

# Software Configuration Management

---



Reto Bonderer

[reto.bonderer@fh-htwchur.ch](mailto:reto.bonderer@fh-htwchur.ch)

University of Applied Sciences Chur

with minor modifications by M. Kropp

University of Applied Sciences Solothurn NWCH

# Learning Goals

---

## ▲ The participant

- ◆ knows why configuration management is important
- ◆ knows what version, release and change management is about
- ◆ knows how to plan configuration management
- ◆ understands how tools support the configuration management processes and can decide whether a tool is feasible for a particular project

# Content

---

- ▲ Version and release management
- ▲ Change management
- ▲ Build management
- ▲ Configuration management planning
- ▲ Configuration identification
- ▲ Subcontractor control and inclusion of COTS parts
- ▲ Tool support

# References

---

- Ian Sommerville: *Software Engineering*, 6<sup>th</sup> edition, Pearson Education 2001
- K. Frühauf, J. Ludewig, H. Sandmayr: *Software – Projektmanagement und – Qualitätssicherung*, 4. durchgesehene Auflage, vdf Zürich 2002 (in German)
- *IEEE Software Engineering Standards Collection*, 1999 edition, IEEE Press, [www.ieee.org](http://www.ieee.org)
- [www.cvshome.org](http://www.cvshome.org)
- [www.sei.cmu.edu](http://www.sei.cmu.edu)



Carnegie Mellon  
Software Engineering Institute

# Acronyms

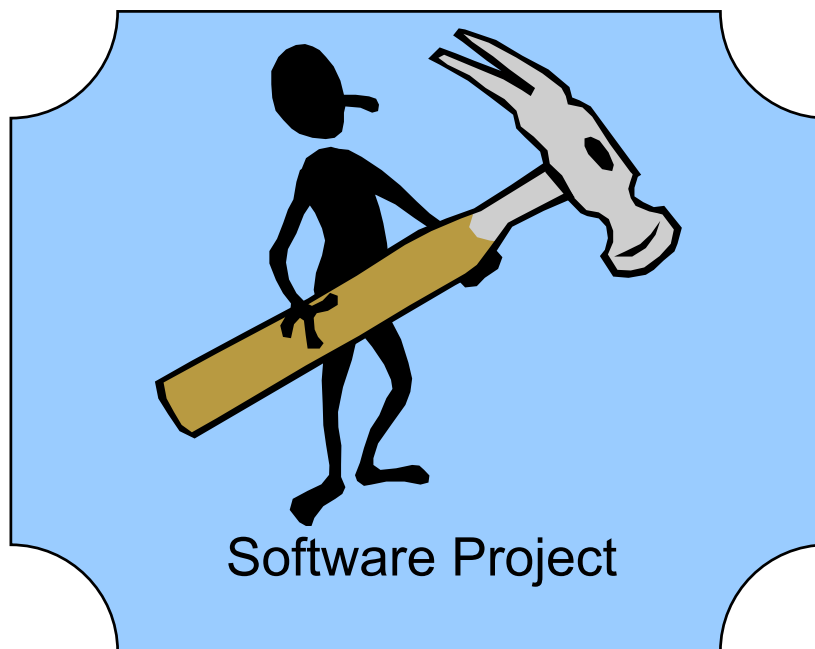
---

▲ CCB	Configuration Control Board
▲ CM	Configuration Management
▲ CMM	Capability Maturity Model
▲ COTS	Commercial Off The Shelf
▲ IEEE	Institute of Electrical and Electronics Engineers
▲ SCM	Software CM
▲ SCMP	SCM Plan
▲ SCR	Software Change Request

# Most Important Software Processes

## Software Project Management

Requirements  
Management



Software  
Quality  
Assurance

## Software Configuration Management

# Symptoms of poor CM

---

- ▲ Bugs that have been corrected reappear
- ▲ Previous releases of software cannot be rebuilt
- ▲ Previous releases of software cannot be found
- ▲ Files get lost
- ▲ Files are “mysteriously” changed
- ▲ The same or similar code exists multiple times in different projects
- ▲ Two developers accidentally change the same file concurrently

# Benefits of an appropriate CM

---

- ▲ Systems are built “automatically” and faster
- ▲ Reintroducing already corrected bugs is avoided  
→ better reputation of company
- ▲ Concurrent development is simplified
- ▲ Appropriate CM is a prerequisite for distributed development



# But: There's No Free Lunch

---

## ▲ There are risks with:

### ◆ Business

- Cost (license fees, training, administration)
- Culture change

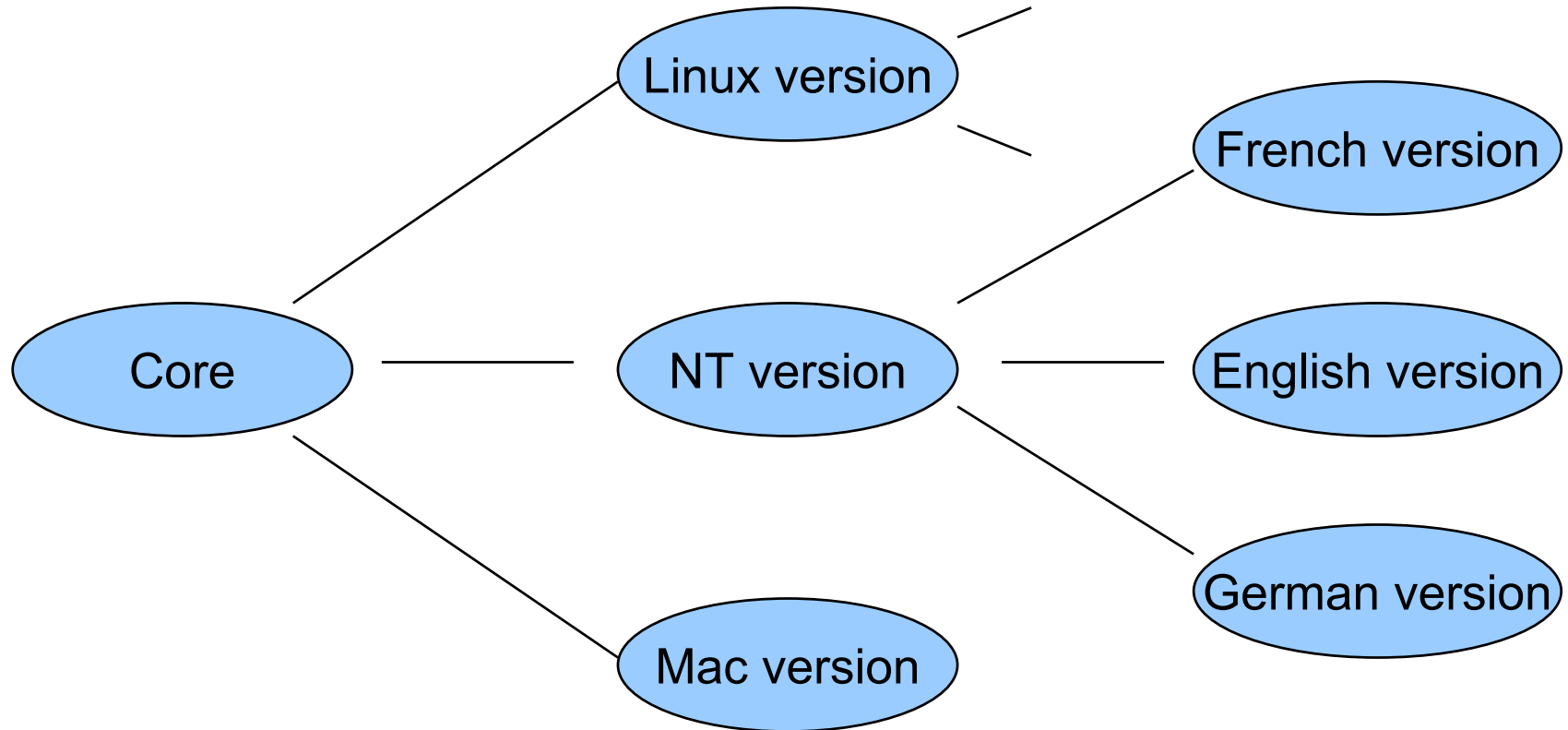
### ◆ People

- Some prefer a different tool
- Resistance to introduce CM (different reasons)
- Cheating (people work around a tool)

### ◆ Technology

- Access and security
- Scalability

# Potential Product Version Tree



# What is SCM?

---

*Software configuration management (SCM)* is responsible to establish and maintain the integrity of the products of the software project throughout the software life cycle.

This includes identifying configuration items, controlling changes and recording and reporting the change implementation status.

# What is a *Configuration*?

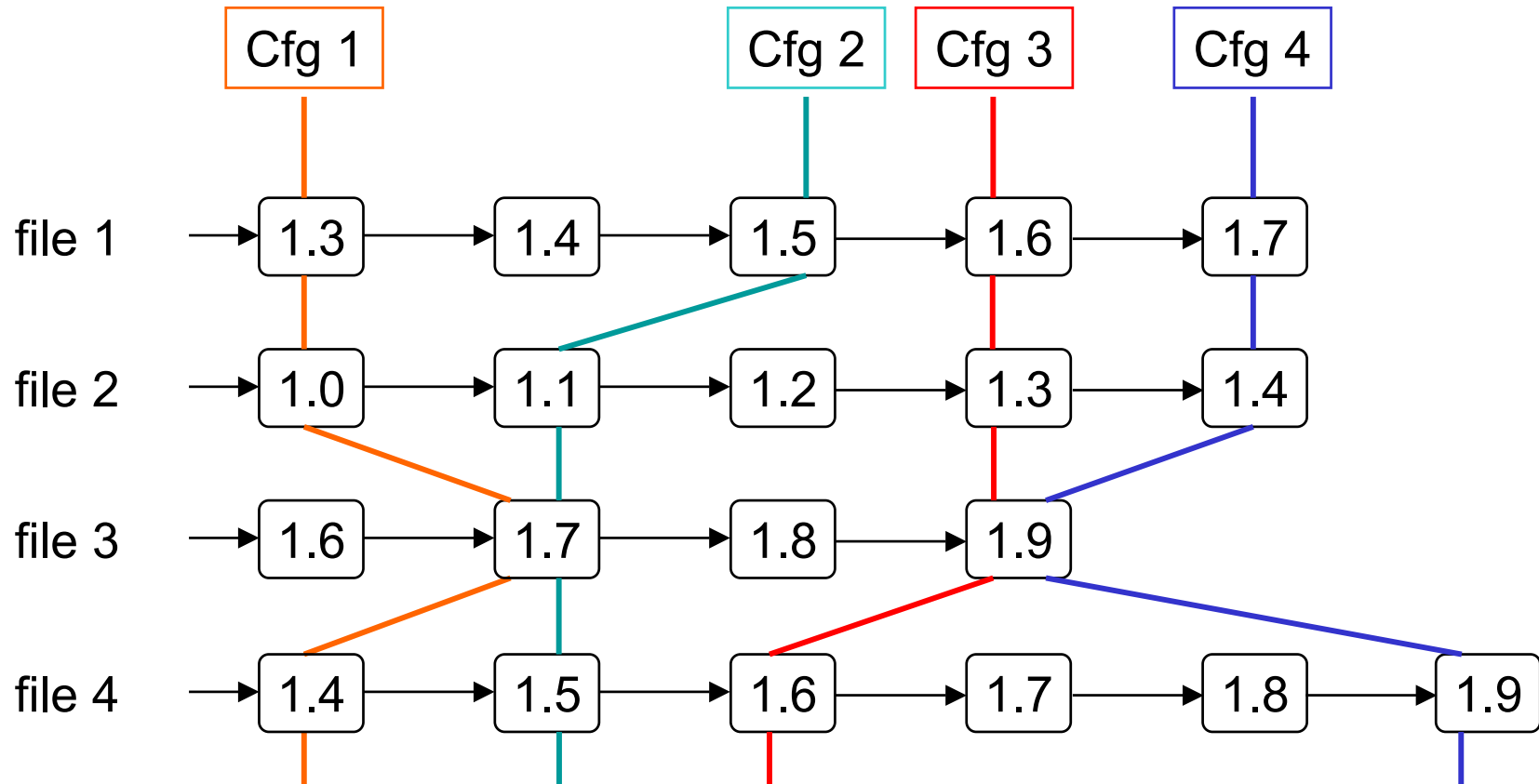
---

*A configuration*

is the functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product.

*IEEE Standard Glossary of Software Engineering Terminology*

# Example: Configurations



# What is a *Configuration item*?

---

*A configuration item*

is an aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process.

*IEEE Standard Glossary of Software Engineering Terminology*

# Example: Driver configuration of a PC SWEED

Gerätename	Treiberversion	Gerätekennung
Laufwerk	5.1.2535.0	IDE\DISKMAXTOR_6L080J4
ISAPnP-Datenleseport	5.1.2600.0	ISAPNP\READDATAPORT\0
Logische Schnittstelle für Druckeranschluss	5.1.2600.0	LPTENUM\MICROSOFTRAWPORT\5&D64CBF1&0&LPT1
NVIDIA 64MB DDR GeForce3 Ti 200 (Dell)	2.1.9.3	PCI\VEN_10DE&DEV_0201&SUBSYS_88511462&REV_A3\
Creative SB Live! Value (WDM)	3511.0.0.0	PCI\VEN_1102&DEV_0002&SUBSYS_80221102&REV_0A\
Creative SBLive!-Gameport	5.1.2535.0	PCI\VEN_1102&DEV_7002&SUBSYS_00201102&REV_0A\
Eicon Diva 2.02 S/T (PCI)	3.5.101.70	PCI\VEN_1133&DEV_E00B&SUBSYS_E00B1133&REV_00\
Intel(R) 82845 Prozessor-zu-E/A-Controller - ...	5.1.2600.0	PCI\VEN_8086&DEV_1A30&SUBSYS_00000000&REV_04\3
Intel(R) 82845 Prozessor-zu-AGP-Controller	5.1.2600.0	PCI\VEN_8086&DEV_1A31&SUBSYS_00000000&REV_04\3
Intel(R) 82801BA LPC-Schnittstellencontroll...	5.1.2600.0	PCI\VEN_8086&DEV_2440&SUBSYS_00000000&REV_05\3
Intel(r) 82801BA/BAM USB universeller Hos...	5.1.2600.0	PCI\VEN_8086&DEV_2442&SUBSYS_01151028&REV_05\3
Intel(R) 82801BA/BAM SMBus-Controller - ...	5.1.2600.0	PCI\VEN_8086&DEV_2443&SUBSYS_01151028&REV_05\3
Intel(r) 82801BA/BAM USB universeller Hos...	5.1.2600.0	PCI\VEN_8086&DEV_2444&SUBSYS_01151028&REV_05\3
Intel(r) 82801BA Ultra ATA Controller	1.1.1.2076	PCI\VEN_8086&DEV_244B&SUBSYS_01151028&REV_05\3
Intel(R) 82801BA/CA PCI-Brücke - 244E	5.1.2600.0	PCI\VEN_8086&DEV_244E&SUBSYS_00000000&REV_05\3
Primary IDE Channel	1.1.1.2076	PCIIDE\IDECHANNEL\4&2AD6B33C&0&0
Secondary IDE Channel	1.1.1.2076	PCIIDE\IDECHANNEL\4&2AD6B33C&0&1
ACPI-Uniprozessor-PC	5.1.2600.0	ROOT\ACPI_HAL\0000
Volume-Manager	5.1.2600.0	ROOT\FTDISK\0000

# Exercise: Car manufacturer

---

Consider a car manufacturer

- ◆ What are the configurations a car manufacturer has to deal with?
- ◆ How does he know how a customer-specific car is built?
- ◆ If an important part (i.e. the brakes) has a bug in a whole production lot: in which cars must the part be replaced?



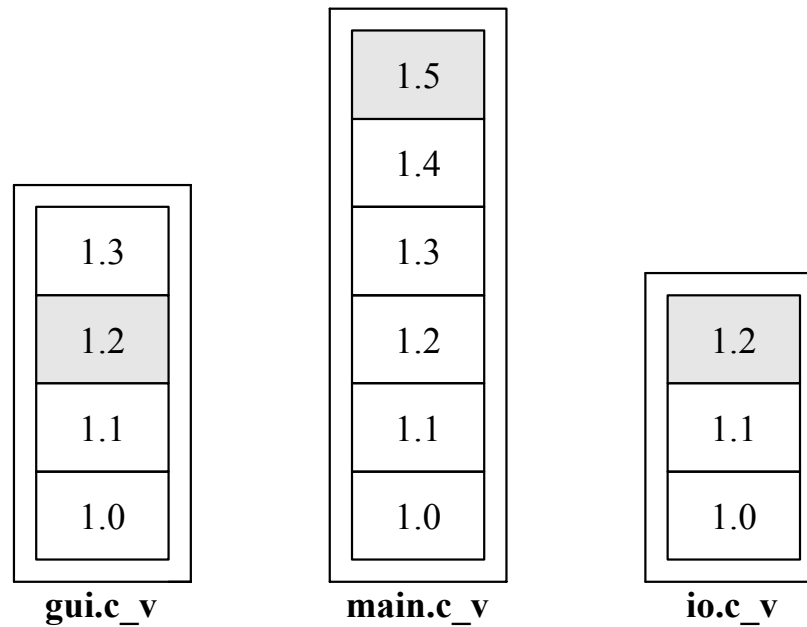
# CM Activities

---

- ▲ Version and release management
- ▲ Change management
- ▲ System building
- ▲ CM planning

# Version and Release Management

---



# Baseline

---

A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

*IEEE Standard Glossary of Software Engineering Terminology*

An initial release or re-release of a computer software **configuration item**, associated with a complete compilation or recompilation of the computer software configuration item.

*IEEE Standard Glossary of Software Engineering Terminology*

The formal notification and distribution of an approved version.

*IEEE Standard for Software Configuration Management Plans*

# Variant

---

A variant is an instance of a system which is *functionally identical* but *non-functionally distinct* from other instances of a system.

*Ian Sommerville*

Examples:

- ◆ Natural language "versions" (English, French, ...)
- ◆ "Versions" for different operating systems
- ◆ Customer specific "versions"

# Release Identification

---

- ▲ Releases are usually identified with a label consisting of three numbers, separated by a dot.

**X.Y.Z**

- ▲ The following conventions usually apply:
  - X represents a major new version of a system (starts with 1)
  - Y represents the update level with minor functional changes (starts with 0)
  - Z is used for bug fix versions (starts with 0)

# 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
  - ◆ Data files needed for system operation
  - ◆ Installation program or shell script
  - ◆ Electronic and paper documentation
  - ◆ Packaging and associated publicity
- ▲ Systems are now normally released on DVD, CD-ROM or as downloadable installation files from the web

# 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

# Exercise: Version and Release Mgmt

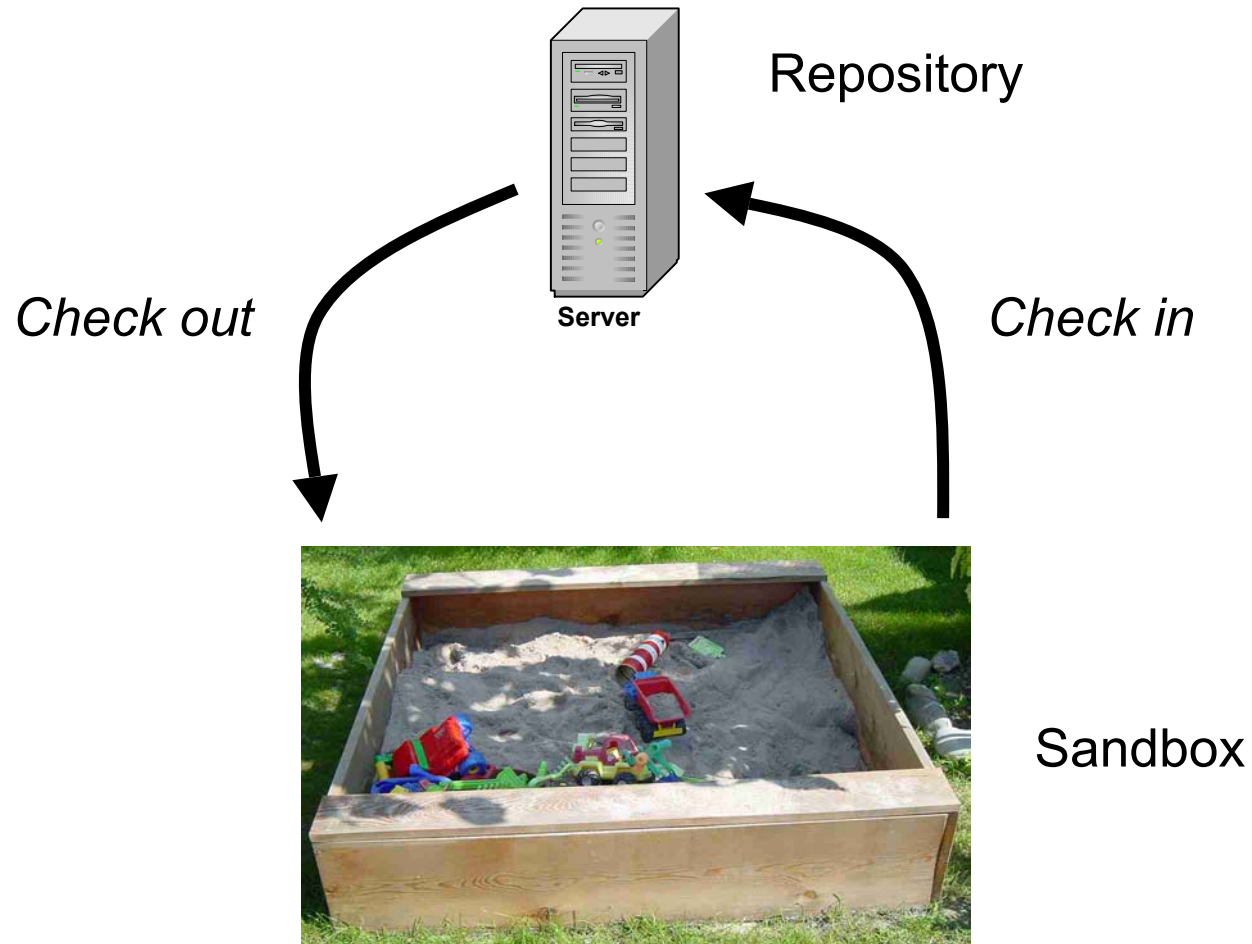
---

- ▲ Discuss whether Windows 98 is new version or just a revision of Windows 95.
- ▲ A software company selling an Office suite wants to ship a new release every 18 months. What is the impact of this fact on the development process?

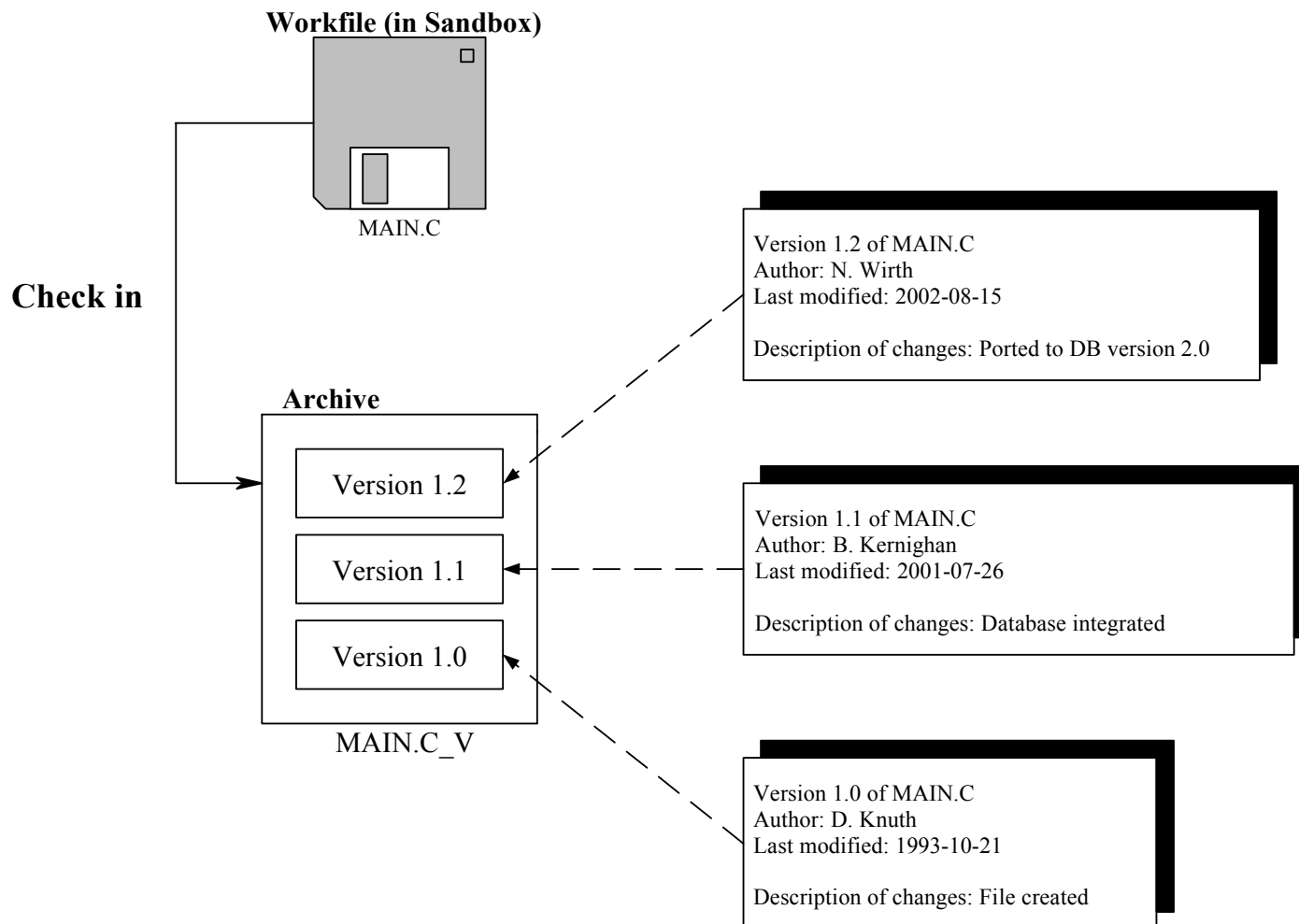
# Version Management Tools

---

- ▲ Automatic version and release identification
- ▲ Efficient storage management.
  - ◆ System stores the differences between versions rather than all the version code
- ▲ Change history recording
- ▲ Independent development
  - ◆ Parallel working on the same version is handled, either by exclusive locking (most tools) or by a so-called unreserved check out model (CVS).



# Content of an Archive



# Concurrent Versions System (CVS)

---



- ▲ Open Source
- ▲ [www.cvshome.org](http://www.cvshome.org)
- ▲ Local or C/S access
- ▲ Commandline based
- ▲ GUIs are available
  - ◆ WinCVS
  - ◆ jCVS
- ▲ May be integrated into other tools
  - ◆ e.g. Together CASE tool

# CVS Commands for Users

---

## ▲ Check out files

- `cvsv checkout filename`

## ▲ Check in files

- `cvsv commit filename`

## ▲ Clean up the working directory

- `cvsv release directoryname`

## ▲ Add files to repository

- `cvsv add filename`

## ▲ Remove files from repository

- `cvsv remove filename`



# CVS Commands for Users (cont'd)

---

## ▲ Show the differences between two versions

- `cv diff filename`

## ▲ Show the log of changes to a file

- `cv log filename`

## ▲ Show the history of each line of a file

- `cv annotate filename`

# CVS Commands for Administrators

---

## ▲ Create a CVS repository

- `cvs -d directoryname init`

# Exercise: CVS

---

- ▲ Copy the provided CVS repository to where you like to have it.
- ▲ Create your sandbox.
- ▲ Now check the whole repository out to your sandbox.
- ▲ Try some CVS commands
  - ◆ See the log of the files, change the files, check them back in, see the log of the files now, create a new file, add it to the repository, ...

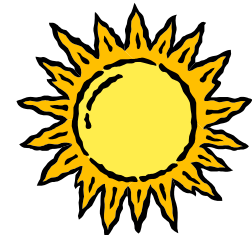
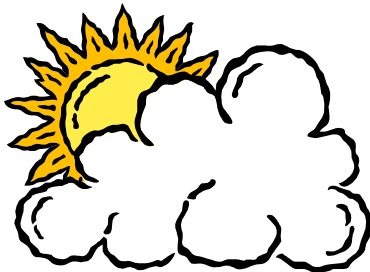
# Exercise: CASE Tool using CVS

---

- ▲ Take a class diagram you created with Together Control Center and put it under version control.
- ▲ Do some changes and check them in again.
- ▲ Try to get the former version back out of the repository.

# Change Management

---

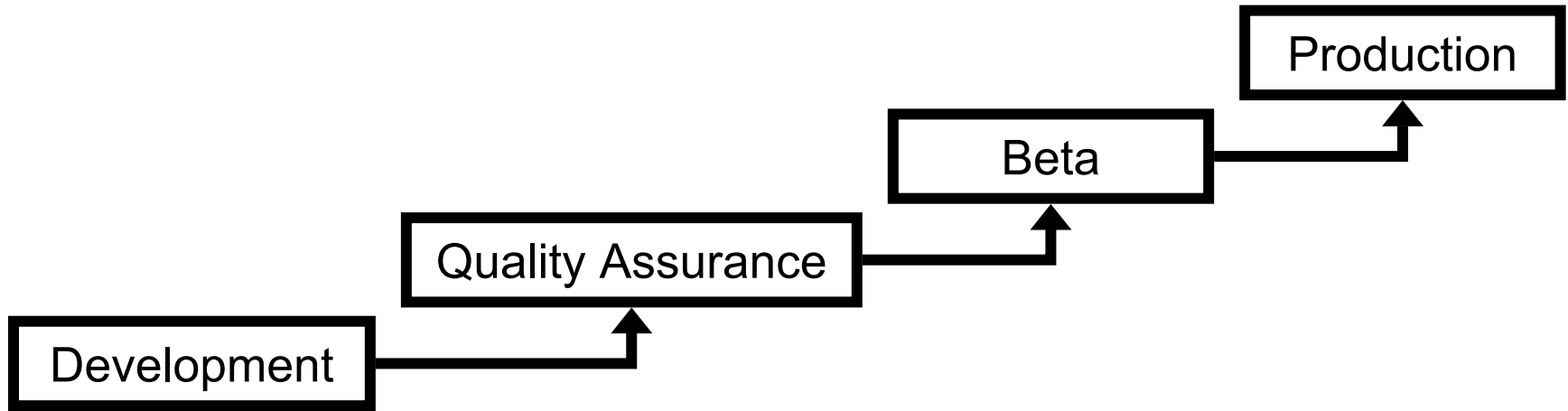


# Change Management

---

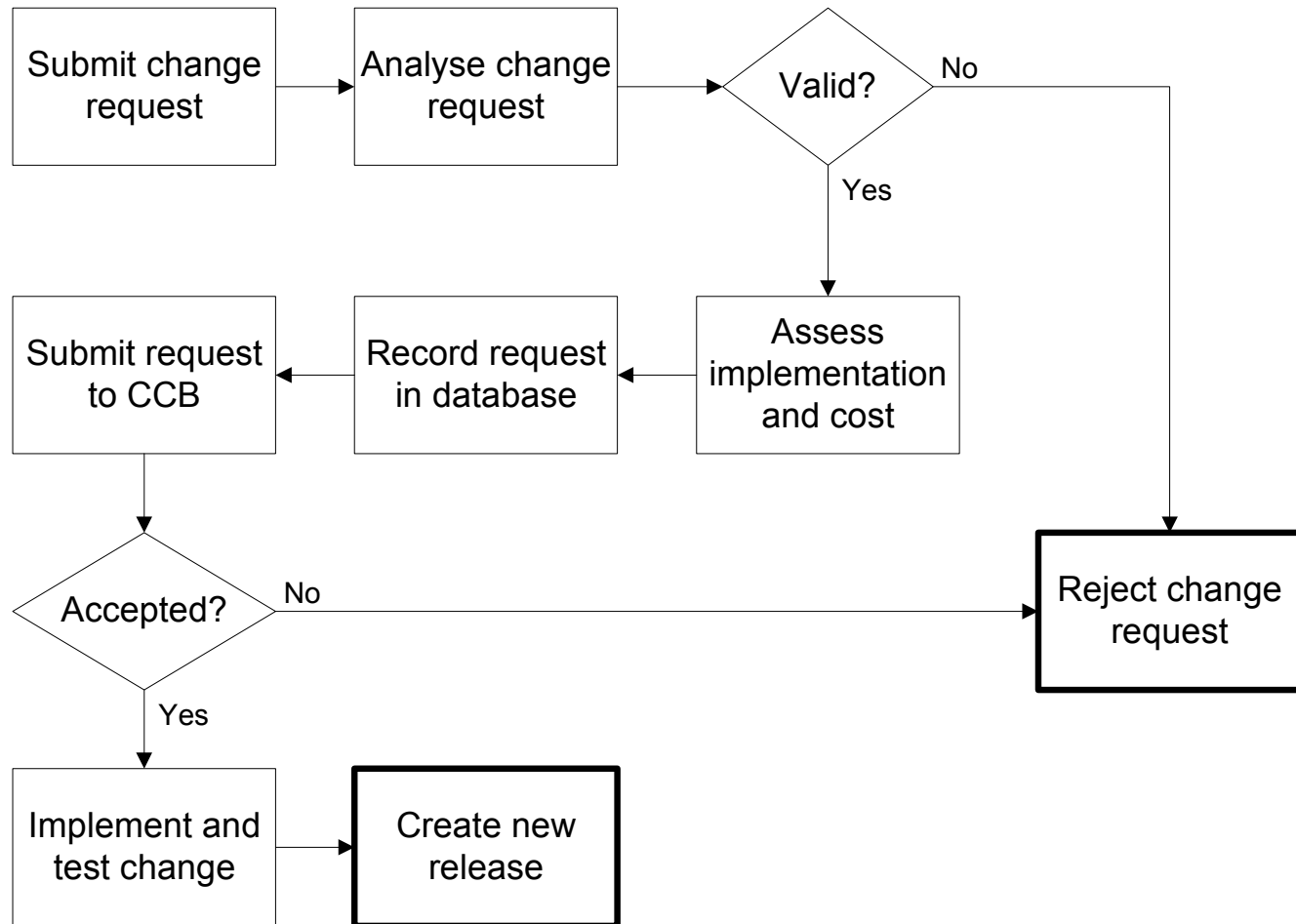
- ▲ Software systems are subject to continual change requests
  - ◆ From users
  - ◆ From developers
  - ◆ From market forces
- ▲ Change management is concerned with managing of these changes and ensuring that they are implemented in the most cost-effective way

# Promotion Model



- ▲ Each life cycle stage is working with a defined baseline.
- ▲ Changes must only be made in the Development stage.
- ▲ No Change management in the Development stage.
- ▲ Baselines are promoted to the next level after (formal) approval by an authorized body.

# Change Management Process





# Change Request Form

---

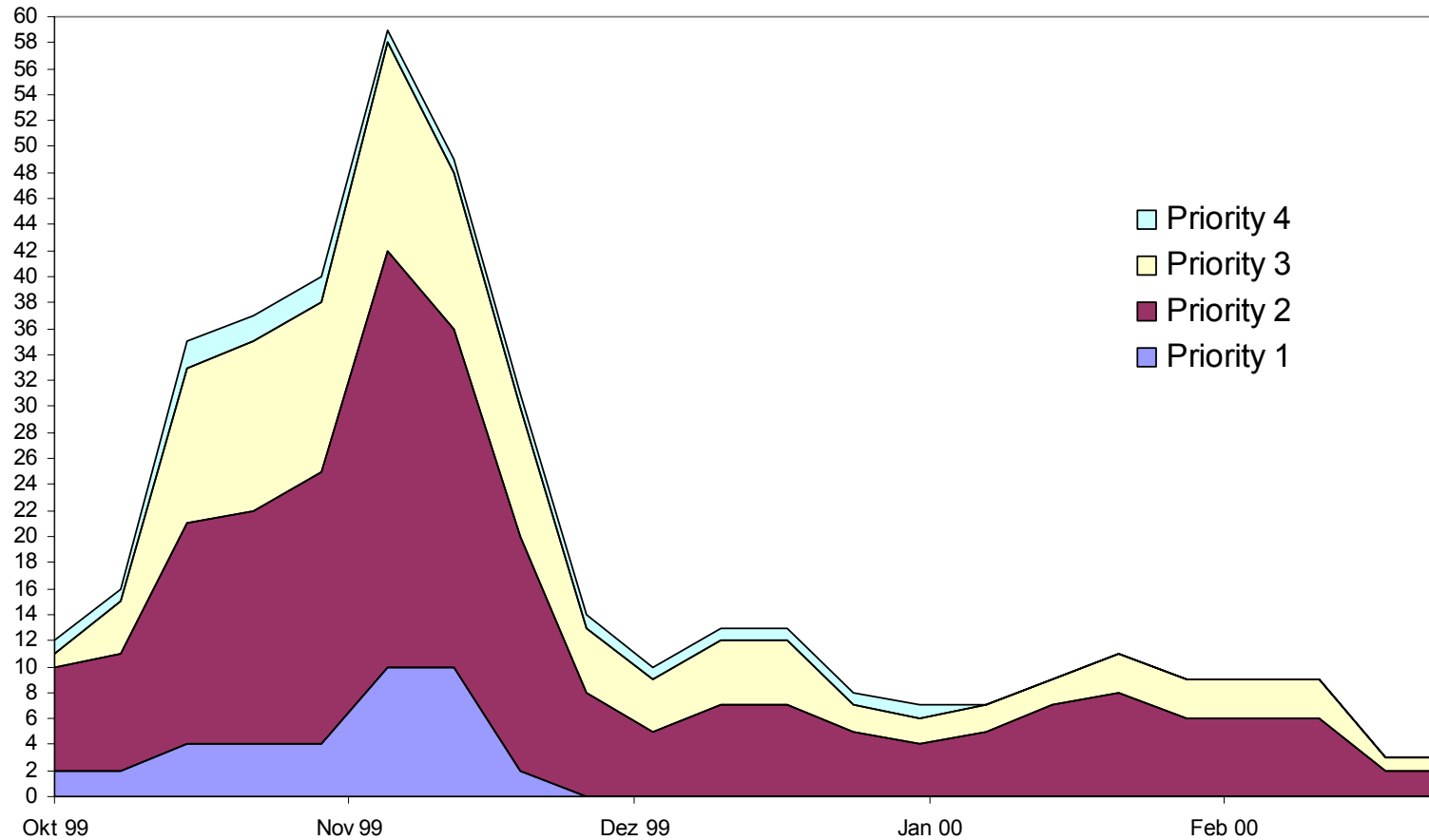
- ▲ Definition of a change request form is part of the CM planning process (may be paper or electronic)
- ▲ Records change required, requestor of change, reason why change was requested and urgency of change
- ▲ Records change evaluation, impact analysis, change cost and recommendations (system maintenance staff)

# Change Tracking Tool

---

- ▲ 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

# A sample problem statistics



# Configuration Control Board (CCB)

---

- ▲ Also called *Change Control Board*
- ▲ Changes should be reviewed by an external group who decide whether or not they are cost-effective from a strategic and organisational viewpoint rather than a technical viewpoint
- ▲ Should be independent of project responsible for system.
- ▲ May include representatives from client and contractor staff

# Revision 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

# 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

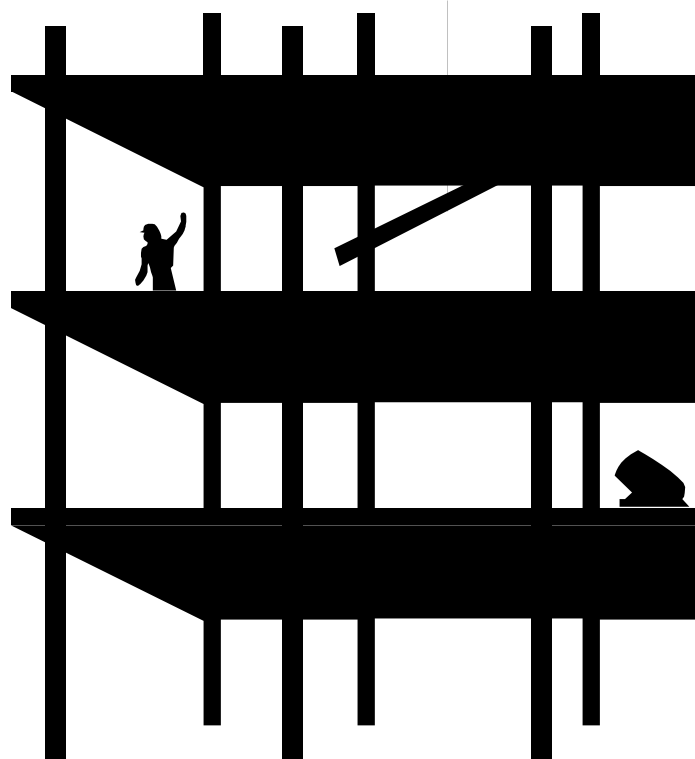
# Exercise: Change Management

---

- ▲ Design an entity-relational model of a configuration database which records information about system components versions, releases and changes
- ▲ Define attributes and queries that might be helpful for a CM database user

# Building a System

---





# System Building

---

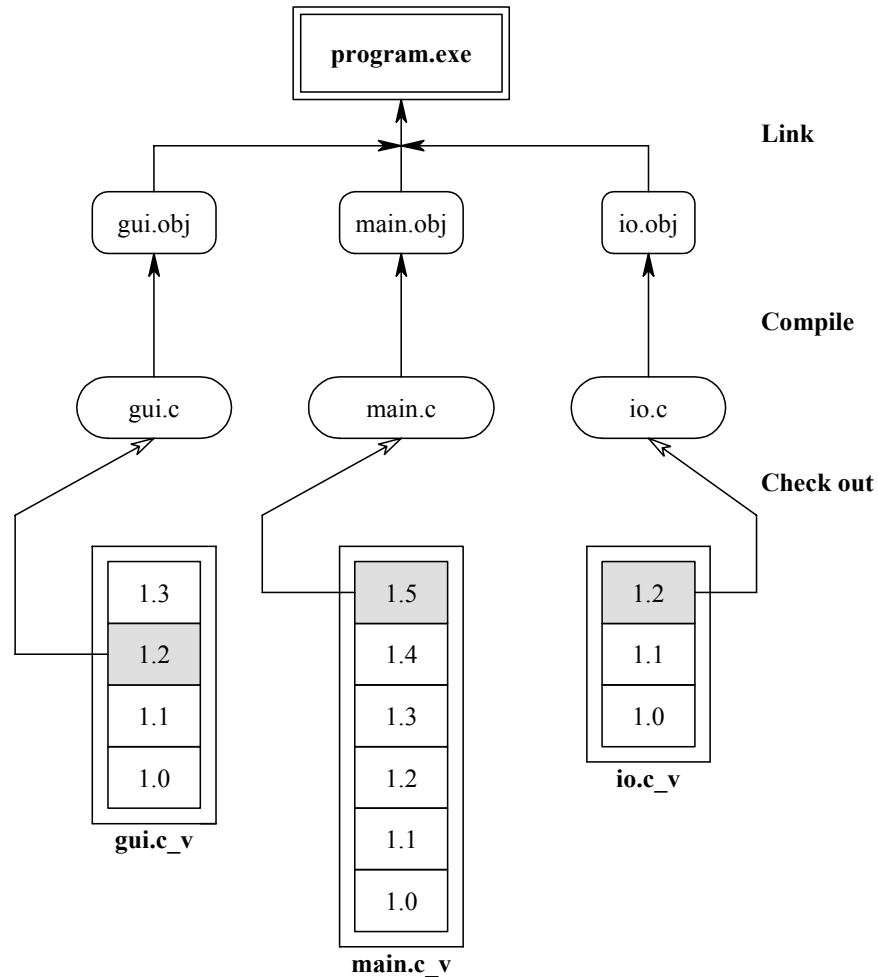
- ▲ The process of compiling and linking software components into an executable system
- ▲ Different systems are built from different combinations of components
- ▲ Different releases are built from different versions of components
- ▲ Invariably supported by automated tools that are driven by 'build scripts'

# 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
  - ◆ Distributed compilation
  - ◆ Derived object management
  - ◆ Integration with version management tools

# Integrated Build Process



# Daily 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

# System Building Problems

---

- ▲ Do the build instructions include all required components?
- ▲ Is the appropriate component version specified?
- ▲ Are all data files available?
- ▲ Is the system being built for the right platform?
- ▲ Is the right version of the compiler and other software tools specified?

# Tool Support

## ▲ Make

- ◆ Comes with many development environments or from GNU
- ◆ [www.gnu.org](http://www.gnu.org)

## ▲ Ant

- ◆ Open source tool from the apache project
- ◆ Based on XML
- ◆ Independent of operating systems
- ◆ [jakarta.apache.org/ant](http://jakarta.apache.org/ant)



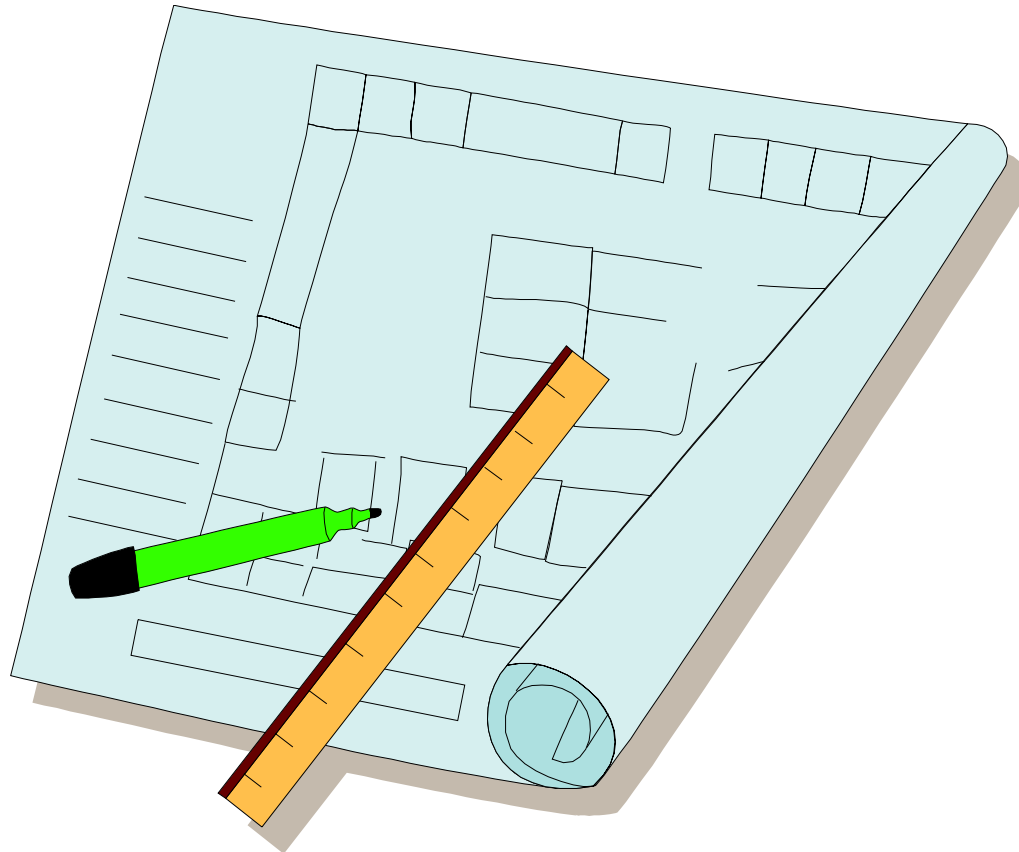
# Exercise: System Building

---

- ▲ Use a make tool (e.g. GNU make) and define a build script for a small development project.

# Configuration Management Planning

---





# 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
- ▲ IEEE standards are recommended
  - IEEE Standard for Software Configuration Management Plans, IEEE Std 828-1998
  - IEEE Guide to Software Configuration Management, IEEE Std 1042-1987

# Products of the Software Process

---

## ▲ Documents

- Specifications
- Manuals

## ▲ Designs

## ▲ Source code

## ▲ Test data

## ▲ Tools

➔ All may have to be managed

# Configuration Management Planning

---

- ▲ Starts during the early phases of the project
  - Recommendation: Plan before you start working
- ▲ Must define the documents or document classes which are to be managed (formal documents)
- ▲ The plan shall be applied to the type and size of the project

# What the CM Plan Defines...

---

- ▲ Items to be managed (Configuration items)
- ▲ Identification of items (naming scheme)
- ▲ Responsibilities for the CM procedures and creation of baselines
- ▲ Policies for change control and version mgmt
- ▲ CM records which must be retained
- ▲ Tools to be used and how to use them
- ▲ CM database to record configuration information

# What the CM Plan May Define...

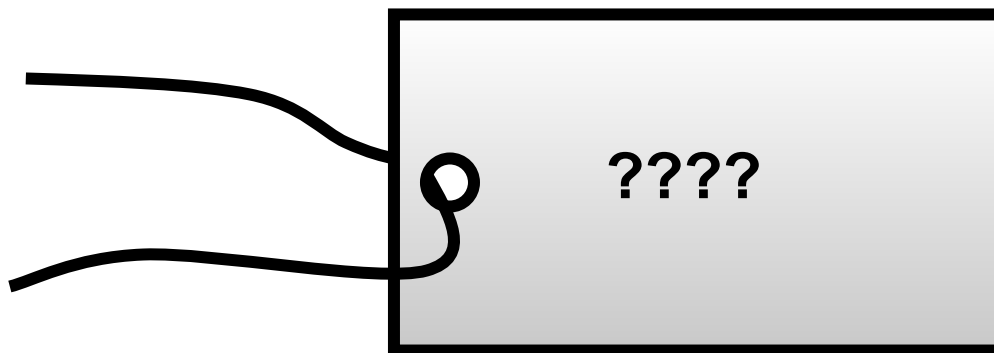
---

- ▲ Configuration status accounting
  - ◆ Configuration auditing
  - ◆ Release process
- ▲ Subcontractor / Supplier management
  - ◆ How to interact with subcontractors / suppliers

# Configuration Item Identification

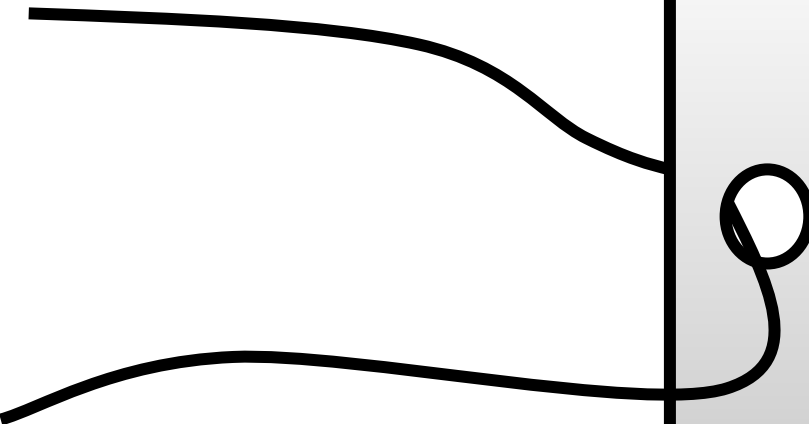
---

- ▲ Each configuration item must unambiguously be identified
- ▲ A naming scheme shall be defined



# Elements of a CI Identification

---

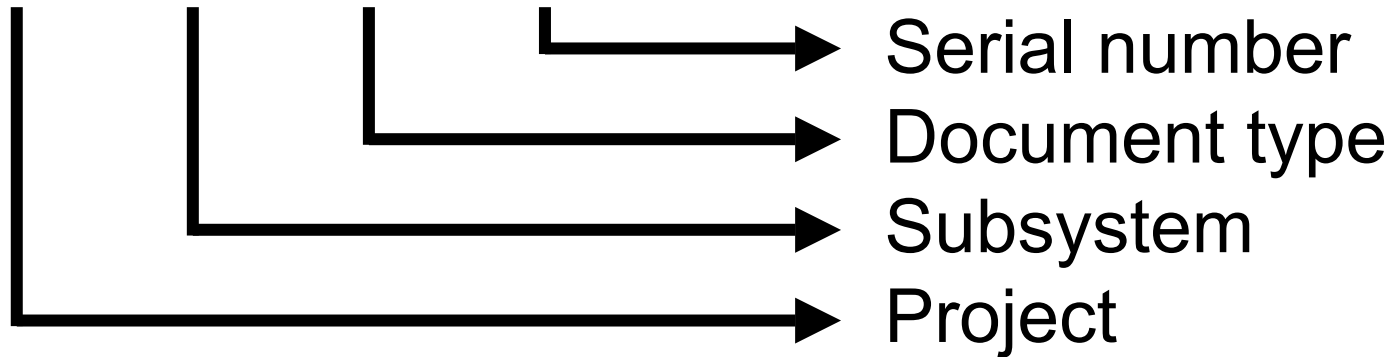
- 
- **Unique identifier**
  - **Version number**
  - **Descriptive name**
  - **(Check sum)**

# Hierarchical Naming Scheme

---

- ▲ A hierarchical scheme with multi-level names is probably the most flexible approach

**PPP.SSS.TTT.nnnn**



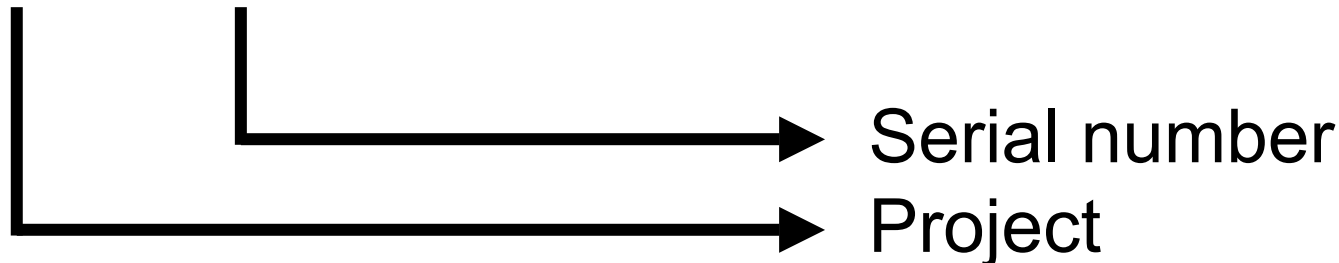


# Linear Naming Scheme

---

- ▲ A linear scheme may be used if the registration is decentralised

**PPP.nnnnnnn**



# Content according to IEEE Std 828

---

- ① **Introduction**  
*Describes the Plan's purpose, scope of application, key terms, and references*
- ② **SCM Management**  
*(Who?) Identifies the responsibilities and authorities for accomplishing the planned activities*
- ③ **SCM Activities**  
*(What?) Identifies all activities to be performed in applying to the project*
- ④ **SCM Schedules**  
*(When?) Identifies the required coordination of SCM activities with the other activities in the project*
- ⑤ **SCM Resources**  
*(How?) Identifies tools and physical and human resources required for execution of the Plan*
- ⑥ **SCM Plan Maintenance**  
*Identifies how the Plan will be kept current while in effect*

# Exercise: Create SCM Plan

---

You already may have worked on a student project. If not, consider a small experimental software project.

**Create a Software Configuration Management Plan (SCMP) for this project.**

# 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 (cont'd)

---

- ▲ 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

# Outlook

---

- ▲ What's missing in this module
  - ◆ Branching of versions
  - ◆ Merging of versions
  - ◆ Managing configurations that consist of firmware and hardware items, too