Lecture 10 – Configuration Management (Sommerville Ch. 29)

Karl R. Wilcox
K.R.Wilcox@reading.ac.uk

The University of Reading

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
- (We will not be covering the relationship of CASE tools to Configuration Management in this course)

- Today's seminar
  - The Design of a Configuration Management Setup

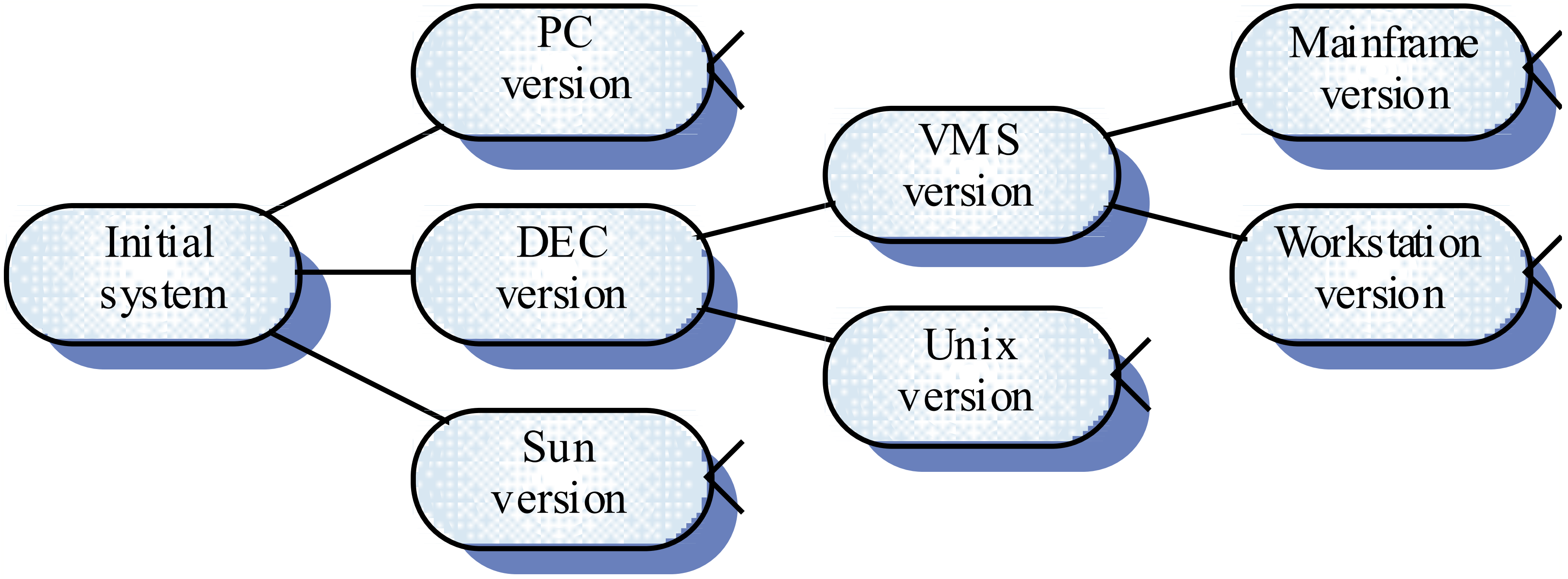
Configuration management

- 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
- 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



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 are generated for a large software system

CM planning

- Starts during the early phases of the project
- Must define the documents or document classes which are to be managed (Formal documents)
- Documents which might be required for future system maintenance should be identified and specified as managed documents

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
- Describes the tools which should be used to assist the CM process
- Defines the CM database used to record configuration information

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

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 managing of these changes and ensuring that they are implemented in the most cost-effective way

Change request form

- Definition of change request form is part of the CM planning process
- Records change required, suggestor of change, reason why change was suggested and urgency of change(from requestor of the change)
- Records change evaluation, impact analysis, change cost and recommendations (System maintenance staff)

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
- May include representatives from client and contractor staff

Derivation history

- Record of changes applied to a document or code component
- Should record, in outline, the change made, the rationale for the change, who made the change and when it was implemented
- 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

```
// PROTEUS project (ESPRIT 6087)
//
// PCL-TOOLS/EDIT/FORMS/DISPLAY/AST-INTERFACE
//
// Object: PCL-Tool-Desc
// Author: G. Dean
// Creation date: 10th November 1998
//
// © Lancaster University 1998
//
// Modification history
// Version      Modifier     Date        Change        Reason
// 1.0          J. Jones     1/12/1998   Add header    Submitted to CM
// 1.1          G. Dean      9/4/1999    New field     Change req. R07/99
```

Version and release management

- Invent identification scheme for system versions
- Plan when 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

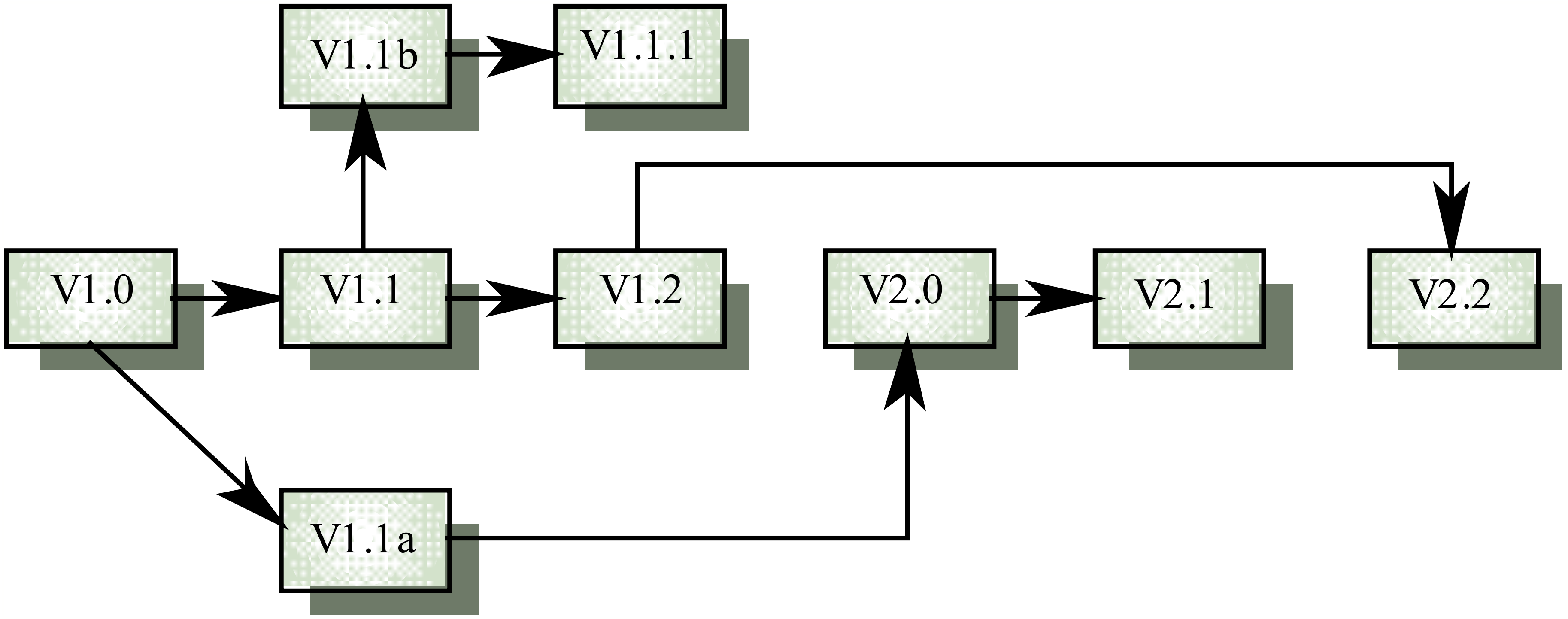
- Procedures for version identification should define an unambiguous way of identifying component versions
- Three basic techniques for component identification
  - Version numbering
  - Attribute-based identification
  - Change-oriented identification

Version numbering

- Simple naming scheme uses a linear derivation e.g. V1, V1.1, V1.2, V2.1, V2.2 etc.
- Actual derivation structure is a tree or a network rather than a sequence
- Names are not meaningful.
- Hierarchical naming scheme may be better

- Marketing may have an influence on version numbering!
  - Consider Microsoft Office & Windows….

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.
- More flexible than an explicit naming scheme for version retrieval; Can cause problems with uniqueness
- 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.
- Example
  - AC3D (language =Java, platform = NT4, date = Jan 1999)

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 CD-ROM 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 must 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 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
- Invariably 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 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

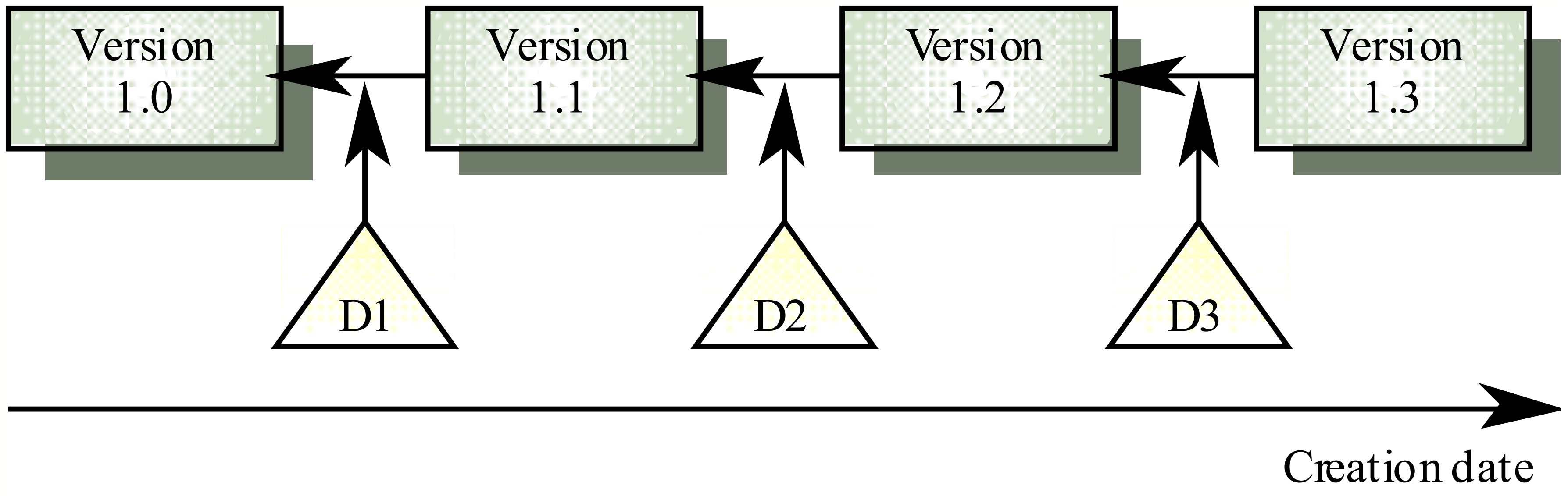
Change management tools

- 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

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

Delta-based versioning



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