Software Risk and Configuration Management

PMIT6111: Software Testing and Quality Assurance

Risk management

Risk management is concerned with identifying risks and drawing up plans to minimise their effect on a project.

A risk is a probability that some adverse circumstance will occur

- Project risks affect schedule or resources;
- Product risks affect the quality or performance of the software being developed;
- Business risks affect the organisation developing or procuring the software.

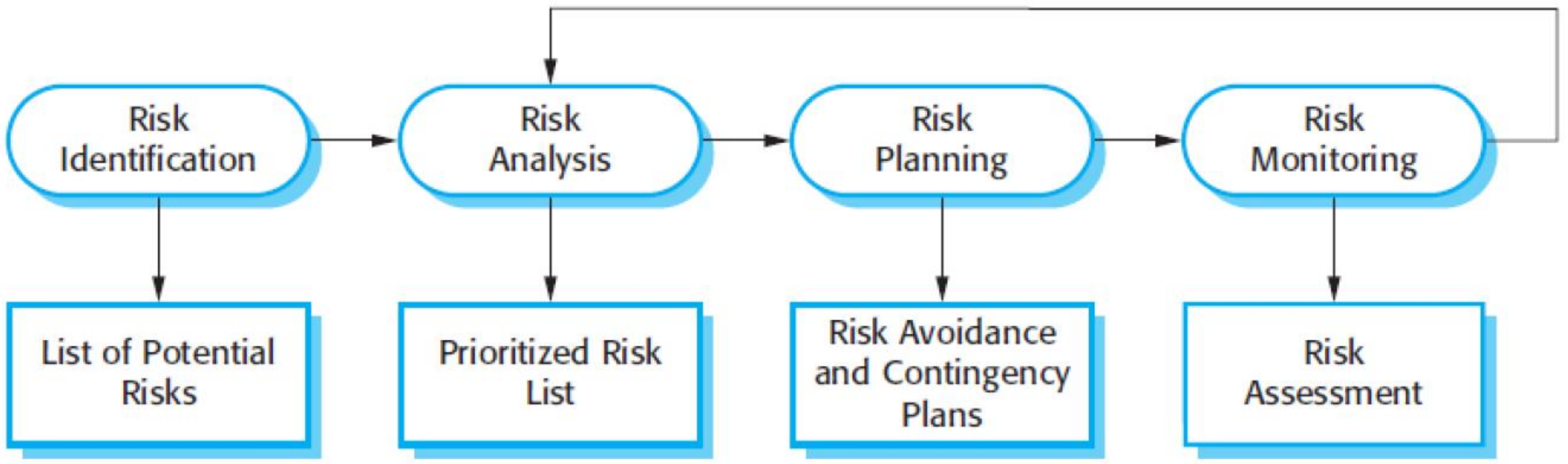
Examples of common project, product, and business risks

Risk	**Affects**	**Description**
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

The risk management process

- Risk identification
  - Identify project, product and business risks;
- Risk analysis
  - Assess the likelihood and consequences of these risks;
- Risk planning
  - Draw up plans to avoid or minimise the effects of the risk;
- Risk monitoring
  - Monitor the risks throughout the project;

The risk management process



Risk identification

May be a team activities or based on the individual project manager’s experience.

A checklist of common risks may be used to identify risks in a project

- Technology risks.
- People risks.
- Organisational risks.
- Requirements risks.
- Estimation risks.

Examples of different risk types

Risk type	Possible risks
Technology	The database used in the system cannot process as many transactions per second as expected. (1) Reusable software components contain defects that mean they cannot be reused as planned. (2)
People	It is impossible to recruit staff with the skills required. (3) Key staff are ill and unavailable at critical times. (4) Required training for staff is not available. (5)
Organizational	The organization is restructured so that different management are responsible for the project. (6) Organizational financial problems force reductions in the project budget. (7)
Tools	The code generated by software code generation tools is inefficient. (8) Software tools cannot work together in an integrated way. (9)
Requirements	Changes to requirements that require major design rework are proposed. (10) Customers fail to understand the impact of requirements changes. (11)
Estimation	The time required to develop the software is underestimated. (12) The rate of defect repair is underestimated. (13) The size of the software is underestimated. (14)

Risk analysis

Assess probability and seriousness of each risk.

Probability may be very low, low, moderate, high or very high.

Risk consequences might be catastrophic, serious, tolerable or insignificant.

Risk types and examples

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget (7).	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project (3).	High	Catastrophic
Key staff are ill at critical times in the project (4).	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused. (2).	Moderate	Serious
Changes to requirements that require major design rework are proposed (10).	Moderate	Serious
The organization is restructured so that different management are responsible for the project (6).	High	Serious
The database used in the system cannot process as many transactions per second as expected (1).	Moderate	Serious

Risk types and examples

Risk	Probability	Effects
The time required to develop the software is underestimated (12).	High	Serious
Software tools cannot be integrated (9).	High	Tolerable
Customers fail to understand the impact of requirements changes (11).	Moderate	Tolerable
Required training for staff is not available (5).	Moderate	Tolerable
The rate of defect repair is underestimated (13).	Moderate	Tolerable
The size of the software is underestimated (14).	High	Tolerable
Code generated by code generation tools is inefficient (8).	Moderate	Insignificant

Risk planning

Consider each risk and develop a strategy to manage that risk.

Avoidance strategies

- The probability that the risk will arise is reduced;

Minimisation strategies

- The impact of the risk on the project or product will be reduced;

Contingency plans

- If the risk arises, contingency plans are plans to deal with that risk;

Strategies to help manage risk

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program generator.

Risk monitoring

Assess each identified risks regularly to decide whether or not it is becoming less or more probable.

Also assess whether the effects of the risk have changed.

Each key risk should be discussed at management progress meetings.

Risk indicators

Risk type	Potential indicators
Technology	Late delivery of hardware or support software; many reported technology problems.
People	Poor staff morale; poor relationships amongst team members; high staff turnover.
Organizational	Organizational gossip; lack of action by senior management.
Tools	Reluctance by team members to use tools; complaints about CASE tools; demands for higher-powered workstations.
Requirements	Many requirements change requests; customer complaints.
Estimation	Failure to meet agreed schedule; failure to clear reported defects.

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.

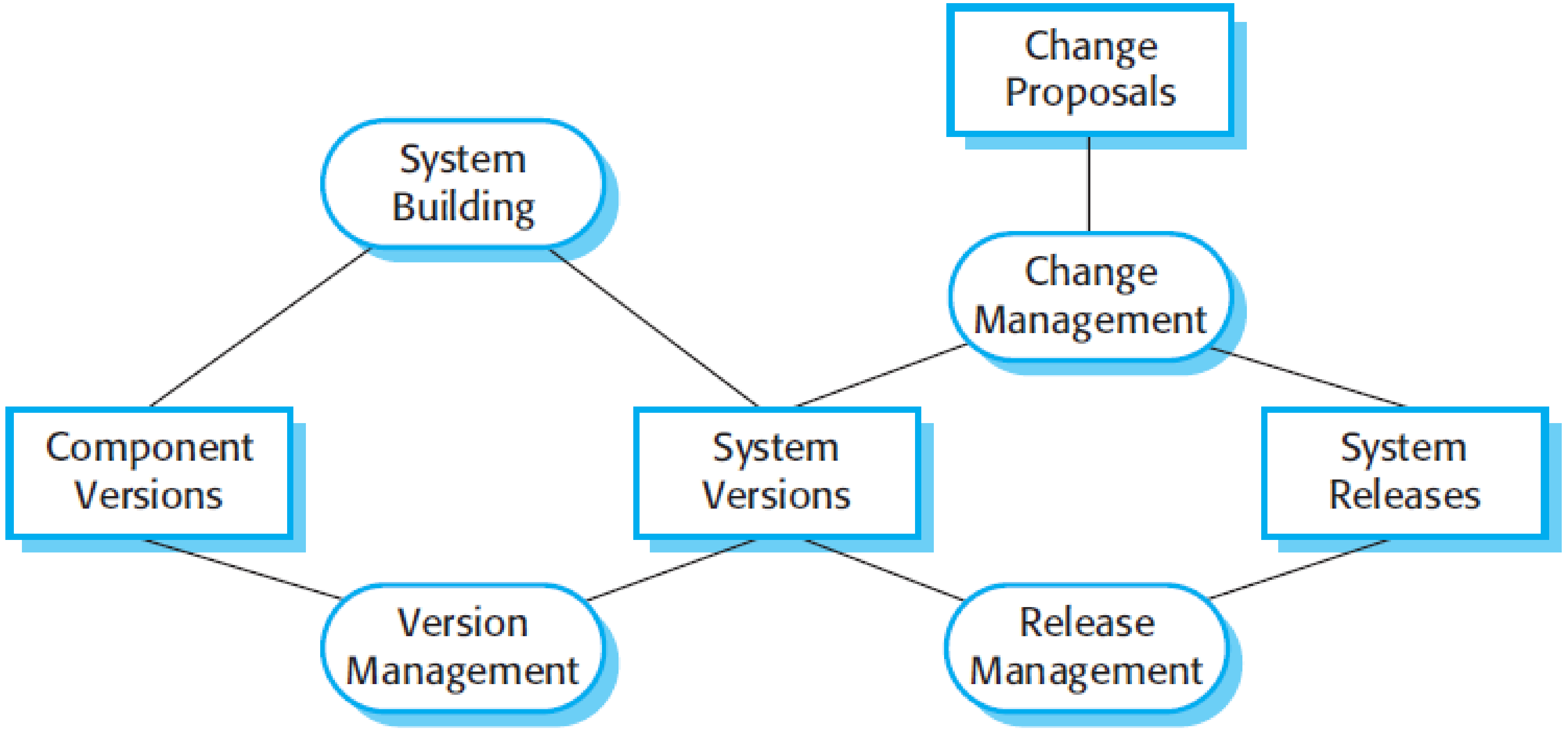
Versions implement proposals for change, corrections of faults, and adaptations for different hardware and operating systems.

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.

SCM Activities

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

SCM Activities



CM terminology

Term	Explanation
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.
Configuration 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.
Version	An instance of a configuration item that differs, in some way, from other instances of that item. Versions always have a unique identifier, which is often composed of the configuration item name plus a version number.
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 should always be possible to recreate a baseline from its constituent components.
Codeline	A codeline is a set of versions of a software component and other configuration items on which that component depends.

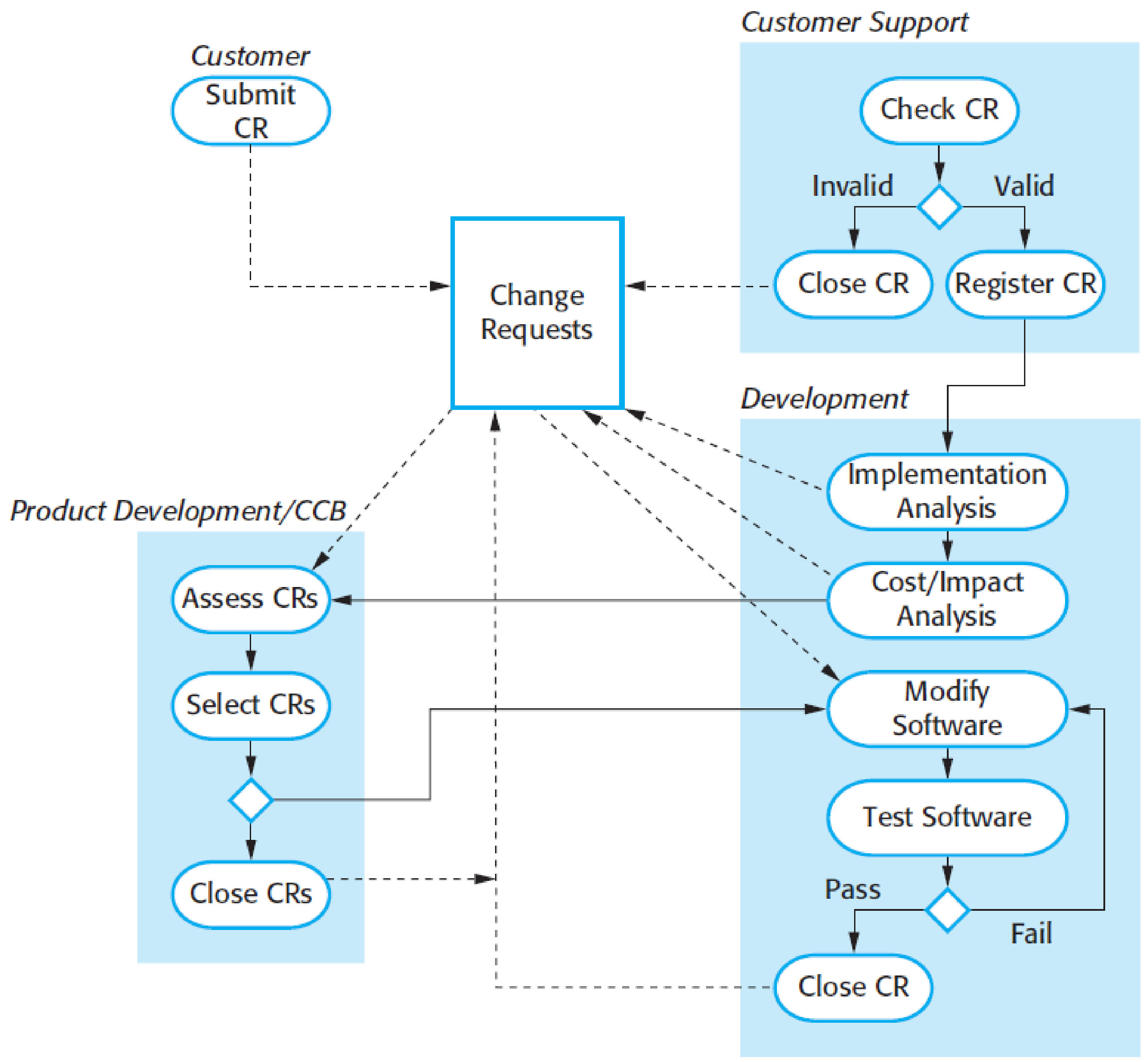
CM terminology

Term	Explanation
Mainline	A sequence of baselines representing different versions of a system.
Release	A version of a system that has been released to customers (or other users in an organization) for use.
Workspace	A private work area where software can be modified without affecting other developers who may be using or modifying that software.
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. These codelines may have been created by a previous branch of one of the codelines involved.
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.

Change Management

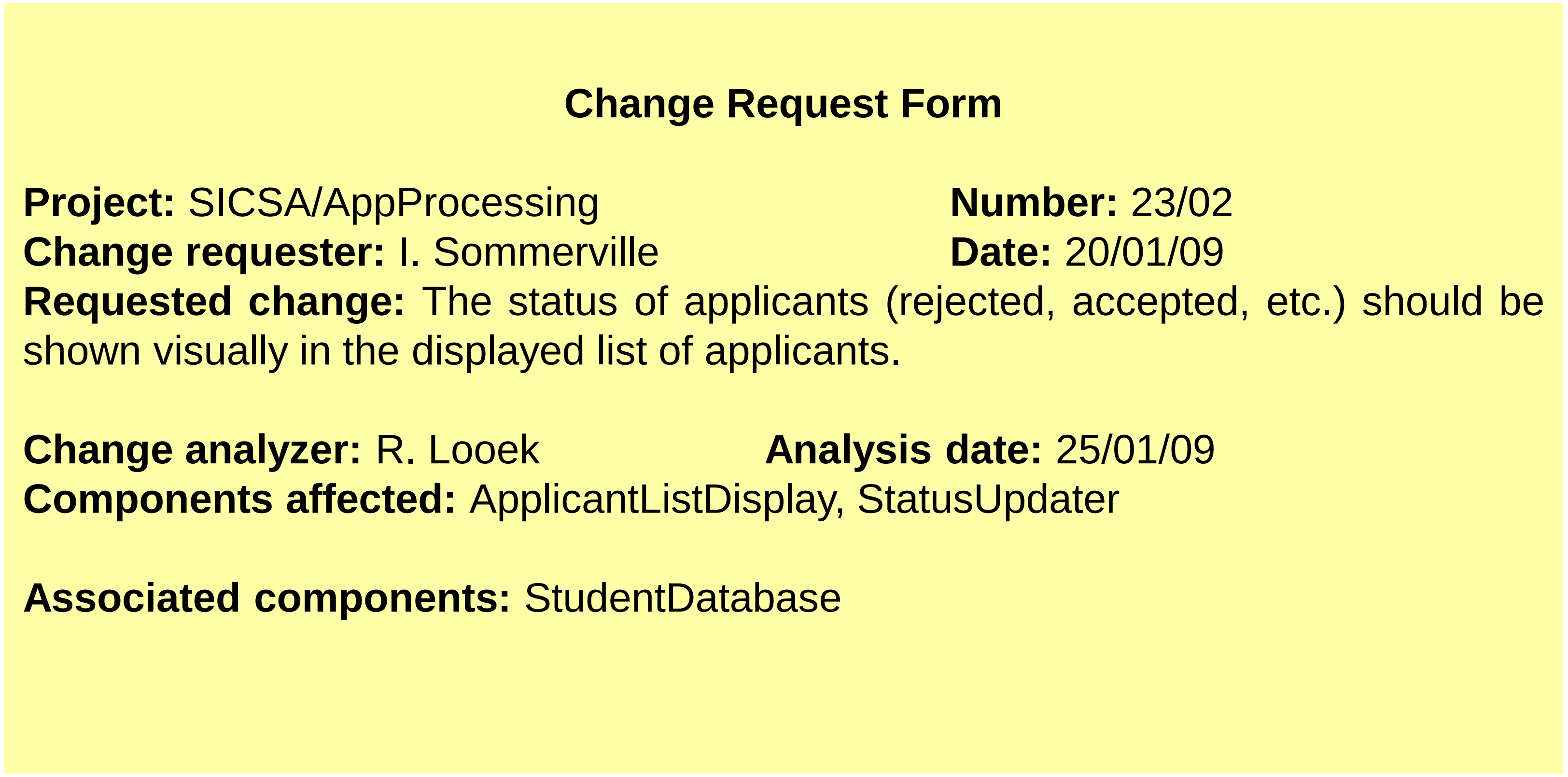
Organizational needs and requirements change during the lifetime of a system, bugs have to be repaired and systems have to adapt to changes in their environment.

Change management is intended to ensure that system evolution is a managed process and that priority is given to the most urgent and cost-effective changes.



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:** 20/01/09
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

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: 28/01/09 **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:

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

Change management and agile methods

In some agile methods, customers are directly involved in change management.

The propose a change to the requirements and work with the team to assess its impact and decide whether the change should take priority over the features planned for the next increment of the system.

Changes to improve the software improvement are decided by the programmers working on the system.

Refactoring, where the software is continually improved, is not seen as an overhead but as a necessary part of the development process.

Derivation history

```
// 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
```

Version management

Version management (VM) is the process of keeping track of different versions of software components or configuration items and the systems in which these components are used.

It also involves ensuring that changes made by different developers to these versions do not interfere with each other.

Therefore version management can be thought of as the process of managing codelines and baselines.

Codelines and baselines

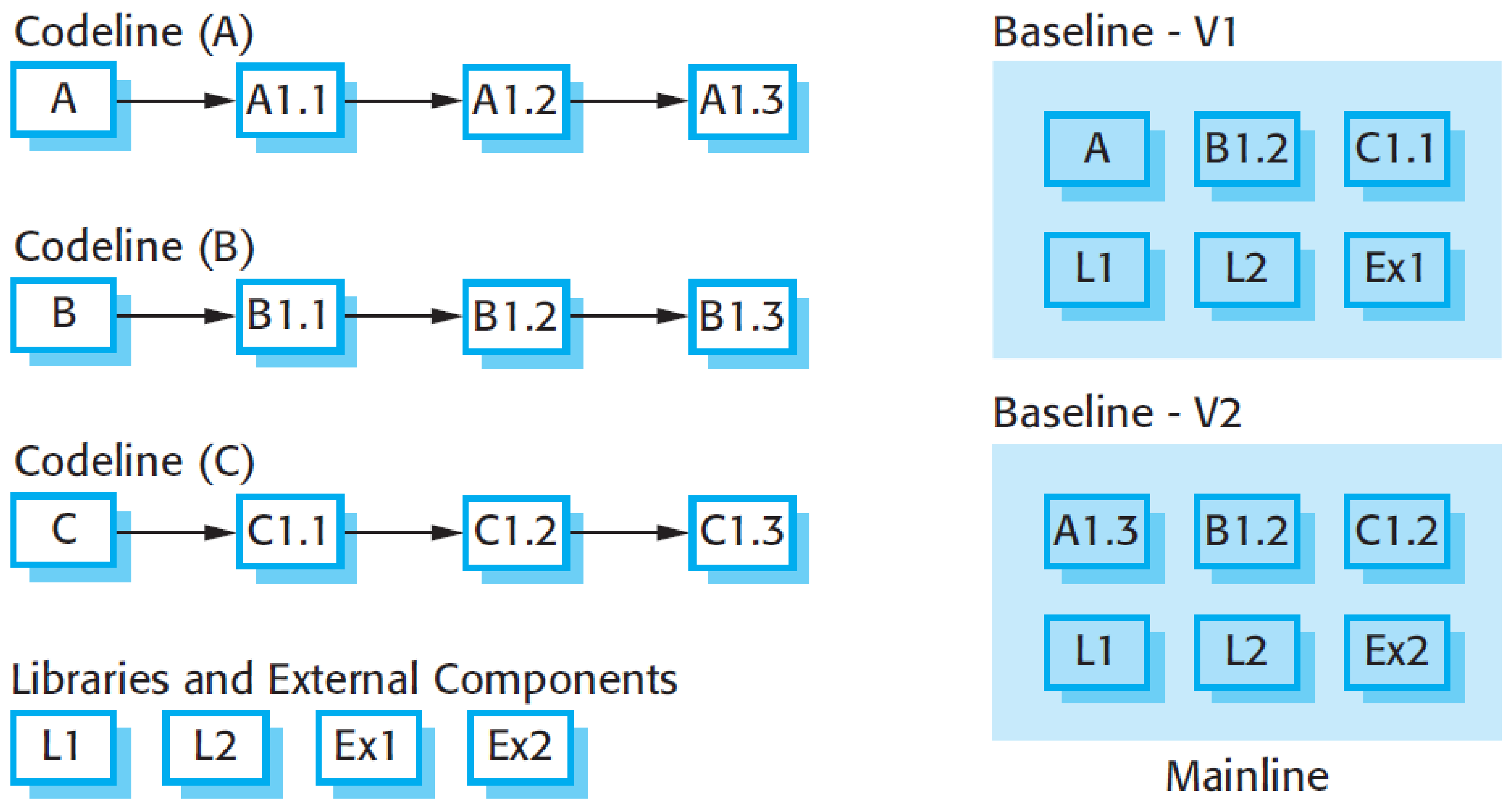
A codeline is a sequence of versions of source code with later versions in the sequence derived from earlier versions.

Codelines normally apply to components of systems so that there are different versions of each component.

A baseline is a definition of a specific system.

The baseline therefore specifies the component versions that are included in the system plus a specification of the libraries used, configuration files, etc.

Codelines and baselines



Baselines

Baselines may be specified using a configuration language, which allows you to define what components are included in a version of a particular system.

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

- For example, a product line may be instantiated so that there are individual system versions for different customers. You may have to recreate the version delivered to a specific customer if, for example, that customer reports bugs in their system that have to be repaired.

Version management systems

Version and release identification

- Managed versions are assigned identifiers when they are submitted to the system.

Storage management

- To reduce the storage space required by multiple versions of components that differ only slightly, version management systems usually provide storage management facilities.

Change history recording

- All of the changes made to the code of a system or component are recorded and listed.

Version management systems

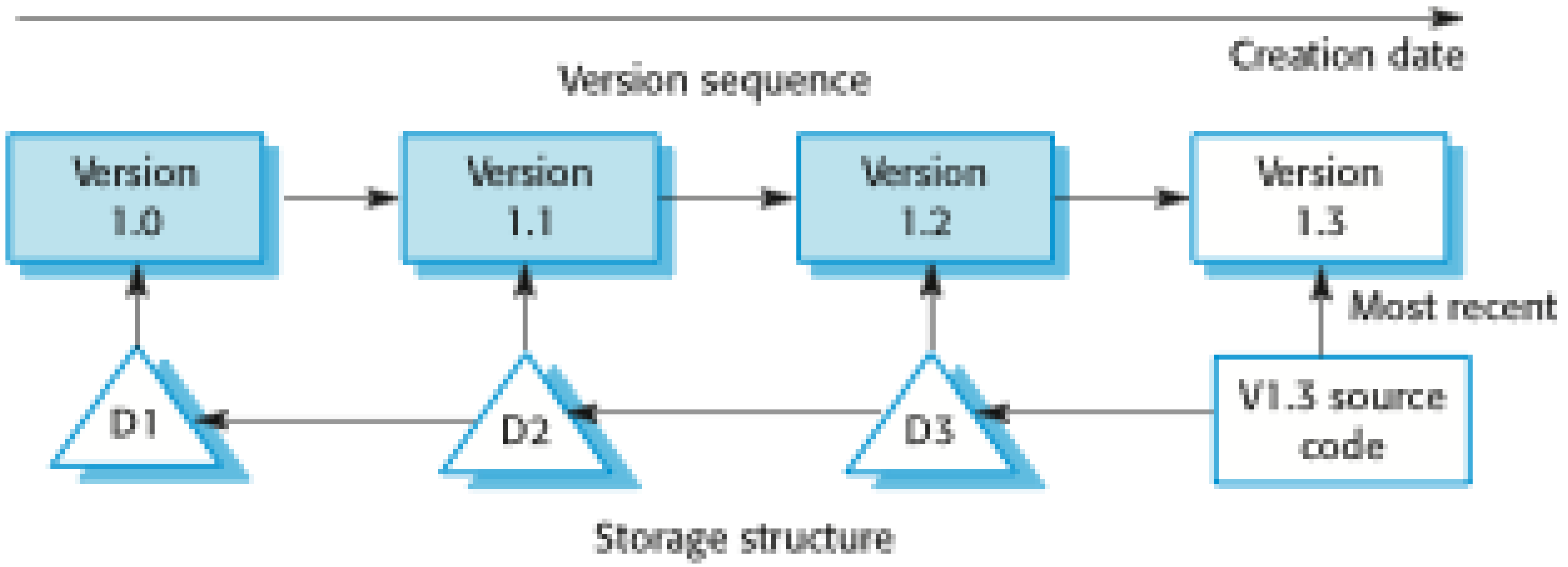
Independent development

- The version management system keeps track of components that have been checked out for editing and ensures that changes made to a component by different developers do not interfere.

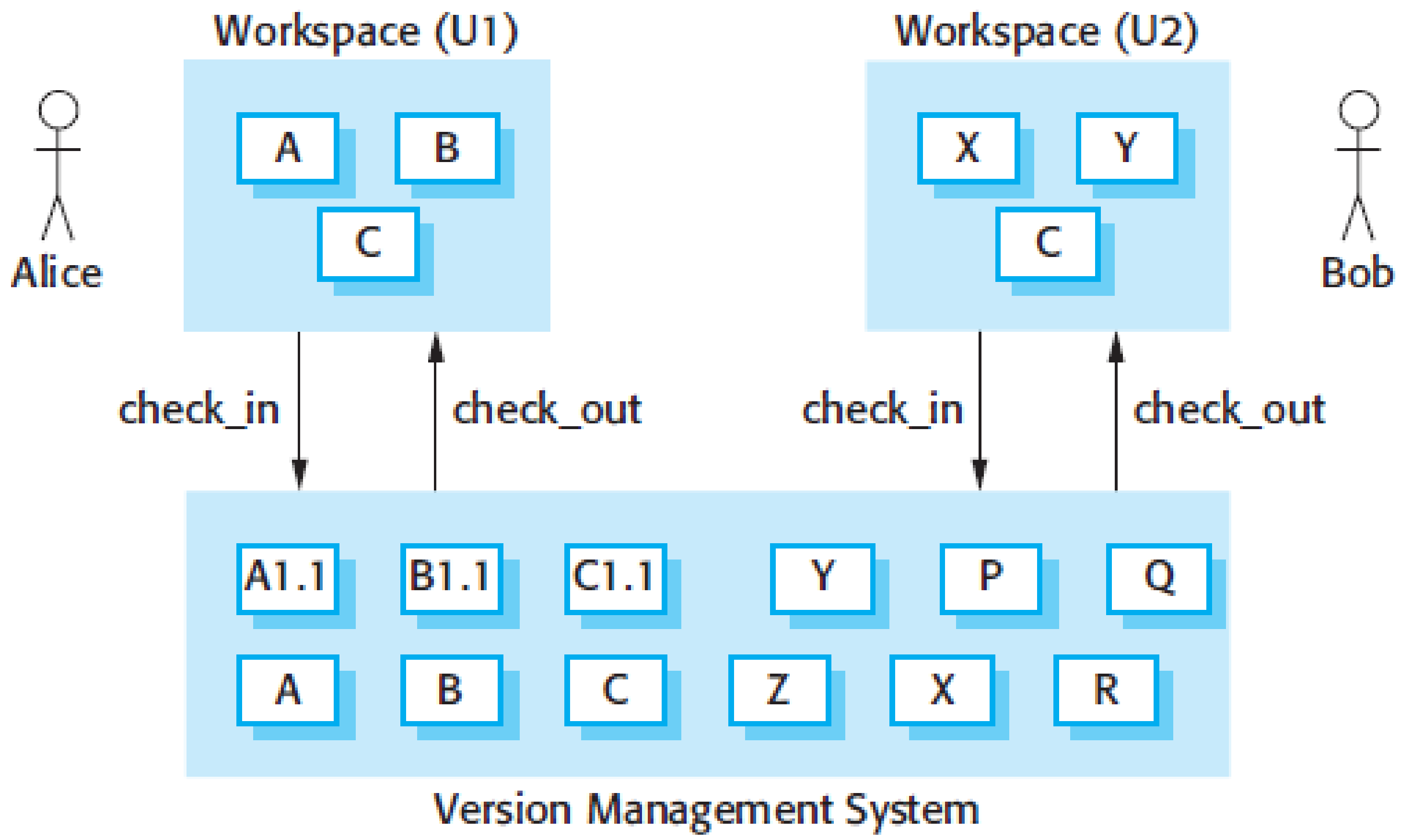
Project support

- A version management system may support the development of several projects, which share components.

Storage management using deltas



Check-in and check-out from a version repository



References

1. Chapter 22 and Chapter 25, Ian Sommerville, “Software Engineering”, 10thEdition.