Software 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

Overview

- What is a configuration?
- Configuration management planning
- Change management
- Version and release management
- System building
- CASE tools for configuration management

What is a configuration?

- A configuration is the source code and documentation for a specific version of an evolving software system.
- Documentation includes:
  - The SRS(software requirement specification) document
  - Design documents
  - Test cases and test data
  - User manuals

Why configuration management ?

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

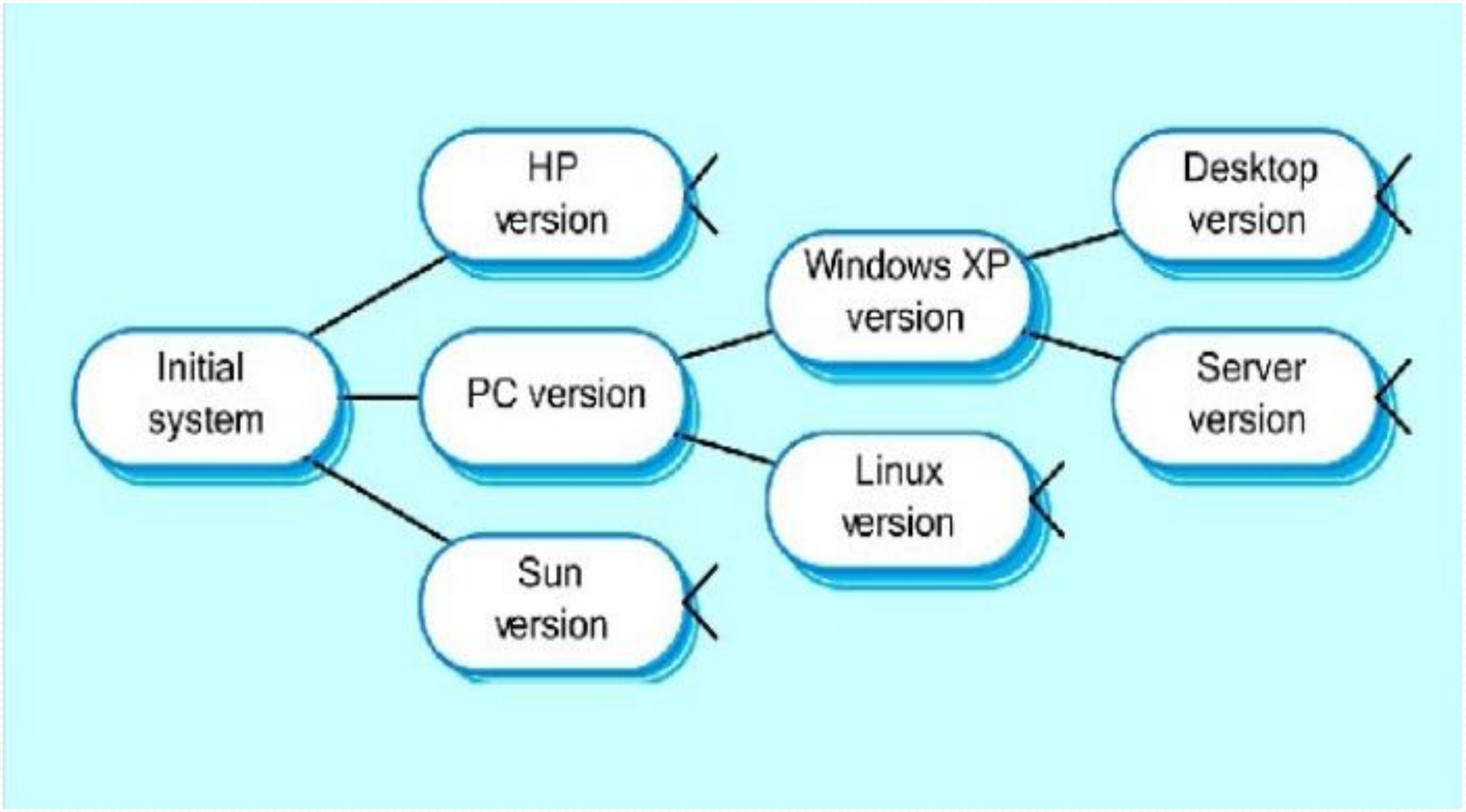
Configuration management (CM)

- The process of software development and maintenance is controlled is called CM.
- 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.
- Involves the development and application of procedures and standards to manage an evolving software product.

CM Activities

- The activity are divided into four broad categories:
    1. The identification of components and changes
    2. The control of the way by which the changes are made
    3. Auditing the changes
    4. Status accounting recording and documenting all the activities that have take place

System Families



CM standards

- CM should always be based on a set of standards which are applied within an organization.
- 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).

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.

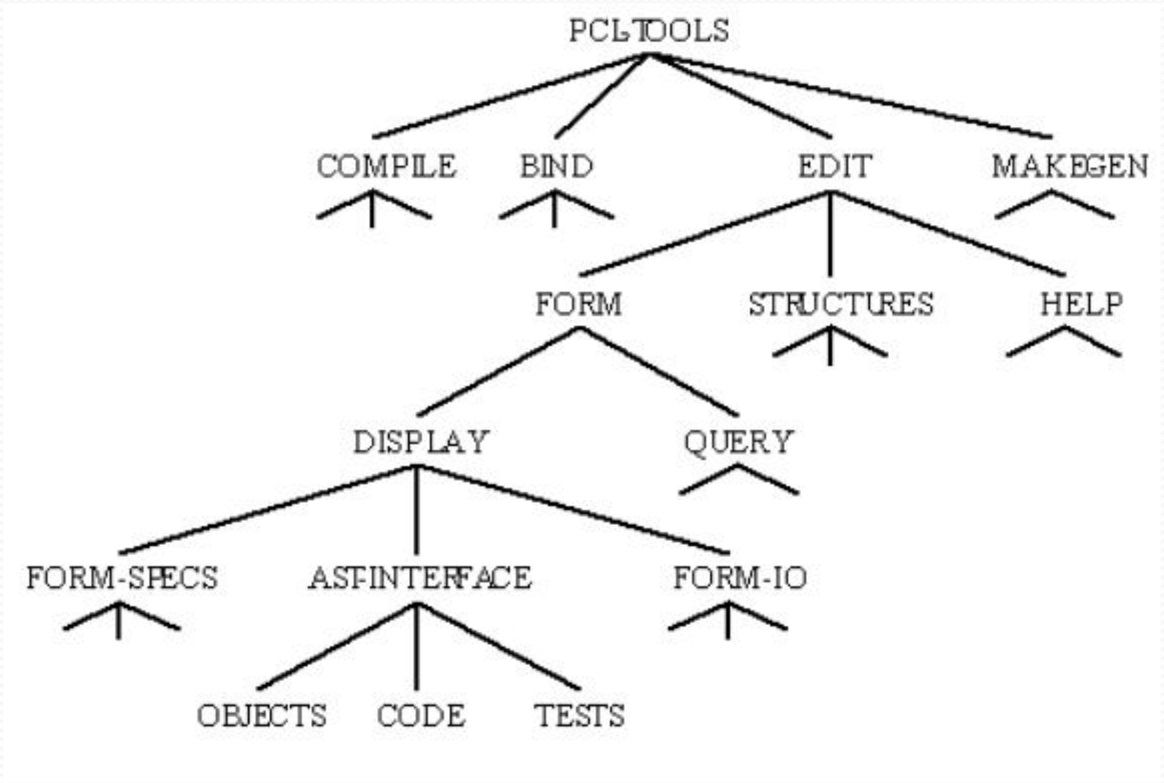
The CM plan

- Defines the configuration items to be managed and identify a naming scheme.
- Defines who takes responsibility for the CM procedures and creation of baselines.
- Defines policies that all team members must use for change control and version management.
- Defines the structure of the CM database.
- Describes the tools which should be used to assist the CM process and any limitations on their use.

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/COD
E

Configuration hierarchy



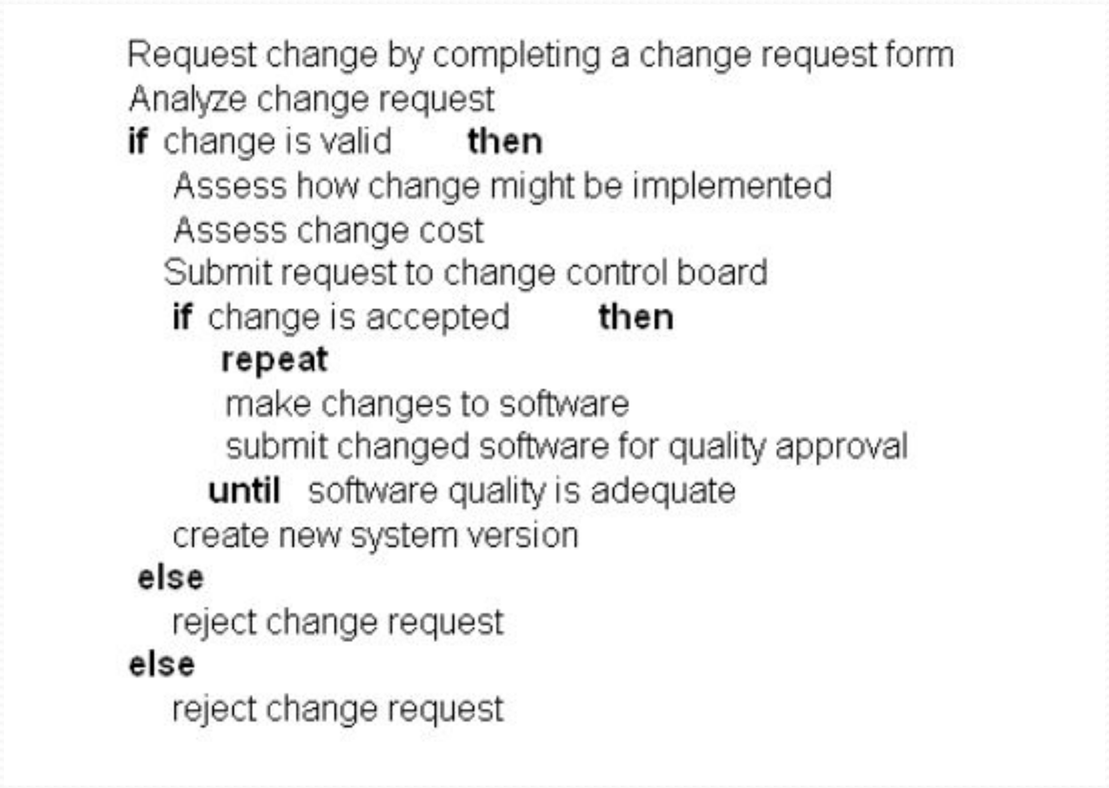
The configuration database

- All CM information should be maintained in a configuration database.
- The goal is to assess the impact of change and generate reports for management.
- 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.

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 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 group responsible for the 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.

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, and 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 designed for different hardware platforms.
- Release An instance of a system which is distributed to users outside of the development team.

Version identification

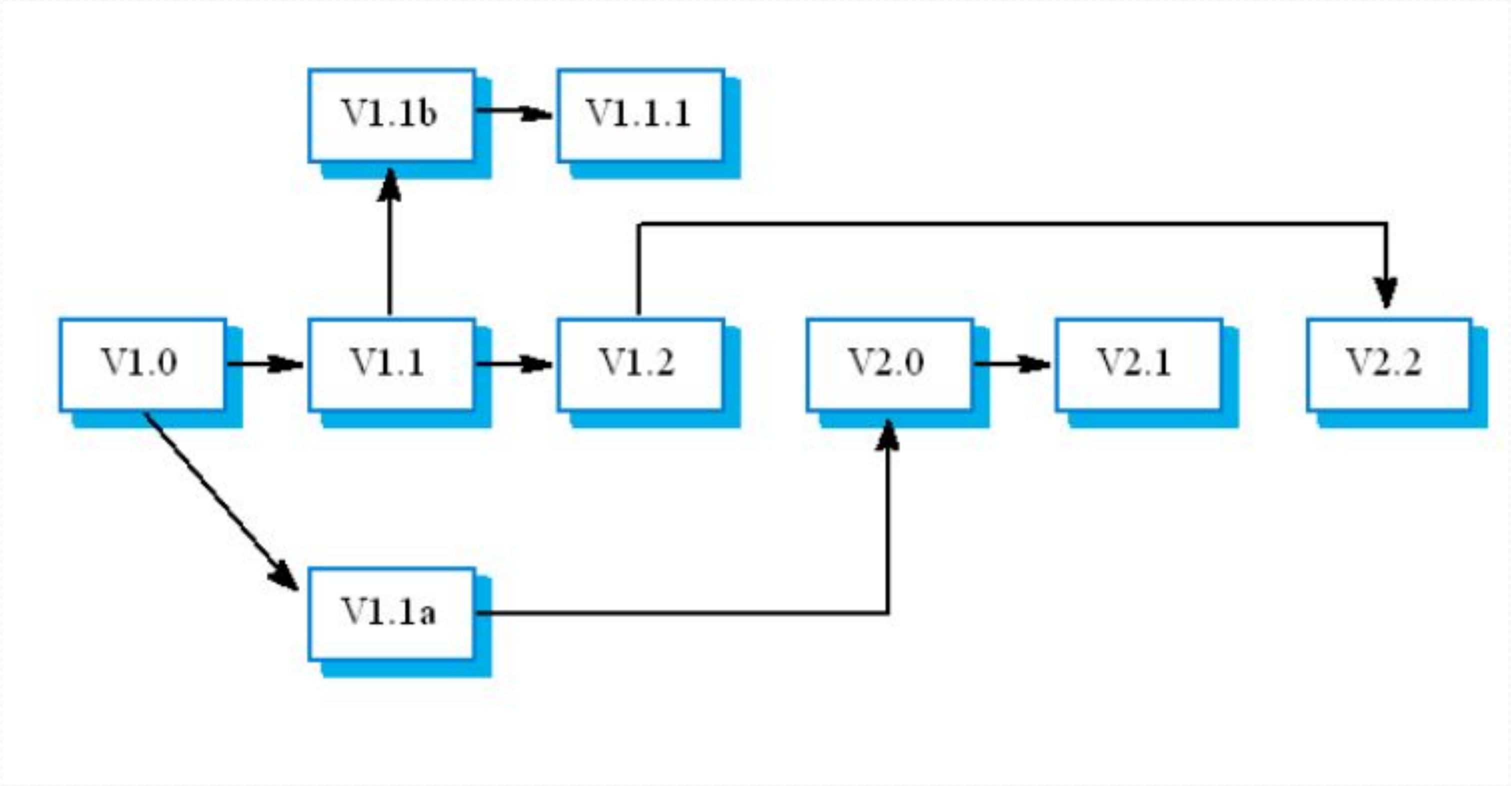
- Procedures for version identification should define an unambiguous way of identifying component versions.
- There are three basic techniques for component identification
  - Version numbering;
  - Attribute-based identification;
  - Change-oriented identification

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 numbering 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, 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.
- Example: a version of the software system AC3D developed in Java for Windows XP in January 2004 would be identified by:
    - AC3D (language=java, platform=XP, date=Jan2004)

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 release

- 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.

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’.

CASE Tools for CM

- CASE stands for Computer Aided Software Engineering. It means, development and maintenance of software projects with help of various automated software tools.
- An instance of software is released under one version. Configuration Management tools deal with –
  - Version and revision management
  - Baseline configuration management
  - Change control management
- CASE tools help in this by automatic tracking, version management and release management.

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.

conclusion

- Configuration management is a necessary tool for managing complex software systems. Lack of configuration management can cause serious problems with reliability, uptime, and the ability to scale a system. Many current software development tools have configuration management features built in. Bitbucket offers a powerful system for configuration management.
- CM tools are Git, Docker, Terraform, Ansible, Salt Stack, Chef and Puppet.

Thank You

References:- This chapter is extracted from Ian Sommerville slides. Text book chapter 29

https://www.atlassian.com/continuous-delivery/principles/configuration-management

https://docplayer.net/5455794-Configuration-management-ian-sommerville-2004-software-engineering-7th-edition-chapter-29-slide-1.html