Software Engineering Theory and Practice

School of Computing	
Title	Software Engineering Theory and Practice
Module Coordinator	Steven Ossont
Email	steven.ossont@port.ac.uk
Code	M30819
Moodle	https://moodle.port.ac.uk/course/view.php?id=11429

U30819: Software Engineering Theory and Practice

Software Configuration Management , Ch25

https://moodle.port.ac.uk/course/view.php?id=11429

Software Configuration Management

- Software systems are constantly changing during development and use.
- Configuration management (CM) is concerned with the policies, processes and tools for managing changing software systems.
- You need Configuration Management because it is easy to lose track of what changes and component versions have been incorporated into each system version.
- Configuration Management is essential for team projects to control changes made by different developers

> 828-2012 - IEEE Standard for Configuration Management in Systems and Software Engineering https://ieeexplore.ieee.org/document/6170935

Software Configuration Management

Configuration management involves four activities:

1. Version management

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.

Software Configuration Management

Configuration management involves four activities:

2. System building

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

Software Configuration Management

Configuration management involves four activities:

3. Change management

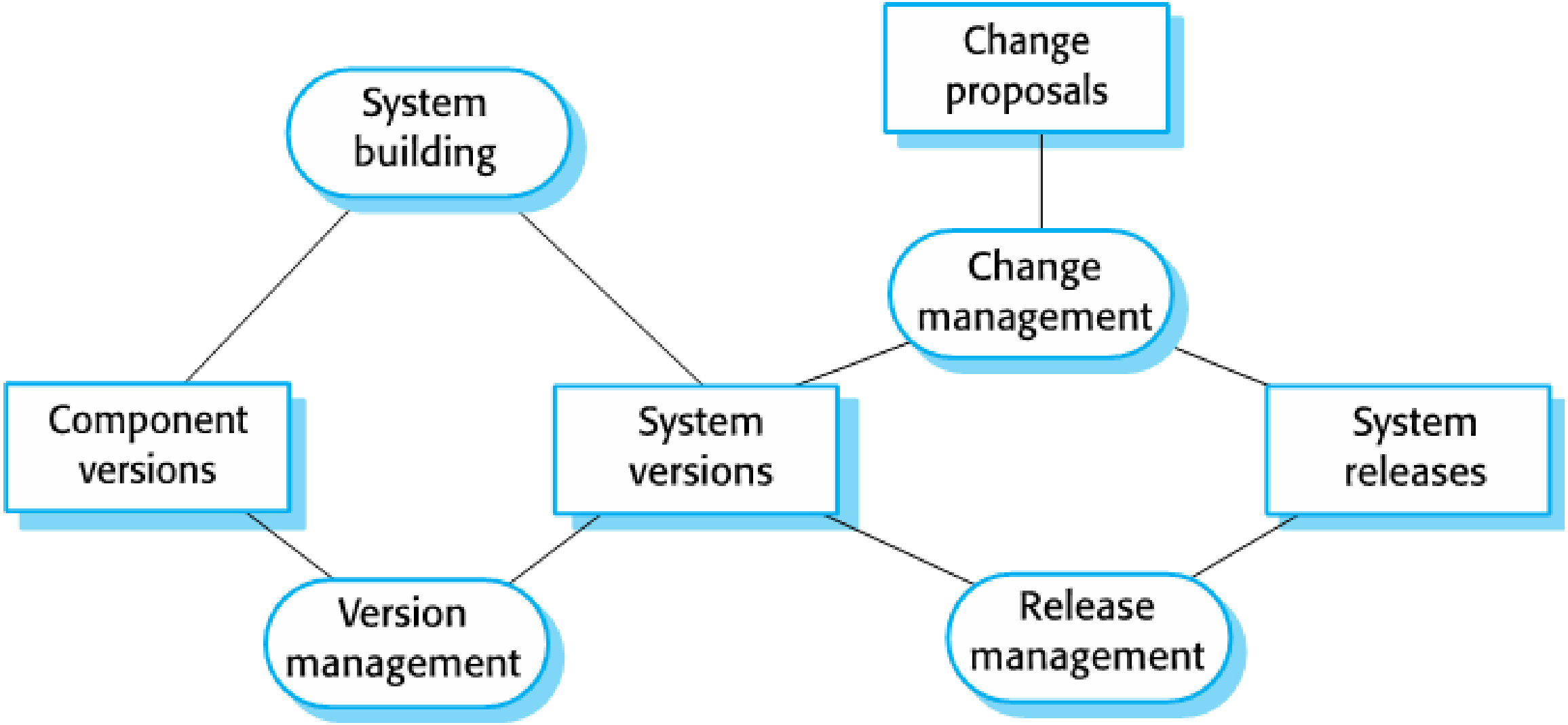
   Keeping track of requests for changes to the software from customers and developers, working out the costs and impact of changes, and deciding the changes should be implemented.

Software Configuration Management

Configuration management involves four activities:

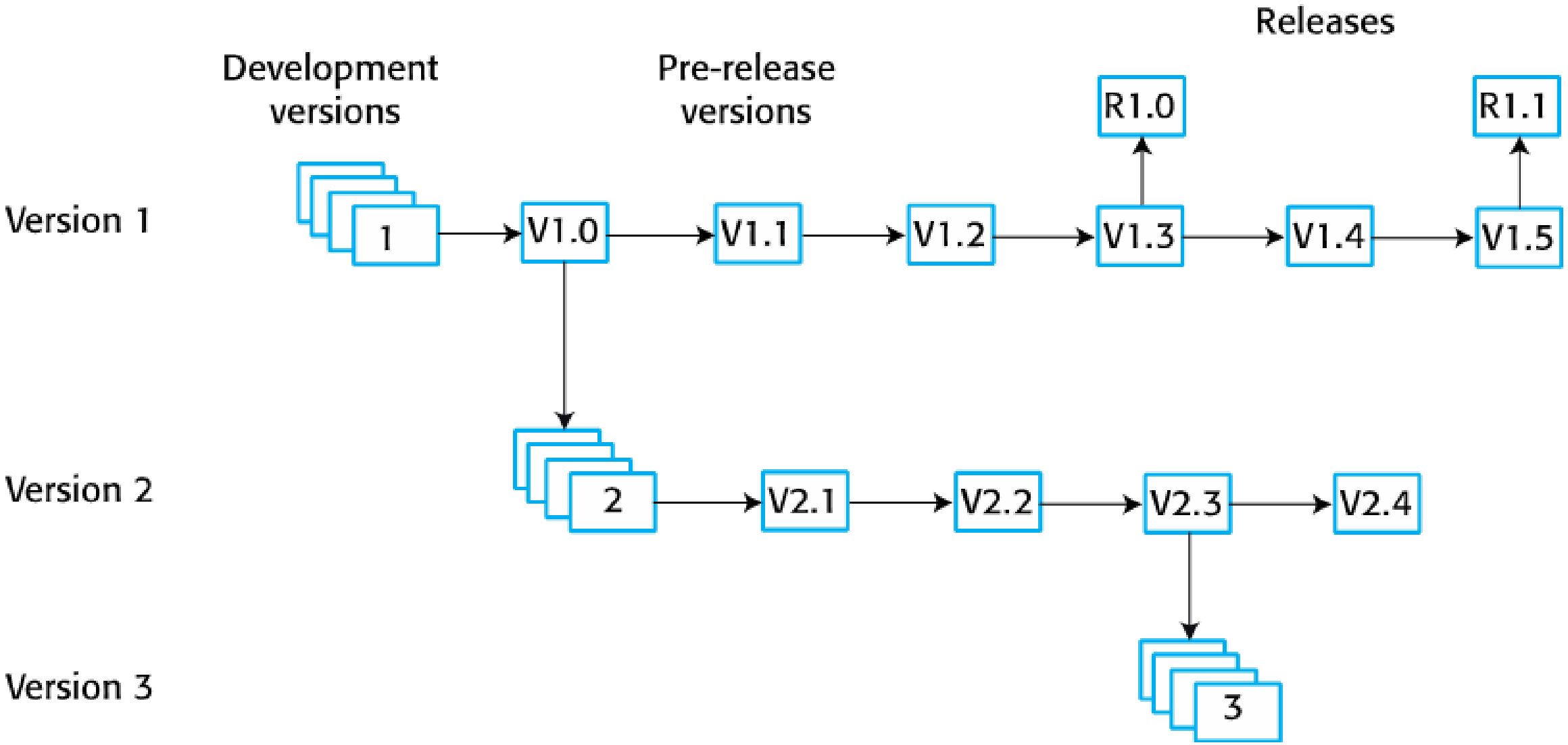
4. Release management

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


Change proposals
System building
Change management
Component versions
System versions
System releases
Version management
Release management

Development phases

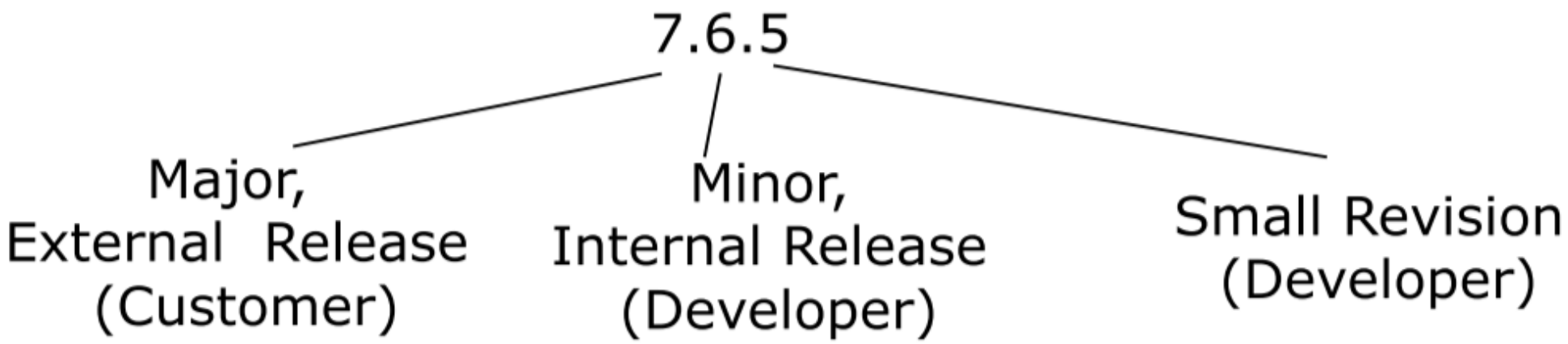
- A development phase where the development team is responsible for managing the software configuration and new functionality is being added to the software.
- A system testing phase where a version of the system is released internally for testing.
  - No new system functionality is added. Changes made are bug fixes, performance improvements and security vulnerability repairs.
- A release phase where the software is released to customers for use.
  - New versions of the released system are developed to repair bugs and vulnerabilities and to include new features.


Releases
Development versions
Pre-release versions
R1.0
R1.1
Version 1
1
V1.0
V1.1
V1.2
V1.3
V1.4
V1.5
Version 2
2
V2.1
V2.2
V2.3
V2.4
Version 3
3

Terminology -- Baseline

- A baseline is a collection of component versions that make up a system. Baselines are controlled, which means that the versions of the components making up the system cannot be changed. This means that it is always possible to recreate a baseline from its constituent components.
- *"A specification or product that has been formally reviewed and agreed to by responsible management, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures"*

Terminology -- Baseline naming schemes



Terminology -- Codeline

A codeline is a set of versions of a software component and other configuration items on which that component depends.

Terminology -- Branching

- The creation of a new codeline from a version in an existing codeline. The new codeline and the existing codeline may then develop independently.

Terminology -- Configuration (version) control

- The process of ensuring that versions of systems and components are recorded and maintained so that changes are managed and all versions of components are identified and stored for the lifetime of the system.

Terminology -- Configuration item or software configuration item (SCI)

Anything associated with a software project (design, code, test data, document, etc.) that has been placed under configuration control. There are often different versions of a configuration item. Configuration items have a unique name.

- Problem statement
- Software project management plan
- Requirements analysis document
- System design document
- Source code
- Functional model
- Unit tests

Terminology -- Configuration item or software configuration item (SCI)

- Integration test strategy
- API specification
- Database(s)
- Testplan
- Testdata
- Support software
- User manual
- Administrator manual

Terminology -- Mainline

- A sequence of baselines representing different versions of a system.

Terminology -- Merging

The creation of a new version of a software component by merging separate versions in different codelines. These codelines may have been created by a previous branch of one of the codelines involved.

Terminology -- Release

- A version of a system that has been released to customers (or other users in an organization) for use.

Terminology -- Repository

- A shared database of versions of software components and meta-information about changes to these components.

Terminology -- System building

- The creation of an executable system version by compiling and linking the appropriate versions of the components and libraries making up the system.

Terminology

- Terminology -- Version
  - An instance of a configuration item that differs, in some way, from other instances of that item. Versions always have a unique identifier.
- Terminology -- Workspace
  - A private work area where software can be modified without affecting other developers who may be using or modifying that software.

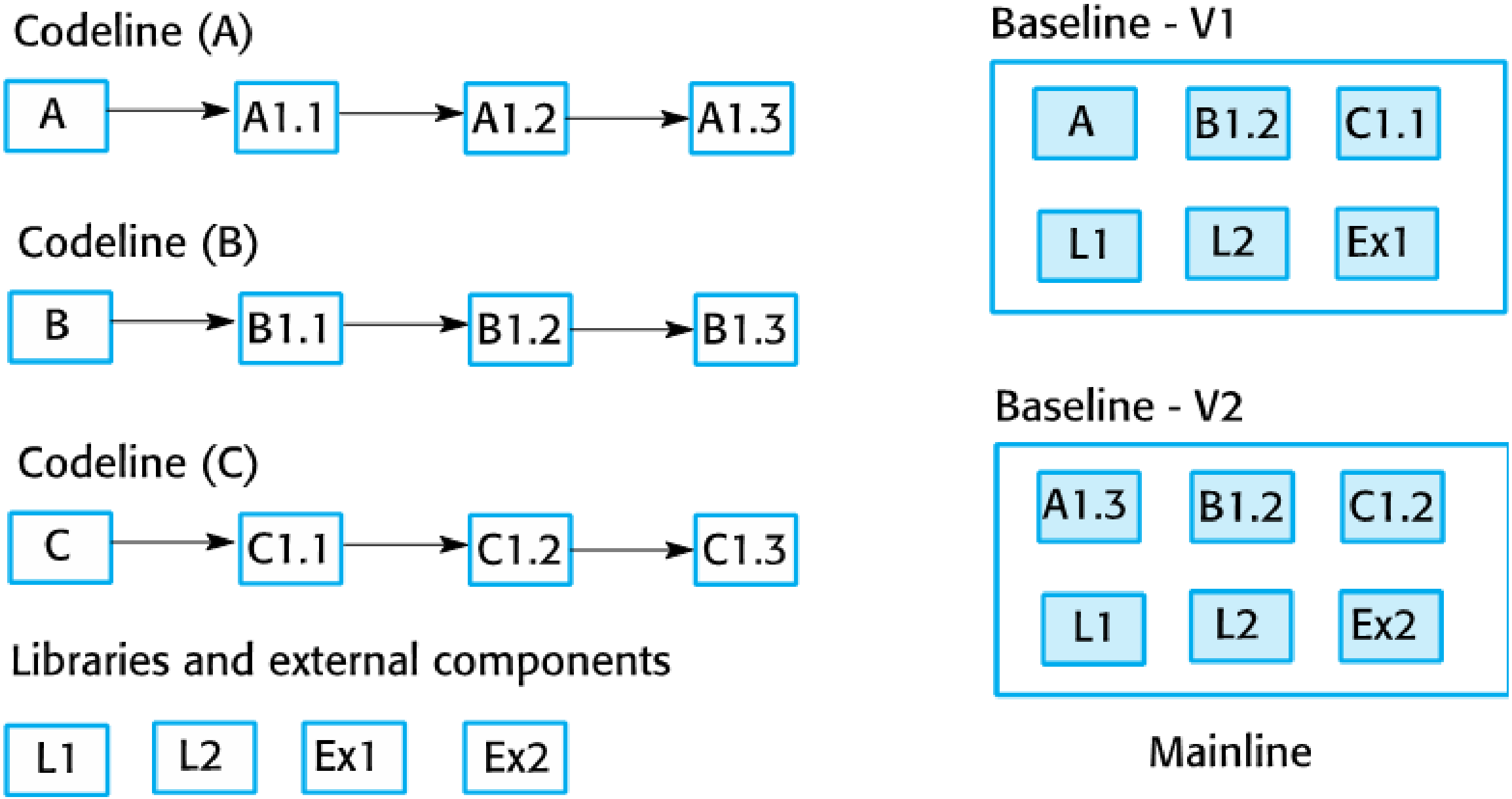
Software Configuration Management - Versioning

> Free book: https://git-scm.com/book/en/v2

The Problem

- Multiple people have to work on software that is changing
- More than one version of the software has to be supported

> Versioning != Backup


Codeline (A)
A
A1.1
A1.2
A1.3
Codeline (B)
B
B1.1
B1.2
B1.3
Codeline (C)
C
C1.1
C1.2
C1.3
Libraries and external components
L1
L2
Ex1
Ex2
Baseline - V1
A
B1.2
C1.1
L1
L2
Ex1
Baseline - V2
A1.3
B1.2
C1.2
L1
L2
Ex2
Mainline

Solution No. 1- AKA The wrong way

- Copy files into another directory
- Rename copies of a file into same folder
- Any other uncontrolled variant

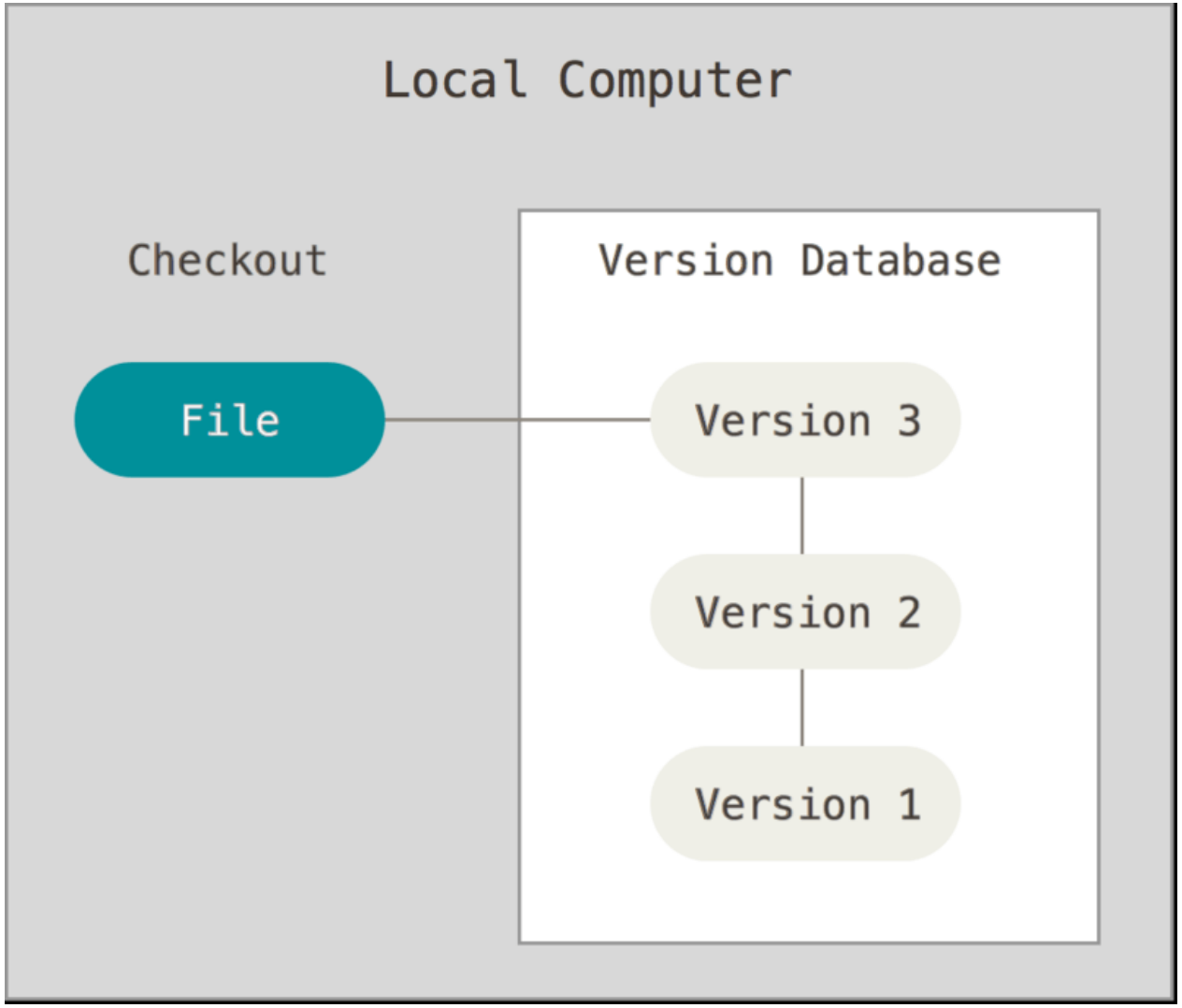
> A truly terrible idea that should never be utilised

Solution No.2 – Local Version control system

Tool: Revision Control System (RCS)
https://www.gnu.org/software/rcs/

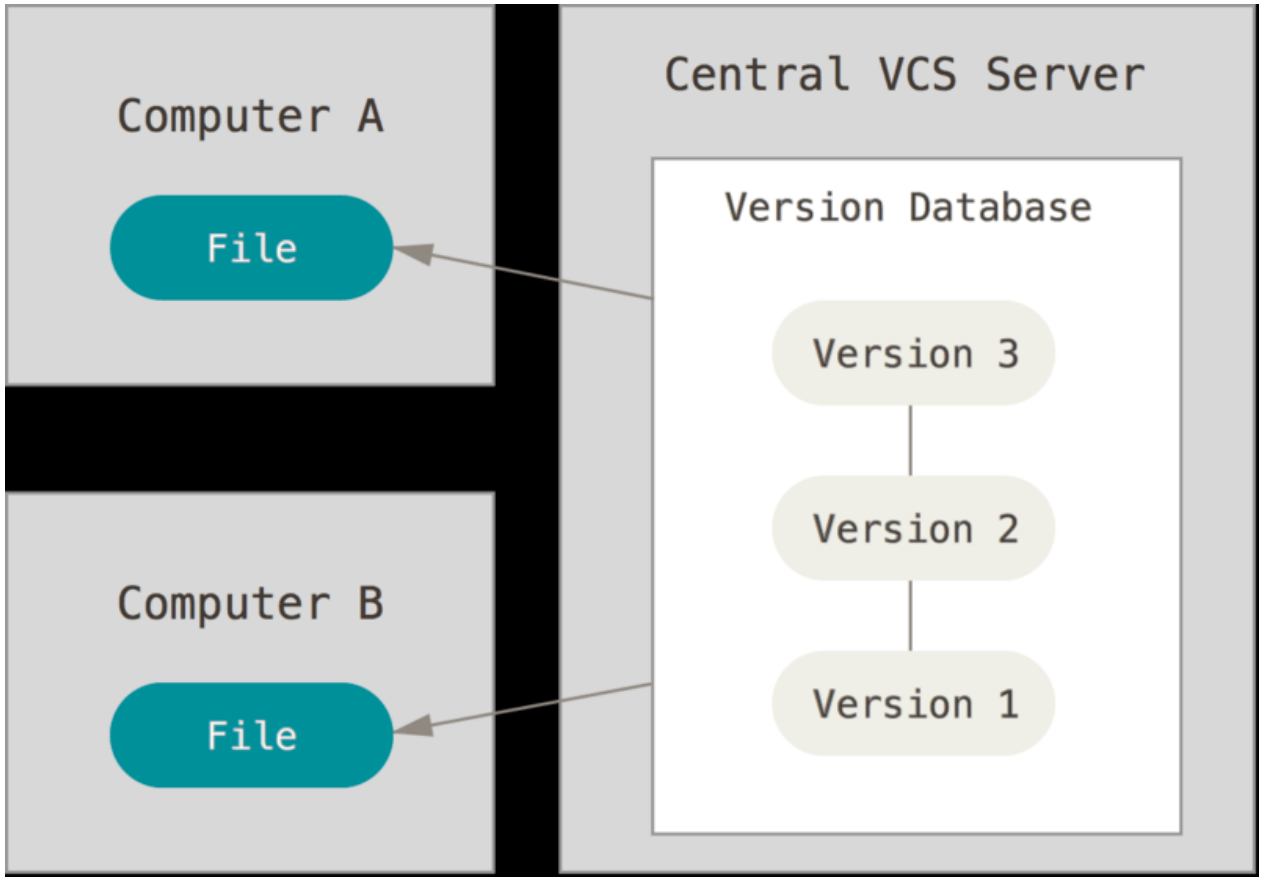
- Very old
- Very limited
- Designed to overcome problems we do not have today (Bandwidth, Storage)

> Very restrictive



Solution No.3 - Centralised Version Control Systems

- Clients check out the latest snapshots of the files
- Modify the files
- Commit the changes back to the **central server**
- Tools: CVS https://savannah.nongnu.org/projects/cvs/; Subversion https://subversion.apache.org/

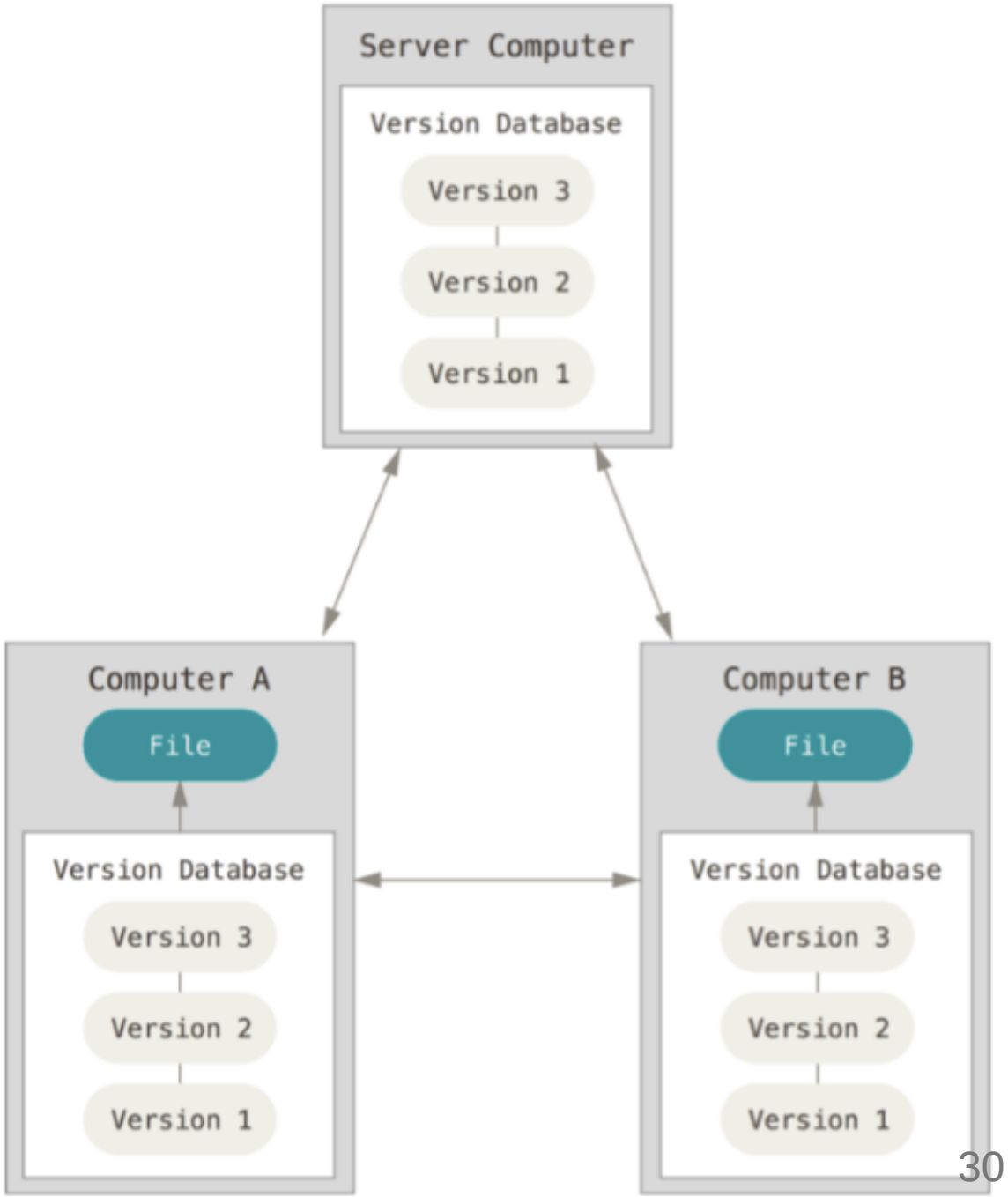


Centralised Version Control Systems

- Everyone knows what everyone else is working on
- Admins control who can do what
- Manage one single central server (not many local ones)
- One single point of failure!
- Server down, no work can be done
- Server corrupted, all history of the project lost

Solution No.4 - Distributed Version Control Systems (DVCS)

- Clients mirror the full repository
- Programmers can make any changes
- Commit the full repository back to the/a central server
- Every clone is a full copy of the whole repository

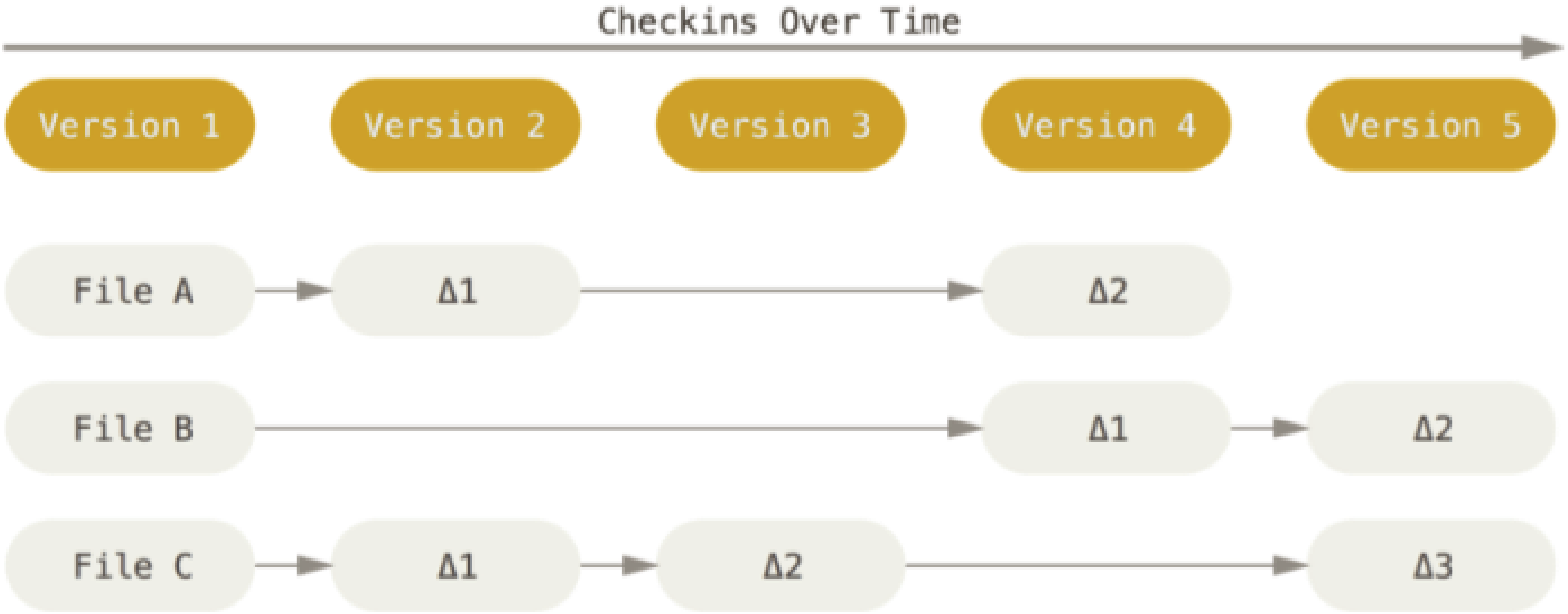


Distributed Version Control Systems (DVCS)

- Tools :
  - Git https://git-scm.com/,
  - Mercurial https://www.mercurial-scm.org/,
  - Bazaar https://bazaar.canonical.com/en/

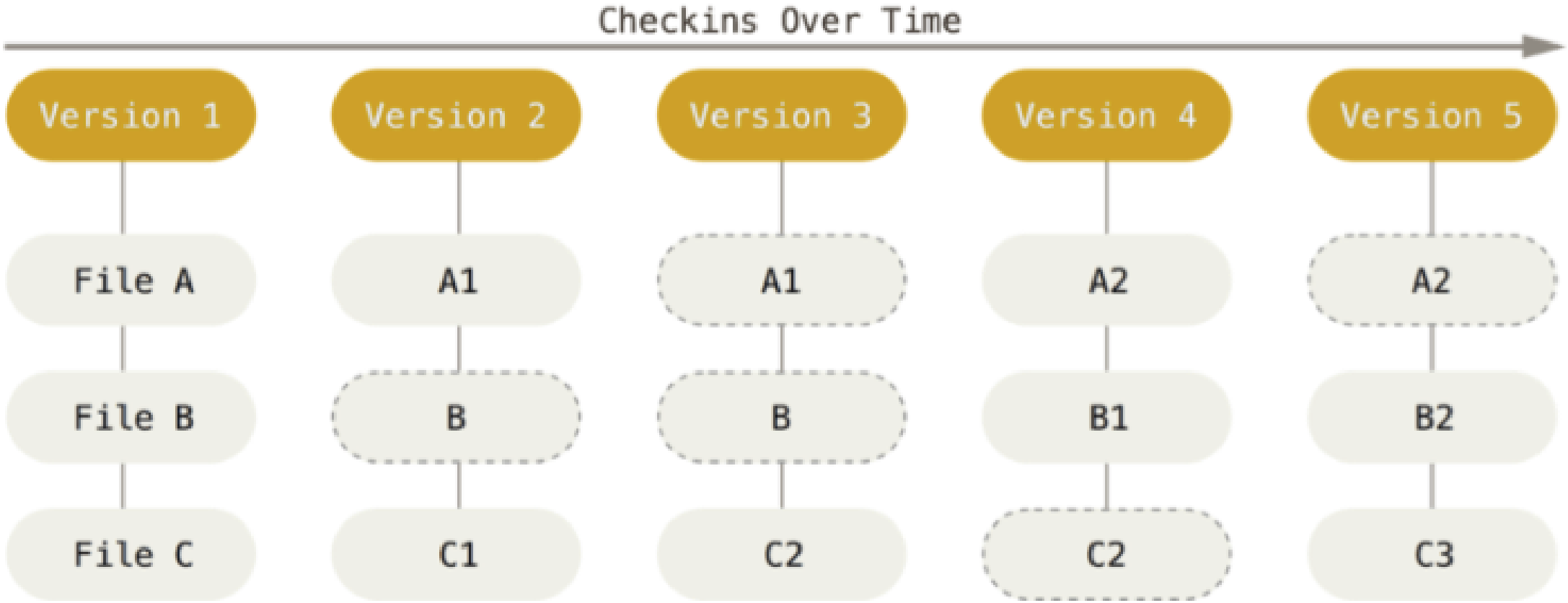
Using a difference technique (Subversion, CVS)

- List of file-based changes (diff)


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
Δ1
Δ2
File B
Δ1
Δ2
File C
Δ1
Δ2
Δ3

Using a file system snapshot (Git)

- Set of snapshots of a miniature file system

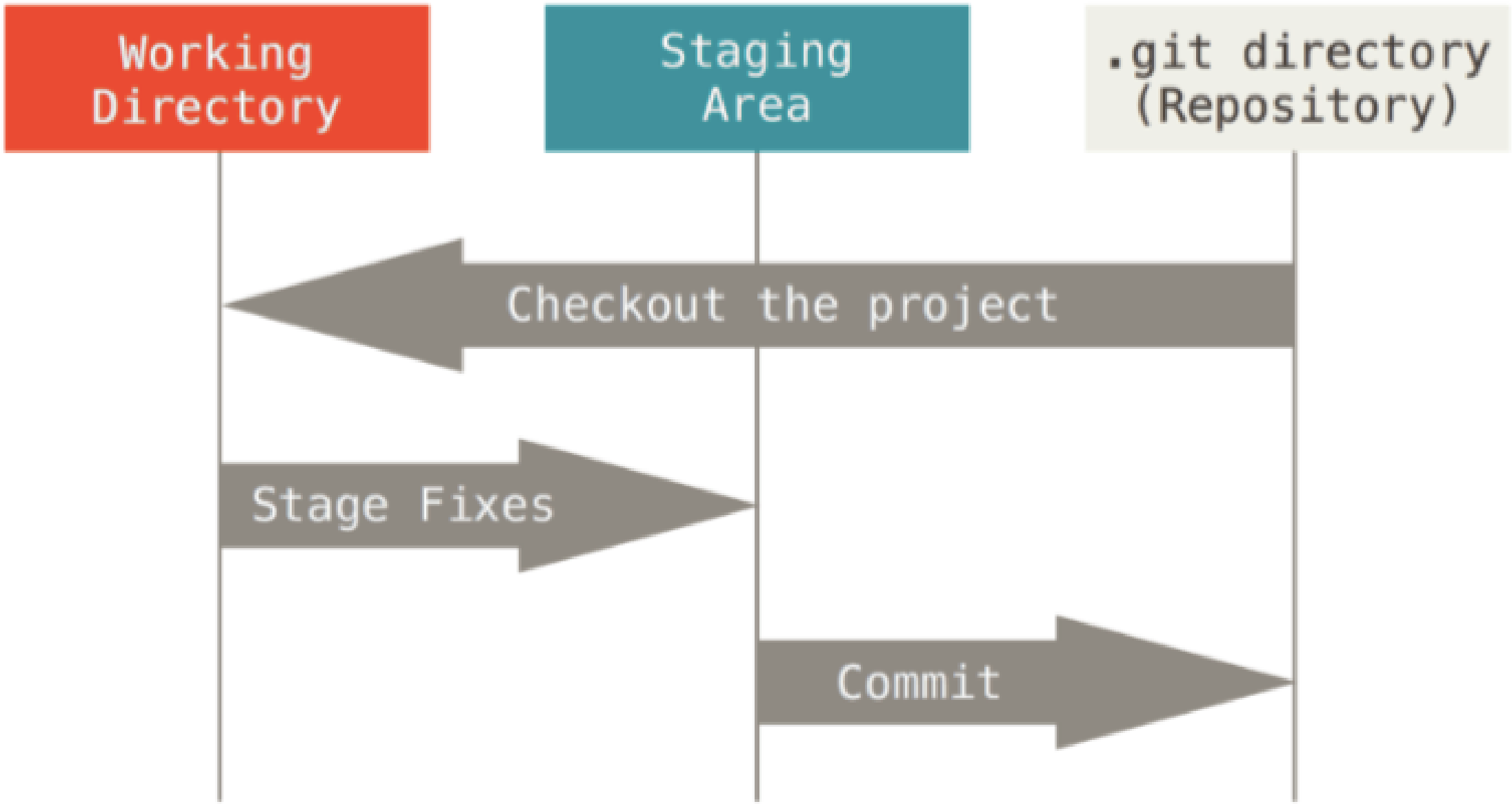

Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
File B
File C
A1
B
C1
A1
B
C2
A2
B1
C2
A2
B2
C3

Git - File State

- ***Committed*** : the file is safely stored in the database
- ***Modified*** : the file has been modified, but it has not yet been committed
- ***Staged*** : the file has been modified file and marked to go into the next commit snapshot

A Git Project

- *.git* folder : stores the metadata and the object database for the project
- Working directory : single checkout of a version of the project; placed on disk for use/modification
- Staging area : stores information about what will go into the next commit


Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit

Basic Git Workflow

1. Modify files in the working directory
2. Stage the files by adding their snapshots to the staging area
3. Commit: takes files from the staging area and stores their snapshots to the Git directory

> Command line (A must) or Graphical Interfaces (Handy once you master the command line)

GitHub

- Web-based Git repository **hosting** service

Software Configuration Management

Configuration management involves four activities:

1. Version control
2. System Building
3. **Change management**
4. Release management

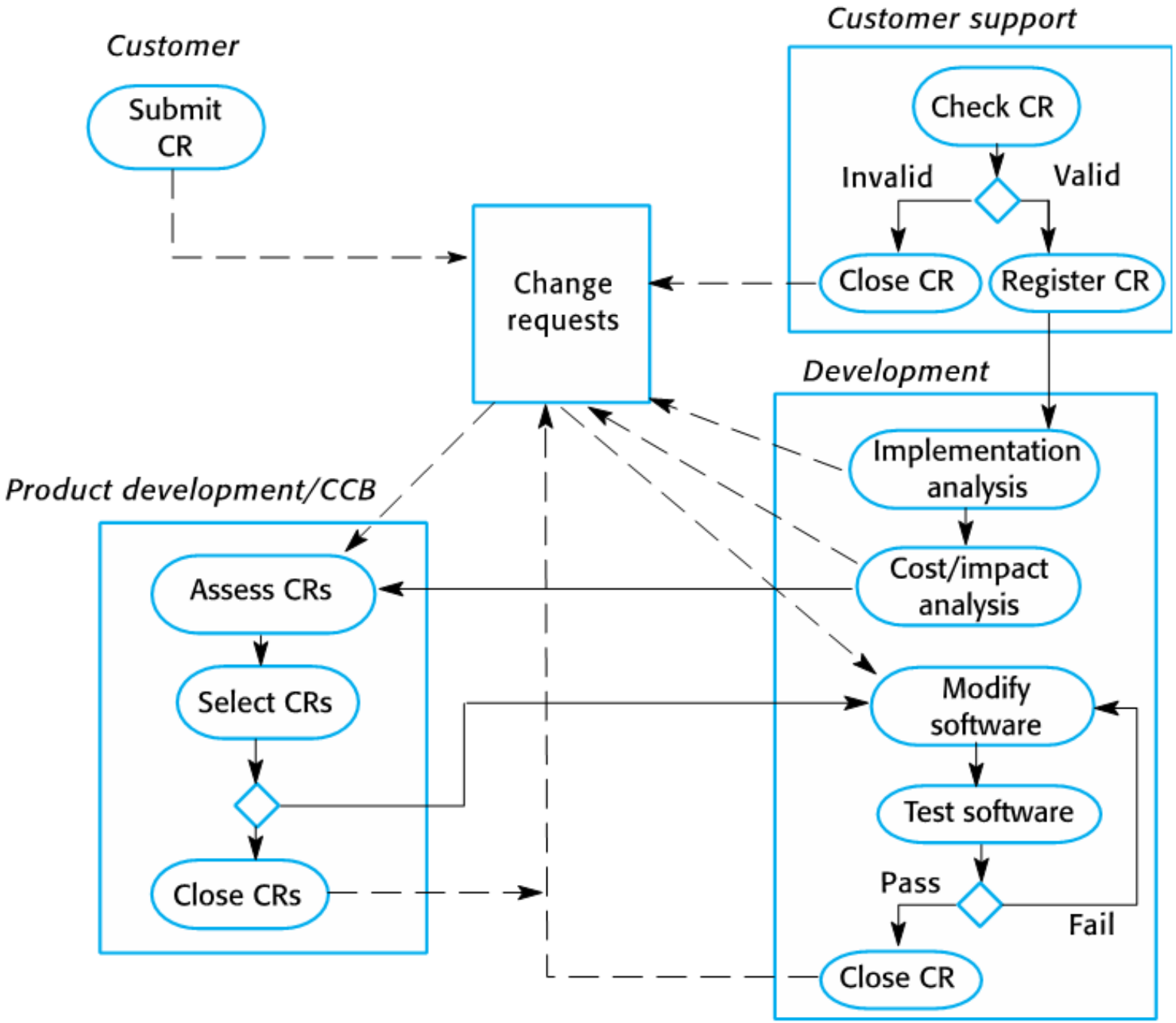
Software Configuration Management -- Change management

- Things change during the lifetime of a system
- Change management controls this evolution
- Costs vs Benefits

Change Management

Handling of change requests

- Change is requested
- Change requested is assessed against requirements and constraints
- Change requests is either accepted or rejected
- If accepted, it is assigned to a developer and implemented
- Implementation change is audited


Customer
Submit CR
Customer support
Check CR
Invalid
Valid
Close CR
Register CR
Change requests
Development
Implementation analysis
Cost/impact analysis
Modify software
Test software
Pass
Fail
Close CR
Product development/CCB
Assess CRs
Select CRs
Close CRs

Factors

- The consequences of not making the change
- The benefits of the change
- The number of users affected by the change
- The costs of making the change
- The product release cycle

Software Configuration Management

Configuration management involves four activities:

1. Version control
2. System Building
3. Change management
4. **Release management**

Software Configuration Management -- Release management

- A system release is a version of a software system that is distributed to customers.
- For mass market software, it is usually possible to identify two types of release: **major releases** which deliver significant new functionality, and **minor releases**, which repair bugs and fix customer problems that have been reported.
- For custom software or software product lines, releases of the system may have to be produced for **each customer** and individual customers may be running several different releases of the system at the same time.

Components

As well as the the executable code of the system, a release may also include:

- configuration files
- data files
- an installation program
- documentation
- packaging and publicity

Why release?

- Completed
- Contract (update on a date)
- OS update
- Faults (not just minor)

The Software Configuration Management

(How will you manage change?)

Software Configuration Management -- Plan

- Software configuration management planning starts during the early phases of a project
- The outcome of planning is the **Software Configuration Management Plan**

Introduction

- Describes the Plan’s purpose, scope of application, key terms, and references

Management (WHO?)

- Identifies the responsibilities and authorities for managing and accomplishing the planned SCM activities
- Who will be responsible for the entire SCM process and creation of baselines.

Activities (WHAT?)

- Identify configuration items
- Tooling (bugs, version)
- types of documents
- Naming convention

Schedule (WHEN?)

- Establishes required coordination of SCM activities with other activities in the project

Resources (HOW?)

- Identifies tools and physical and human resources required for the execution of the Plan

Plan maintenance

- Identifies how the Plan will be kept current while in effect

> 828-2012 - IEEE Standard for Configuration Management in Systems and Software Engineering https://ieeexplore.ieee.org/document/6170935

Questions?

- References
  - https://git-scm.com/book/en/v2

> Ensure your are familiar with this excellent book!