Configuration Management

Topics

- Change management
- Version management
- System building
- Release management

Configuration management

- Reminder: software **changes** frequently
- Systems are a set of versions
- Each version has to be
  - maintained
  - managed

Versions

- Version implementation
  - proposals for change
  - corrections of faults
  - adaptations for different hardware and operating systems.

Configuration management (CM)

- Policies
- Processes
- Tools

for managing changing software systems.

- CM helps **track** which **changes** and **component** versions have been incorporated into each system **version**.

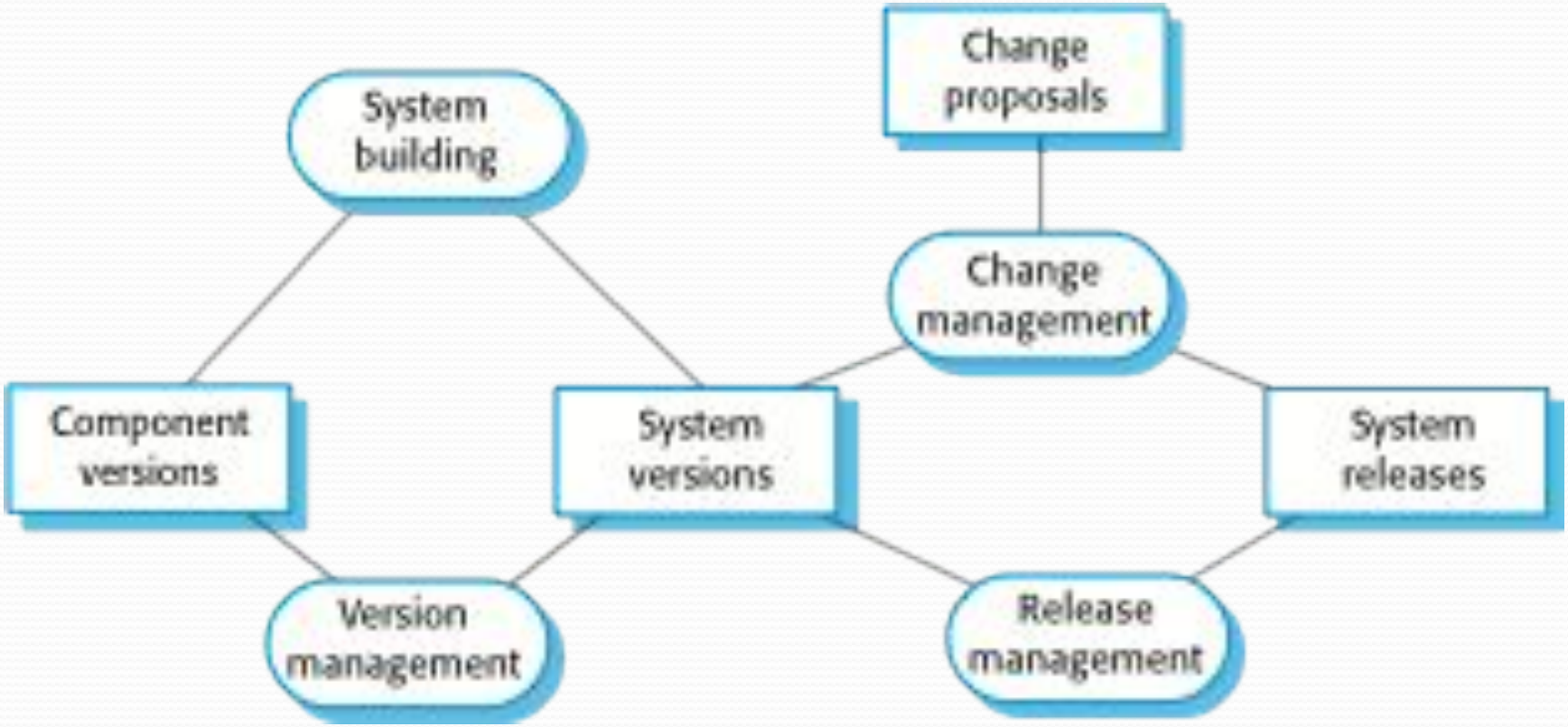
CM activities

- **Change management**
  - Track requests for changes
  - Estimate costs and impact of changes
  - Decide the changes to be implemented
- **Version management**
  - Track multiple versions of system components
  - Ensure that changes made to components by different developers do not interfere with each other.

CM Activities (‘cont).

- **System building**
  - Assemble
    - program components
    - data
    - libraries
  - Compile to create an **executable system**
- **Release management**
  - Prepare software for external release
  - Track system versions released for customer use

Configuration management activities



CM terminology

Version

- An instance of a configuration item
- Differs, in some way, from other instances of that item
- Have a unique identifier, i.e.
  - configuration item name + version number.

Configuration control

- The process of ensuring that versions of systems and components are
  - Recorded
  - Maintained
- All versions of components are
  - Identified
  - Stored for the lifetime of the system.

Configuration Item (CI)

- Anything associated with a software project that has been placed under configuration control
  - design
  - code
  - test data
  - Document
  - etc.
- Configuration items have a unique name
- Different versions of a configuration item

Workspace

A **private work area** where software can be modified without affecting other developers who may be using or modifying that software.

Release

A version of a system that

has been **released** to **customers** for use.

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.

Merging

- The creation of a
  - new version of a software component
  - by merging separate versions in different codelines.

System building

- Creation of an executable system version
- By compiling and linking the appropriate versions of the components and libraries making up the system.

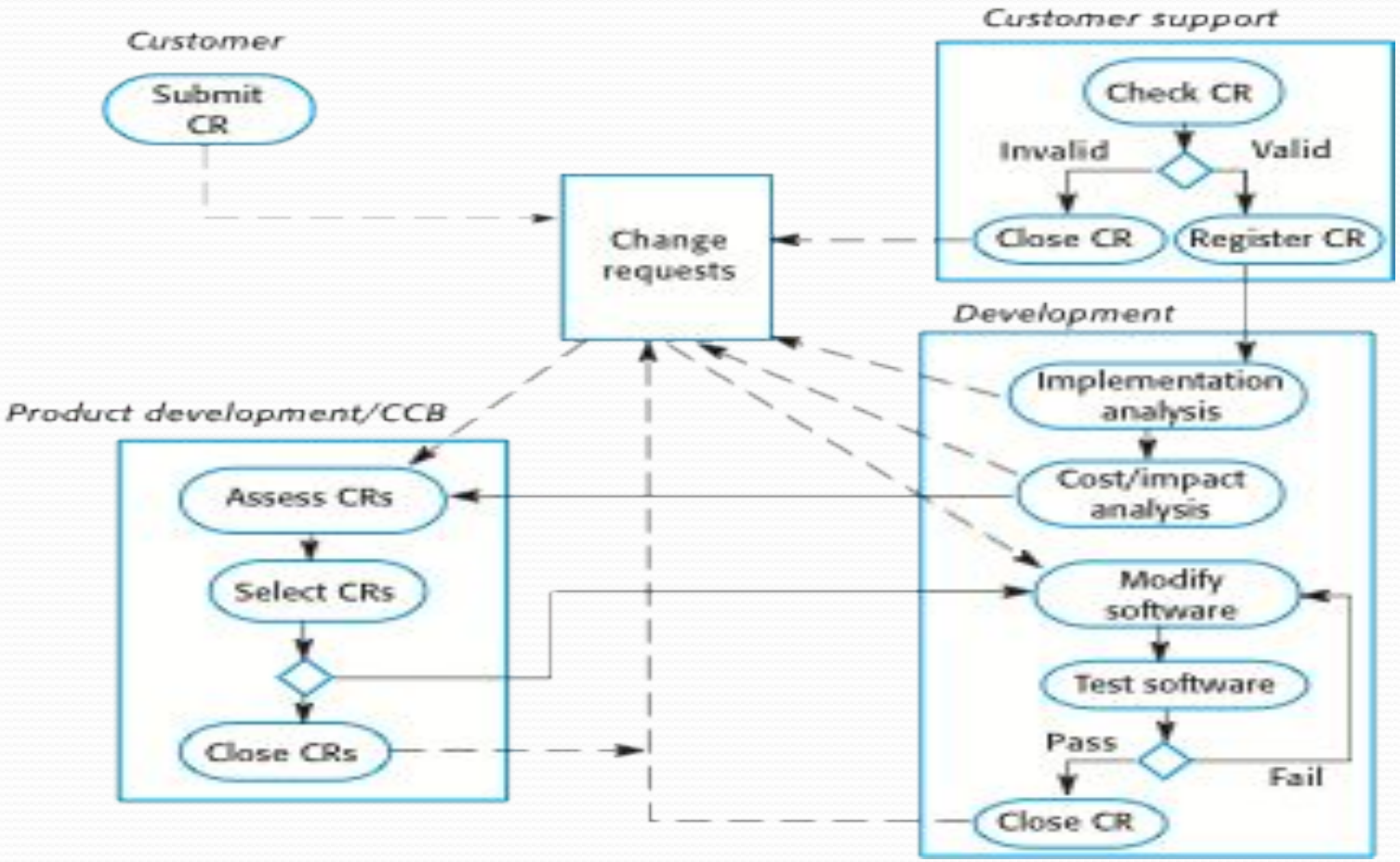
Change management

- Organizational needs and requirements change
- Bugs have to be repaired
- Adapt to changes in their environment
- Change management is managing system evolution
  - Priority to most urgent and cost-effective changes.

Change management process

- Analyze costs & benefits of proposed changes
- Approve changes that are worthwhile
- Track which components have been changed

The change management process



A partially completed change request form (a)

Change Request Form

Project: SICSA/AppProcessing **Number:** 23/02
Change requester: I. Sommerville **Date:** 20/01/12

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:** 25/01/12

Components affected: ApplicantListDisplay, StatusUpdater

Associated components: StudentDatabase

A partially completed change request form (b)

Change Request Form

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: 28/01/12 **CCB decision date:** 30/01/12
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:

Factors in change analysis

- 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

CM and agile methods

- In some agile methods
  - Customers are involved in change management
  - Propose a **change** to the **requirements**
  - Work with the team to **assess its impact**
  - Decide change priority
    - Is it more important than features planned for the next increment of the system

CM and agile methods

- Changes to improve the software
  - Decided by the programmers working on the system.
- Refactoring
  - software is continually improved
  - Not an overhead!!!
  - Necessary part of the development process.

Derivation history

```
// Ozurler project (OB 1203)
//
// APP-SYSTEM/AUTH/REPORT/USER_ROLE
//
// Object: currentRole
// Author: R. Yazar
// Creation date: 13/4/2012
//
// © Boun
//
// Modification history
// Version    ModifierDate          Change      Reason
// 1.0     A. Yazar     11/4/2012   Add header Submitted to CM
// 1.1     R. Zambak  13/4/2012   New field     Change req. R07/02
```

Baseline

- Collection of component versions of a system.
- Baselines are controlled
  - versions of the components of the system cannot be changed.
- Must be possible to **recreate** a baseline from
  - **component** versions that are included in the system
  - **libraries** used
  - **configuration files**, etc.

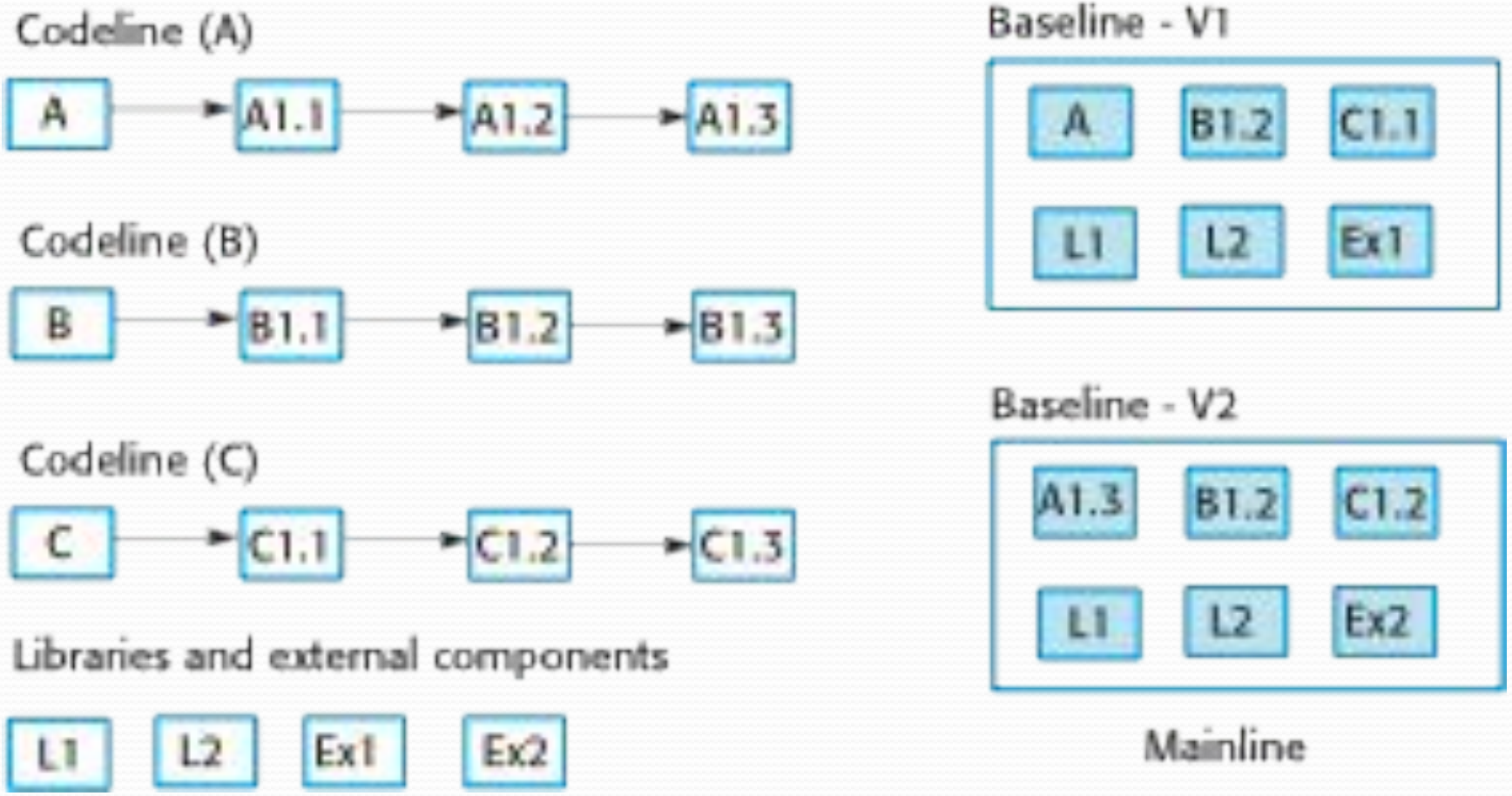
Codeline

- A codeline is
  - a set of versions of a software component
  - other configuration items on which the components depend on

Mainline

Mainline is a sequence of baselines representing different versions of a system.

Codelines and baselines



Baselines

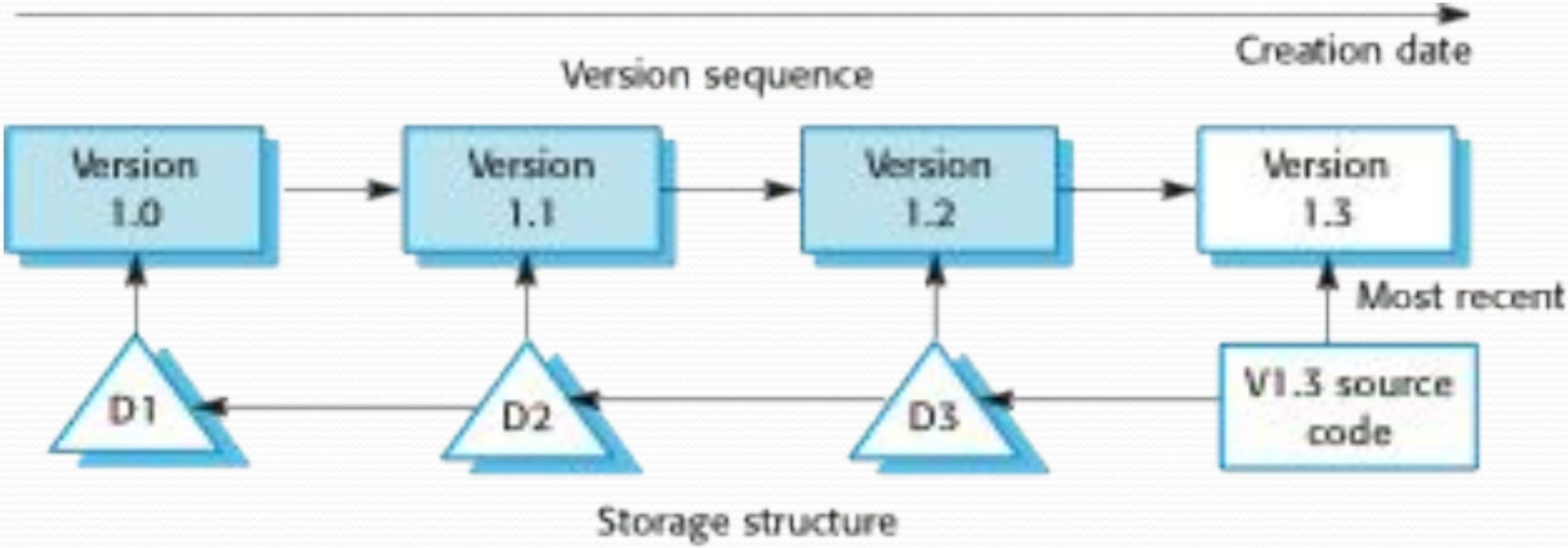
- Baselines are important because you often have to **recreate** a specific version of a complete system.

- For example a product line
  - individual versions for different customers
  - recreated for a specific customer (customer reports bugs)

Version management systems

- Version and release **identification**
- Storage management
  - Reduce storage required by multiple versions of components that differ only slightly
- Change history recording
  - All of the changes made to the code of a system or component are recorded and listed.

Storage management using deltas



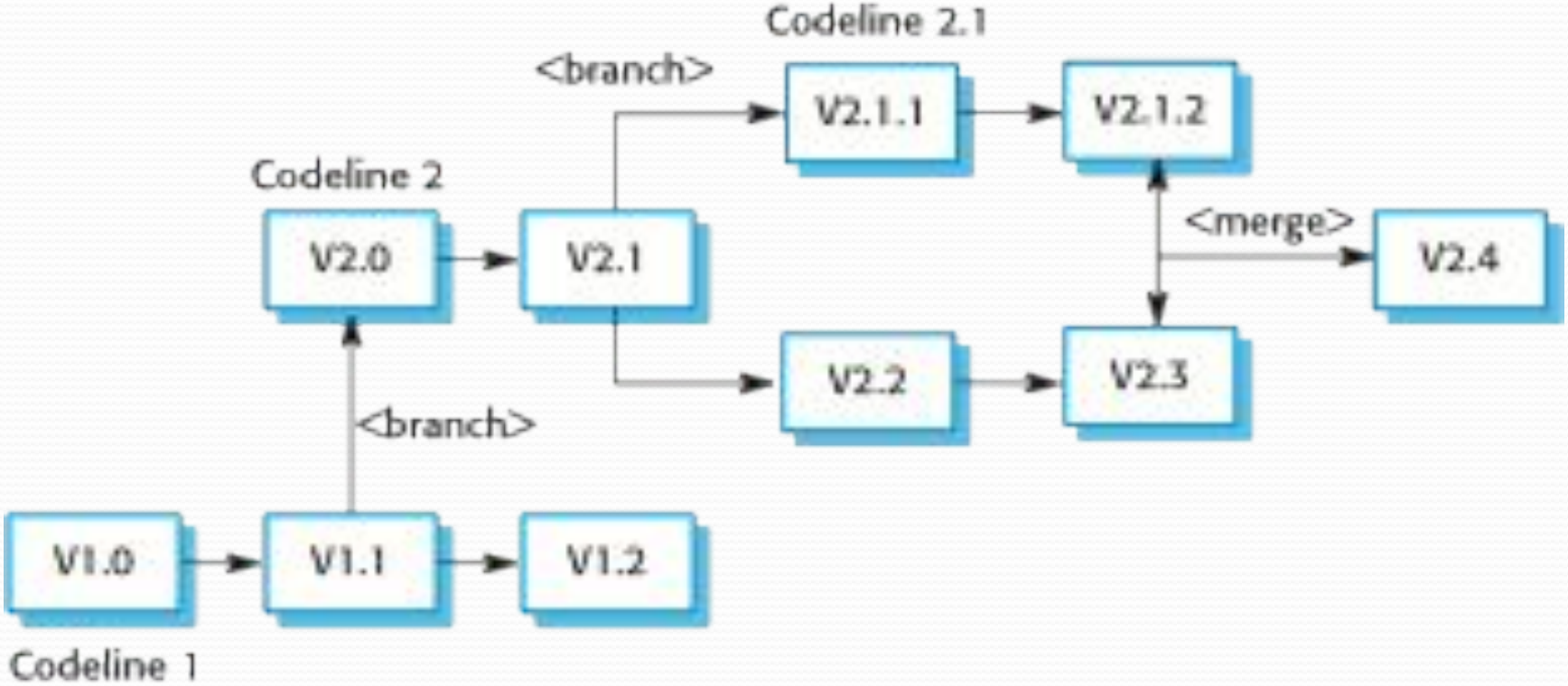
Codeline branches

- There may be several **independent** sequences
  - different developers work independently on different versions of the source code
  - change it in different ways

Codeline Merge

- When codeline branches need to be **merged**
  - create a new version of a component
  - includes all changes that have been made
  - If the **changes** made involve **different parts** of the code, the component versions may be merged **automatically** by combining the **deltas** that apply to the code.

Branching and merging



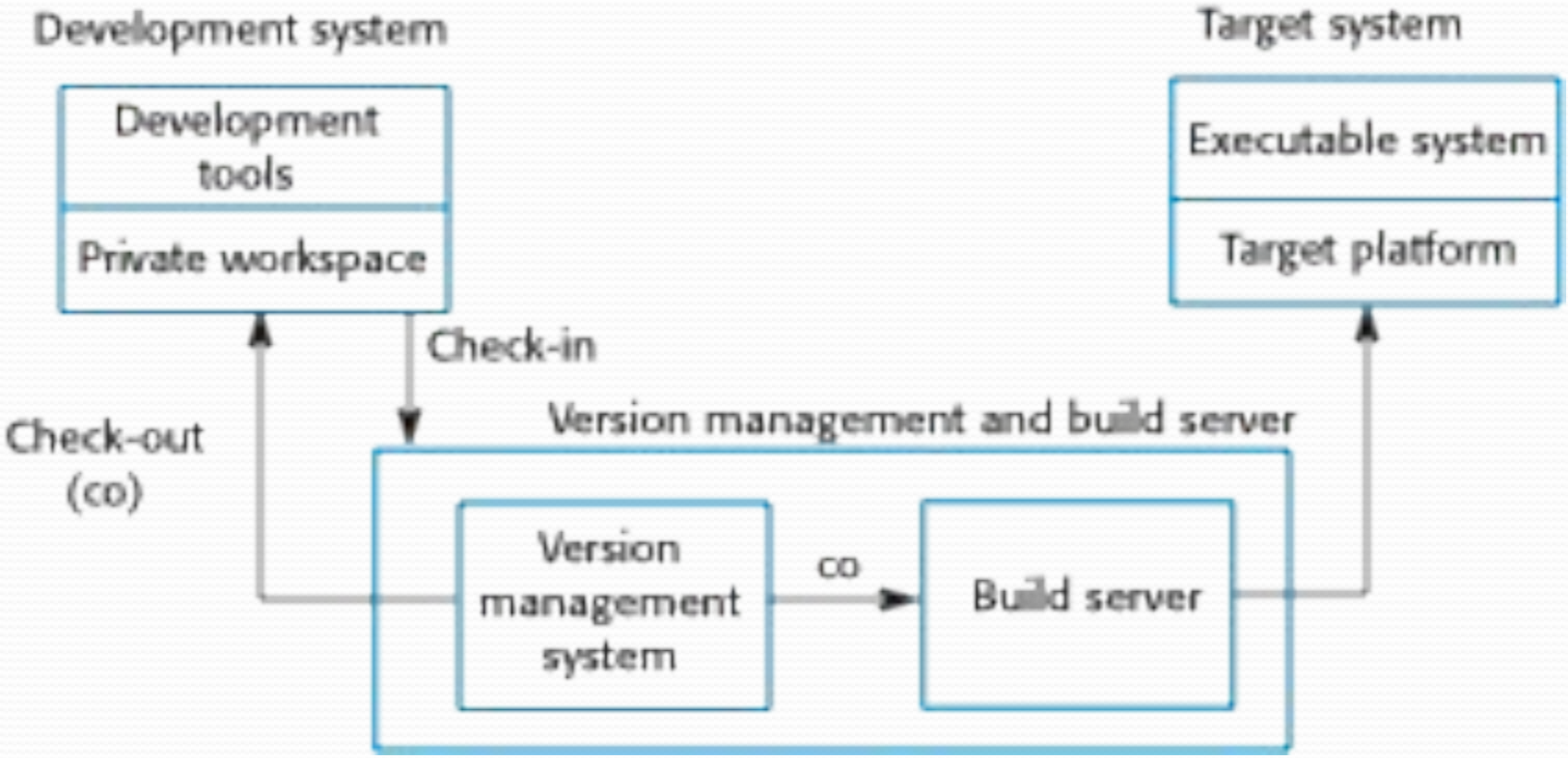
System building

- Process of creating a complete, executable system by compiling and linking the system components, external libraries, configuration files, etc.
- System building tools and version management tools must communicate as the build process involves checking out component versions from the repository managed by the version management system.
- The configuration description used to identify a baseline is also used by the system building tool.

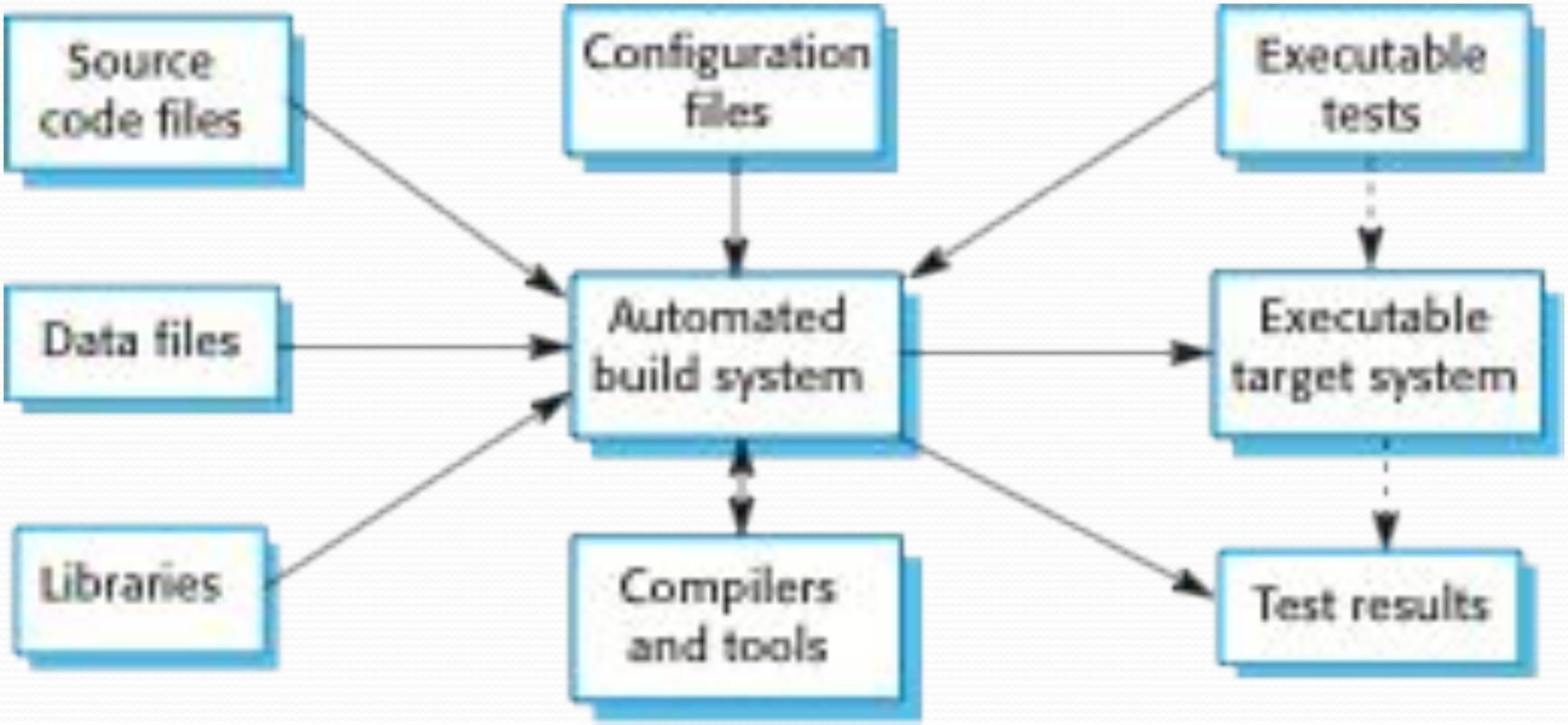
Build platforms

- The **development system**:
  - tools such as compilers, source code editors, etc.
- Developers check out code from the version management system into a **private workspace**
- The **build server** used to build executable versions of the system
- The **target environment** is the platform on which the system executes.

Development, build, and target platforms



System building



Build system functionality

- Build script generation
- Version management system integration
- Minimal re-compilation
- Executable system creation
- Test automation
- Reporting
- Documentation generation

Note: Tools like Jenkins (https://jenkins.io/) are very useful in managing the process

Minimizing recompilation

- Checks if a compiled version of a component is available. If so, doesn't recompile
- A unique signature identifies each source and object code version and is changed when the source code is edited.
- By comparing the signatures on the source and object code files, it is possible to decide if the source code was used to generate the object code component.

File identification

- Modification timestamps
  - If the source code file of a component has been modified after the related object code file → recompile
- Source code checksums
  - The signature on the source code file is a checksum calculated from data in the file. A checksum function calculates a unique number using the source text as input.
  - Source code files with different checksums are **actually** different.

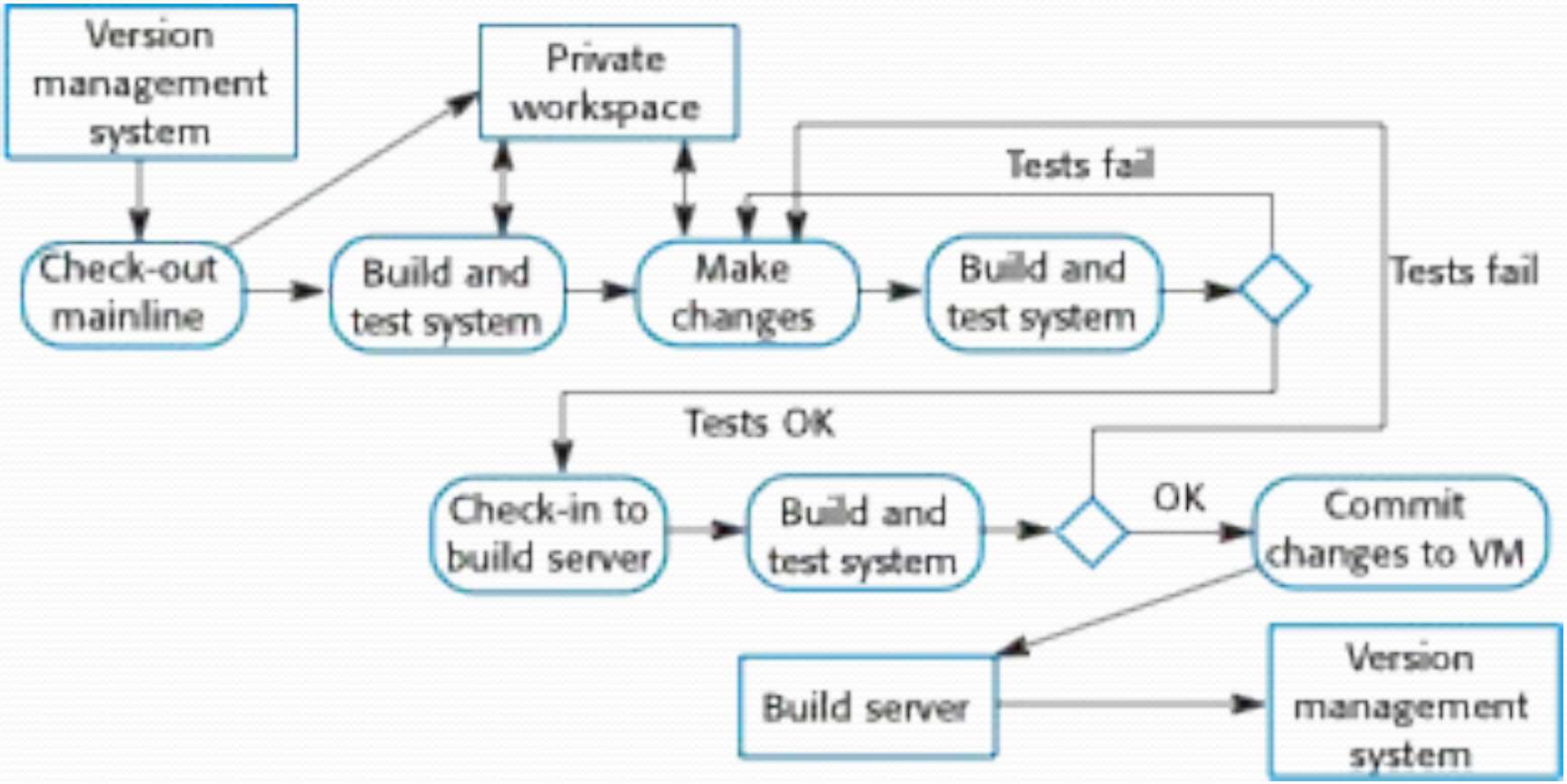
Agile building

- Check out the mainline system from the version management system into the developer's private workspace.
- Build the system
- Run automated tests to ensure that the built system passes all tests.
- If it doesn't
  - build the build is **broken**
  - inform whoever checked in the last baseline system
  - s/he is responsible for repairing the problem!
- Make the changes to the system components.
- Build the system in the private workspace and rerun system tests. If the tests fail, continue editing.

Agile building

- When the system has passed its tests
  - check it into the build system
  - do not commit it as a new system baseline.
  - build the system on the build server
  - run the tests, if fails
    - check out the components that have failed
    - Edit so that tests pass on your private workspace.
- If the system passes its tests on the build system, then commit the changes you have made as a new baseline in the system mainline.

Continuous integration



Daily building

- The development organization sets a delivery time (say 2 p.m.) for system components.
  - If developers have new versions of the components, they must deliver them by that time.
  - A new version of the system is built from these components
  - 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.
  - These faults are fixed in a subsequent version of the component.

Release management

- A system release is a version of a software system that is **distributed to customers**.
- For mass market software
  - major releases which deliver significant new functionality
  - minor releases – bug repair 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.

Release tracking

- In the event of a problem, it may be necessary to reproduce exactly the software that has been delivered to a particular customer.
- When a system release is produced, it must be documented to ensure that it can be re-created exactly in the future.
- This is particularly important for customized, long-lifetime embedded systems, such as those that control complex machines.
  - Customers may use a single release of these systems for many years and may require specific changes to a particular software system long after its original release date.

Release reproduction

- Document the specific versions of the source code components that were used to create the executable code.
- Must keep copies of the source code files, corresponding executables and all data and configuration files.
- Record the versions of the operating system, libraries, compilers and other tools used to build the software.

Release planning

- As well as the technical work involved in creating a release distribution, advertising and publicity material have to be prepared and marketing strategies put in place to convince customers to buy the new release of the system.
- Release timing
  - If releases are too frequent or require hardware upgrades, customers may not move to the new release, especially if they have to pay for it.
  - If system releases are too infrequent, market share may be lost as customers move to alternative systems.

Release components

- In addition to executable code, a release may also include:
  - configuration files defining how the release should be configured for particular installations;
  - data files, such as files of error messages, that are needed for successful system operation;
  - an installation program that is used to help install the system on target hardware;
  - electronic and paper documentation describing the system;
  - packaging and associated publicity that have been designed for that release.

Factors influencing system release planning

- Technical quality of the system
  - If serious system faults are reported
    - fault repair release
  - Minor system faults
    - may repair by issuing patches
    - applied to current release
    - distributed over the Internet

What we learned

- Software consists of ever changing components
- Configuration management is essential in controlling and tracking quality software