
Configuration management

Objectives

- To explain the importance of software configuration management (CM)
- To describe key CM activities namely CM planning, change management, version management and system building
- To discuss the use of CASE tools to support configuration management processes

Topics covered

- Configuration management planning
- Change management
- Version and release management
- System building
- CASE tools for configuration management

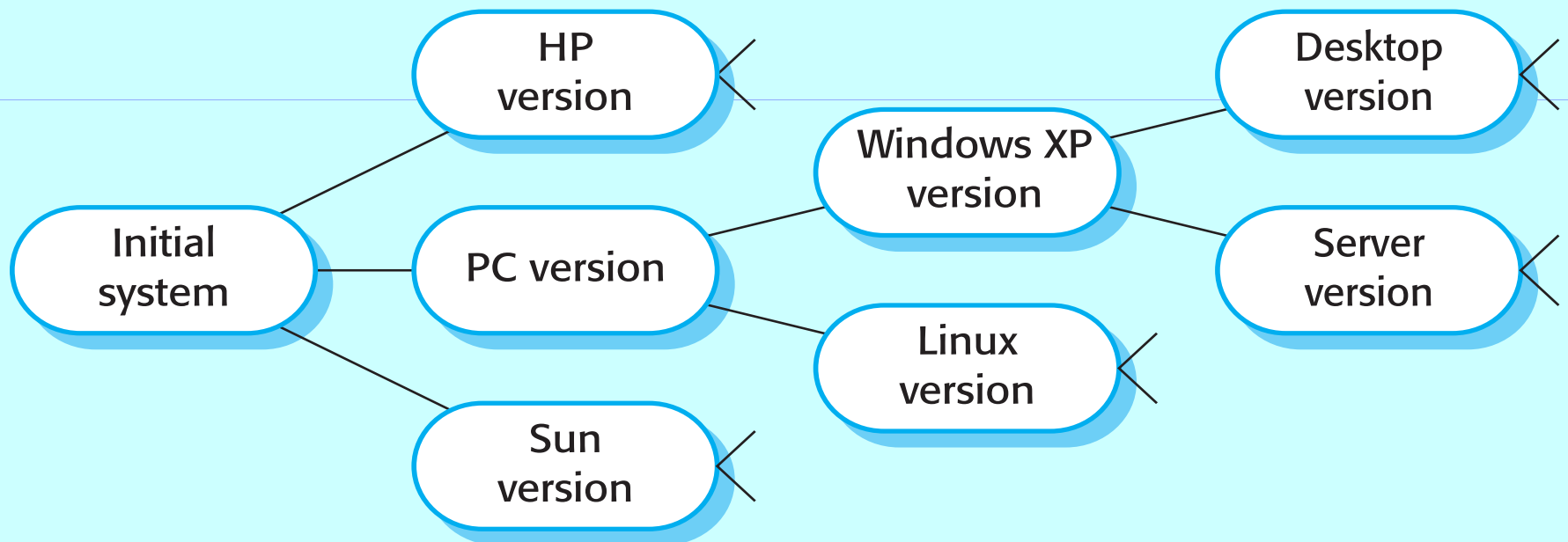
Configuration management

- New versions of software systems are created as they change:
 - For different machines/OS;
 - Offering different functionality;
 - Tailored for particular user requirements.
- Configuration management is concerned with **managing evolving software systems**:
 - System change is a team activity;
 - CM aims to control the costs and effort involved in making changes to a system.

Configuration management

- Involves the **development and application of procedures and standards** to manage an evolving software product.
- CM may be seen as part of a more general **quality management process**.
- When released to CM, software systems are sometimes called **baselines** as they are a starting point for further development.

System families



CM standards

- CM should always be **based on a set of standards** which are applied within an organisation.
- Standards should define **how items are identified, how changes are controlled and how new versions are managed.**
- Standards may be based on **external CM standards** (e.g. IEEE standard for CM).
- Some existing standards are based on a waterfall process model - new CM standards are needed for evolutionary development.

Concurrent development and testing

- A time (say 2pm) for delivery of system components is agreed.
- A new version of a system is built from these components by compiling and linking them.
- This new version is delivered for testing using pre-defined tests.
- Faults that are discovered during testing are documented and returned to the system developers.

Frequent system building

- It is easier to find problems that stem from **component interactions** early in the process.
- This encourages thorough unit testing - developers are under pressure not to 'break the build'.
- A stringent **change management process** is required to keep track of problems that have been discovered and repaired.

Configuration management planning

- All products of the software process may have to be managed:
 - Specifications;
 - Designs;
 - Programs;
 - Test data;
 - User manuals.
- Thousands of separate documents may be generated for a large, complex software system.

The CM plan

- Defines the types of documents to be managed and a document naming scheme.
- Defines who takes responsibility for the CM procedures and creation of baselines.
- Defines policies for change control and version management.
- Defines the CM records which must be maintained.

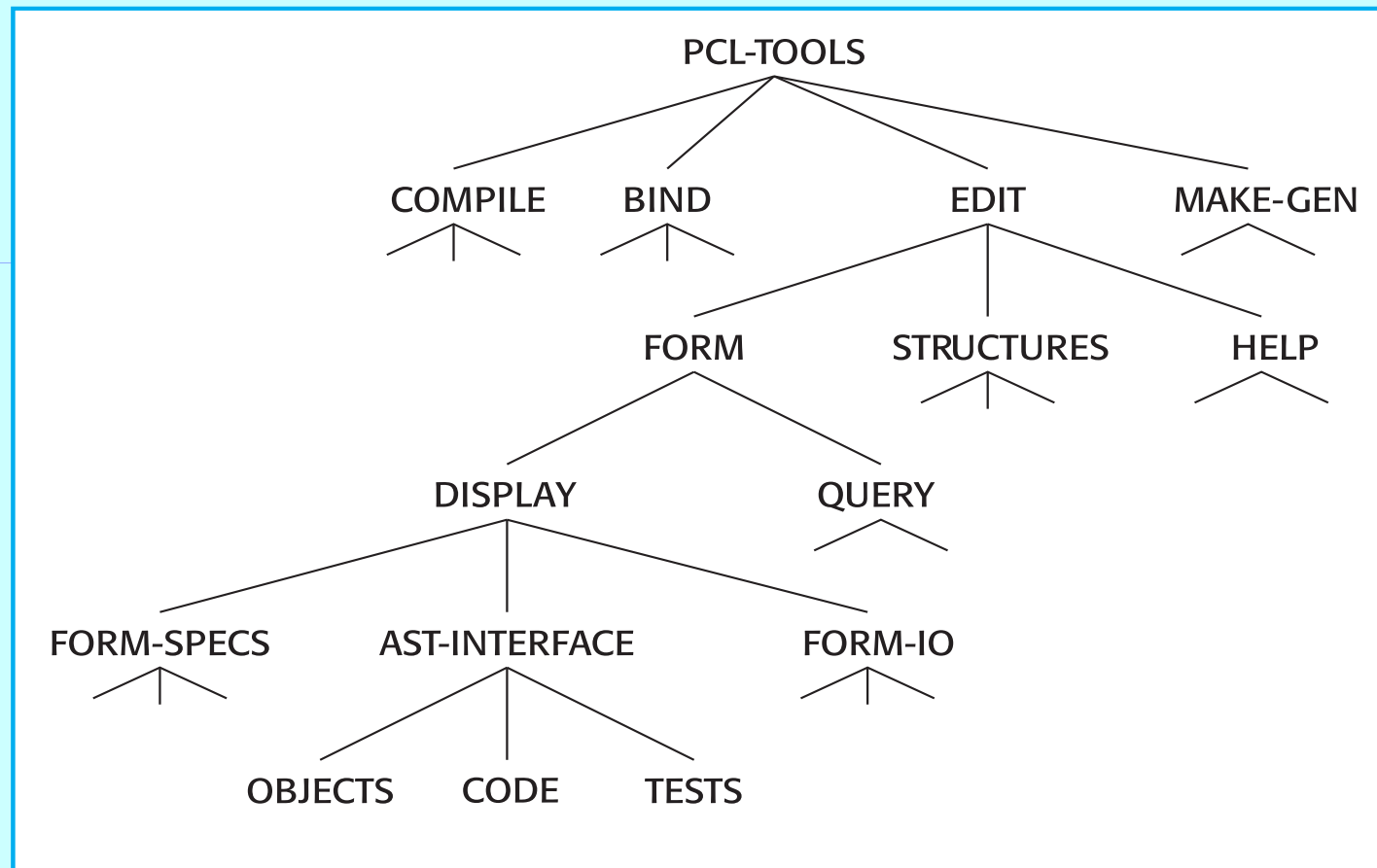
The CM plan

- Describes the tools which should be used to assist the CM process and any limitations on their use.
- Defines the process of tool use.
- Defines the CM database used to record configuration information.
- May include information such as the CM of external software, process auditing, etc.

Configuration item identification

- Large projects typically produce **thousands of documents which must be uniquely identified**.
- Some of these documents must be maintained for the lifetime of the software.
- Document **naming scheme should be defined** so that related documents have related names.
- A hierarchical scheme with multi-level names is probably the most flexible approach.
 - PCL-TOOLS/EDIT/FORMS/DISPLAY/AST-INTERFACE/CODE

Configuration hierarchy



The configuration database

- All CM information should be maintained in a configuration database.
- This should allow queries about configurations to be answered:
 - Who has a particular system version?
 - What platform is required for a particular version?
 - What versions are affected by a change to component X?
 - How many reported faults in version T?
- The CM database should preferably be linked to the software being managed.

CM database implementation

- May be part of an integrated environment to support software development.
 - The CM database and the managed documents are all maintained on the same system
- CASE tools may be integrated with this so that there is a close relationship between the CASE tools and the CM tools.
- More commonly, the CM database is maintained separately as this is cheaper and more flexible.

Change management

- Software systems are subject to continual change requests:
 - From users;
 - From developers;
 - From market forces.
- **Change management** is concerned with keeping track of these changes and ensuring that they are implemented in the most cost-effective way.

The change management process

Request change by completing a change request form

Analyze change request

if change is valid **then**

 Assess how change might be implemented

 Assess change cost

 Submit request to change control board

if change is accepted **then**

repeat

 make changes to software

 submit changed software for quality approval

until software quality is adequate

 create new system version

else

 reject change request

else

 reject change request

Change request form

- The definition of a change request form is part of the CM planning process.
- This form records the change proposed, requestor of change, the reason why change was suggested and the urgency of change (from requestor of the change).
- It also records change evaluation, impact analysis, change cost and recommendations (System maintenance staff).

Change request form

Change Request Form

Project: Proteus/PCL-Tools

Number: 23/02

Change requester: I. Sommerville

Date: 1/12/02

Requested change: When a component is selected from the structure, display the name of the file where it is stored.

Change analyser: G. Dean

Analysis date: 10/12/02

Components affected: Display-Icon.Select, Display-Icon.Display

Associated components: FileTable

Change assessment: Relatively simple to implement as a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.

Change priority: Low

Change implementation:

Estimated effort: 0.5 days

Date to CCB: 15/12/02

CCB decision date: 1/2/03

CCB decision: Accept change. Change to be implemented in Release 2.1.

Change implementor:

Date of change:

Date submitted to QA:

QA decision:

Date submitted to CM:

Comments

Change tracking tools

- A major problem in change management is tracking change status.
- Change tracking tools keep track the status of each change request and automatically ensure that change requests are sent to the right people at the right time.
- Integrated with E-mail systems allowing electronic change request distribution.

Change control board

- Changes should be reviewed by an external group who decide whether or not they are cost-effective from a strategic and organizational viewpoint rather than a technical viewpoint.
- Should be independent of project responsible for system. The group is sometimes called a change control board.
- The CCB may include representatives from client and contractor staff.

Derivation history

- This is a record of changes applied to a document or code component.
- It should record, in outline, the change made, the rationale for the change, who made the change and when it was implemented.
- It may be included as a comment in code. If a standard prologue style is used for the derivation history, tools can process this automatically.

Component header information

```
// BANKSEC project (IST 6087)
//
// BANKSEC-TOOLS/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: N. Perwaiz
// Creation date: 10th November 2002
//
// © Lancaster University 2002
//
// Modification history
// Version      Modifier Date          Change          Reason
// 1.0          J. Jones          1/12/2002      Add header      Submitted to CM
// 1.1          N. Perwaiz          9/4/2003      New field       Change req. R07/02
```


Version and release management

- Invent an identification scheme for system versions.
- Plan when a new system version is to be produced.
- Ensure that version management procedures and tools are properly applied.
- Plan and distribute new system releases.

Versions/variants/releases

- **Version** An instance of a system which is functionally distinct in some way from other system instances.
- **Variant** An instance of a system which is functionally identical but non-functionally distinct from other instances of a system.
- **Release** An instance of a system which is distributed to users outside of the development team.

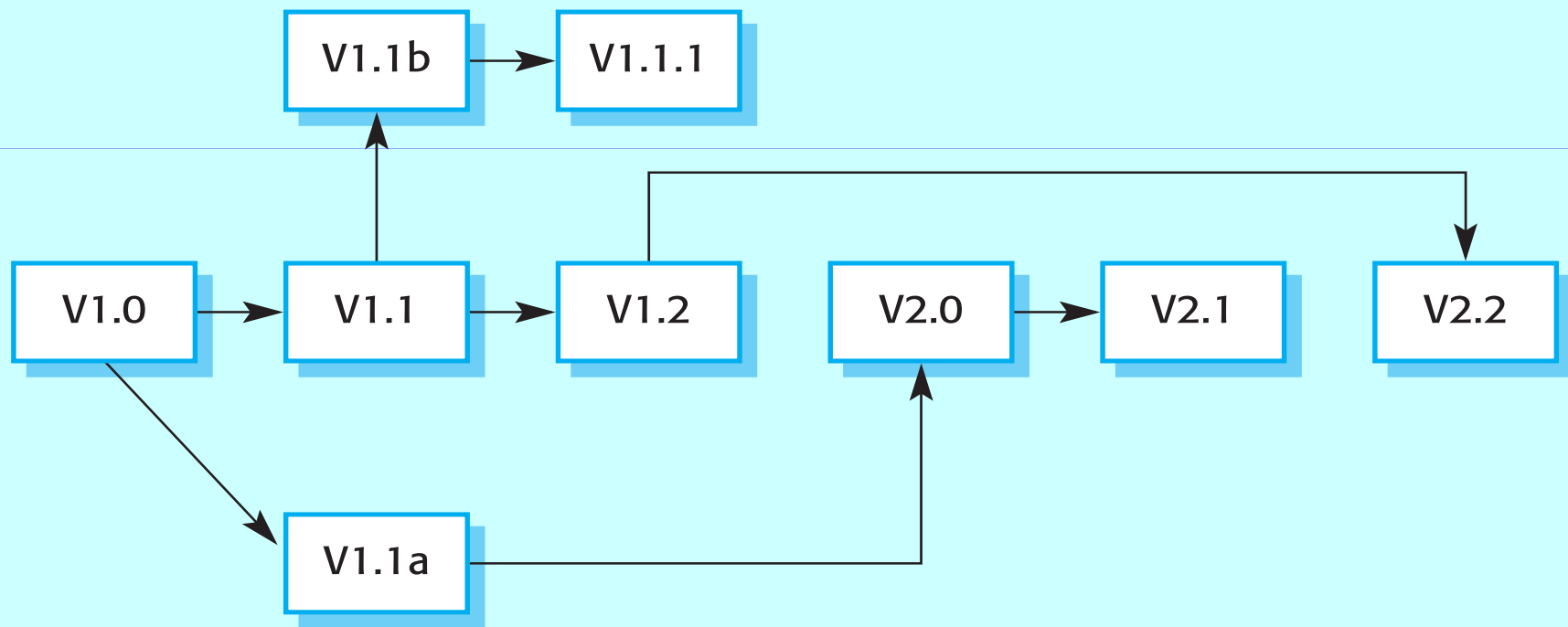
Version identification

- Procedures for version identification should define an unambiguous way of identifying component versions.
- There are three basic techniques for component identification
 - Version numbering;
 - Attribute-based identification;
 - Change-oriented identification.

Version numbering

- Simple naming scheme uses a linear derivation
 - V1, V1.1, V1.2, V2.1, V2.2 etc.
- The actual derivation structure is a tree or a network rather than a sequence.
- Names are not meaningful.
- A hierarchical naming scheme leads to fewer errors in version identification.

Version derivation structure



Attribute-based identification

- Attributes can be associated with a version with the combination of attributes identifying that version
 - Examples of attributes are Date, Creator, Programming Language, Customer, Status etc.
- This is more flexible than an explicit naming scheme for version retrieval; However, it can cause problems with uniqueness - the set of attributes have to be chosen so that all versions can be uniquely identified.
- In practice, a version also needs an associated name for easy reference.

Attribute-based queries

- An important advantage of attribute-based identification is that it can support queries so that you can find ‘the most recent version in Java’ etc.
- The query selects a version depending on attribute values
 - AC3D (language =Java, platform = XP, date = Jan 2003).

Change-oriented identification

- Integrates versions and the changes made to create these versions.
- Used for systems rather than components.
- Each proposed change has a change set that describes changes made to implement that change.
- Change sets are applied in sequence so that, in principle, a version of the system that incorporates an arbitrary set of changes may be created.

Release management

- Releases must incorporate changes forced on the system by errors discovered by users and by hardware changes.
- They must also incorporate new system functionality.
- Release planning is concerned with when to issue a system version as a release.

System releases

- Not just a set of executable programs.
- May also include:
 - Configuration files defining how the release is configured for a particular installation;
 - Data files needed for system operation;
 - An installation program or shell script to install the system on target hardware;
 - Electronic and paper documentation;
 - Packaging and associated publicity.
- Systems are now normally released on optical disks (CD or DVD) or as downloadable installation files from the web.

Release problems

- Customer may not want a new release of the system
 - They may be happy with their current system as the new version may provide unwanted functionality.
- Release management should not assume that all previous releases have been accepted. All files required for a release should be re-created when a new release is installed.

Release decision making

- Preparing and distributing a system release is an expensive process.
- Factors such as the technical quality of the system, competition, marketing requirements and customer change requests should all influence the decision of when to issue a new system release.

System release strategy

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. However, minor system faults may be repaired by issuing patches (often 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 21)	This suggests that the increment of functionality that is included in each release is approximately constant. Therefore, if there has been a system release with significant new functionality, then it may have to be followed by a repair release.
Competition	A new system release may be necessary because a competing product is available.
Marketing requirements	The marketing department of an organisation may have made a commitment for releases to be available at a particular date.
Customer change proposals	For customised 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.

Release creation

- Release creation involves collecting all files and documentation required to create a system release.
- Configuration descriptions have to be written for different hardware and installation scripts have to be written.
- The specific release must be documented to record exactly what files were used to create it. This allows it to be re-created if necessary.

System building

- The process of compiling and linking software components into an executable system.
- Different systems are built from different combinations of components.
- This process is now always supported by automated tools that are driven by 'build scripts'.

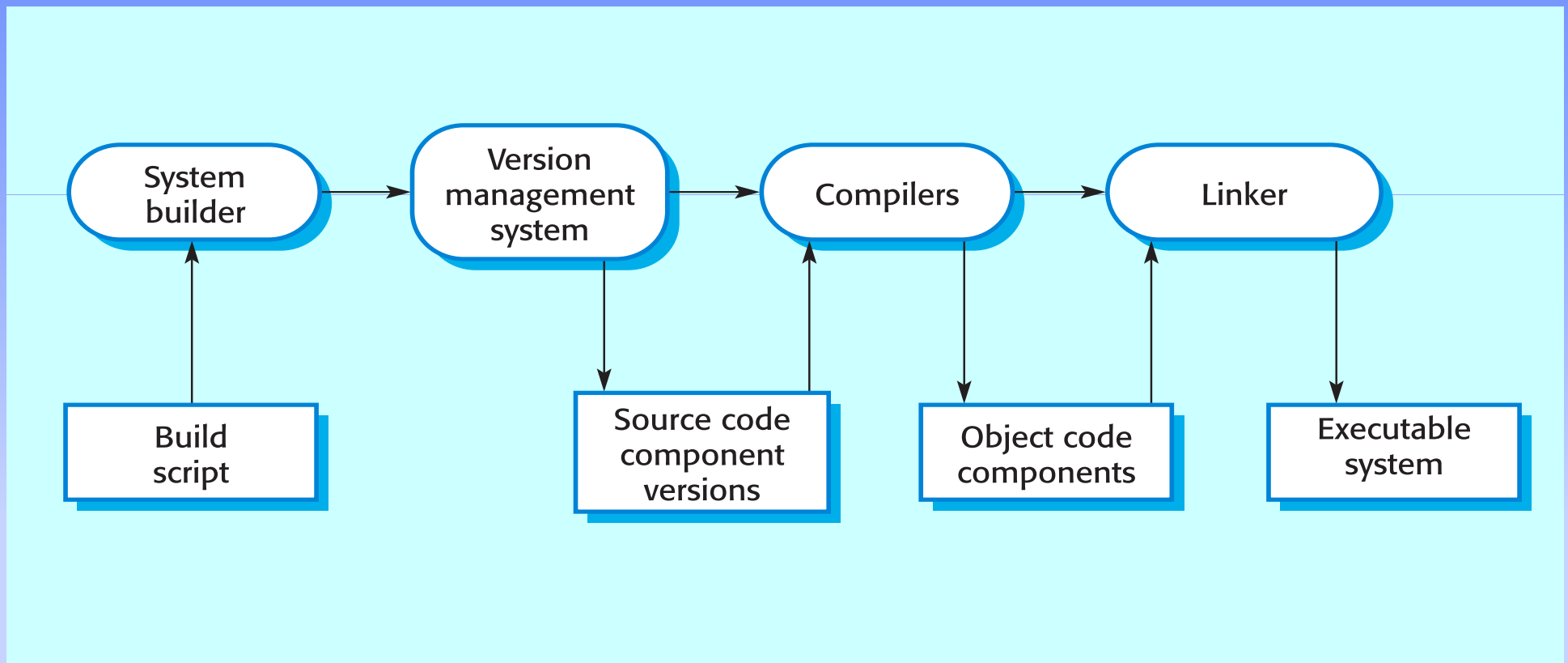
System building problems

- Do the build instructions include all required components?
 - When there are many hundreds of components making up a system, it is easy to miss one out. This should normally be detected by the linker.
- Is the appropriate component version specified?
 - A more significant problem. A system built with the wrong version may work initially but fail after delivery.
- Are all data files available?
 - The build should not rely on 'standard' data files. Standards vary from place to place.

System building problems

- Are data file references within components correct?
 - Embedding absolute names in code almost always causes problems as naming conventions differ from place to place.
- Is the system being built for the right platform
 - Sometimes you must build for a specific OS version or hardware configuration.
- Is the right version of the compiler and other software tools specified?
 - Different compiler versions may actually generate different code and the compiled component will exhibit different behaviour.

System building



CASE tools for configuration management

- CM processes are standardised and involve applying pre-defined procedures.
- Large amounts of data must be managed.
- CASE tool support for CM is therefore essential.
- Mature CASE tools to support configuration management are available ranging from stand-alone tools to integrated CM workbenches.

CM workbenches

- **Open workbenches**
 - Tools for each stage in the CM process are integrated through organizational procedures and scripts. Gives flexibility in tool selection.
- **Integrated workbenches**
 - Provide whole-process, integrated support for configuration management. More tightly integrated tools so easier to use. However, the cost is less flexibility in the tools used.

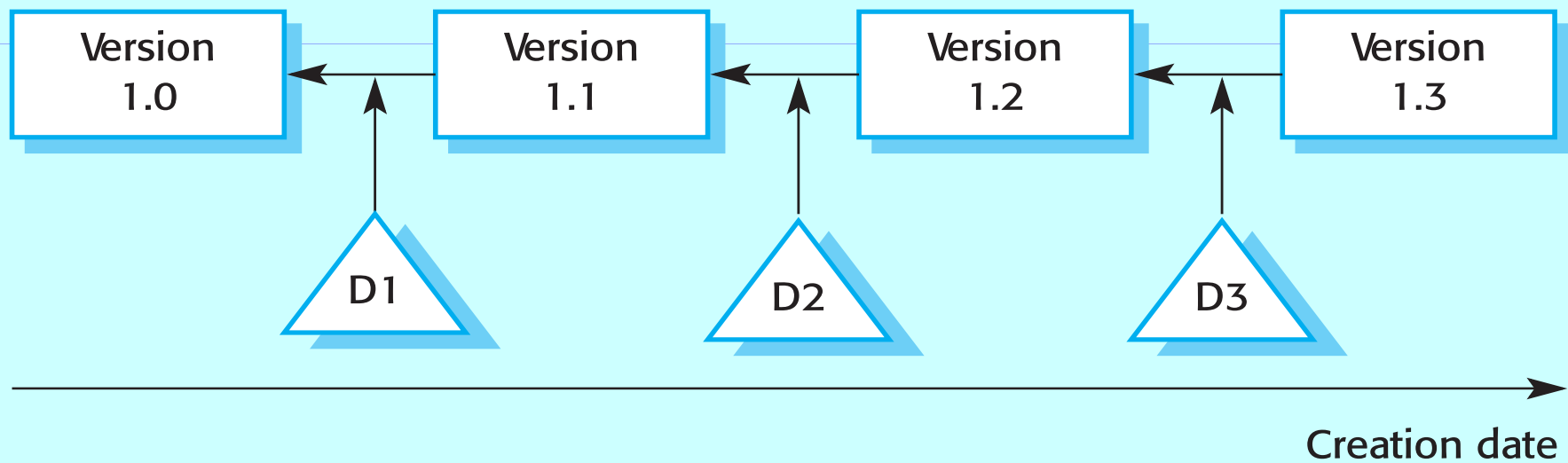
Change management tools

- Change management is a procedural process so it can be modelled and integrated with a version management system.
- Change management tools
 - Form editor to support processing the change request forms;
 - Workflow system to define who does what and to automate information transfer;
 - Change database that manages change proposals and is linked to a VM system;
 - Change reporting system that generates management reports on the status of change requests.

Version management tools

- **Version and release identification**
 - Systems assign identifiers automatically when a new version is submitted to the system.
- **Storage management**
 - System stores the differences between versions rather than all the version code.
- **Change history recording**
 - Record reasons for version creation.
- **Independent development**
 - Only one version at a time may be checked out for change. Parallel working on different versions.
- **Project support**
 - Can manage groups of files associated with a project rather than just single files.

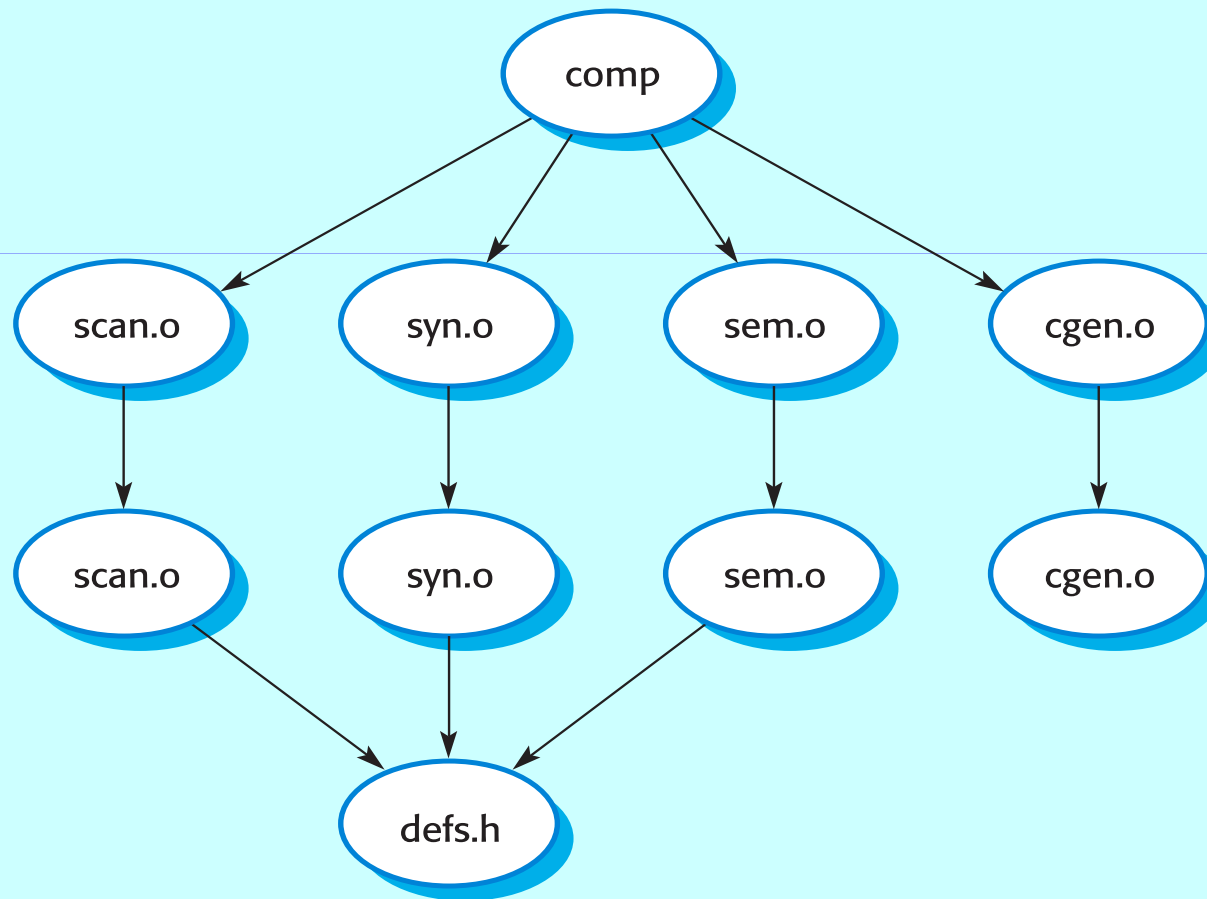
Delta-based versioning



System building

- 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;
 - Derived object management.

Component dependencies



Key points

- Configuration management is the management of system change to software products.
- A formal document naming scheme should be established and documents should be managed in a database.
- The configuration data base should record information about changes and change requests.
- A consistent scheme of version identification should be established using version numbers, attributes or change sets.

Key points

- System releases include executable code, data, configuration files and documentation.
- System building involves assembling components into a system..
- CASE tools are available to support all CM activities
- CASE tools may be stand-alone tools or may be integrated systems which integrate support for version management, system building and change management.