Aalto University
School of Science

Software Evolution and Configuration Management

Casper Lassenius

26.2.2013

Topics covered

- Evolution processes
  - Change processes for software systems
- Program evolution dynamics
  - Understanding software evolution
- Software maintenance
  - Making changes to operational software systems
- Legacy system management
  - Making decisions about software change

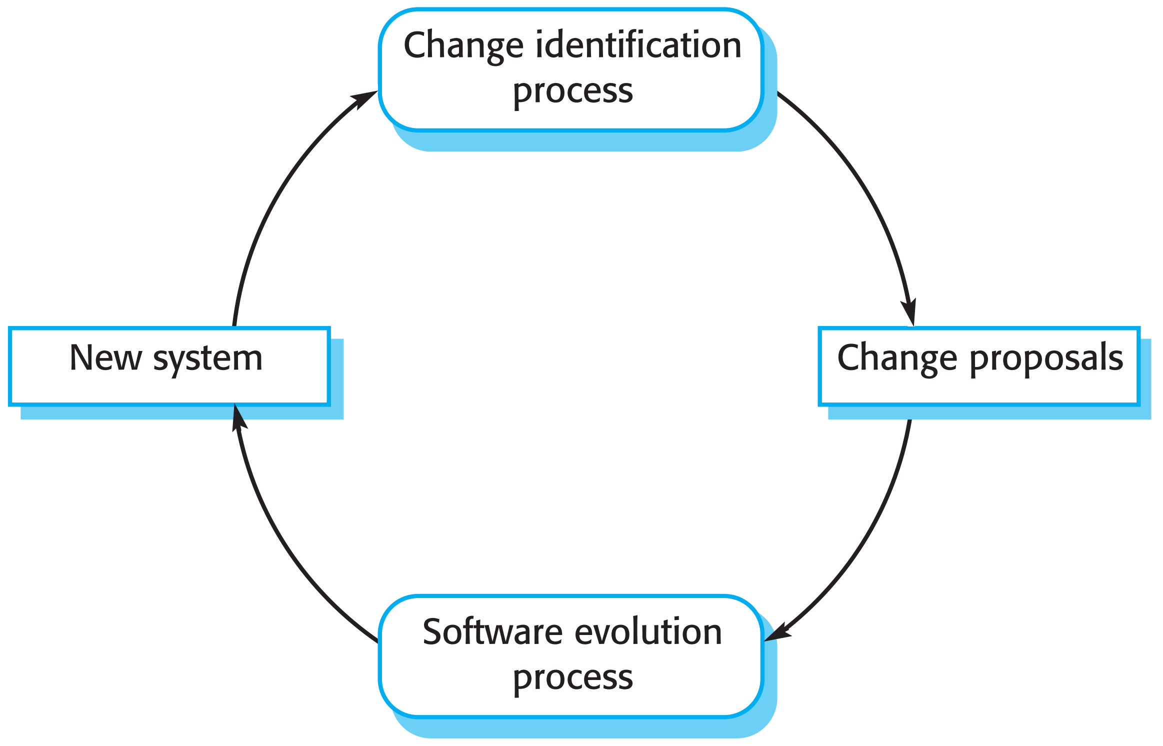
Software change

- ✧ Software change is inevitable
  - New requirements emerge when the software is used;
  - The business environment changes;
  - Errors must be repaired;
  - New computers and equipment is added to the system;
  - The performance or reliability of the system may have to be improved.
- ✧ A key problem for all organizations is implementing and managing change to their existing software systems.

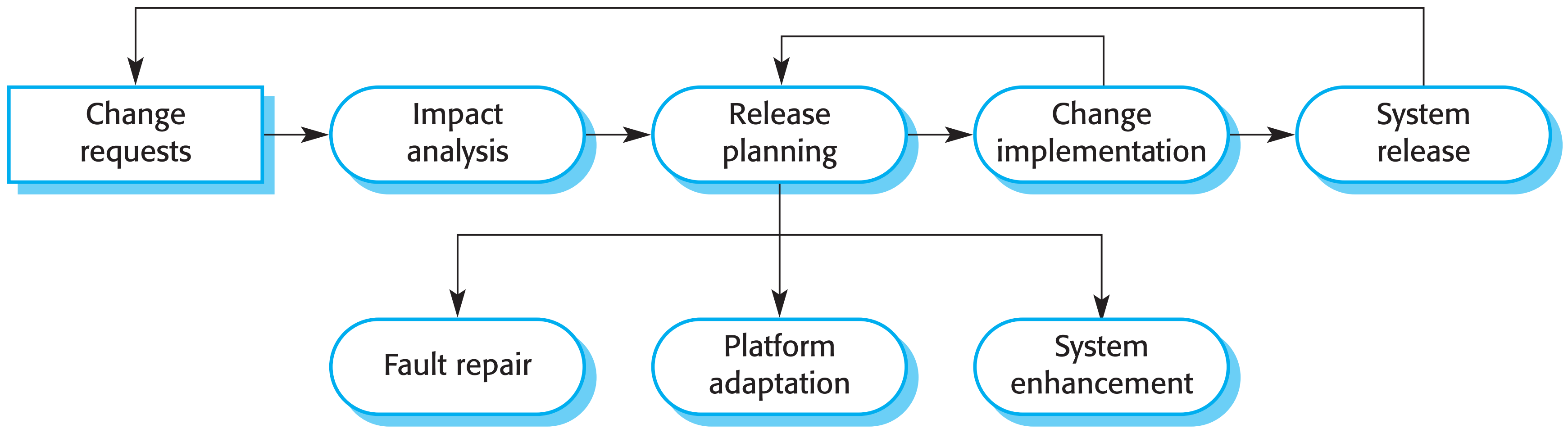
Evolution processes

- Software evolution processes depend on
  - The type of software being maintained;
  - The development processes used;
  - The skills and experience of the people involved.
- Proposals for change are the driver for system evolution.
  - Should be linked with components that are affected by the change, thus allowing the cost and impact of the change to be estimated.
- Change identification and evolution continues throughout the system lifetime.

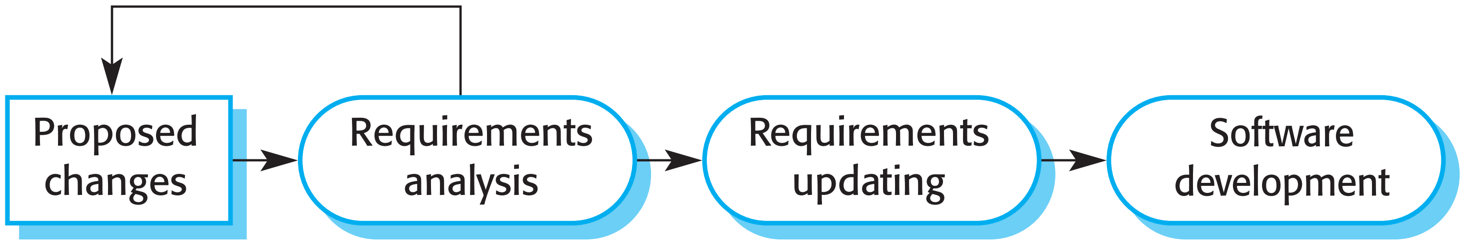
Change identification and evolution processes



The software evolution process



Change implementation

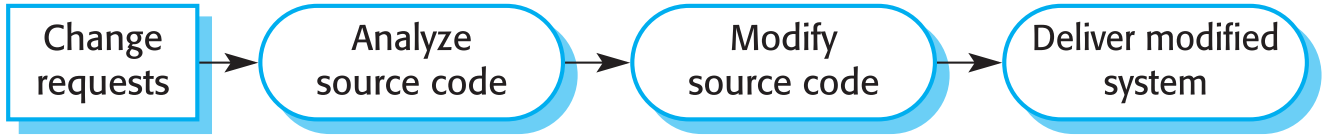


Urgent change requests

✧ Urgent changes may have to be implemented without going through all stages of the software engineering process

- If a serious system fault has to be repaired to allow normal operation to continue;
- If changes to the system's environment (e.g. an OS upgrade) have unexpected effects;
- If there are business changes that require a very rapid response (e.g. the release of a competing product).

The emergency repair process



Agile methods and evolution

- ✧ Agile methods are based on incremental development so the transition from development to evolution is a seamless one.
  - Evolution is simply a continuation of the development process based on frequent system releases.
- ✧ Automated regression testing is particularly valuable when changes are made to a system.
- ✧ Changes may be expressed as additional user stories.

Handover problems

- Where the development team have used an agile approach but the evolution team is unfamiliar with agile methods and prefer a plan-based approach.
  - The evolution team may expect detailed documentation to support evolution and this is not produced in agile processes.
- Where a plan-based approach has been used for development but the evolution team prefer to use agile methods.
  - The evolution team may have to start from scratch developing automated tests and the code in the system may not have been refactored and simplified as is expected in agile development.

Key points

- ✧ Software development and evolution can be thought of as an integrated, iterative process that can be represented using a spiral model.
- ✧ For custom systems, the costs of software maintenance usually exceed the software development costs.
- ✧ The process of software evolution is driven by requests for changes and includes change impact analysis, release planning and change implementation.

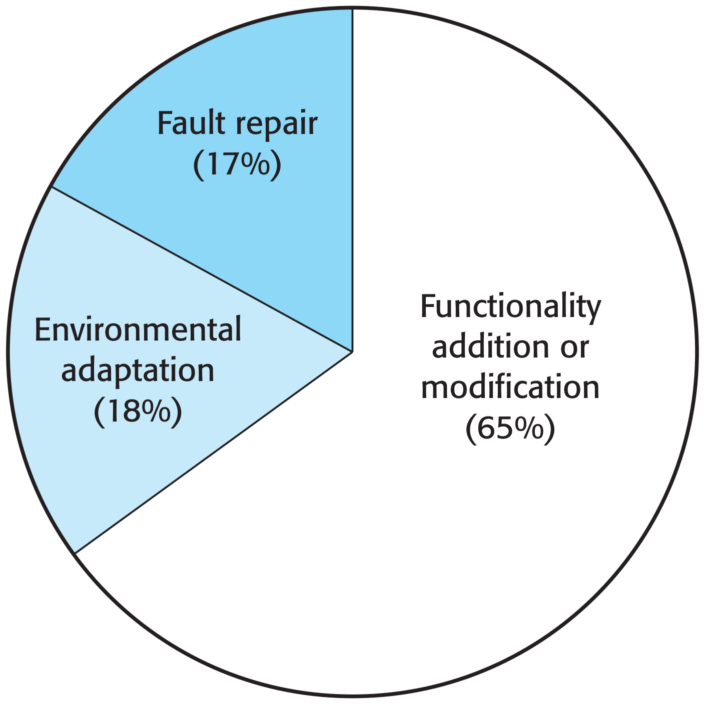
Software maintenance

- ✧ Modifying a program after it has been put into use.
- ✧ The term is mostly used for changing custom software. Generic software products are said to evolve to create new versions.
- ✧ Maintenance does not normally involve major changes to the system’s architecture.
- ✧ Changes are implemented by modifying existing components and adding new components to the system.

Types of maintenance

- ✧ Maintenance to repair software faults
  - Changing a system to correct deficiencies in the way meets its requirements.
- ✧ Maintenance to adapt software to a different operating environment
  - Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation.
- ✧ Maintenance to add to or modify the system’s functionality
  - Modifying the system to satisfy new requirements.

Figure 9.8 Maintenance effort distribution



Maintenance costs

- ✧ Usually greater than development costs (2* to 100* depending on the application).
- ✧ Affected by both technical and non-technical factors.
- ✧ Increases as software is maintained. Maintenance corrupts the software structure so makes further maintenance more difficult.
- ✧ Ageing software can have high support costs (e.g. old languages, compilers etc.).

System re-engineering

- ✧ Re-structuring or re-writing part or all of a legacy system without changing its functionality.
- ✧ Applicable where some but not all sub-systems of a larger system require frequent maintenance.
- ✧ Re-engineering involves adding effort to make them easier to maintain. The system may be re-structured and re-documented.

Advantages of reengineering

- Reduced risk
  - There is a high risk in new software development. There may be development problems, staffing problems and specification problems.
- Reduced cost
  - The cost of re-engineering is often significantly less than the costs of developing new software.

The reengineering process

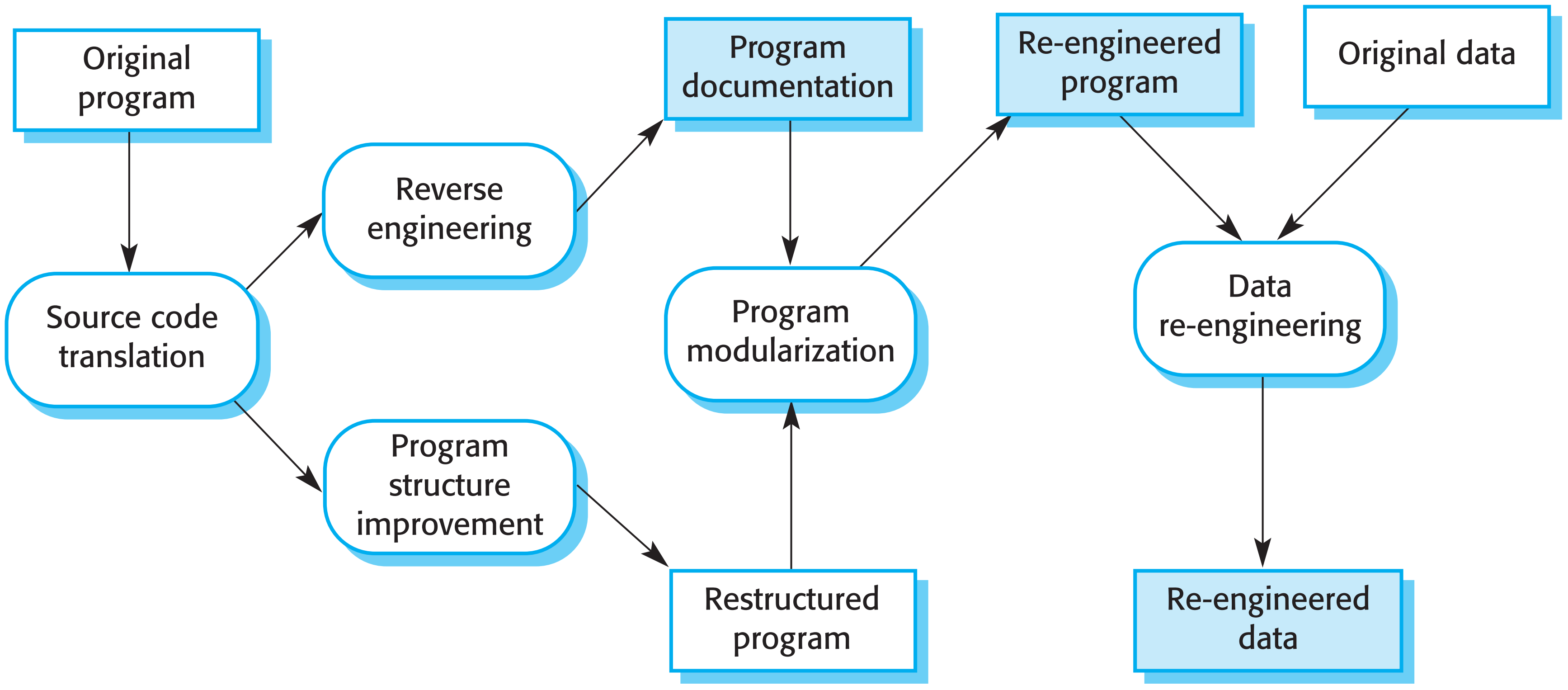
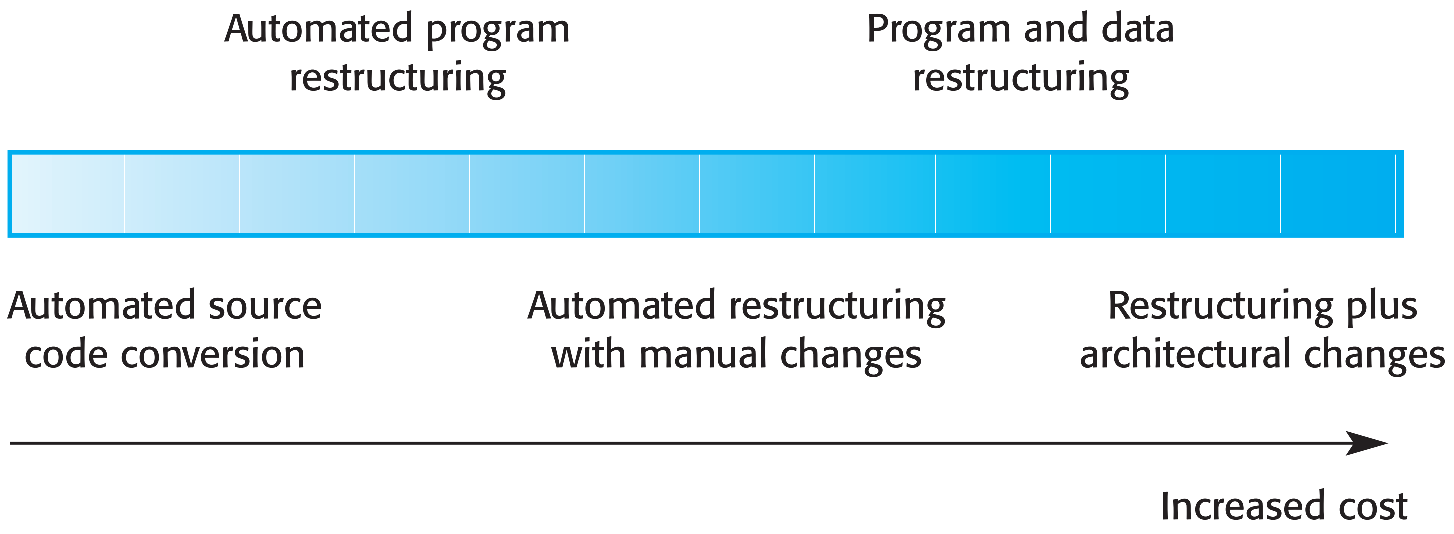


Figure 9.12 Reengineering approaches


Automated program restructuring
Program and data restructuring
Automated source code conversion
Automated restructuring with manual changes
Restructuring plus architectural changes
Increased cost

Preventative maintenance by refactoring

- ✧ Refactoring is the process of making improvements to a program to slow down degradation through change.
- ✧ You can think of refactoring as ‘preventative maintenance’ that reduces the problems of future change.
- ✧ Refactoring involves modifying a program to improve its structure, reduce its complexity or make it easier to understand.
- ✧ When you refactor a program, you should not add functionality but rather concentrate on program improvement.

Refactoring and reengineering

- ✧ Re-engineering takes place after a system has been maintained for some time and maintenance costs are increasing. You use automated tools to process and re-engineer a legacy system to create a new system that is more maintainable.
- ✧ Refactoring is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system.

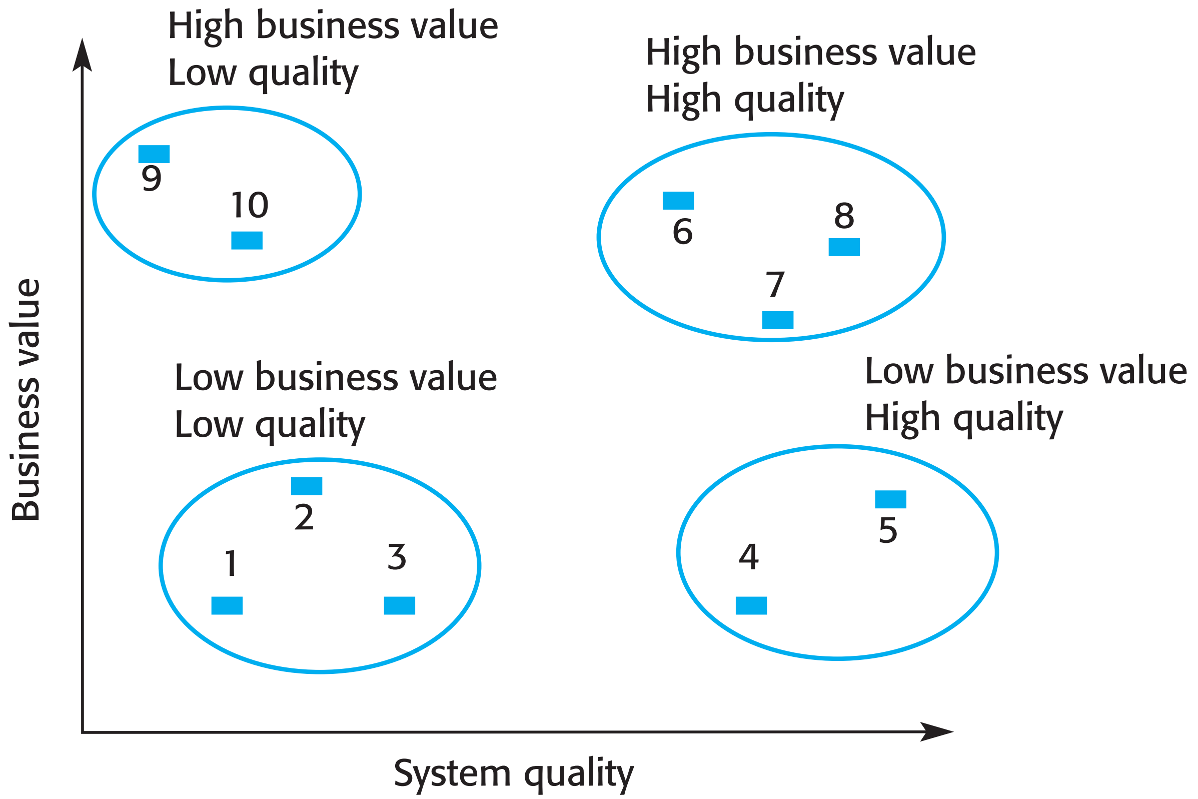
‘Bad smells’ in program code

- Duplicate code
  - The same or very similar code may be included at different places in a program. This can be removed and implemented as a single method or function that is called as required.
- Long methods
  - If a method is too long, it should be redesigned as a number of shorter methods.
- Switch (case) statements
  - These often involve duplication, where the switch depends on the type of a value. The switch statements may be scattered around a program. In object-oriented languages, you can often use polymorphism to achieve the same thing.

Legacy system management

- ✧ Organisations that rely on legacy systems must choose a strategy for evolving these systems
  - Scrap the system completely and modify business processes so that it is no longer required;
  - Continue maintaining the system;
  - Transform the system by re-engineering to improve its maintainability;
  - Replace the system with a new system.
- ✧ The strategy chosen should depend on the system quality and its business value.

Figure 9.13 An example of a legacy system assessment



Key points

- There are 3 types of software maintenance, namely bug fixing, modifying software to work in a new environment, and implementing new or changed requirements.
- Software re-engineering is concerned with re-structuring and re-documenting software to make it easier to understand and change.
- Refactoring, making program changes that preserve functionality, is a form of preventative maintenance.
- The business value of a legacy system and the quality of the application should be assessed to help decide if a system should be replaced, transformed or maintained.

Configuration Management

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

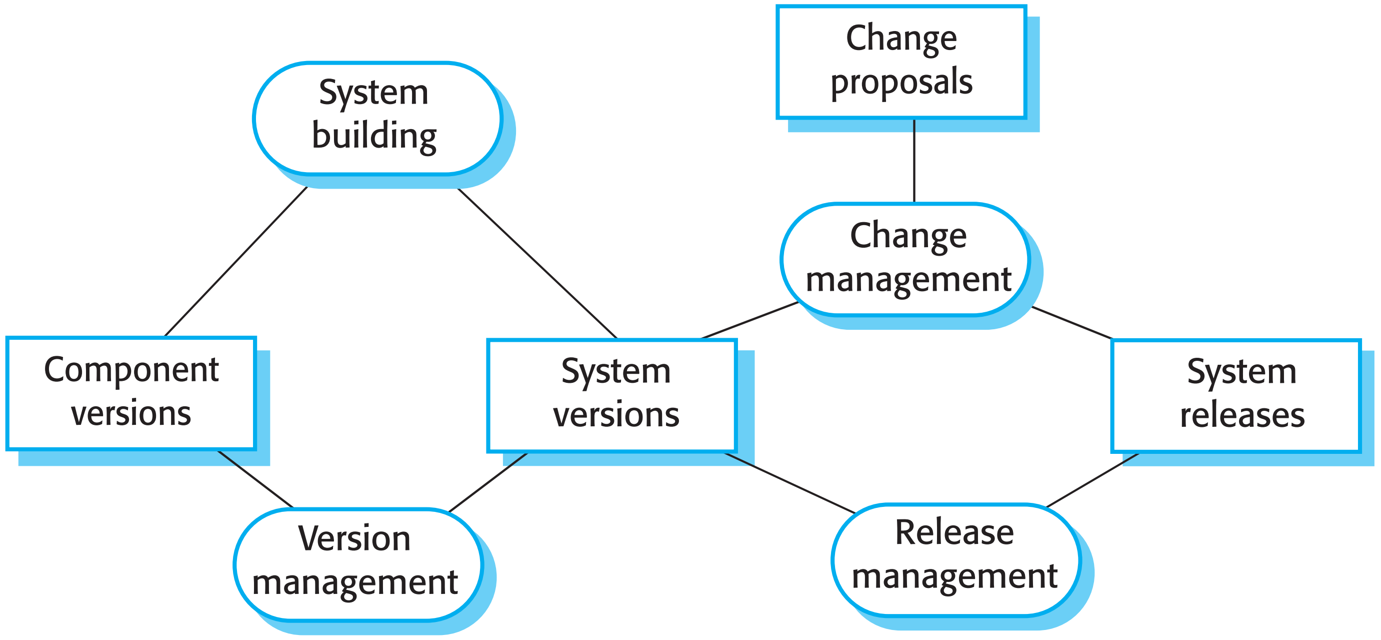
Configuration management

- ✧ 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.

CM activities

- 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.
- 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
  - The process of assembling program components, data and libraries, then compiling 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.

Configuration management 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.

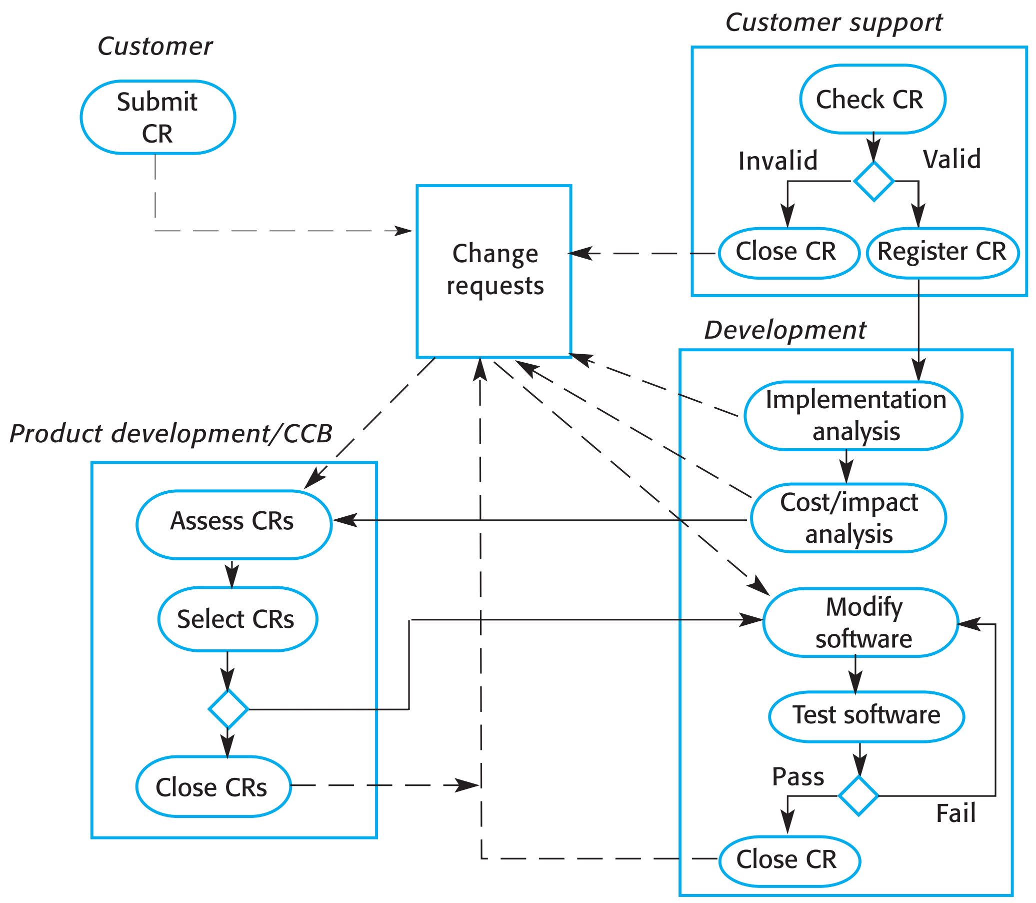
CM terminology

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 is concerned with analyzing the costs and benefits of proposed changes, approving those changes that are worthwhile and tracking which components in the system have been changed.

The change management process



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.

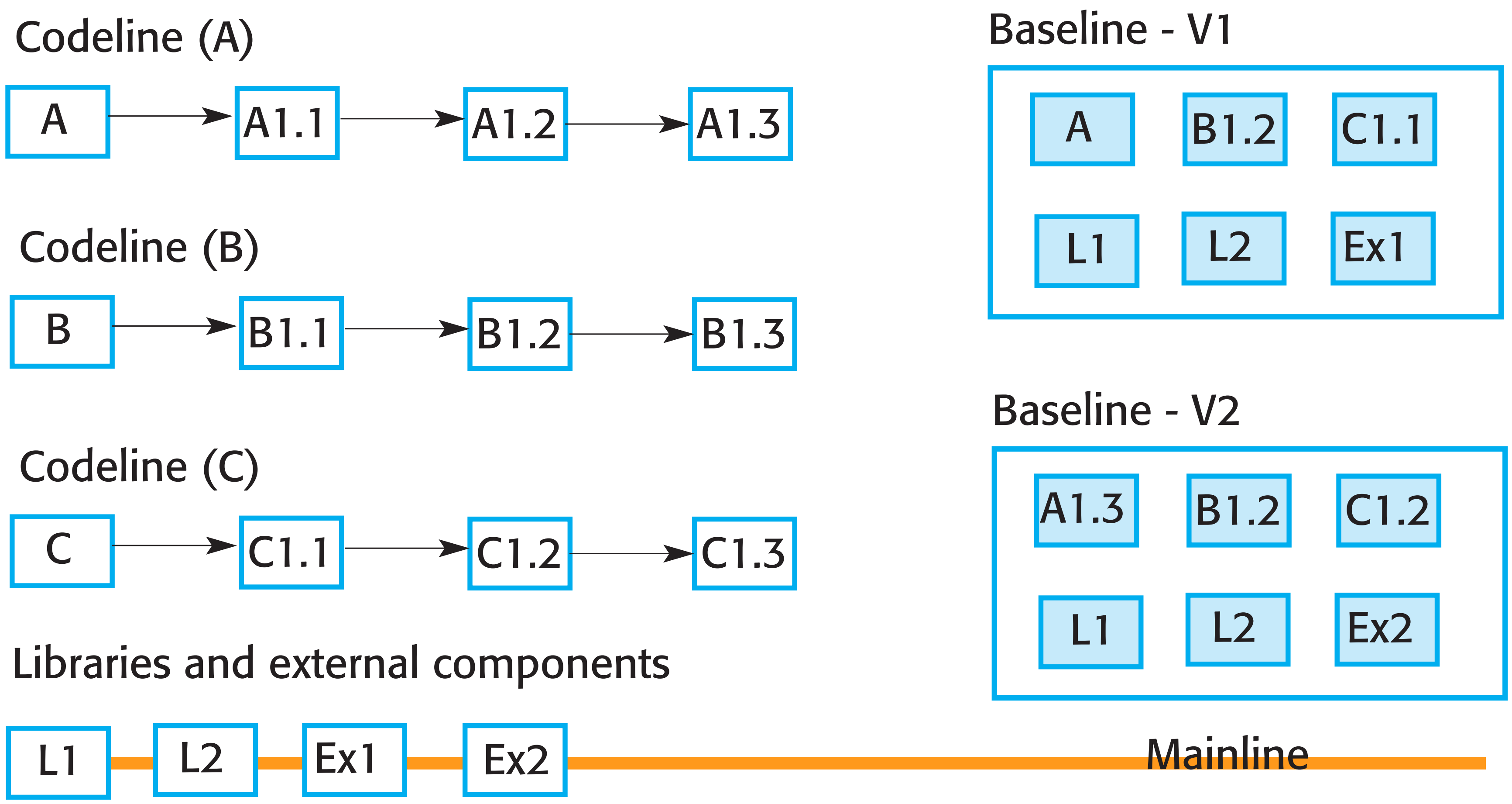
Version management

- 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.

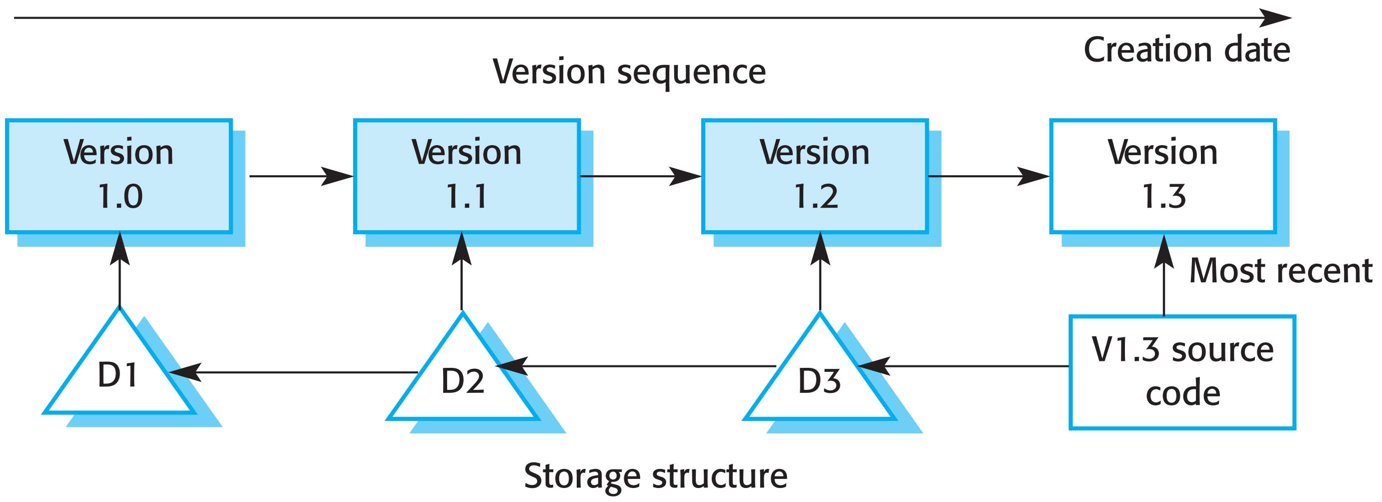
Version management systems

- 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.

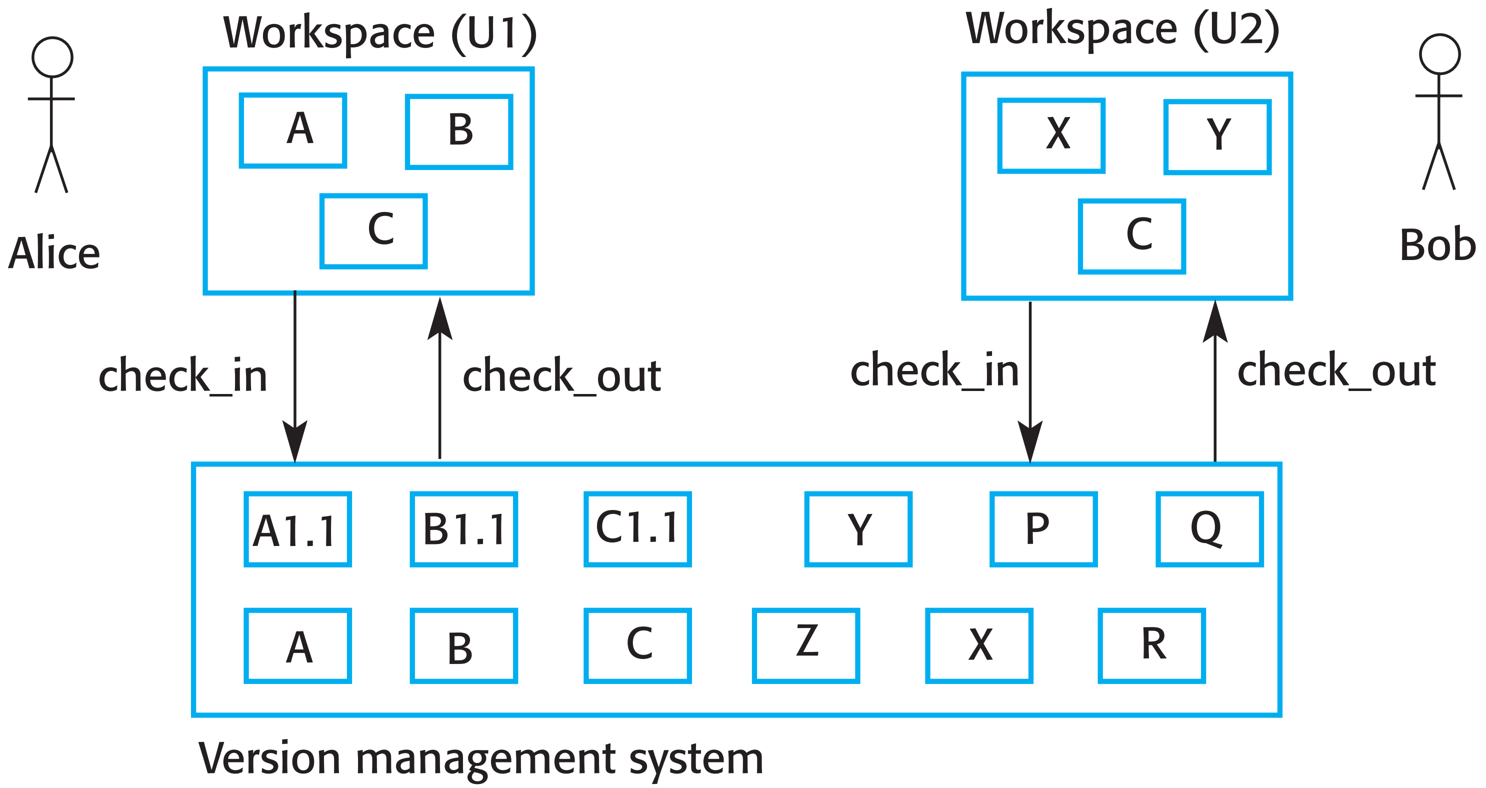
Version management systems

- ✧ 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



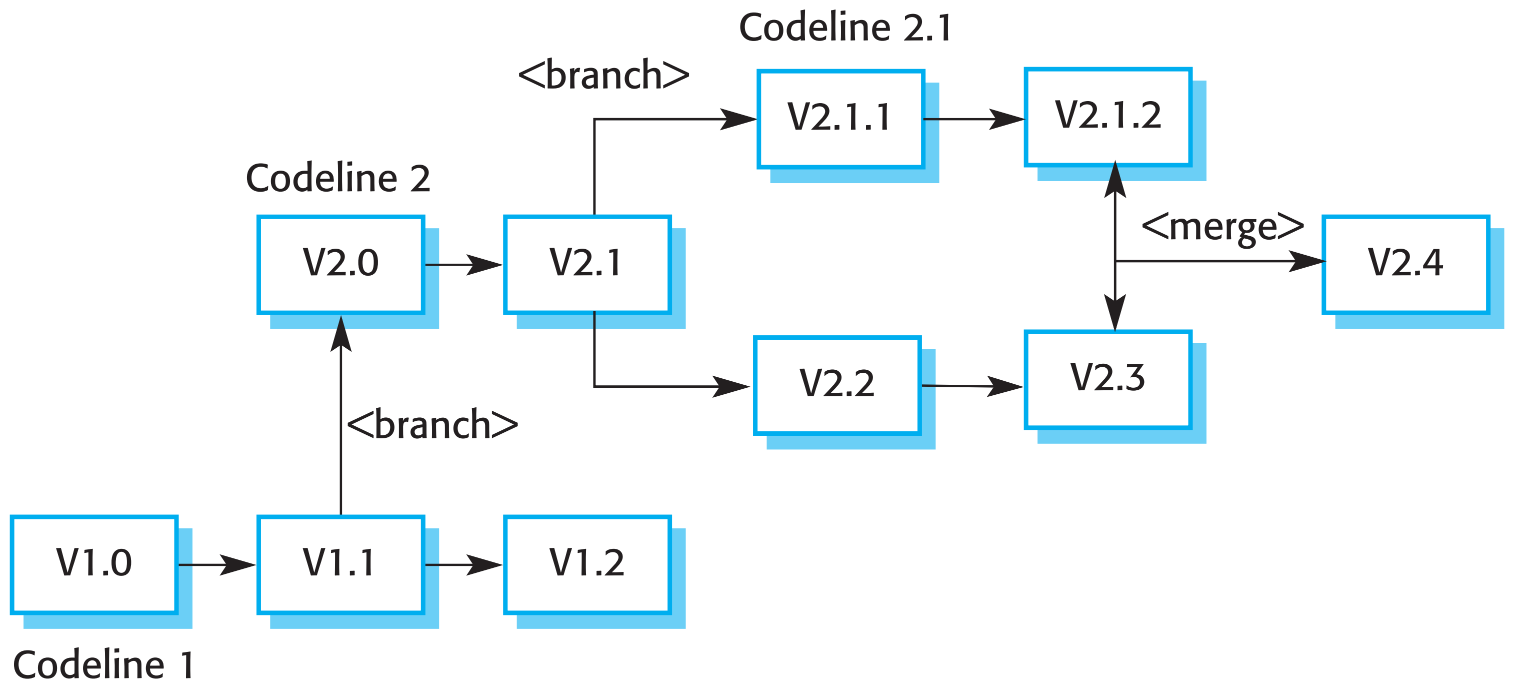
Check-in and check-out from a version repository



Codeline branches

- ✧ Rather than a linear sequence of versions that reflect changes to the component over time, there may be several independent sequences.
  - This is normal in system development, where different developers work independently on different versions of the source code and so change it in different ways.
- ✧ At some stage, it may be necessary to merge codeline branches to create a new version of a component that 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



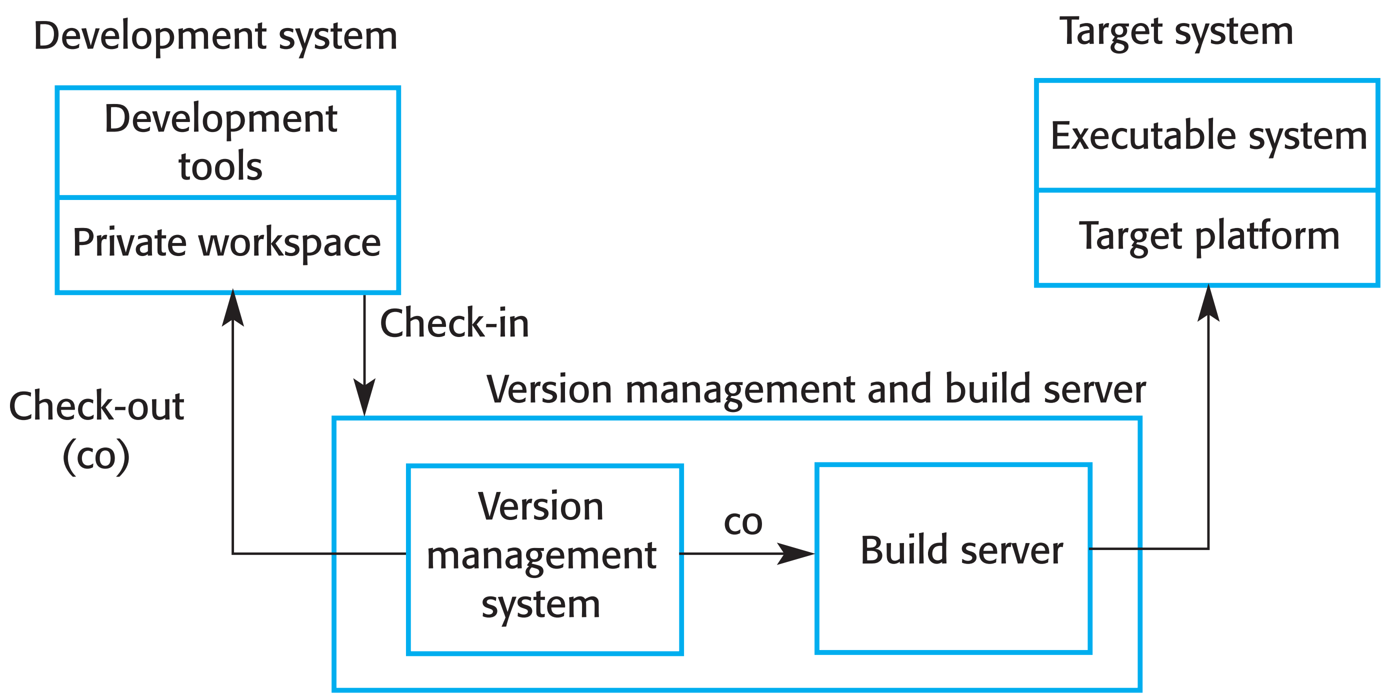
Key points

- ✧ Configuration management is the management of an evolving software system. When maintaining a system, a CM team is put in place to ensure that changes are incorporated into the system in a controlled way and that records are maintained with details of the changes that have been implemented.
- ✧ The main configuration management processes are change management, version management, system building and release management.
- ✧ Change management involves assessing proposals for changes from system customers and other stakeholders and deciding if it is cost-effective to implement these in a new version of a system.
- ✧ Version management involves keeping track of the different versions of software components as changes are made to them.

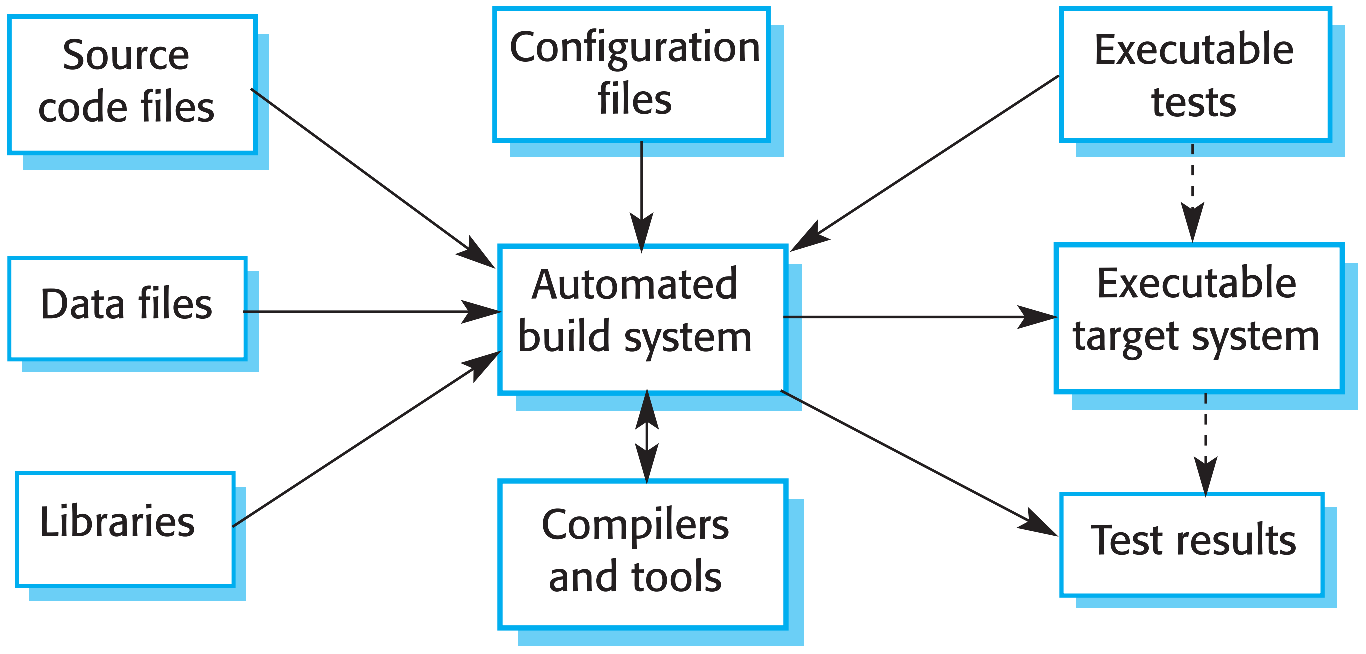
System building

- ✧ System building is 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.

Development, build, and target platforms



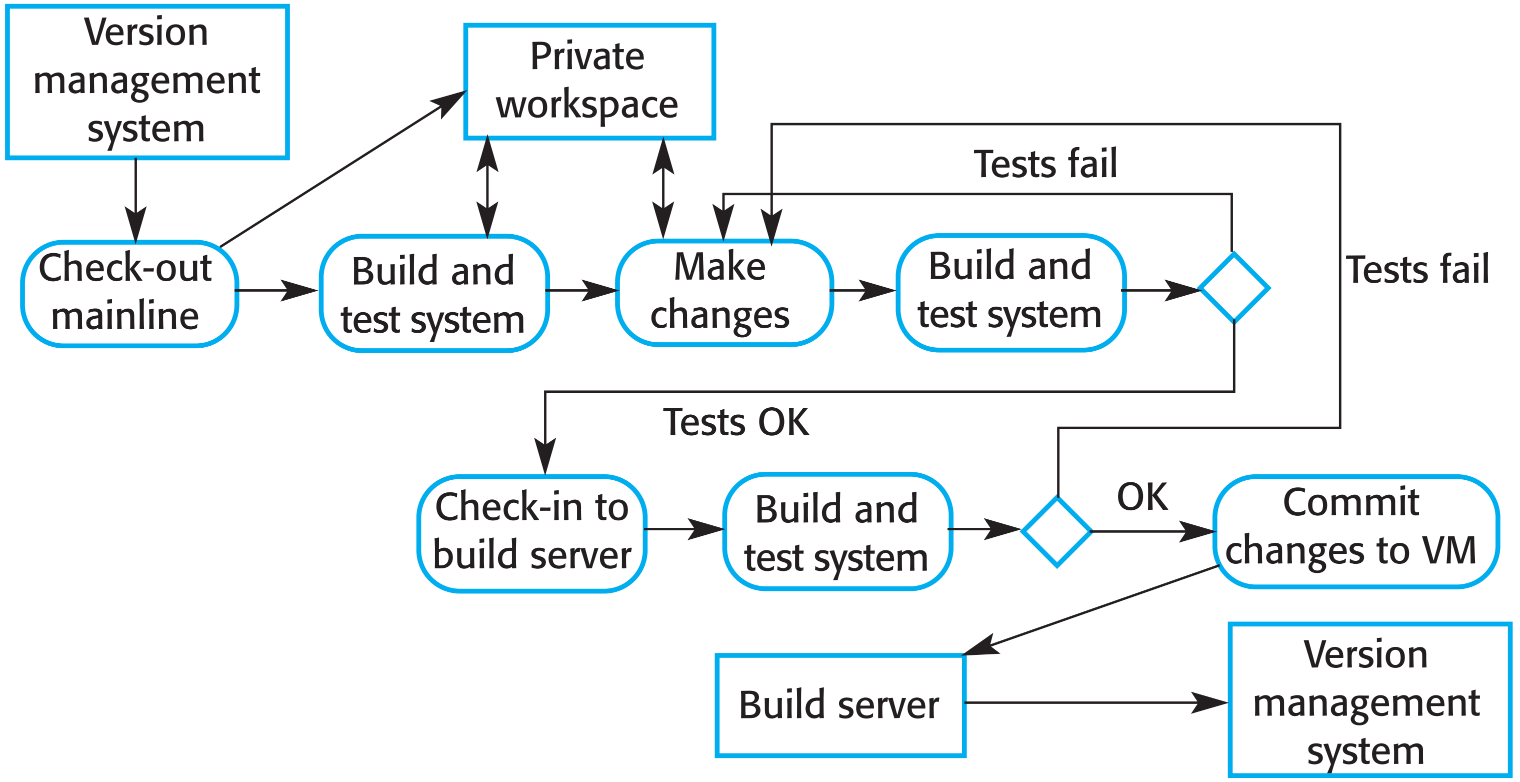
System building



Build system functionality

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

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 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.

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.

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 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.

Key points

- ✧ System building is the process of assembling system components into an executable program to run on a target computer system.
- ✧ Software should be frequently rebuilt and tested immediately after a new version has been built. This makes it easier to detect bugs and problems that have been introduced since the last build.
- ✧ System releases include executable code, data files, configuration files and documentation. Release management involves making decisions on system release dates, preparing all information for distribution and documenting each system release.

Questions?