Software Engineering

Configuration Management
(CM)
Objective

- Introduce the concept of Configuration Management (CM) and Change management
- Describe the change management process
- Explain system release strategies
- Explain version management tools
- Introduce Delta-based versioning
What is Configuration Management

• Managing the products of system changes
Configuration management

- Procedures and standards to manage an evolving software product
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
The configuration database

- All CM information should be maintained in a configuration database

- Configuration database should allow queries about configurations to be answered on
  - 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?
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
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
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
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 may be:
  - released on CD-ROM or
  - downloaded 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 decision making

When to issue a new system release

- Preparing and distributing a system release is an expensive process.

- System release strategy: Factors affecting “When to issue a new system release“:
  - technical quality of the system,
  - competition,
  - marketing requirements, and
  - customer change requests.
# System release strategy

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical quality of the system</td>
<td>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.</td>
</tr>
<tr>
<td>Lehman’s fifth law (see Chapter 27)</td>
<td>This suggests that the increment of functionality which 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.</td>
</tr>
<tr>
<td>Competition</td>
<td>A new system release may be necessary because a competing product is available.</td>
</tr>
<tr>
<td>Marketing requirements</td>
<td>The marketing department of an organisation may have made a commitment for releases to be available at a particular date.</td>
</tr>
<tr>
<td>Customer change proposals</td>
<td>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.</td>
</tr>
</tbody>
</table>
Release creation

- Release creation involves collecting all files and documentation required to create a system release.

- Configuration descriptions and installation scripts have to be written for different hardware.

- 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
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 system built with the wrong version may work initially but fail after delivery

• Are all data files available?
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

• 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

- System builder
- Version management system
- Compilers
- Linker

- Build script
- Source code component versions
- Object code components
- Executable system
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
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

- Version 1.0
- Version 1.1
- Version 1.2
- Version 1.3

D1 D2 D3

Creation date

V1.2 V1.3 + D3
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
Key CM 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 database 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 CM 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