AGILE SOFTWARE ENGINEERING CONFIGURATION MANAGEMENT & DEVOPS

JOHN STOUBY PERSSON

AALBORG UNIVERSITY
DENMARK

Last lecture’s objectives

Knowledge about testing in software engineering

Skills in organizing and conducting software test processes.

Competencies to manage testing activities in agile software engineering.

Lecture objectives

Knowledge about the software engineering problems that configuration management and DevOps helps to solve.

Skills in defining and managing processes for changes, versions, builds, and releases of software.

Competencies for managing software configurations in software engineering.

Configuration Management (CM)

… is concerned with the policies, processes and tools for managing changing software systems.

Involves four closely related activities:

1. **Change management**
2. **Version management**
3. **System building**
4. **Release management**

- Plan-driven vs agile CM

- Key terms: Branch, Merge, Baseline (not as defined by GIT!)

1. Change Management

… Change is inevitable.

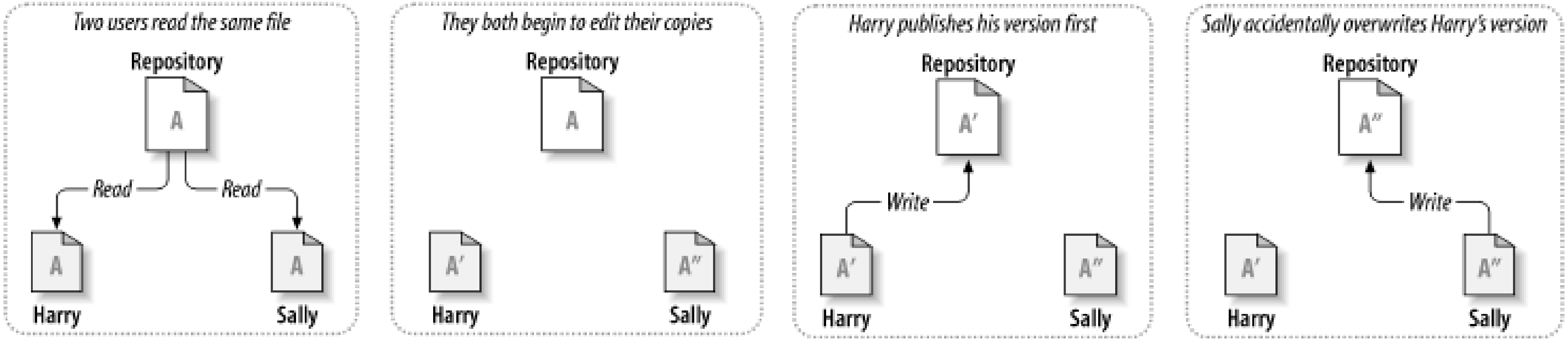
1. Keep track of requests for change to software from customers and developers
2. Work out the cost and impact of making these changes
3. Decide if and when changes should be implemented

Often submitted in a change request form.

Changes are managed by a Change Control Board (CCB) or product development group.

2. Version Control

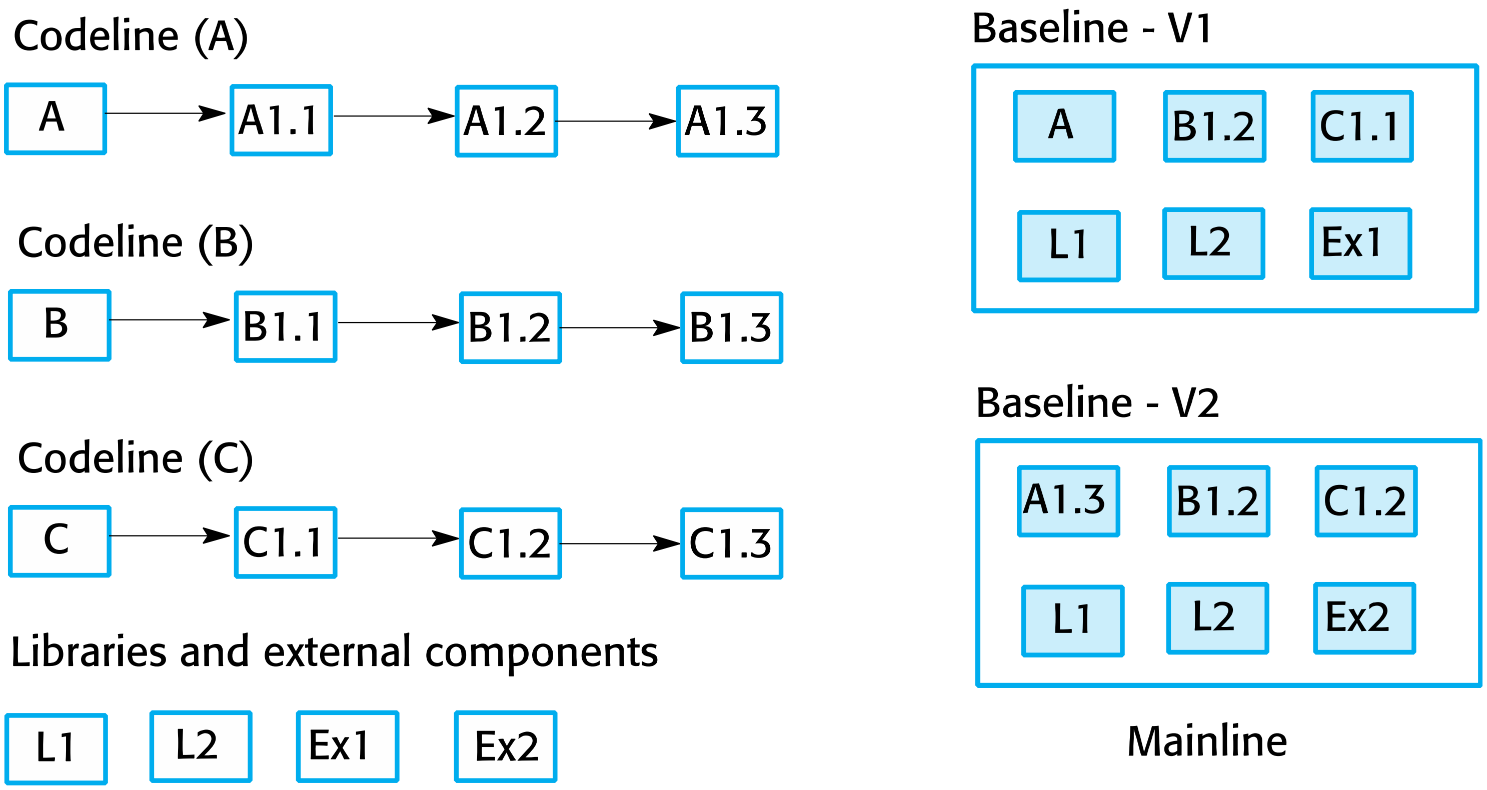
Problem: Two or more people need to work on the same file


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

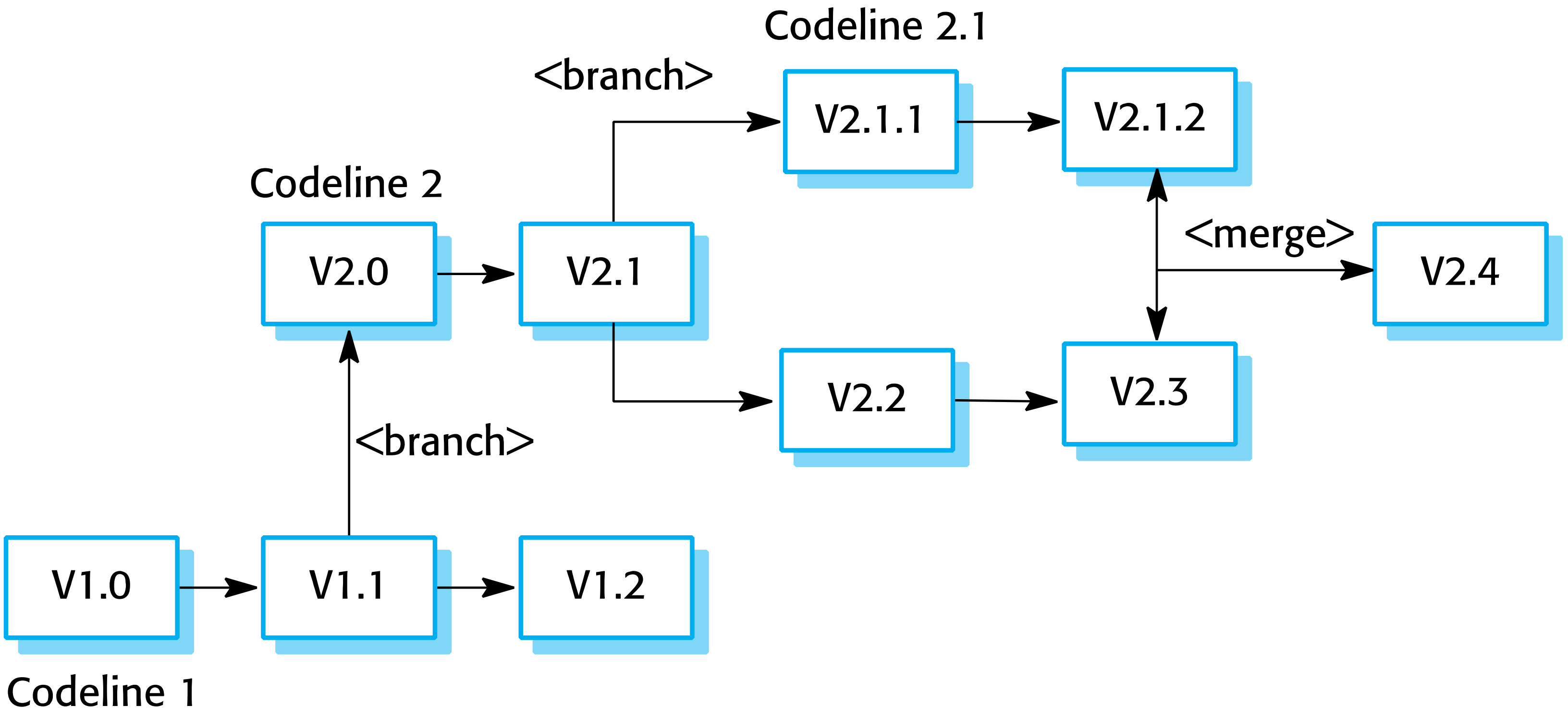
Solutions:

- Pessimistic: file locking
- Optimistic: version merging

2. Version Control: Codelines and baselines



2. Version Control: Branching and merging



Configuration Management terminology (Differs from Git)

Term	Explanation
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.
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.
Codeline	A codeline is a set of versions of a software component and other configuration items on which that component depends.
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.
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.
Mainline	A sequence of baselines representing different versions of a system.

CM terminology

Term	Explanation
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.
Release	A version of a system that has been released to customers (or other users in an organization) for use.
Repository	A shared database of versions of software components and meta-information about changes to these components.
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.
Version	An instance of a configuration item that differs, in some way, from other instances of that item. Versions always have a unique identifier.
Workspace	A private work area where software can be modified without affecting other developers who may be using or modifying that software.

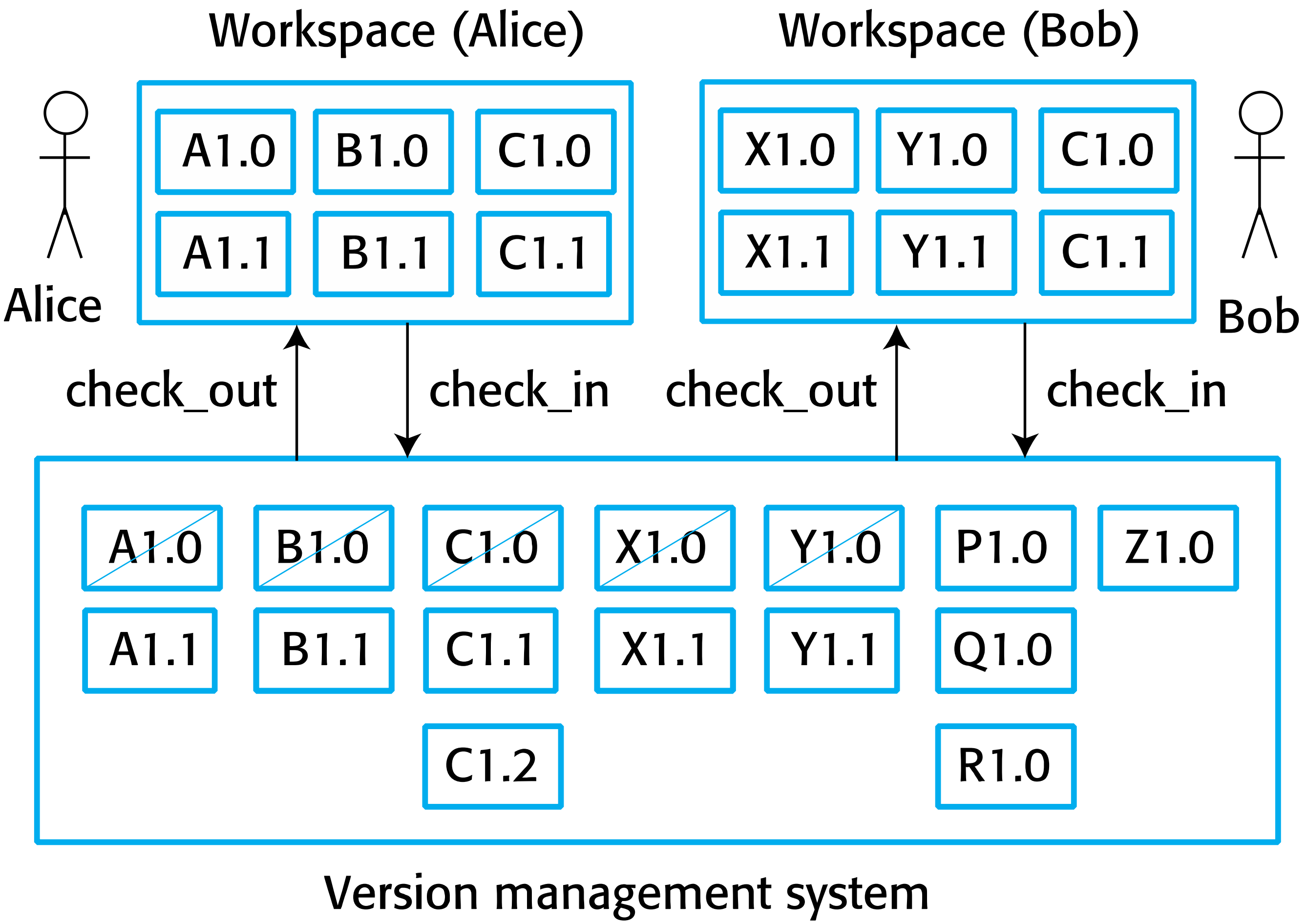
2. Version Control: Systems

Version control (VC) systems identify, store and control access to the different versions of components. There are two types of modern version control system

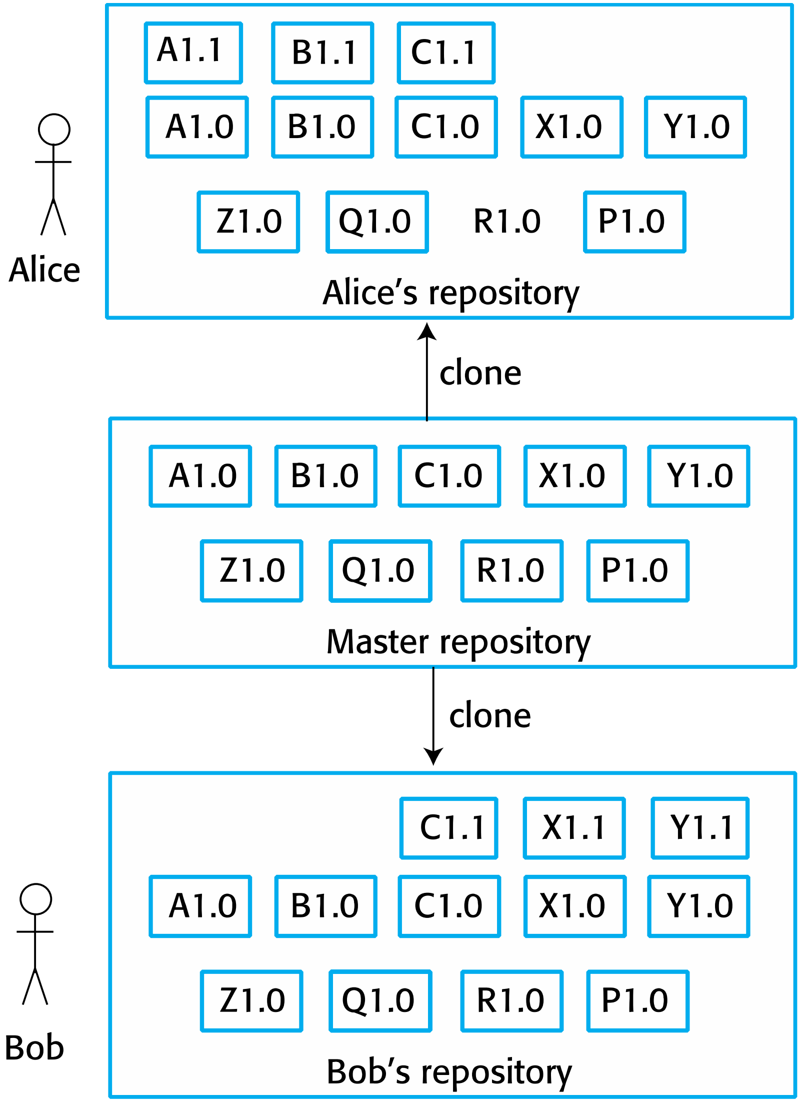
- Centralized systems, where there is a single master repository that maintains all versions of the software components that are being developed. Subversion is a widely used example of a centralized VC system.

- Distributed systems, where multiple versions of the component repository exist at the same time. Git is a widely-used example of a distributed VC system.

2. Version Control: Centralized Check-in/Check-out



2. Version Control: Decentralized repository cloning



Alice

A1.1 B1.1 C1.1

A1.0 B1.0 C1.0 X1.0 Y1.0

Z1.0 Q1.0 R1.0 P1.0

Alice’s repository

clone

A1.0 B1.0 C1.0 X1.0 Y1.0

Z1.0 Q1.0 R1.0 P1.0

Master repository

clone

C1.1 X1.1 Y1.1

A1.0 B1.0 C1.0 X1.0 Y1.0

Z1.0 Q1.0 R1.0 P1.0

Bob’s repository

Bob

2. Version Control: Benefits of distributed version control

It provides a backup mechanism for the repository.

- If the repository is corrupted, work can continue, and the project repository can be restored from local copies.

It allows for off-line working so that developers can commit changes if they do not have a network connection.

Project support is the default way of working.

- Developers can compile and test the entire system on their local machines and test the changes that they have made.

Essential in open source development! Why?

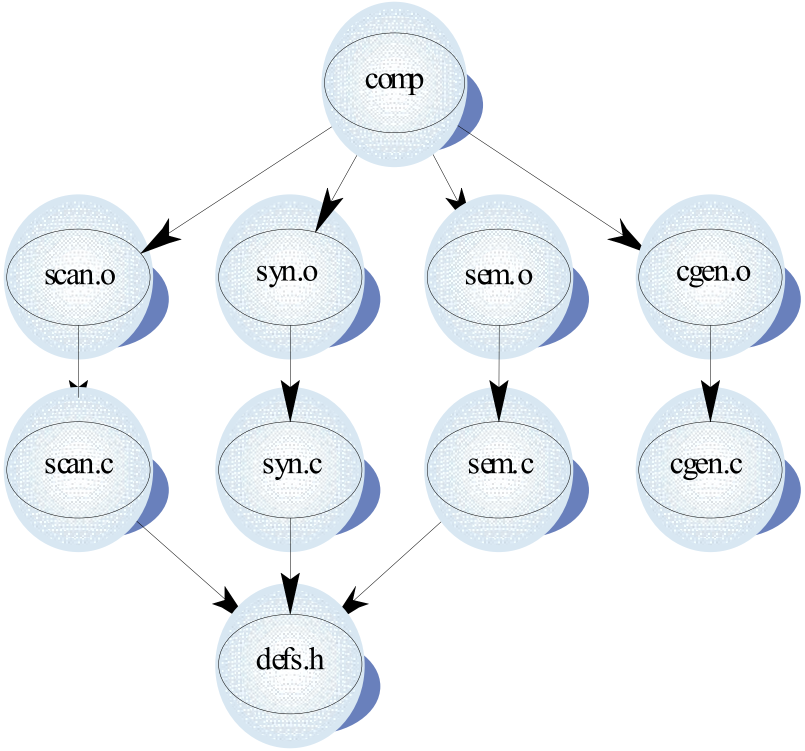
3. System Building: Tools

Building a large system is computationally expensive and may take several hours

Hundreds of files may be involved

System building tools may provide:

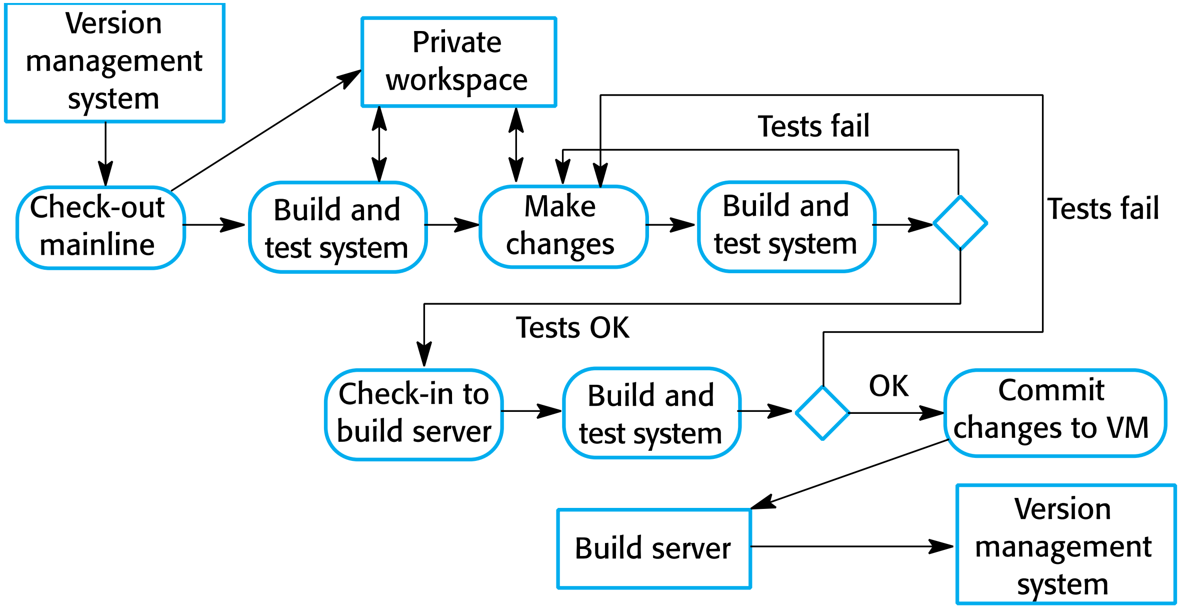
- a dependency specification language and interpreter
- tool selection and instantiation support
- distributed compilation



Examples:

Scons, Jenkins CI

3. System Building: Continuous integration



3. System Building: Daily building or (weekly, monthly, …)

The development organization sets a delivery time (say 2 p.m.) for system components.

- If developers have new versions of the components that they are writing, they must deliver them by that time.
- A new version of the system is built from these components by compiling and linking them to form a complete system.
- This system is then delivered to the testing team, which carries out a set of predefined system tests
- Faults that are discovered during system testing are documented and returned to the system developers. They repair these faults in a subsequent version of the component.

4. Release Management

… A system release is a version of a software system that is distributed to customers.

1. The release must be documented to ensure that it can be re-created
2. A release Baseline is that documentation. It includes all versions of codefiles, libraries, documentation as it was when the release was created

NOTE: This is the definition of ”baseline” in context of configuration management. The word baseline is used in many other context, baseline cost in project planning. GIT also uses the term baseline for something else.

Plan-driven vs agile configuration management

Plan-driven configuration management

- Configuration management is the organizational and tool supported management of change to software products
- Formal document naming schemes and management in a database
- The configuration database should record information about changes and change requests
- A consistent scheme of version identification should be established using version numbers, attributes or change sets
- System releases include executable code, data, configuration files and documentation
- System building involves assembling components into a system
- Tools are available to support all CM activities
- Tools may be stand-alone or may be integrated systems which integrate support for version management, system building and change management

Plan-driven and agile configuration management

	Plan-driven	**Agile**
Focus	documents and code	code
Activities	CM practice, change management, version control, automated build	version control, automated build
Responsible	CM team, CM board	programmers
Process	formal, managed	informal and integrated with practice environment
Outcome	CM audit (documented product control)	next release
Importance	indispensible in medium and large projects	indispensible

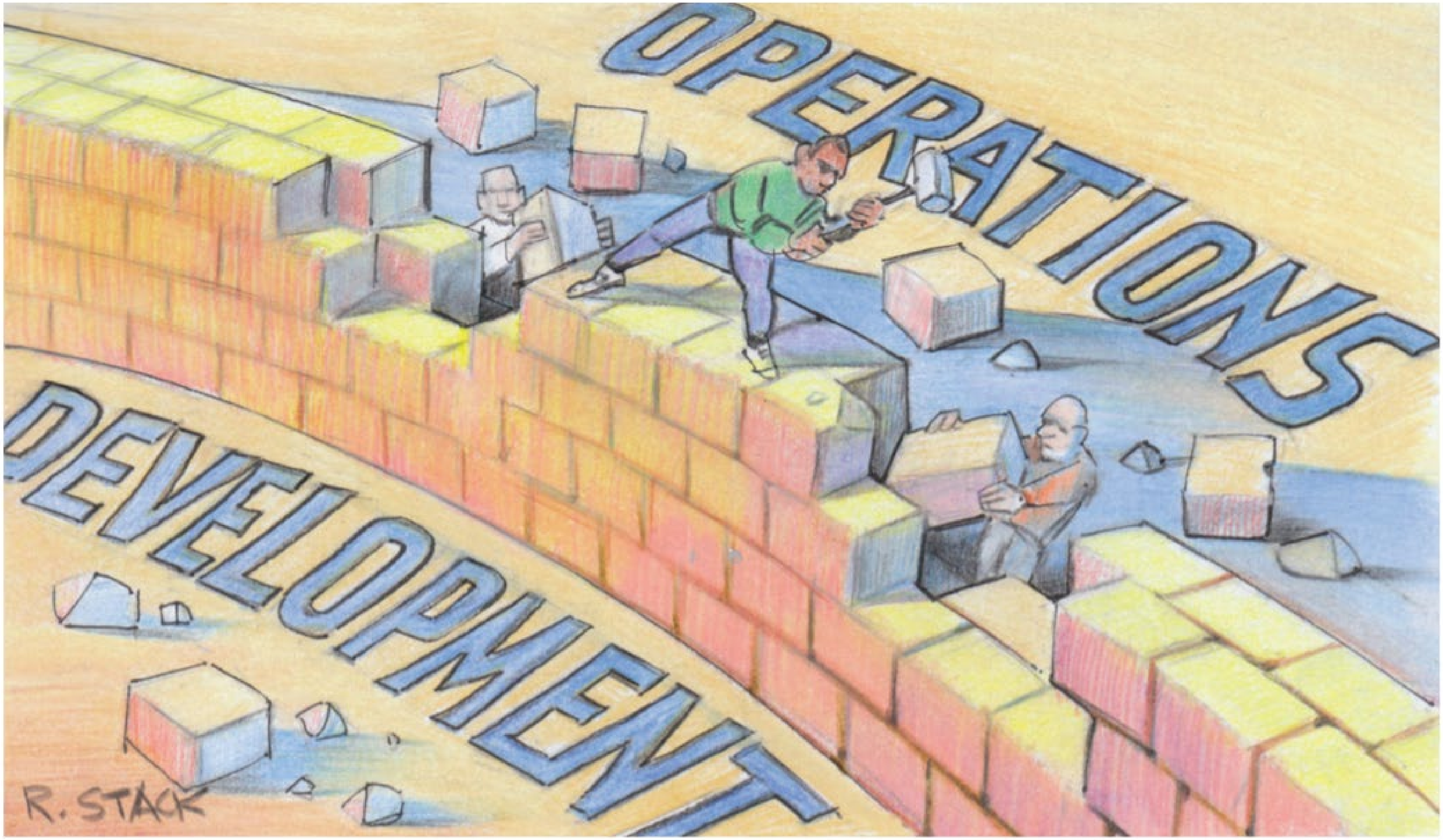
Agile development and CM

Agile development, where components and systems are changed several times per day, is impossible without using CM tools.

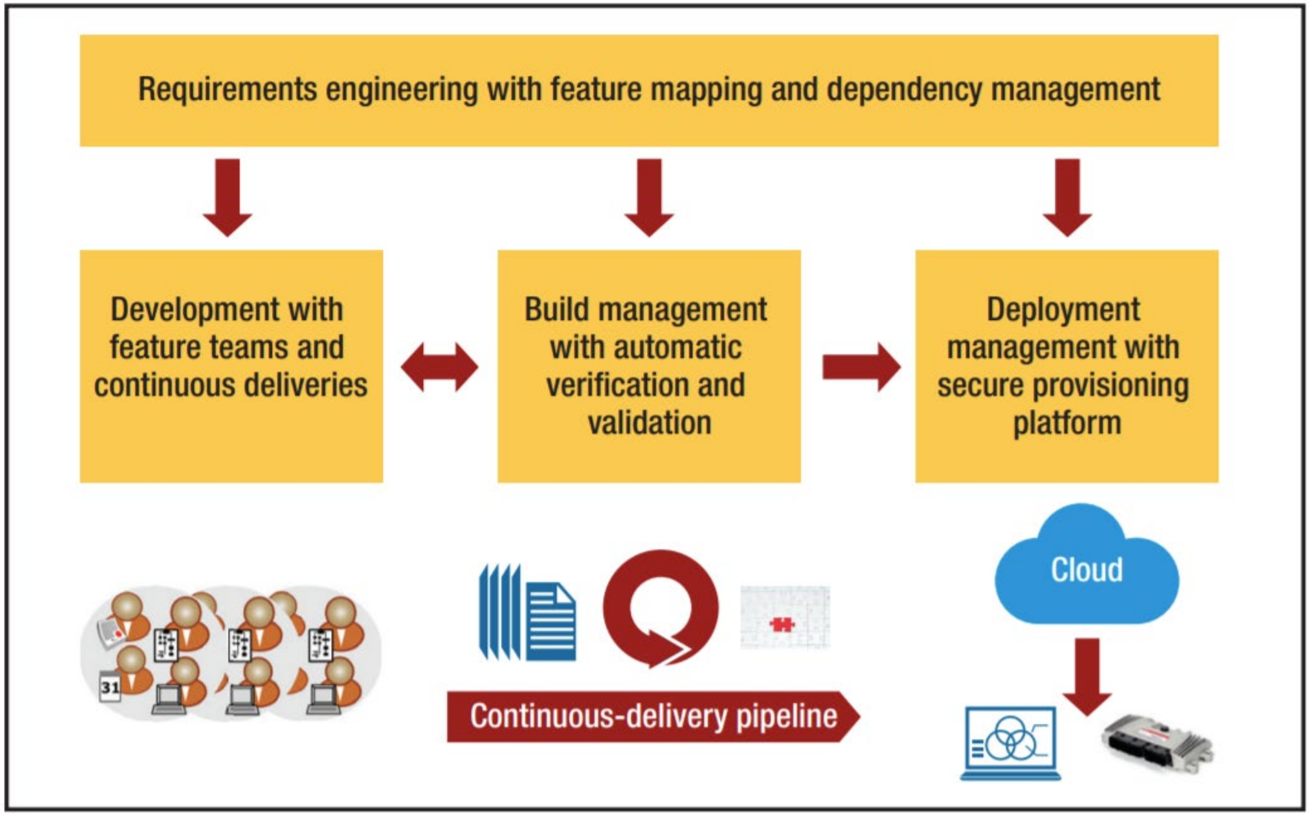
The definitive versions of components are held in a shared project repository and developers copy these into their own workspace.

They make changes to the code then use system building tools to create a new system on their own computer for testing. Once they are happy with the changes made, they return the modified components to the project repository.

DevOps



DevOps



Requirements engineering with feature mapping and dependency management

Development with feature teams and continuous deliveries

Build management with automatic verification and validation

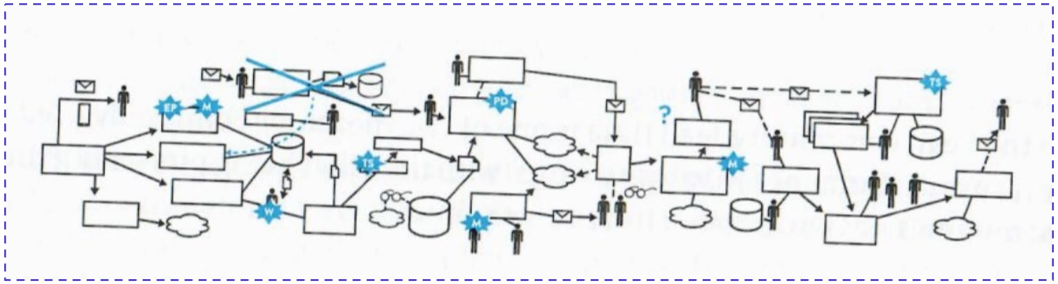
Deployment management with secure provisioning platform

FIGURE 1. The generic DevOps production and delivery process. It aims to better integrate the development, production, and operations business processes with adequate technology.

(Ebert et al. 2016, p. 95)

DevOps and LEAN process thinking

Value stream



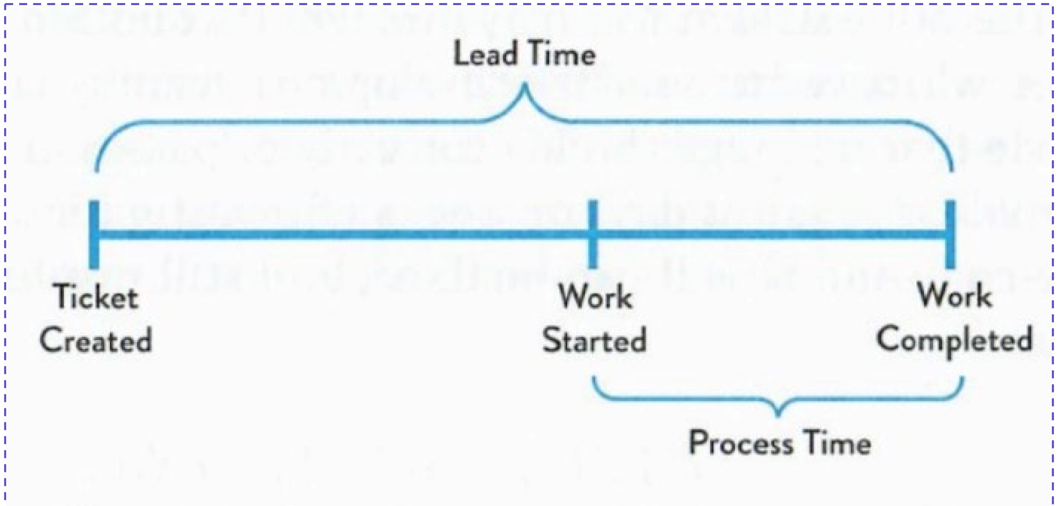
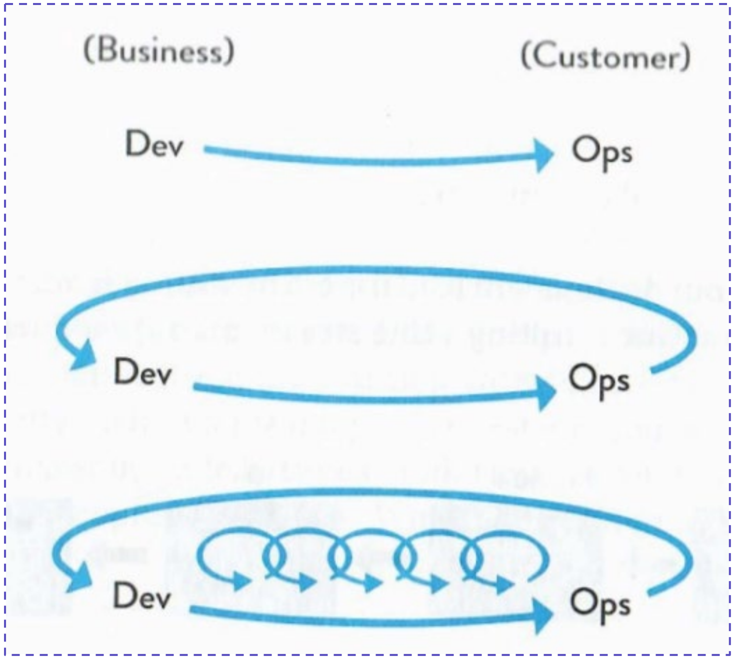


Figure 2. *Lead time vs. process time of a deployment operation*



Group exercises

Exercise 1: Configuration management in your project.

Lecture objectives

Knowledge about the software engineering problems that configuration management and DevOps helps to solve.

Skills in defining and managing processes for changes, versions, builds, and releases of software.

Competencies for managing software configurations in software engineering.