Tim Pengajar IF2250

IF2250 – Rekayasa Perangkat Lunak

Software Configuration Management

SEMESTER II TAHUN AJARAN 2023/2024

KNOWLEDGE & SOFTWARE ENGINEERING

Configuration Management

- Because software changes frequently, systems, can be thought of as a set of versions, each of which has to be maintained and managed.
- Versions implement proposals for change, corrections of faults, and adaptations for different hardware and operating systems.
- Configuration management (CM) is concerned with the policies, processes and tools for managing changing software systems. You need CM because it is easy to lose track of what changes and component versions have been incorporated into each system version.

Software Configuration Items (SCI)

- Computer programs
  - both source and executable
- Documentation
  - both technical and user
- Data
  - contained within the program or external to it

Configuration Management System Elements

- Component elements
  - set of tools coupled within a file management system to enable access to and management of each SCI
- Process elements
  - collection of procedures that define an effective approach to change management for all stakeholders
- Construction elements
  - set of tools that automates the construction of software to ensure a set of validated components is assembled
- Human elements
  - team uses a set of tools and process features encompassing other CM elements

KNOWLEDGE & SOFTWARE ENGINEERING

Baselines

- A baseline is a milestone in software development that is marked by the delivery of one or more configuration items.
- A work product becomes a baseline only after it is reviewed and approved.
- Once a baseline is established each change request must be evaluated and verified by a formal procedure before it is processed.

SCM Repository Functions

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

SCM Tool Features

- Versioning
  - control changes to all work products before and after release to customer
- Dependency tracking and change management
  - tracking relationships among multiple versions of work products to enable efficient changes (link management)
- Requirements tracing
  - depends on link management, provides the ability to track all work products that result from a specific requirements specification (forward tracing) and to identify which requirement generated by any given work product (backward tracing)
- Configuration management
  - works closely with link management and versioning facilities to keep track of a series of configurations representing project milestones or production releases
- Audit trails
  - establishes additional information about when, where, why, and by whom changes were made

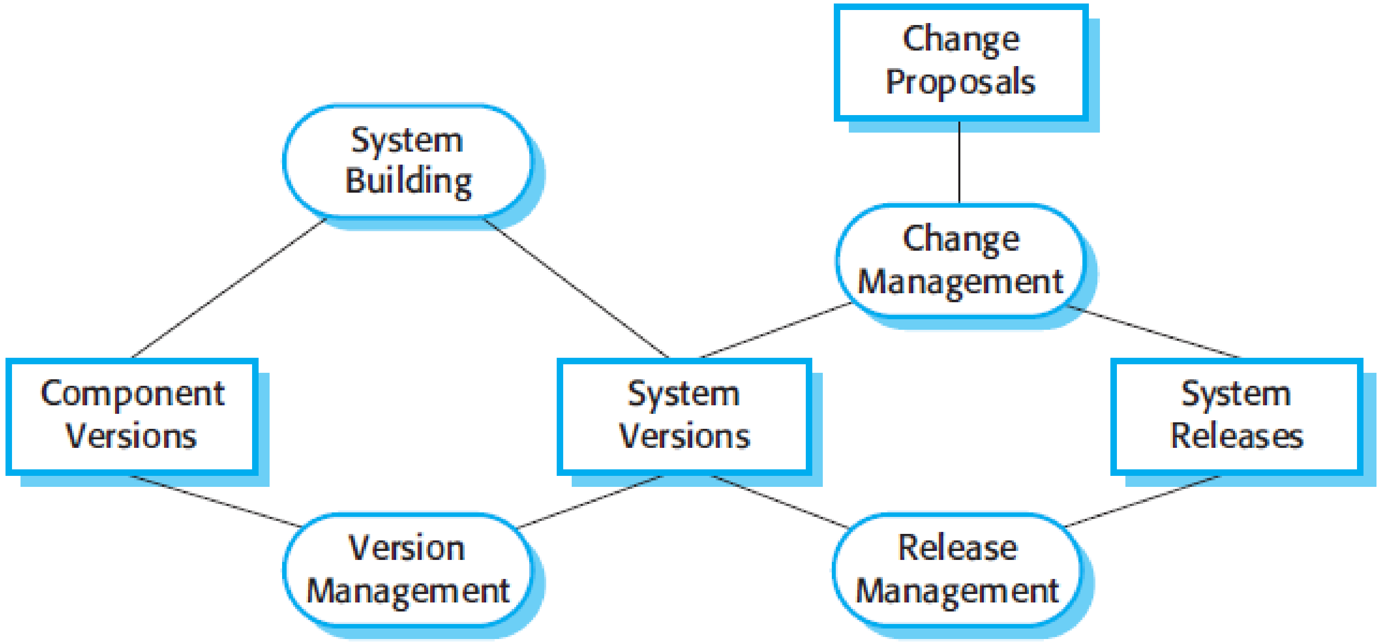
SCM Process Objectives

- **Identify all items** that define the software configuration
- **Manage changes** to one or more configuration items
- **Facilitate construction of different versions** of a software application
- **Ensure that software quality is maintained** as configuration evolves

SCM Tasks

- Identification
    - tracking multiple versions to enable efficient changes
- Version control
    - control changes before and after release to customer
- Change control
    - authority to approve and prioritize changes
- Configuration auditing
    - ensure changes made properly
- Reporting
    - tell others about changes made

SCM activities

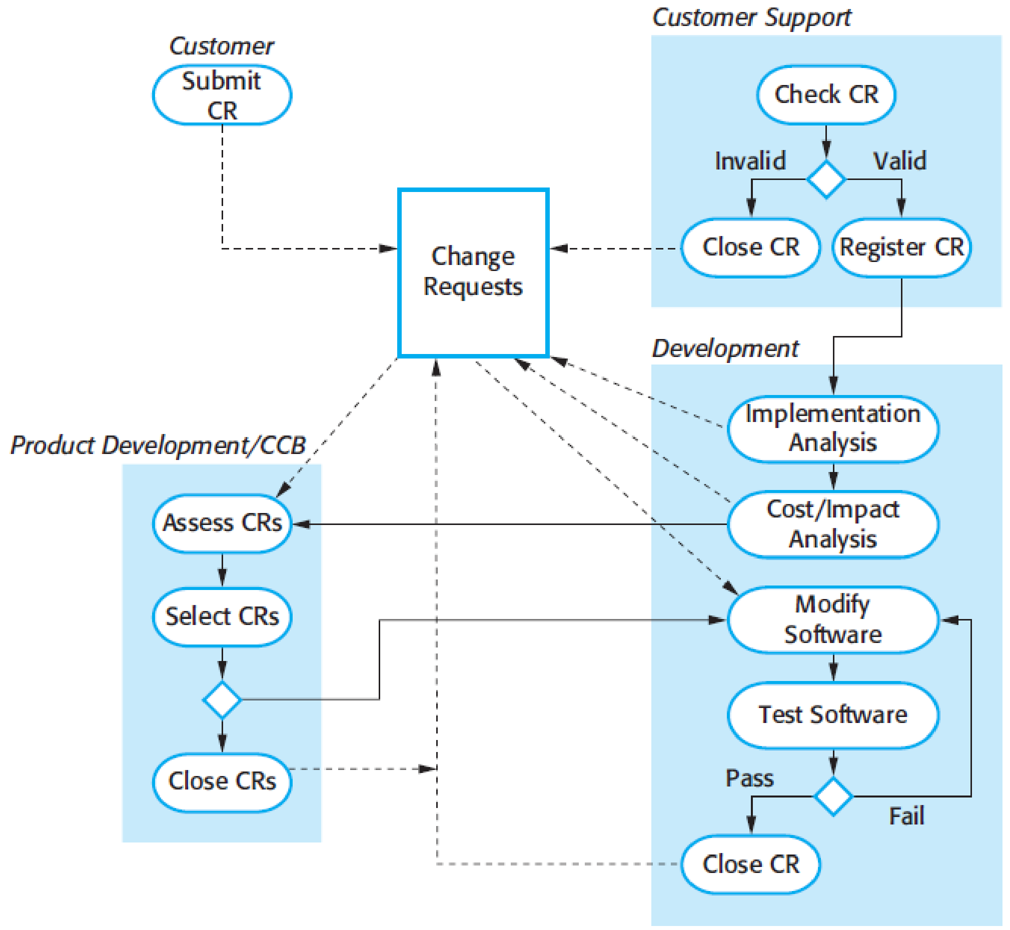


SCM activities

- *Change management*
  - keeping track of requests for changes from customers and developers, working out the costs and impact of making these changes, and deciding if and when the changes should be implemented
- *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
- *System building*
  - assembling program components, data, and libraries, and then compiling and linking these to create an executable system
- *Release management*
  - preparing software for external release and keeping track of the system versions that have been released for customer use

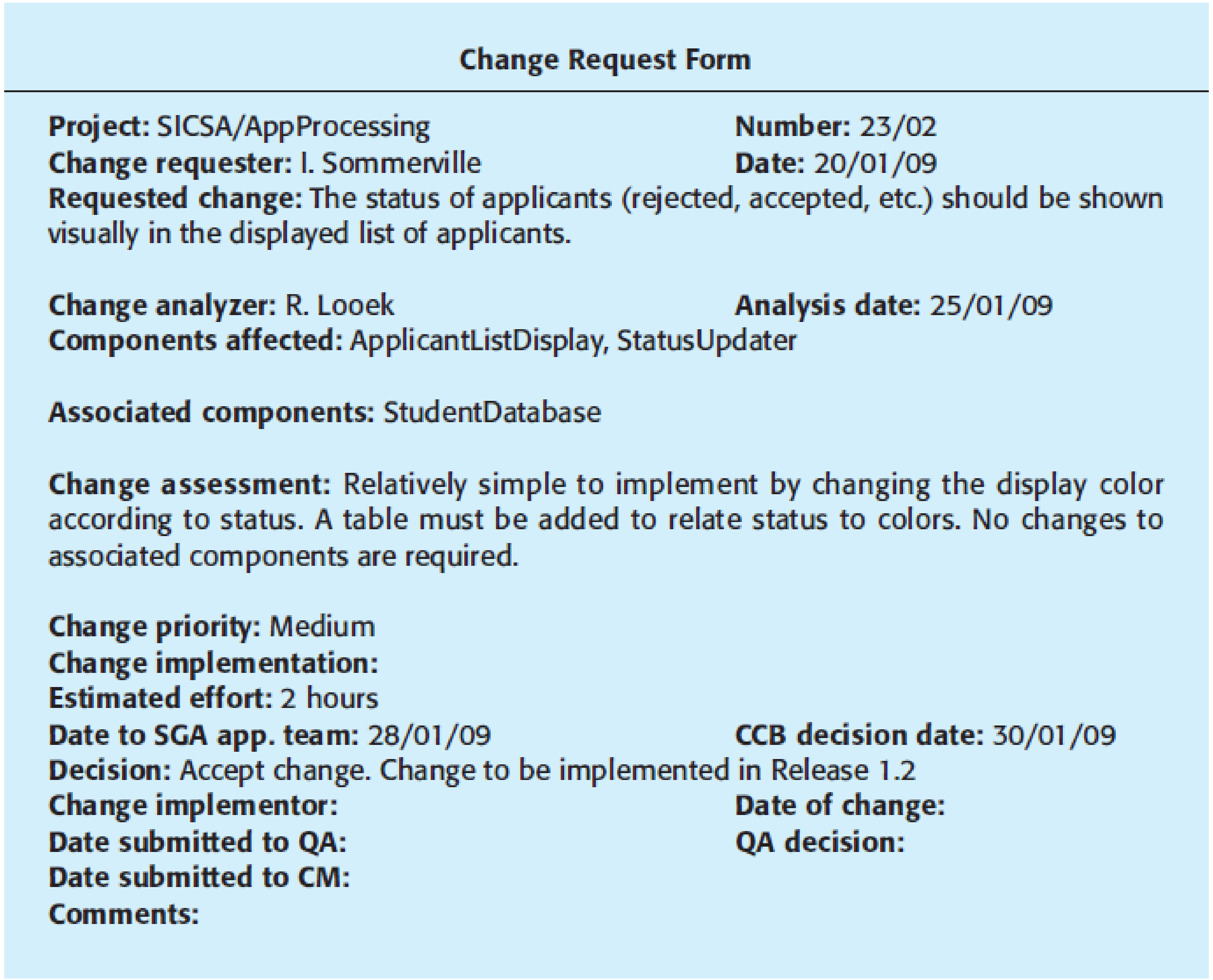
Change management process

used after a system has been released to customers



Change management process (2)

- Checking, approving, costing…
- Initiated when a 'customer' completes and submits a change request
  - a bug report, where the symptoms of the bug are described,
  - a request for additional functionality to be added to the system
- Submitted using a ***change request form*** (CRF)
- Checked to ensure that it is valid
  - not all change requests require action
- The required changes to the system modules are assessed
- The cost of making the change is estimated



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:

Change management process (3)

- The CCB or product development group considers the impact of the change from a strategic and organizational view
- Significant 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*

Change management for software products

- Change requests come from the customer support team, the company marketing team and the developers
- The customer support team may submit change requests associated with bugs that have been discovered and reported by customers
  - Customers may use a webpage or e-mail to report bugs.
  - A bug management team translates them into formal system change requests
- Marketing staff meet with customers and investigate competitive products
  - They may suggest changes that should be included to make it easier to sell a new version of a system to new and existing customers
- The system developers themselves may have some good ideas about new features that can be added to the system

Derivation history

- As the development team changes software components, they should maintain a record of the changes made to each component, called the derivation history of a component

```
// SICSA project (XEP 6087)
//
// APP-SYSTEM/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: R. Looek
// Creation date: 13/11/2009
//
// © St Andrews University 2009
//
// Modification history
// Version   Modifier   Date         Change       Reason
// 1.0       J. Jones   11/11/2009   Add header   Submitted to CM
// 1.1       R. Looek   13/11/2009   New field    Change req. R07/02
```

Change management in agile methods

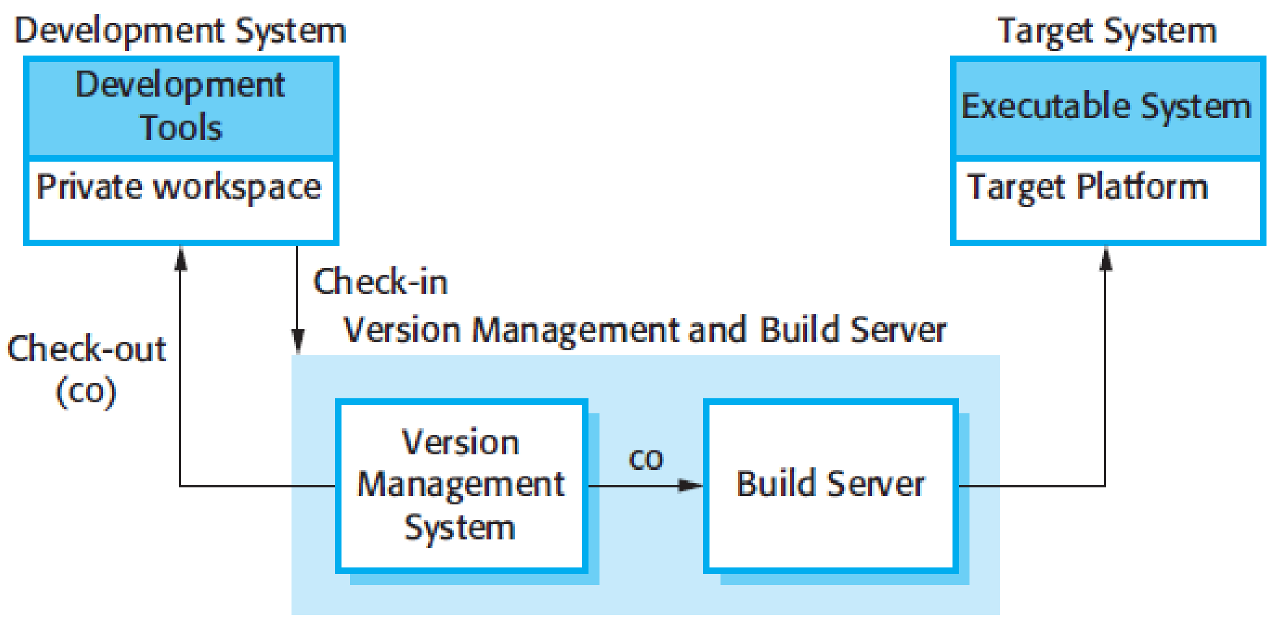
- Customers are directly involved in deciding whether a change should be implemented
  - When they propose a change to the system requirements, they work with the team to assess the impact of that change and then decide whether the change should take priority over the features planned for the next increment of the system.
- Changes that involve software improvement are left to the discretion of the programmers working on the system
  - Refactoring, where the software is continually improved, is not seen as an overhead but rather as a necessary part of the development process.

System building

- The 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

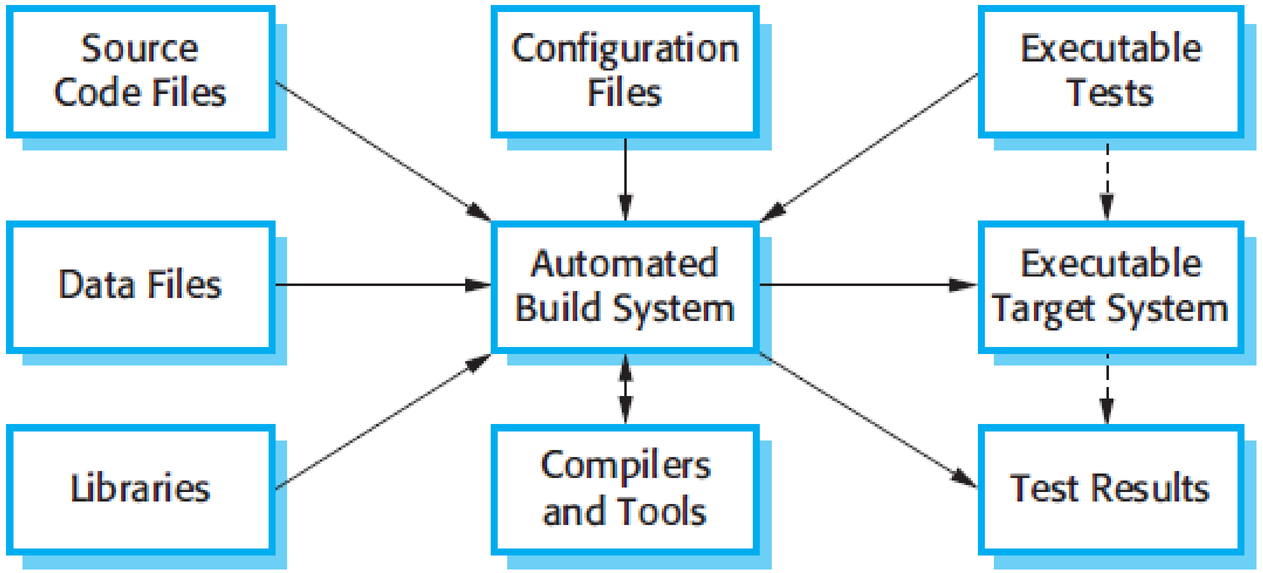
System building (2)

- Three different system platforms involved
  - The development system
  - The build server
  - The target environment



System building (3)

- involves assembling a large amount of information about the software and its operating environment
  - it always makes sense to use an **automated build tool**



Built system features

- *Build script generation*
  - Analyze the program that is being built
  - Identify dependent components
  - Automatically generate a build script (called a configuration file)
  - Support the manual creation and editing of build scripts
- *Version management system integration*
  - Check out the required versions of components from the version management system
- *Minimal recompilation*
  - Work out what source code needs to be recompiled and set up compilations if required
  - The corresponding object code, which has been compiled from the source code, has a related **signature** that identifies each source code version and is changed when the source code is edited

Built system features (2)

- *Executable system creation*
  - Link the compiled object code files with each other and with other required files, such as libraries and configuration files, to create an executable system
- *Test automation*
  - Can automatically run automated tests using test automation tools such as JUnit
- *Reporting*
  - Provide reports about the success or failure of the build and the tests that have been run
- *Documentation generation*
  - Able to generate release notes about the build and system help pages

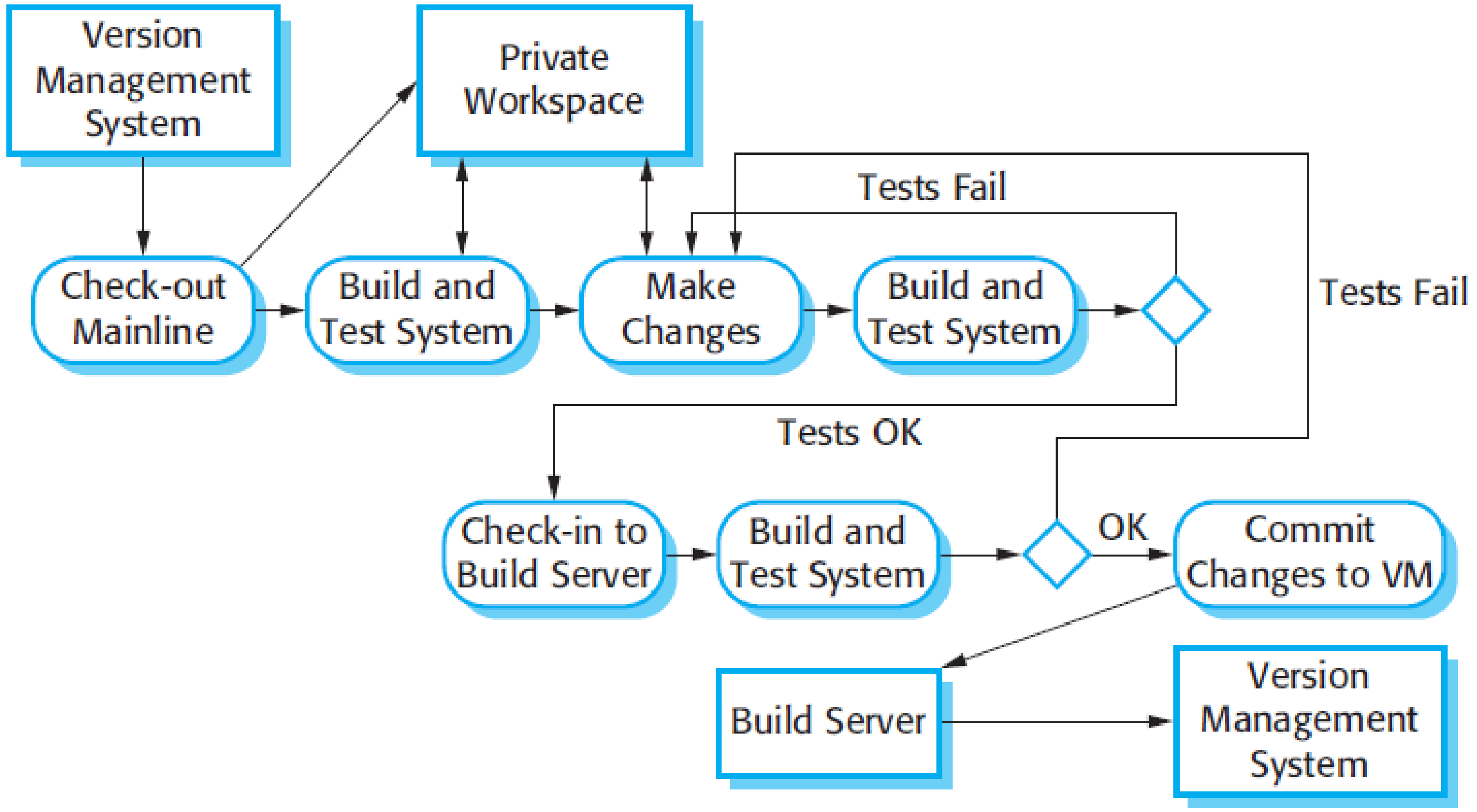
Object code signatures

- *Modification timestamps*
  - The signature on the source code file is the **time** and **date** when that file was modified
  - If the source code file of a component has been modified after the related object code file, then the system assumes that recompilation to create a new object code file is necessary
  - Only the **most recently compiled** object code file is maintained in the system
- *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. If you change the source code (even by one character), this will generate a different checksum
  - Allowing many different versions of the object code of a component to be maintained at the same time

Continuous Integration

- Agile methods recommend that very **frequent system builds** should be carried out with **automated testing** (sometimes called smoke tests) to discover software problems
- Frequent builds may be part of a process of continuous integration

Continuous Integration (2)



The steps of CI

1. Check out the mainline system from the version management system into the developer’s private workspace
2. Build the system and run automated tests to ensure that the built system passes all tests
3. Make the changes to the system components
4. Build the system in the private workspace and rerun system tests. If the tests fail, continue editing
5. Once the system has passed its tests, check it into the build system but do not commit it as a new system baseline
6. Build the system on the build server and run the tests
7. 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

Daily build system

- For <u>large systems</u> or for <u>systems where the execution platform is not the same</u> as the development platform, continuous integration may be impractical
- Can use a daily build system:
  - The development organization sets a delivery time (say 2 P.M.) for system components
  - 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

Advantages of frequent building

- the chances of finding problems stemming from component interactions early in the process are increased
- encourages thorough unit testing of components
- developers are put under pressure not to 'break the build'
  - reluctant to deliver new component versions that have not been properly tested
  - less time is spent during system testing; discovering and coping with software faults that could have been found by the developer

Release Management

- A system release is a version of a software system that is distributed to customers
- Two types of release:
  - Major releases, which deliver significant new functionality
  - Minor releases, which repair bugs and fix customer problems that have been reported

Managing system release in SPL

- Special 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
- May have to manage tens or even hundreds of different releases of software product
- Has to provide information about which customers have which releases of the system and the relationship between releases and system versions
- It must be documented to ensure each release can be re-created exactly in the future

Managing system release in SPL (2)

- You must keep copies of the source code files, corresponding executables, and all data and configuration files
- You should also record the versions of the operating system, libraries, compilers, and other tools used to build the software
- You have to store copies of the platform software and the tools used to create the system in the version management system along with the source code of the target system

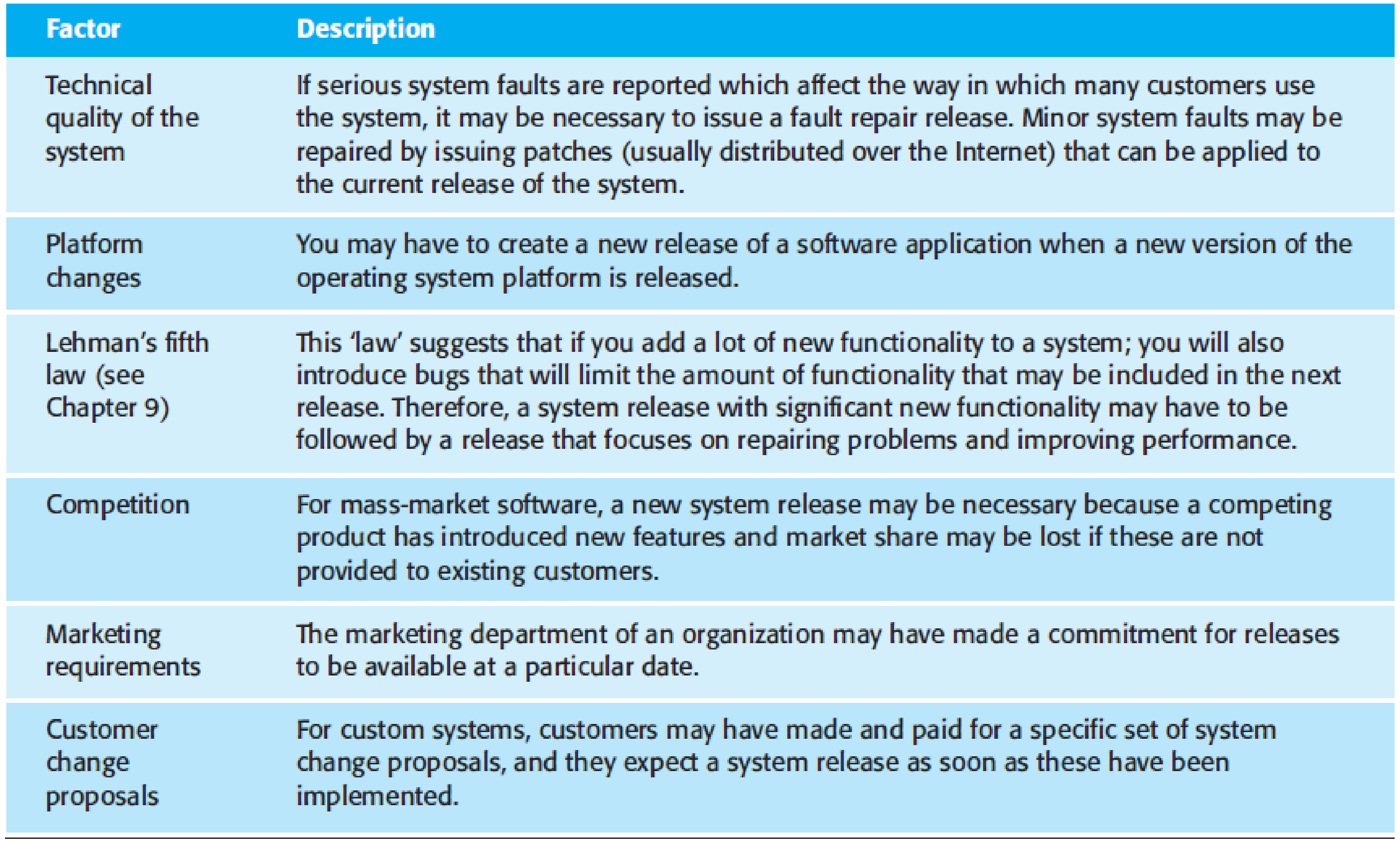
Preparing and distributing a system release

- Creating a release distribution
- Preparing advertising and publicity material
- Preparing marketing strategies to convince customers to buy the new release of the system

Release timing

- Careful thought must be given to 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

Factor influencing release planning



Factor	Description
Technical quality of the system	If serious system faults are reported which affect the way in which many customers use the system, it may be necessary to issue a fault repair release. Minor system faults may be repaired by issuing patches (usually distributed over the Internet) that can be applied to the current release of the system.
Platform changes	You may have to create a new release of a software application when a new version of the operating system platform is released.
Lehman's fifth law (see Chapter 9)	This 'law' suggests that if you add a lot of new functionality to a system; you will also introduce bugs that will limit the amount of functionality that may be included in the next release. Therefore, a system release with significant new functionality may have to be followed by a release that focuses on repairing problems and improving performance.
Competition	For mass-market software, a new system release may be necessary because a competing product has introduced new features and market share may be lost if these are not provided to existing customers.
Marketing requirements	The marketing department of an organization may have made a commitment for releases to be available at a particular date.
Customer change proposals	For custom systems, customers may have made and paid for a specific set of system change proposals, and they expect a system release as soon as these have been implemented.

System release content

- Configuration files defining how the release should be configured for particular installations
  - configuration descriptions may have to be written for different hardware and operating systems and instructions prepared for customers who need to configure their own systems
- Data files, such as files of error messages, that are needed for successful system operation
  - all associated data files must be identified in the version management system and tagged with the release identifier
- An installation program that is used to help install the system on target hardware
  - scripts for the installation program may have to be written

System release content (2)

- Electronic and paper documentation describing the system
    - if machine-readable manuals are distributed, electronic copies must be stored with the software
- Packaging and associated publicity that have been designed for that release
    - an executable master image of the software must be prepared and handed over for distribution to customers or sales outlets

Changing to a new release

- You cannot assume that customers will always install new system releases
  - Some system users may be happy with an existing system
  - They may consider that it is not worth the cost of changing to a new release
- New releases of the system cannot rely on the installation of previous releases
- The software distributor cannot assume that the files required for release 3 have already been installed in all sites
  - Some sites may go directly from release 1 to release 3, skipping release 2

Minor release to repair problem

- The software vendors make patches to repair the existing software available on a website to be downloaded by customers
  - The problem with using downloadable patches is that many customers may never discover the existence of these problem repairs and may not understand why they should be installed
  - They may instead continue using their existing, faulty system with the consequent risks to their business
  - In some situations, where the patch is designed to repair security loopholes, the risks of failing to install the patch can mean that the business is susceptible to external attacks

Minor release to repair problem (2)

- Mass-market software vendors, such as Adobe, Apple, and Microsoft, usually implement **automatic updating** where systems are updated whenever a new minor release becomes available
- However, this **does not usually work for custom systems** because these systems do not exist in a standard version for all customers