

  "The items that comprise all information produced as part of the software process"

CM – What?

- CM: Managing software configuration

  + Managing changing software systems

- Why ?
  - New versions of software systems are created as they change
    - For different machines/OS
    - Offering different functionality or tailored for particular user requirements

Consequences of poor CM practices [Babich 86]

- The latest version of source code cannot be found.
- No one knows which modules comprise the software system delivered to the customer.
- A difficult defect fixed at great expense suddenly reappears.
- A developed and tested feature is mysteriously missing.
- A fully tested program suddenly does not work.
  - **Avalon OS**: Microsoft, Codename: Longhorn, 2003 ~ 2005
  - Back to Server 2003 codebase !
  - ☛ Loss of several thousand Person-Years effort
- Programmers are working on the wrong version of the code.
- There is no traceability between the software requirements, documentations, and code.

Table of Contents

29.1 Software Configuration Management

29.1.1 An SCM Scenario

29.1.2 Elements of a Configuration Management System

29.1.3 Baselines

29.1.4 Software Configuration Items

29.1.5 Management of Dependencies and Changes

29.1.1 An SCM Scenario

- **Project manager**: in charge of a software group
- **Configuration manager**: in charge of the CM procedures and policies
- **Software engineers**: responsible for developing and maintaining the software product
- **Customer**: uses the product

29.1.2 Elements of a Configuration Management System

- Component elements — Tools coupled within a file management system (e.g., a database) that enables management of each software configuration item.
- Process elements — Procedures and tasks for all constituencies involved in the management, engineering and use of computer software.
- Construction elements — Tools that automate the construction of software by ensuring that the proper set of components (i.e., the correct version) have been assembled.
- Human elements — The software team uses a set of tools and processes (encompassing other CM elements)

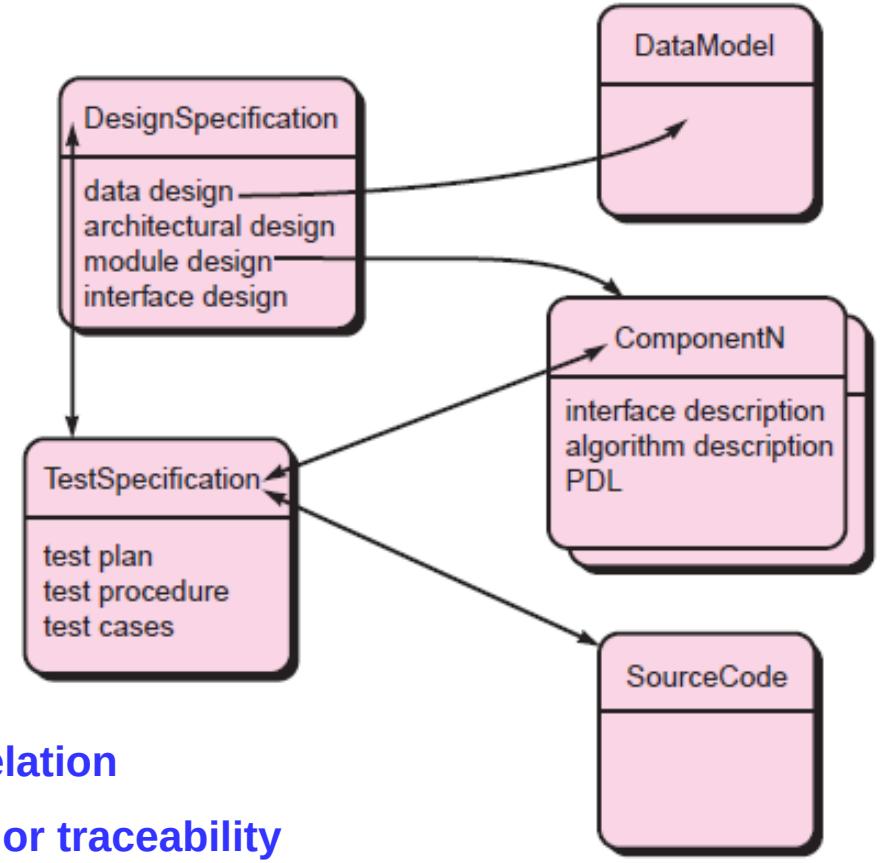
29.1.3 Baselines

- 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."
- Baseline: A milestone in the development of software
  (Milestone: an end-point of a process activity.)
  - Delivered as one or more software configuration items
  - Approval of these SCIs is obtained through a formal technical review

29.1.4 SCI

SCI: Software Configuration Items

Which are the basic objects and which are the aggregate objects?



: Compositional relation

: Interrelationship or traceability

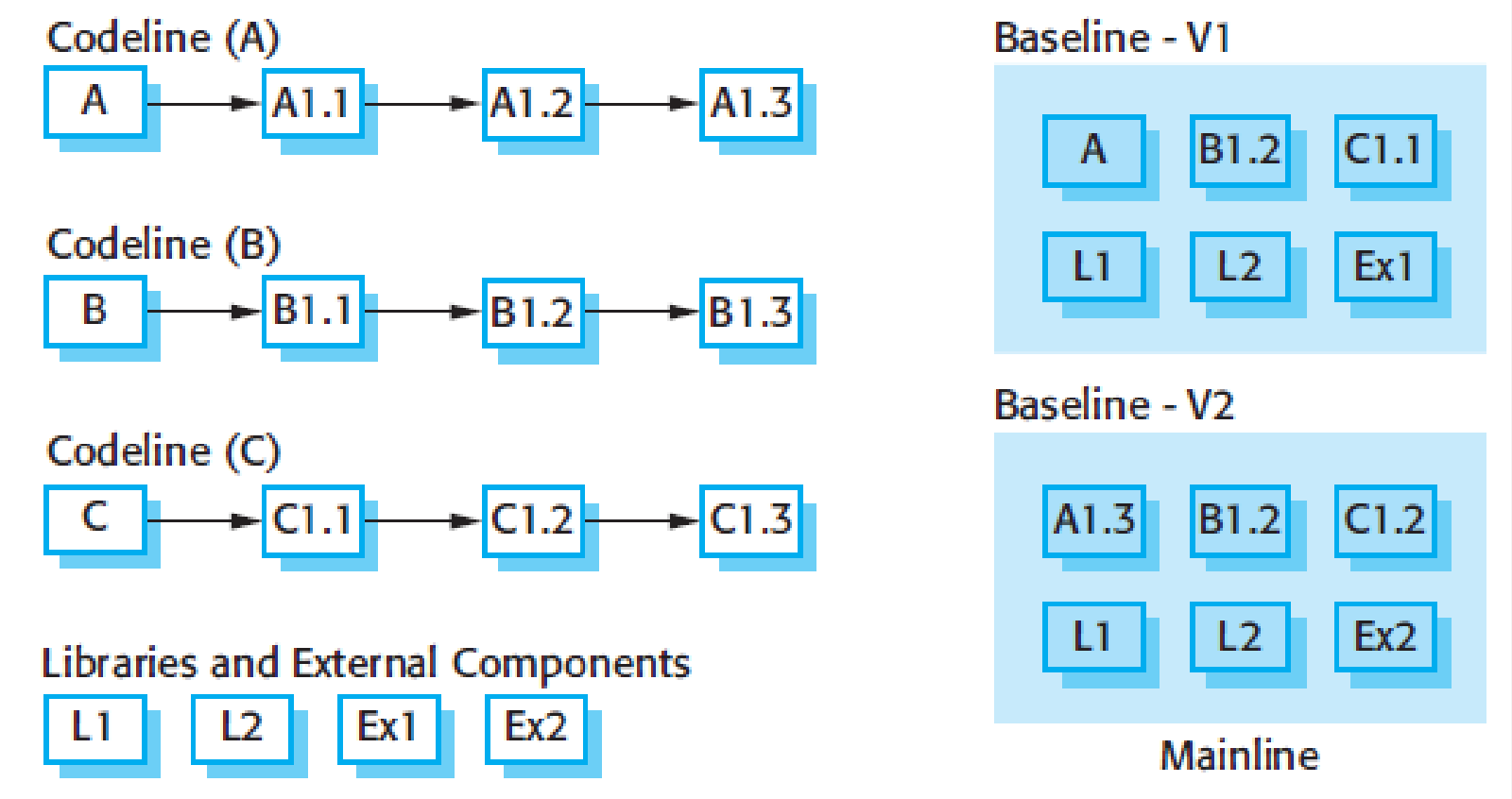
Codelines and Baselines (1/4)

- Codeline: a sequence of versions of source code with later versions in the sequence derived from earlier versions.
  - Normally apply to components of systems where each component has different versions

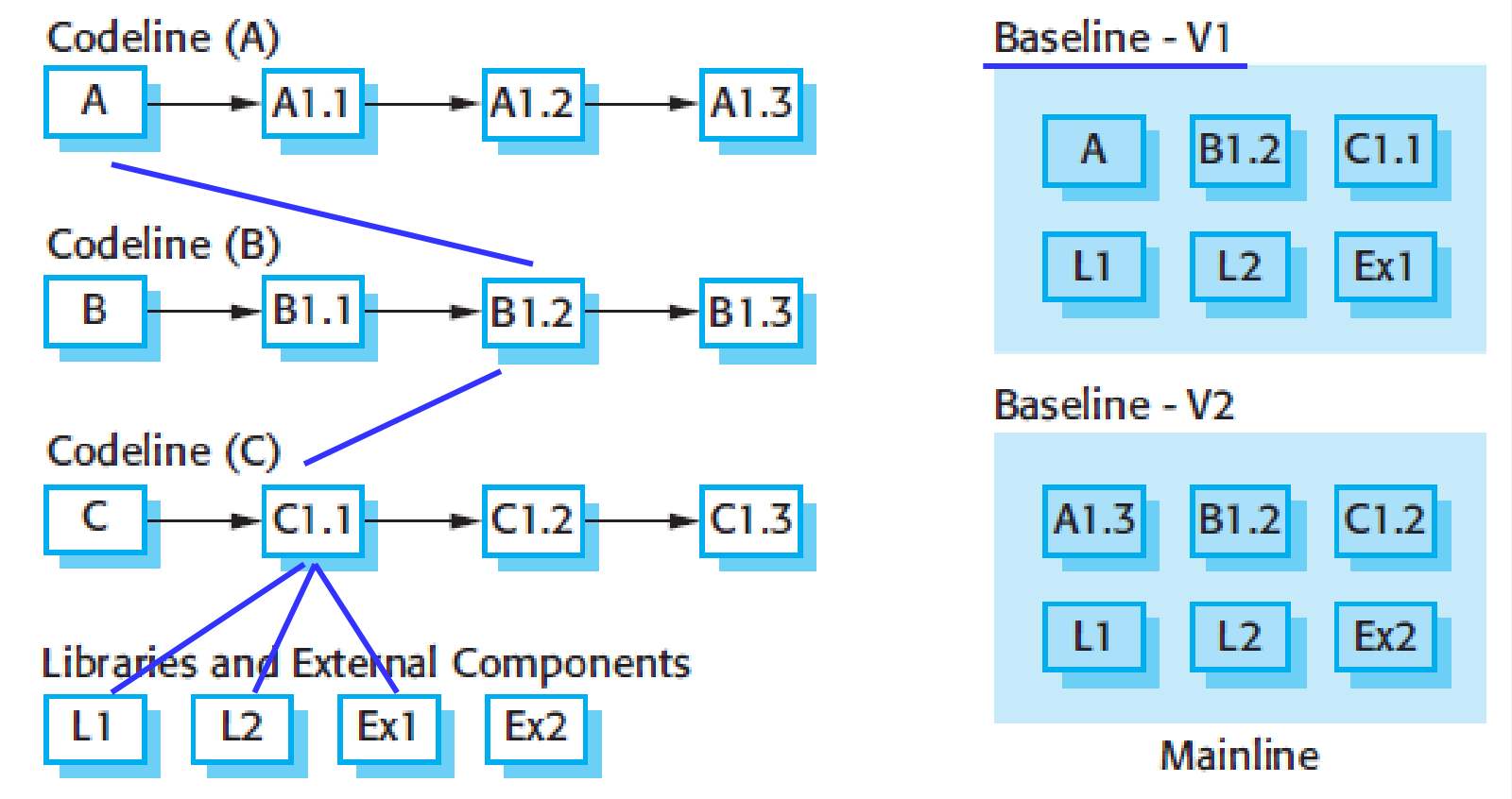
- Baseline: an *approved* snapshot of the system at appropriate points in the development life cycle.
  - requirements specification
  - design specification
  - a partial system
  - a product formally reviewed and agreed upon

- System Baseline vs. Code Baseline (See next slide)

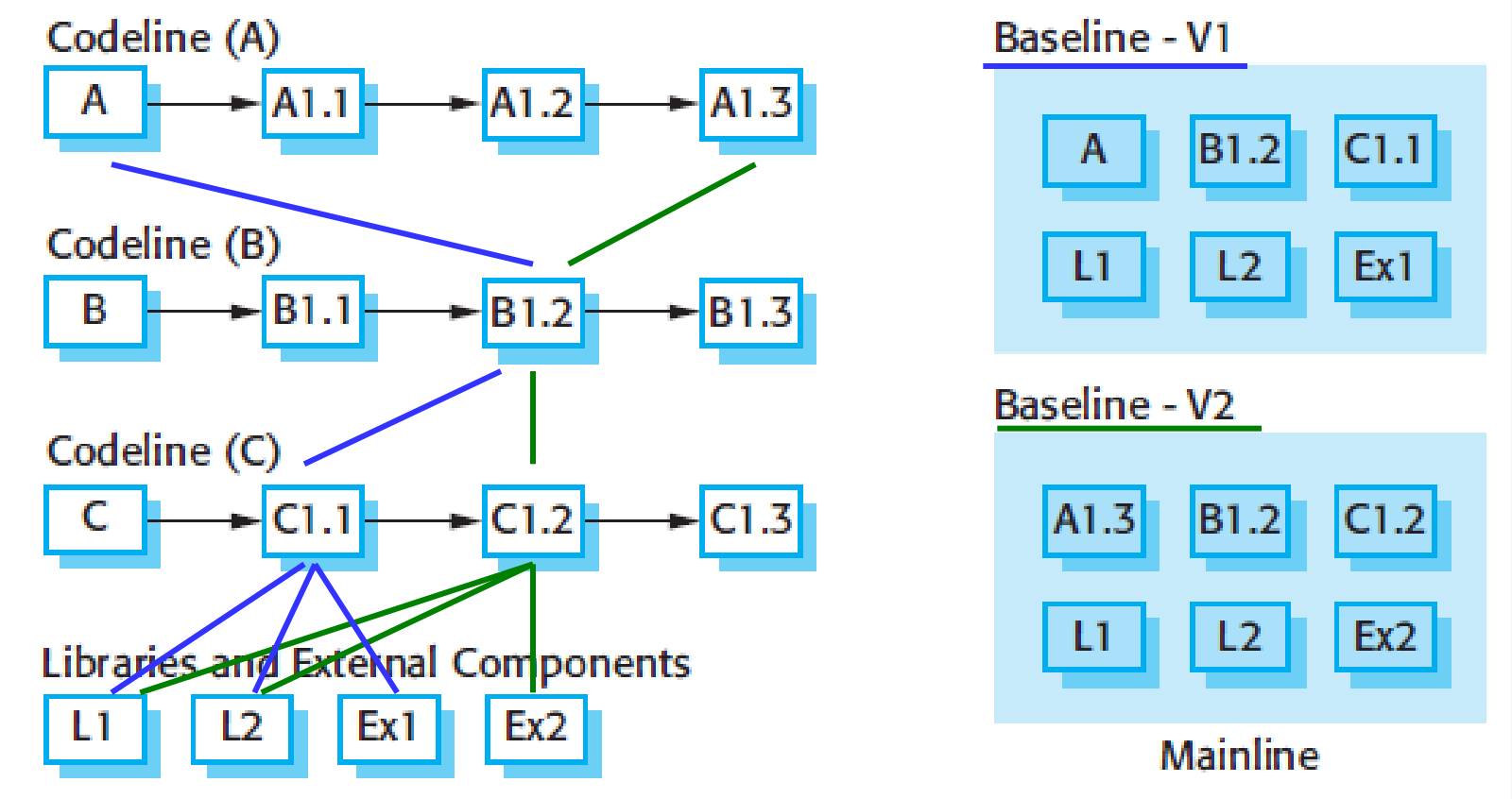
Codelines and (Code) Baselines (2/4)



Codelines and Baselines (3/4)

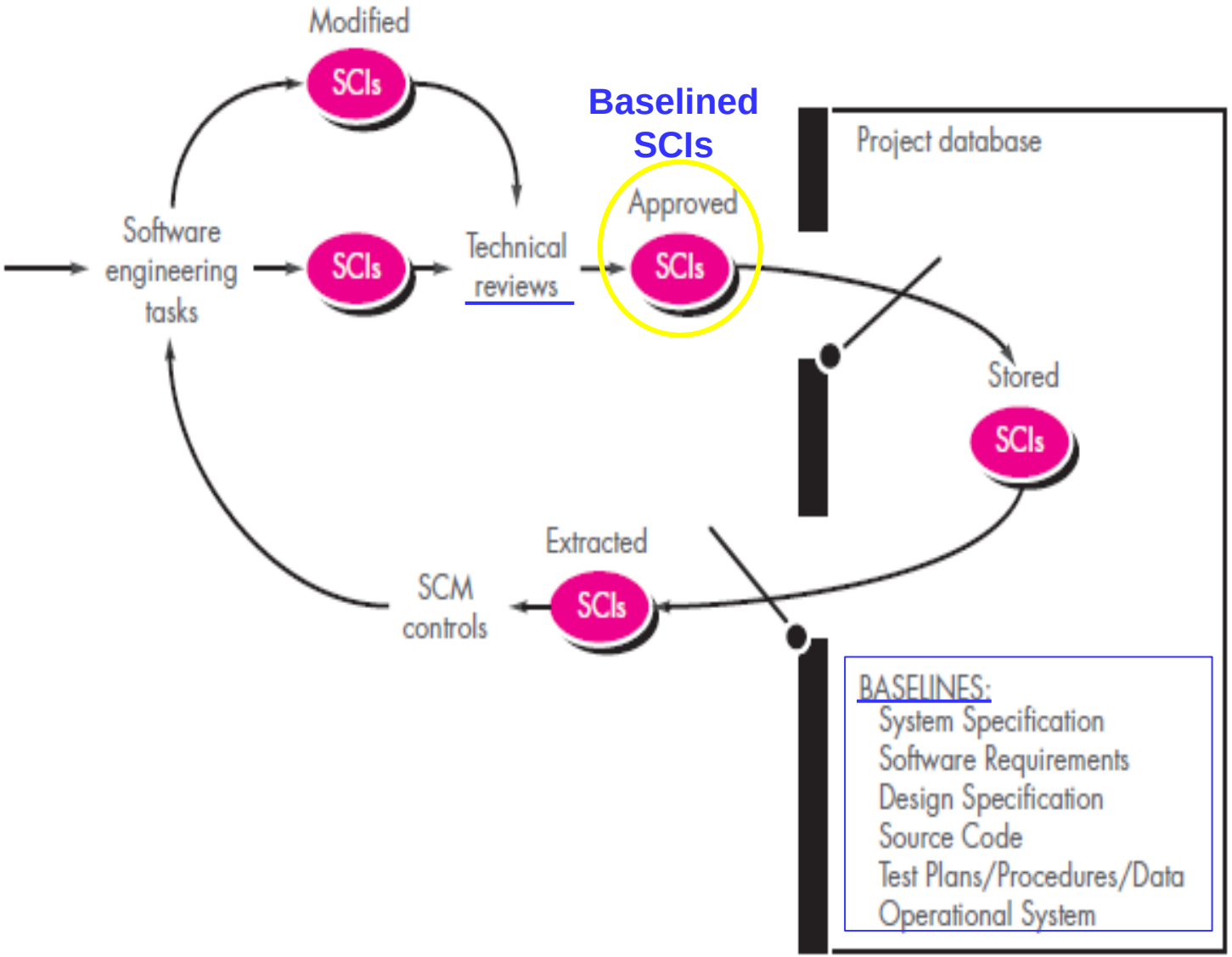


Codelines and Baselines (4/4)



System Baseline or Code Baseline ?

Baselined SCIs and the project database



System Baseline or Code Baseline ?

29.2 The SCM Repository

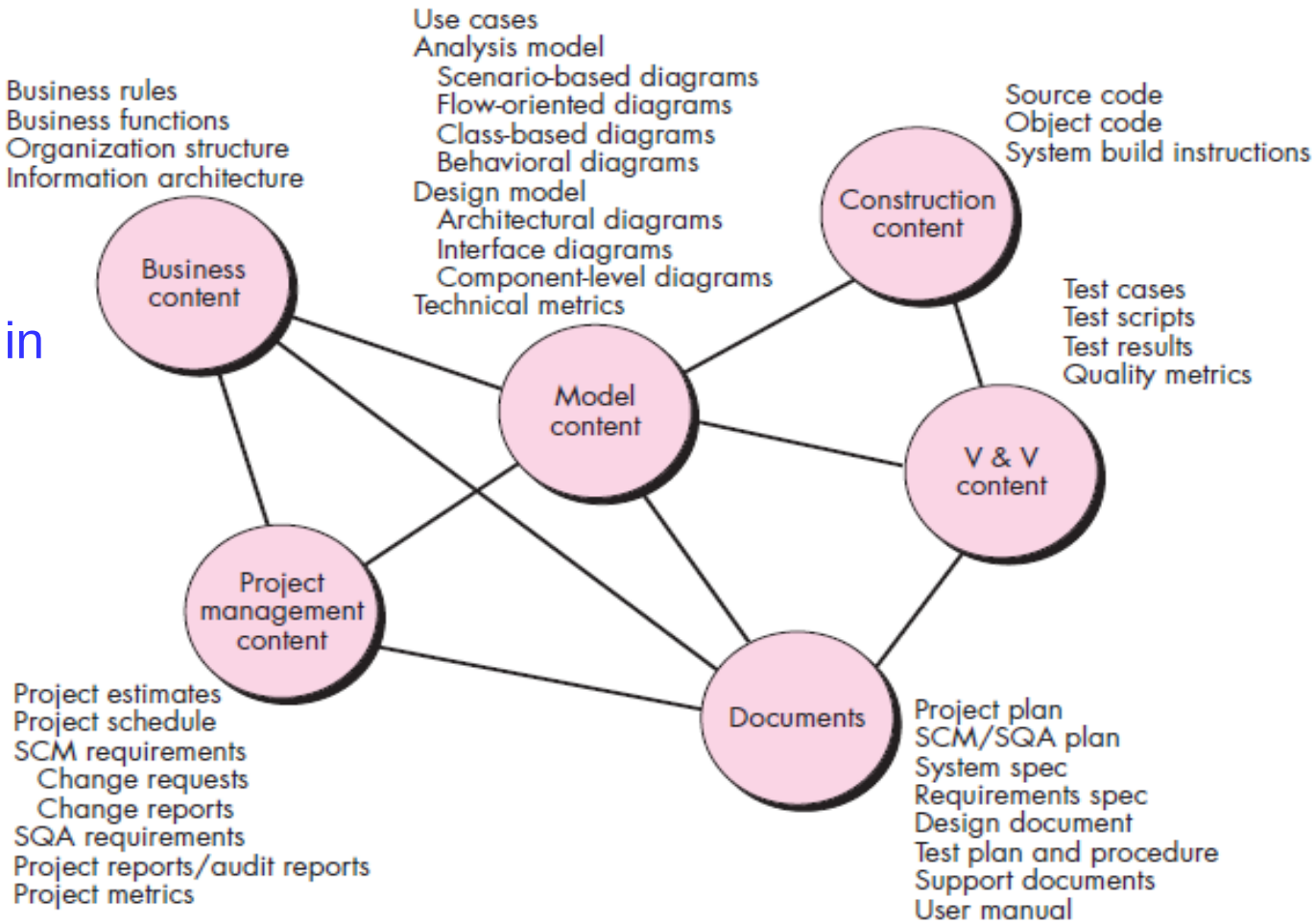
- The SCM repository
  - The set of mechanisms and data structures that
    - allow a software team to manage change in an effective manner

- The Role of the Repository [For89]:
  - Data integrity
  - Information sharing
  - Tool integration
  - Data integration
  - Methodology enforcement
  - Document standardization

29.2.1 General Features and Content

Content of the repository

Kinds of things that are stored in the repository



29.2.2 SCM Features (supported by the Repository)

- **Versioning**

  - Saves all of the versions

- **Dependency tracking and change management**

  -The repository manages a wide variety of relationships among the data elements stored in it.

- **Requirements tracing**

  - 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
  - Link management keeps track of interdependencies.

- **Audit trails**

  - Establishes information about when, why, and by whom changes are made.

29.3 The SCM Process

29.3.1 Identification of Objects in the Software Configuration

29.3.2 Version Control

29.3.3 Change Control

29.3.4 Impact Management

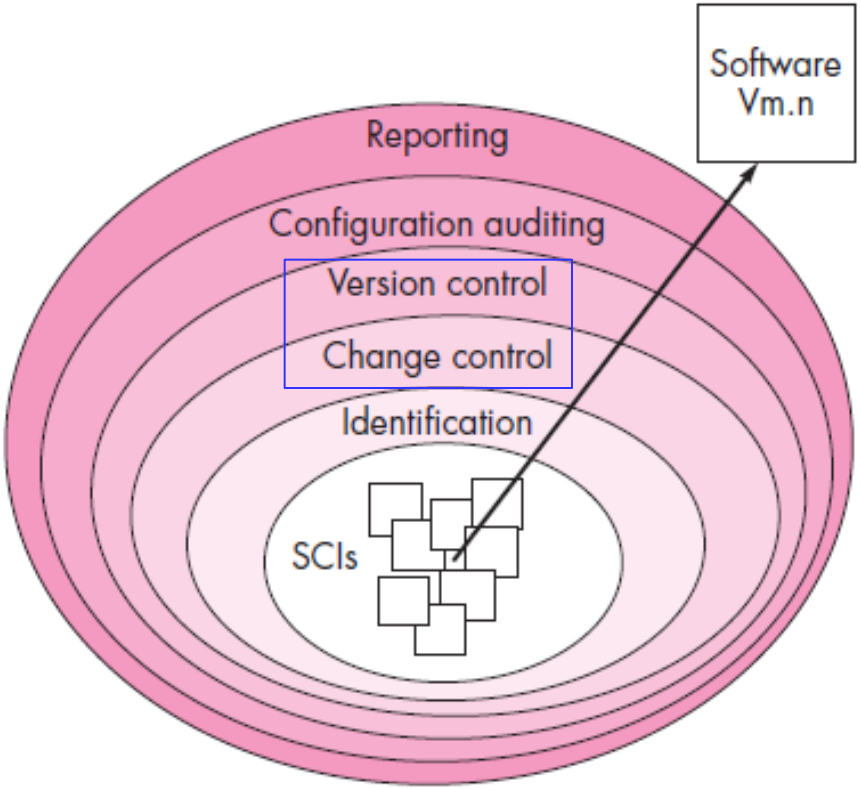
29.3.5 Configuration Audit

29.3.6 Status Reporting

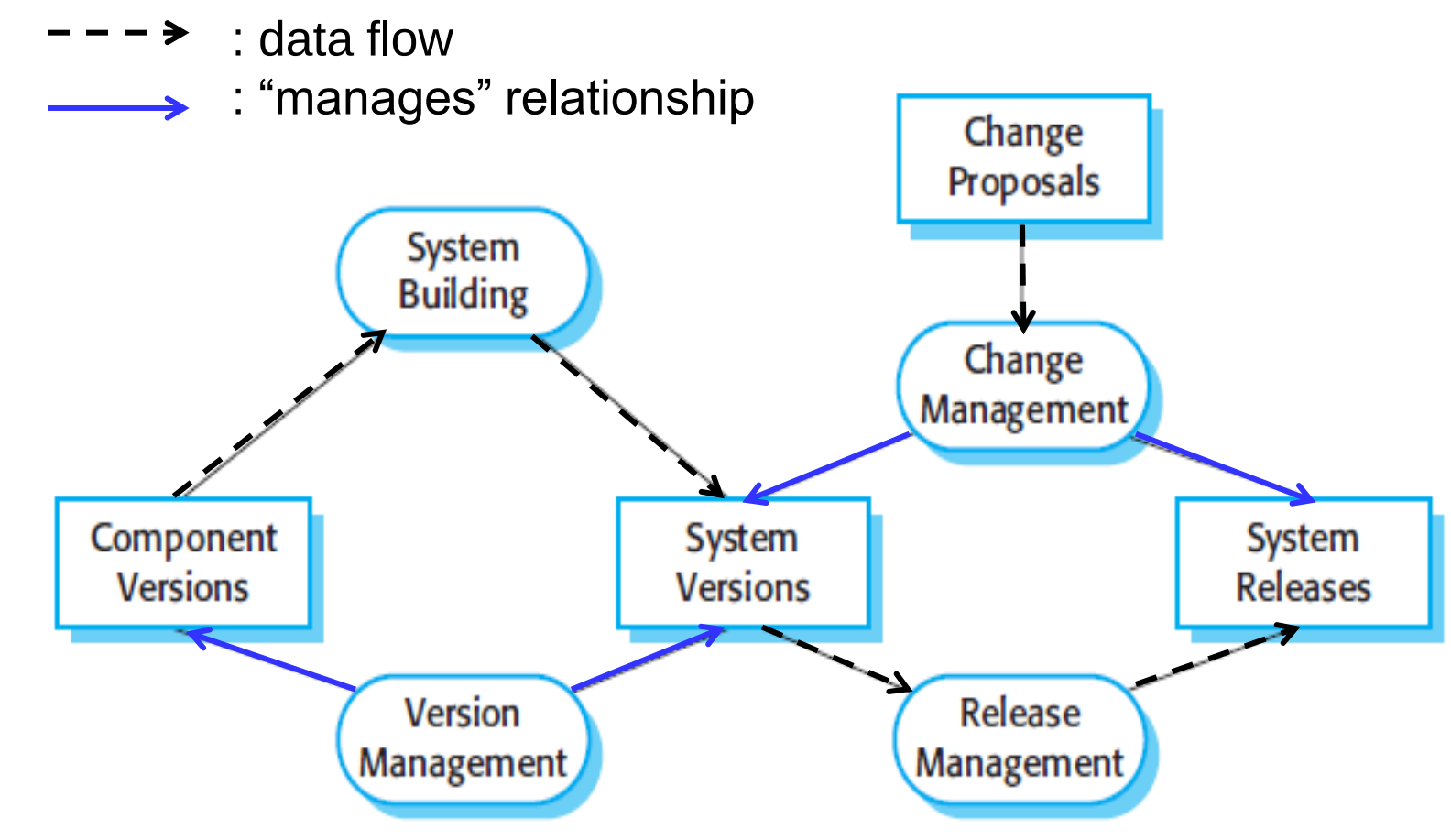
The SCM Process addresses …

- How does a software team identify the elements of a software configuration?
- How does an organization manage the existing versions of a program (and its documentation)?
- How does an organization control changes before and after software is released?
- Who has responsibility for approving and ranking changes?
- How can we ensure that changes have been made properly?
- What mechanism is used to inform others of changes that are made?

Layers of the SCM process



CM Activities (1/2)



CM Activities (2/2)

- System building

  - The process of assembling program components, data and libraries, then compiling them to create an executable system.

- Change management

  - Keeping track of requests for changes to the software from customers and developers, working out the costs and impact of changes, and deciding the changes should be implemented.

- Version management (VM)

  - Keeping track of the multiple versions of system components and ensuring that changes made to components by different developers do not interfere with each other.

- Release management (RM)

  - Preparing software for external release and keeping track of the system versions that have been released for customer use.

29.3.1 Identification of Objects in the Software Configuration

- To control and manage SCIs, each should be separately named and then organized.

- Two types of SCIs:
  - basic objects
  - aggregate objects

29.3.2 Version Control

- Uses **procedures** and **tools**

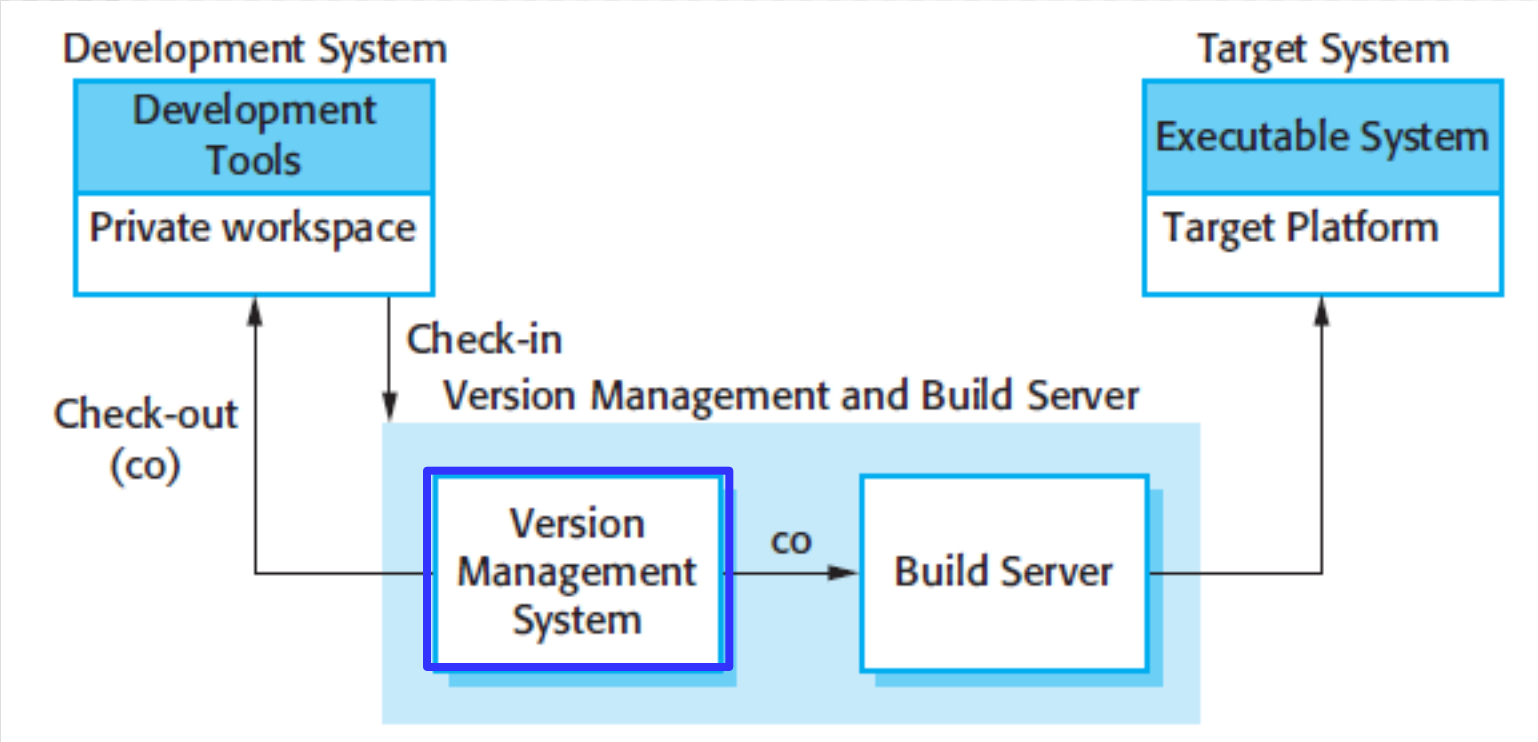
  Example CVS, Subversion, Git

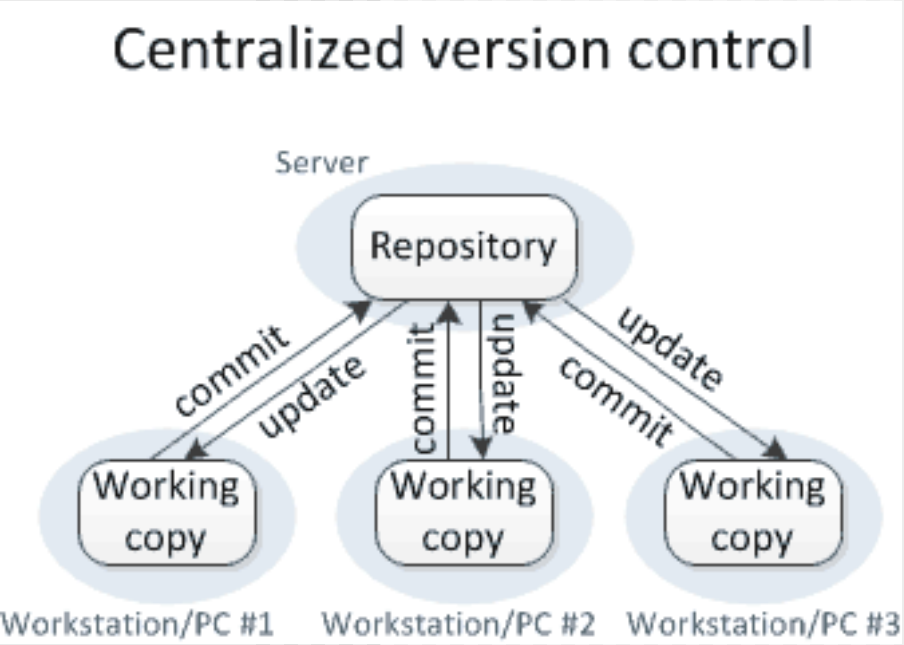
- A version control system implements:
  - project database (repository)
    - stores all relevant configuration objects
  - version management capability
    - stores all versions of a configuration object
  - make facility
    - collect all relevant configuration objects and construct a specific version of the software.

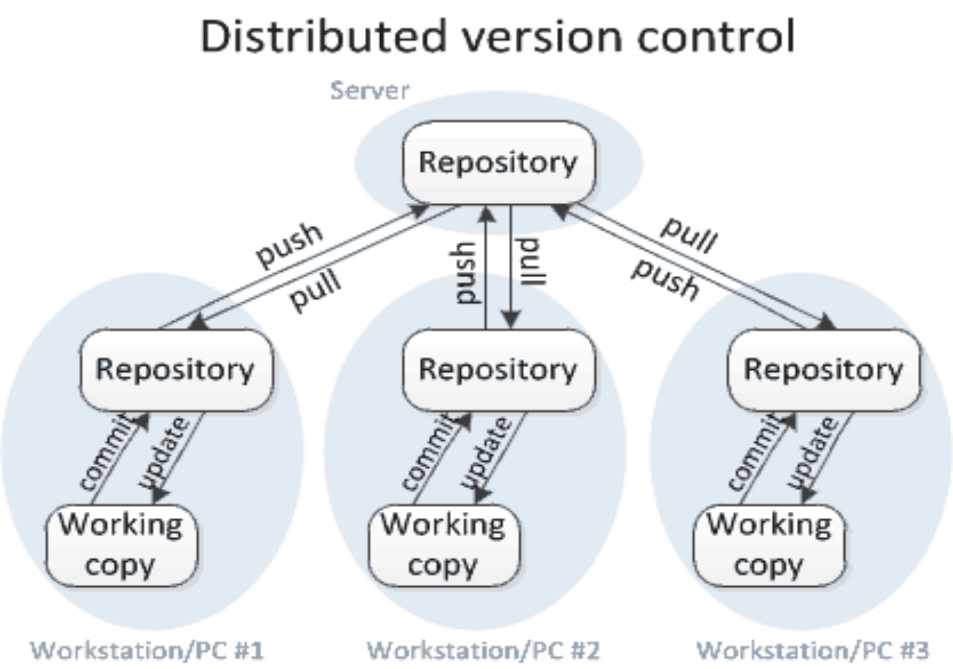
- A version control system is directly integrated with
  - an issues tracking system (also called *bug tracking system*)
    - tracks all outstanding issues associated with each configuration object.

      Example Bugzilla, Jira, Redmine

Development, build and target platforms

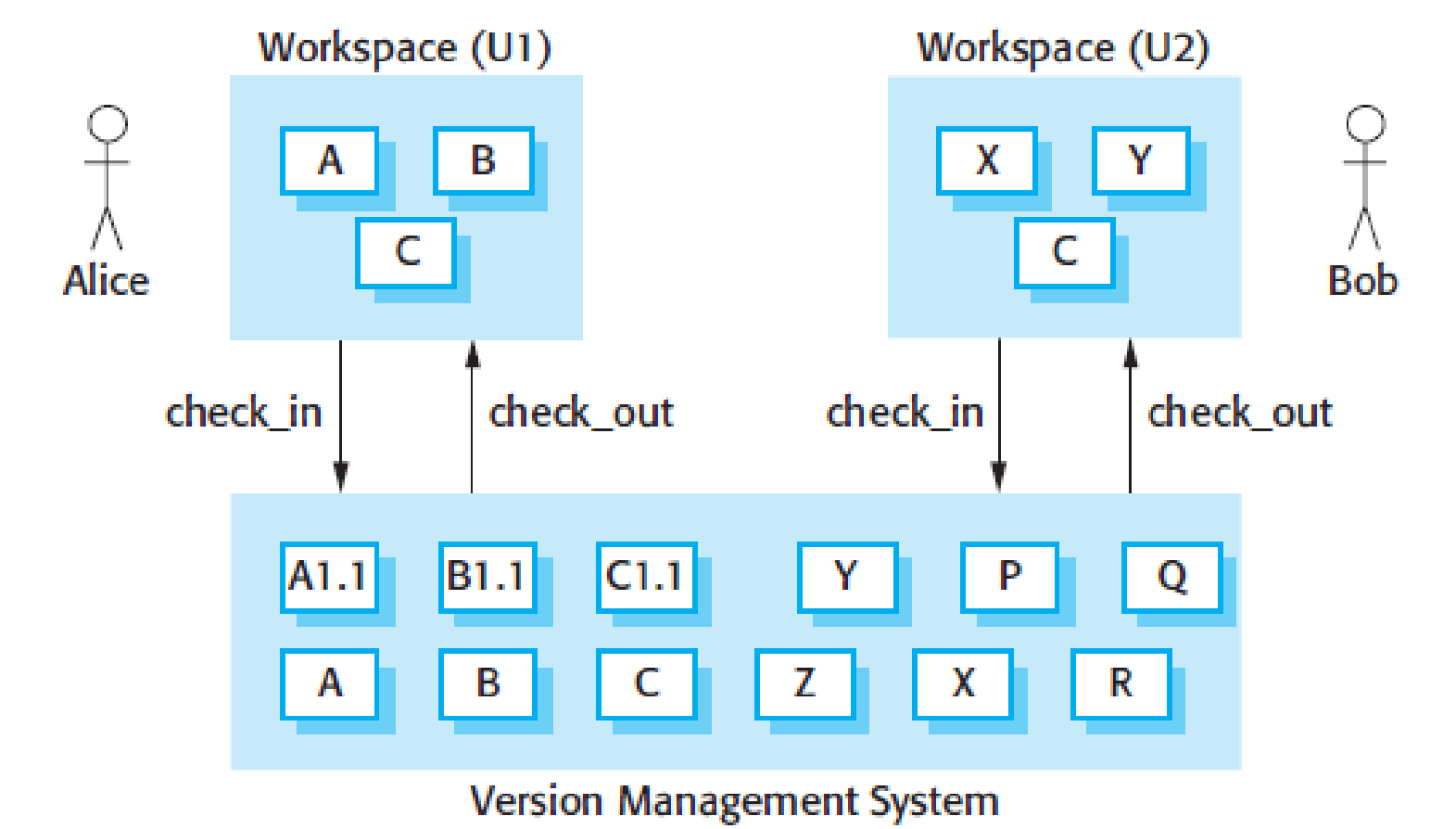



Centralized version control
Server
Repository
commit
update
commit
update
commit
update
Working copy
Working copy
Working copy
Workstation/PC #1
Workstation/PC #2
Workstation/PC #3


Distributed version control
Server
Repository
push
pull
push
pull
pull
push
Repository
Repository
Repository
commit
update
commit
update
commit
update
Working copy
Working copy
Working copy
Workstation/PC #1
Workstation/PC #2
Workstation/PC #3

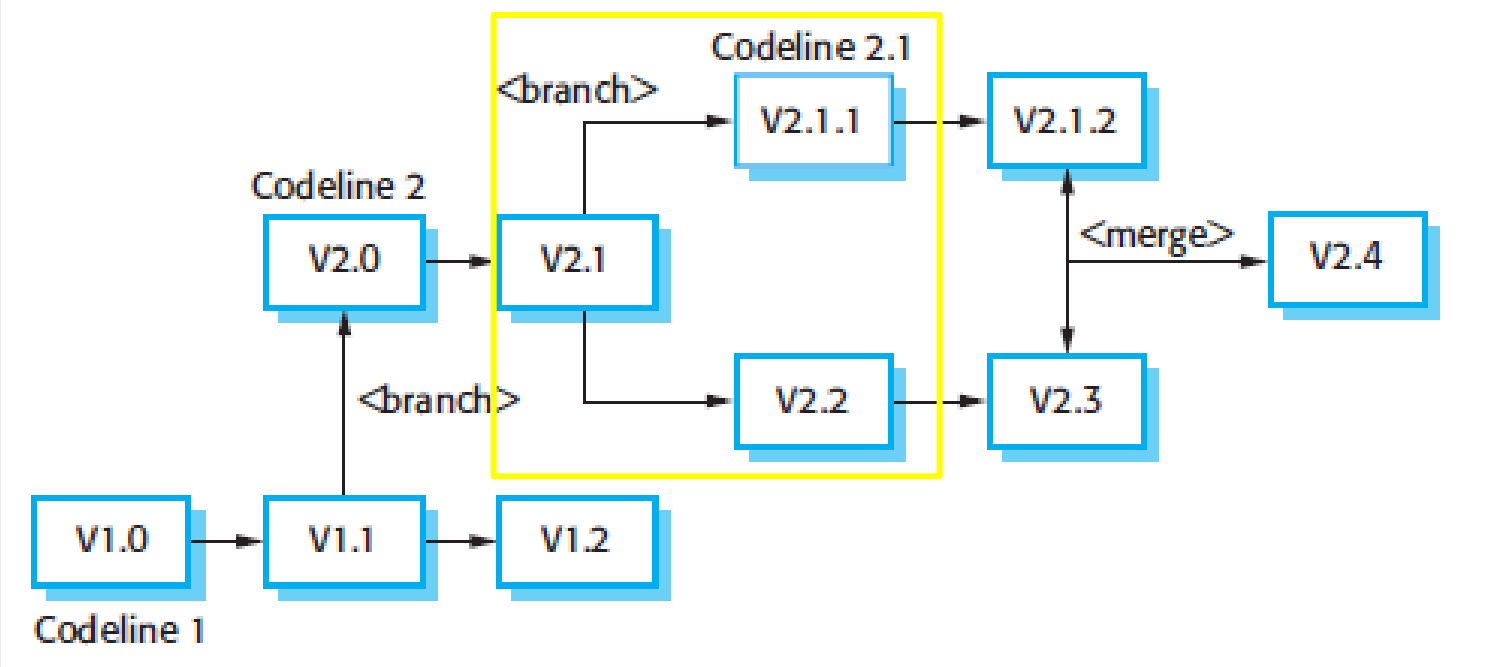
VM Systems

- Check-in and check-out from a version repository



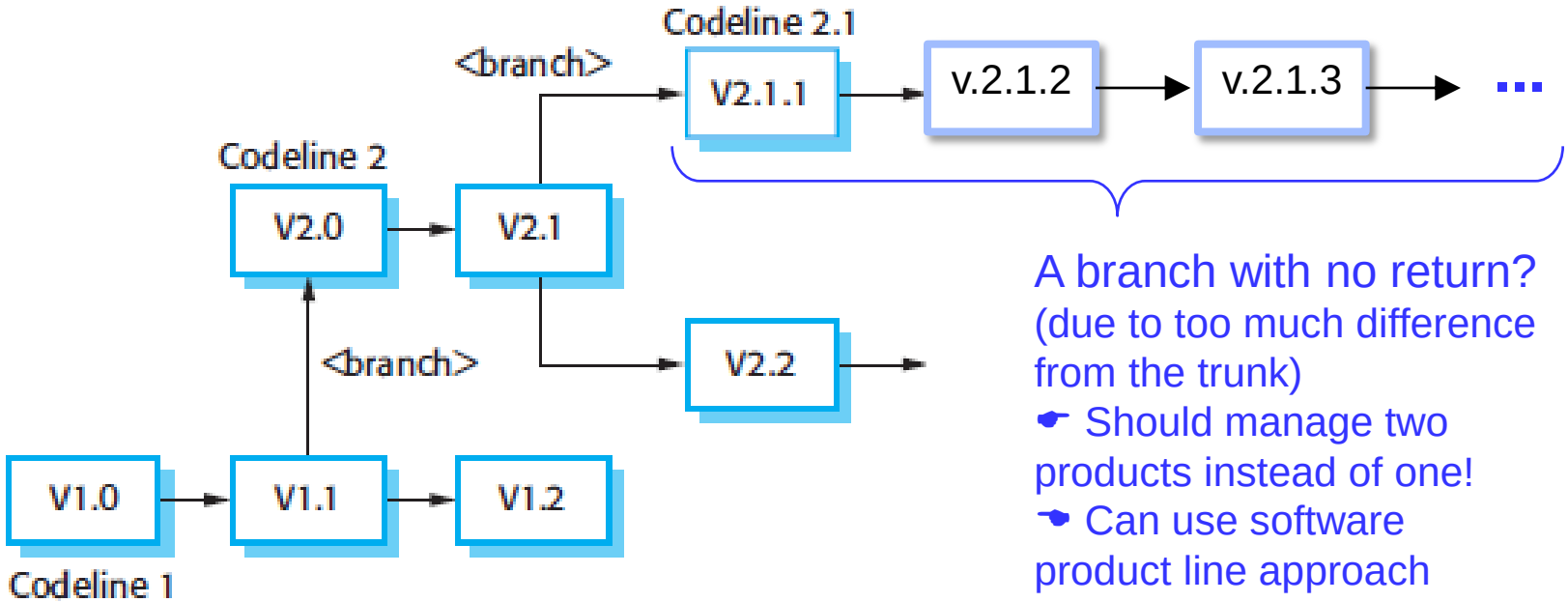
VM Systems

- If two developers check out the same version and make changes to it, they are assigned different version numbers.

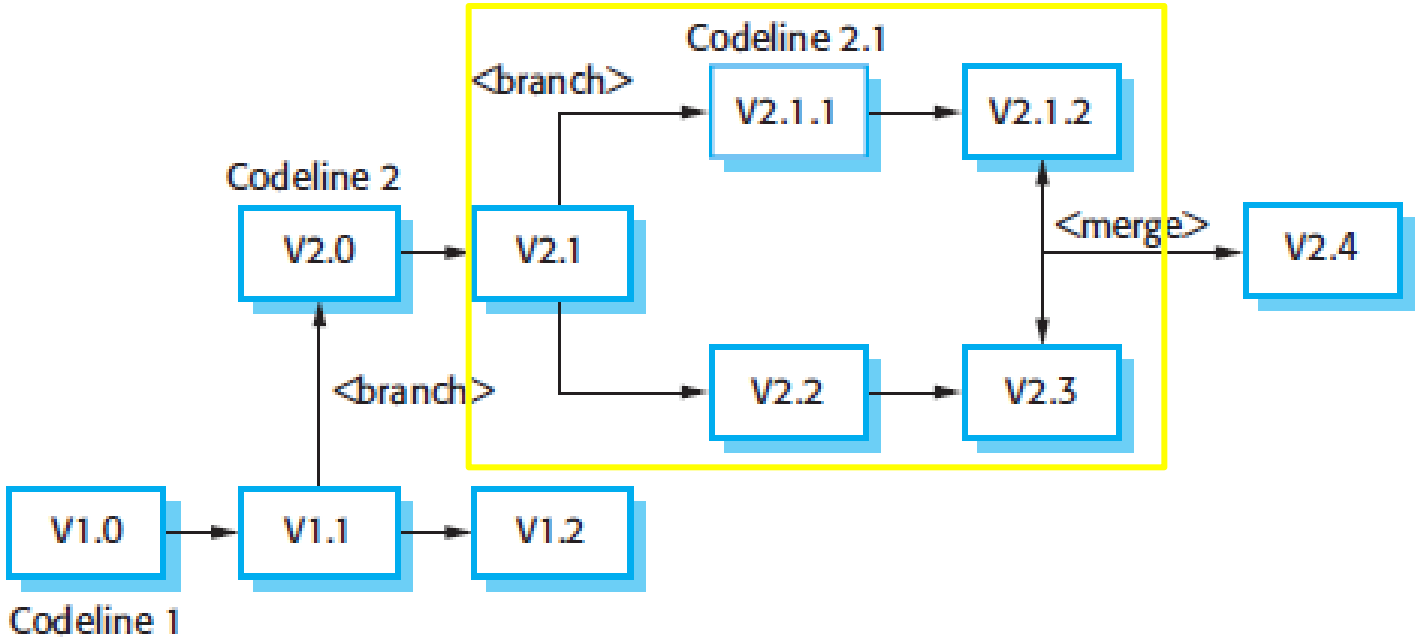


Branch

- Is needed when
  - Maintenance (bug fixing) and upgrade (adding new feature) are done at the same time
  - Extensive upgrade on the release product
  - A customer specific feature is added

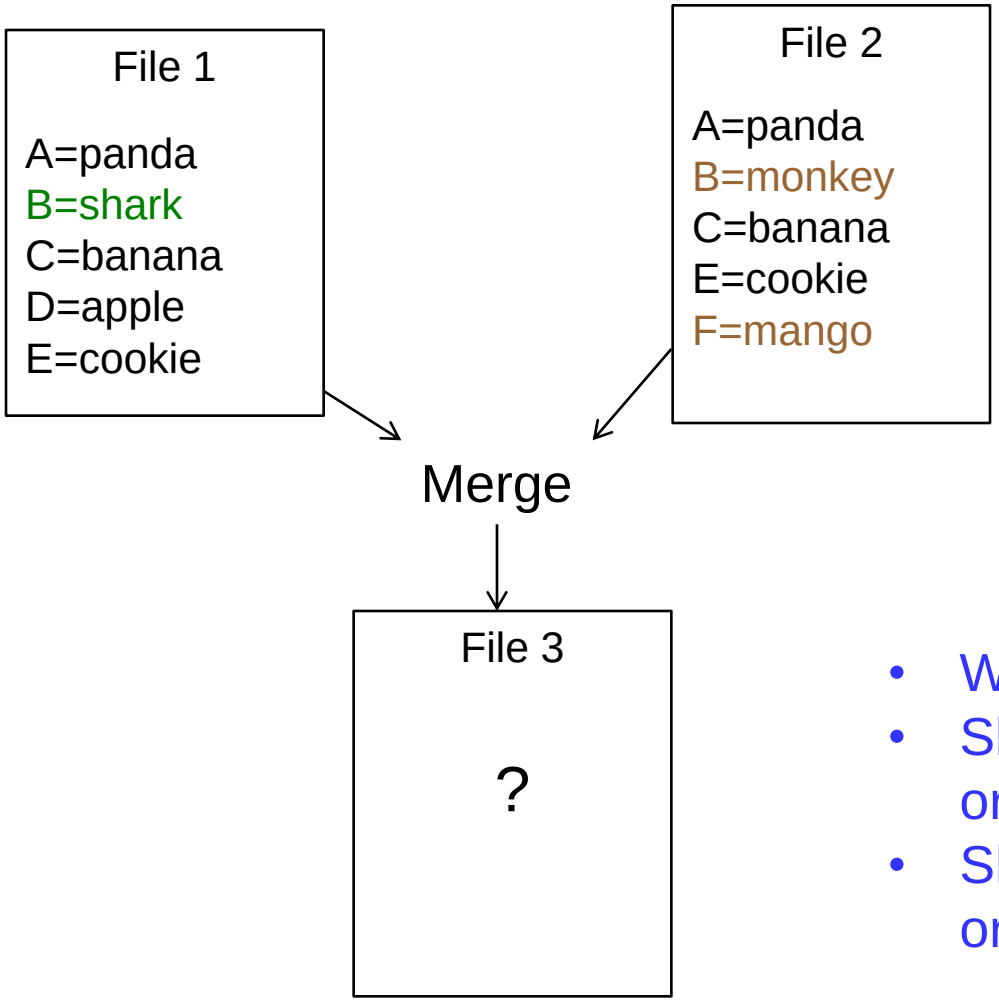


Branch and Merge



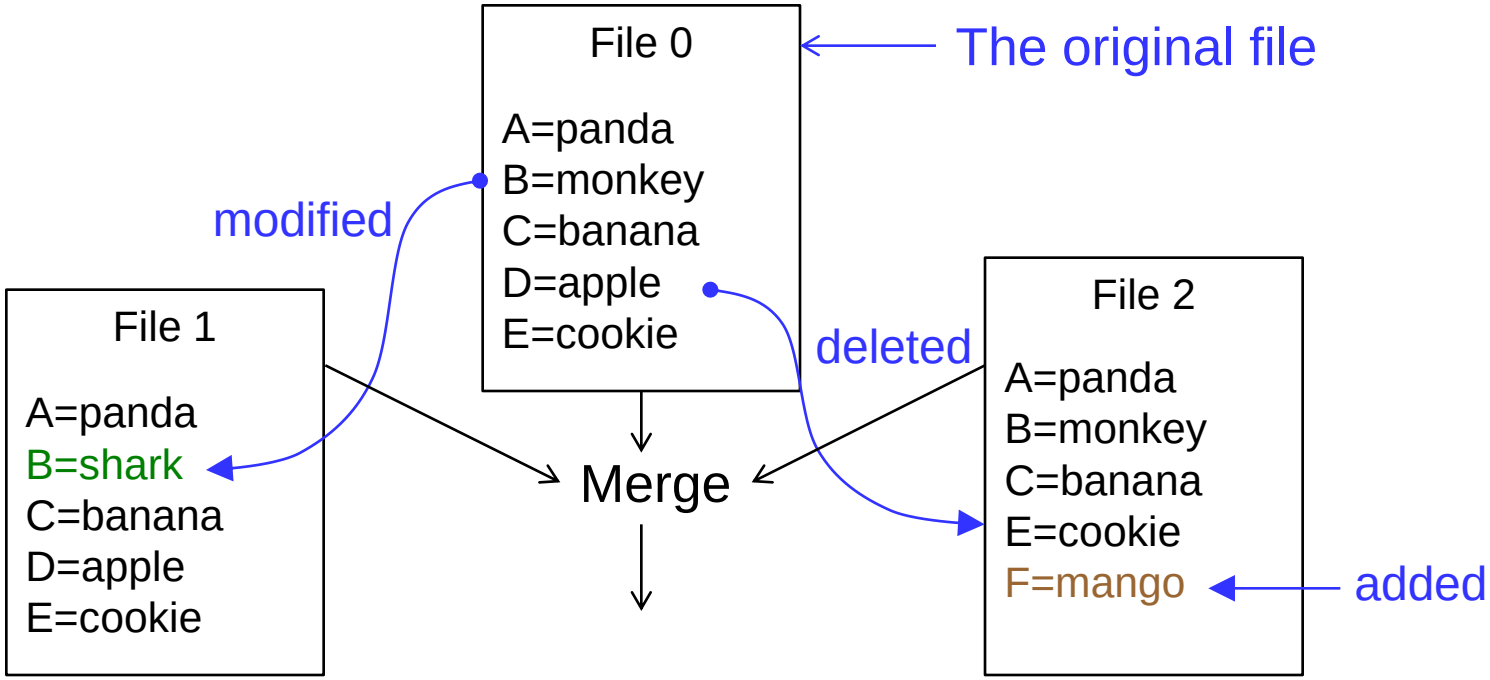
How can we ensure consistency?

2-Way Merge



- What should B be?
- Should D be included or excluded?
- Should F be included or excluded?

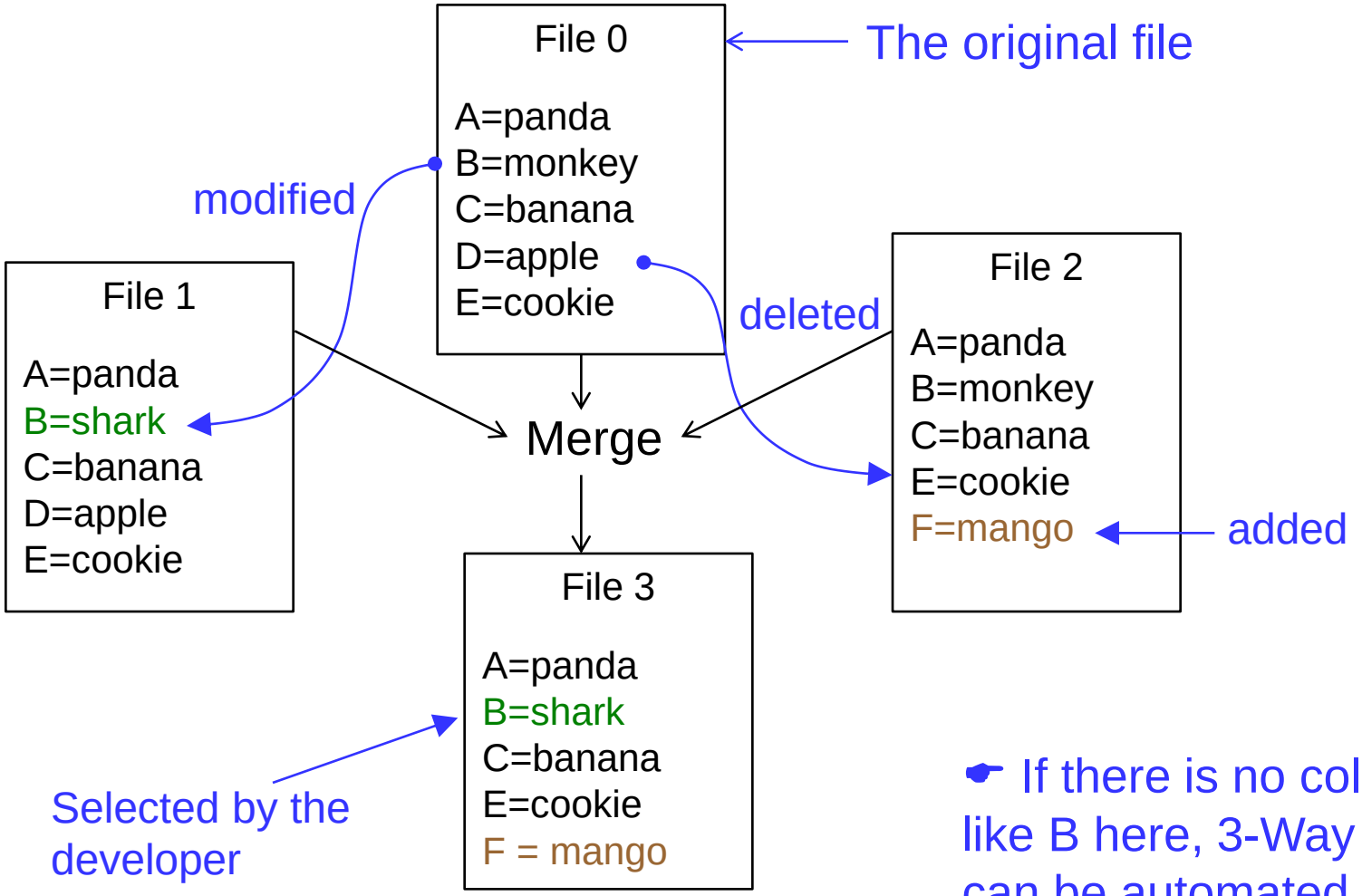
3-Way Merge (1/2)



?

☛ If there is no collision like B here, 3-Way merge can be automated.

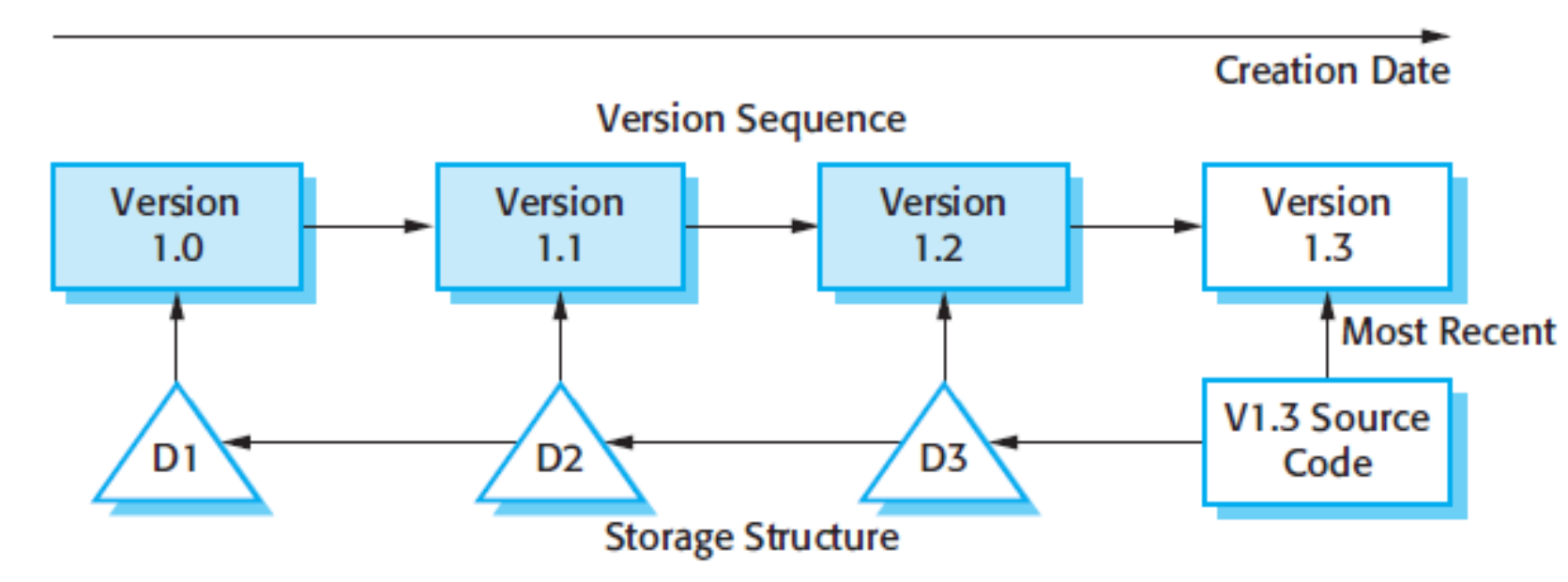
3-Way Merge (2/2)



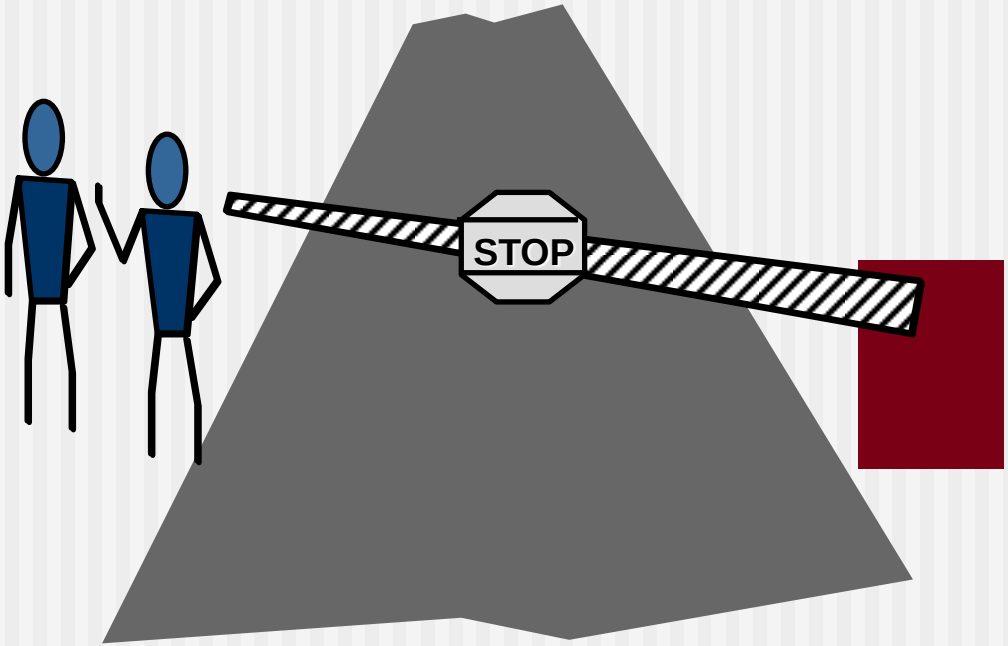
☛ If there is no collision like B here, 3-Way merge can be automated.

VM Systems

- Storage management using deltas



29.3.3 Change Control



A partially completed change request form

Change Request Form

Project: SICSA/AppProcessing **Number:** 23/02
Change requester: I. Sommerville **Date:** 20/01/09
Requested change: The status of applicants (rejected, accepted, etc.) should be shown visually in the displayed list of applicants.

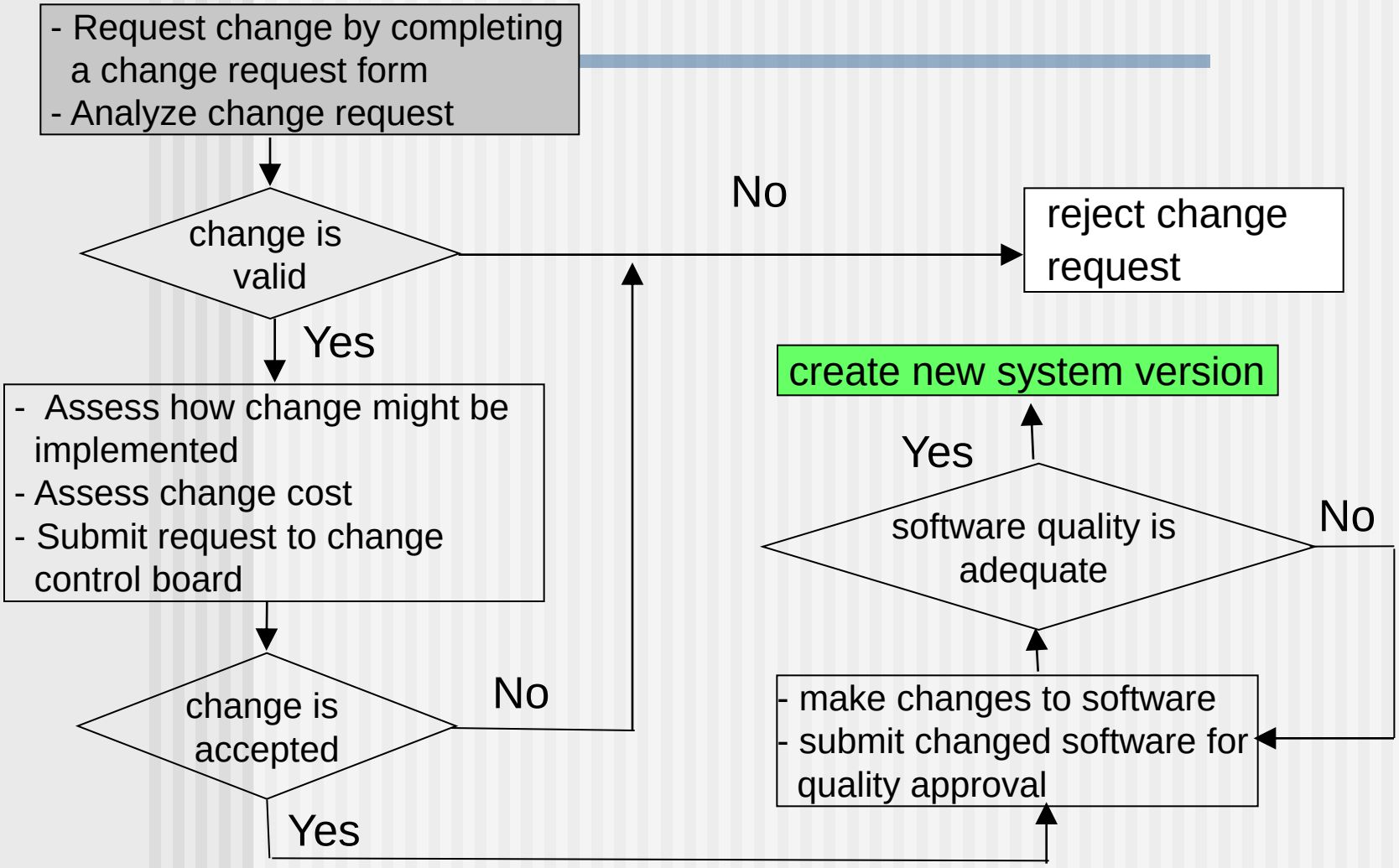
Change analyzer: R. Looek **Analysis date:** 25/01/09
Components affected: ApplicantListDisplay, StatusUpdater

Associated components: StudentDatabase

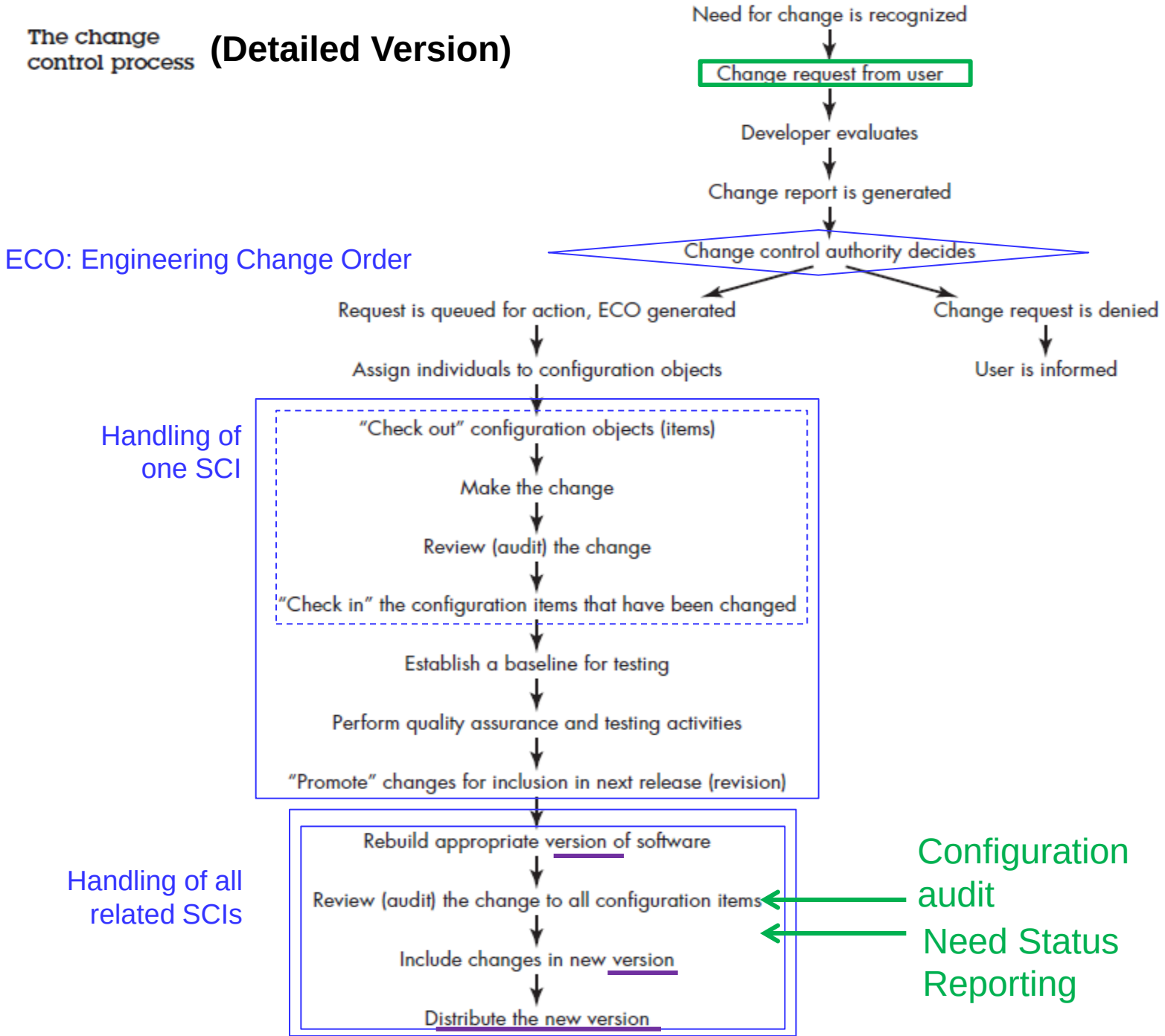
Change assessment: Relatively simple to implement by changing the display color according to status. A table must be added to relate status to colors. No changes to associated components are required.

Change priority: Medium
Change implementation:
Estimated effort: 2 hours
Date to SGA app. team: 28/01/09 **CCB decision date:** 30/01/09
Decision: Accept change. Change to be implemented in Release 1.2
Change implementor: **Date of change:**
Date submitted to QA: **QA decision:**
Date submitted to CM:
Comments:

The Change Management Process (Brief Version)



The change control process **(Detailed Version)**



29.3.4 Impact Management

- Three actions

(1) Impact network

- identifies the members of a software team (and other stakeholders)

☛ Clear definition of the software architecture is important

(2) Forward impact management

- assesse the impact of your own changes on the members of the impact network

(3) Backward impact management

- examines changes that are made by other team members and their impact on your work.

29.3.5 Configuration Audit

How can a software team ensure that

the change has been properly implemented?

(1) Technical review

(2) Configuration Audit

Asks …

1. Has the change specified in the ECO been made?
   Have any additional modifications been incorporated?

2. Has a technical review been conducted to assess technical correctness?

3. Has the software process been followed and have software engineering standards been properly applied?

4. Has the change been “highlighted” in the SCI?
   Have the change date and change author been specified?
   Do the attributes of the configuration object reflect the change?

5. Have SCM procedures for noting the change, recording it, and reporting it been followed?

6. Have all related SCIs been properly updated?

29.3.6 Status Reporting

- = Status Accounting

- Asks the following questions:
  - What happened?
  - Who did it?
  - When did it happen?
  - What else will be affected?

☛ Each time **configuration audit** is conducted, the results are reported as part of the “configuration status reporting” (See the Change Control Process)