INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Summer Quarter, 2008
Michele Rousseau

Lecture Notes 6 – Configuration Management

(Some notes adapted from Sommerville 2000, Scott Miller, Susan E. Sim & UML Distilled)

Announcements

- **Assignment #2 has been posted**
  - TA will cover it in discussion
- **Read: Van Vliet Ch. 4 - CM & 10 – Modeling**
  - (if you haven't already)
  - Other info on UML that might be useful: http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/
  - Argo UML Info:

    http://argouml.tigris.org/
    - Other info on UML that might be useful:
      - http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/
  - Some books on UML:
    - Fowler (2004). UML Distilled: Third Edition: A Brief Guide to the Standard Object Modeling Language, Addison-Wesley, 2004
    - Larman (2005) .Applying UML and Patterns, Third Edition. Prentice Hall PTR, 2005
- **Quiz #1 – Regrades due today by the end of class**
  - Please have a cover sheet

Previously in INF 111/CSE121…

- **Equivalence Partitioning & Boundary Value Analysis**
- **Integration Testing**
  - Top-Down
  - Bottom Up

Today's Lecture

- **Configuration Management**
  - Version Control
- **Modeling**
  - OOAD
    - UML – Part 1

Configuration Management

- **Manages software artifacts**
- **Change happens → CM manages that change**
  - Change requests
  - Bugs fixed
  - Etc..
  - Different versions co-exist
  - What about – different configurations and versions of the system?

CM - Baseline

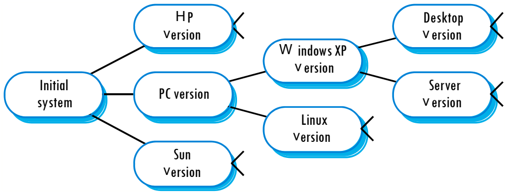
- **Start with a completed version of the system**

  Includes all Configuration items
  - All documentation
    - Requirements Specification
    - Design Document
    - Test Plan
    - Test Results
    - User Manual
  - Source code
  - Test Cases
  - Could include hardware
- **Thoroughly tested and completed**

CM – Different Versions

- **As change happens → new versions**
  - Different machines/OS
  - Offering different functionality
  - Tailored for particular user requirements.
- **CM Manages these changes**
  - CM is a team (sometimes assoc. w/ QA)
  - Controls
    - Costs
    - Effort
    - .. Maintains all changes & documents

System families



CM-Team

- **Creates Procedures for change**
- **Standards**

  Defines..
  - How items are identified
  - How changes are controlled
  - How new versions are managed
  - May be based on external standards (DOD, IEEE)

You need a CM Plan!

- **Define:**
  - Documents
    - What is to be managed (which docs)
    - Document naming scheme
  - Who is responsible for..
    - Procedures
    - Creation of Baselines
  - Polices for…
    - Change Control
    - Version Mgmt
  - Which CM records must be maintained

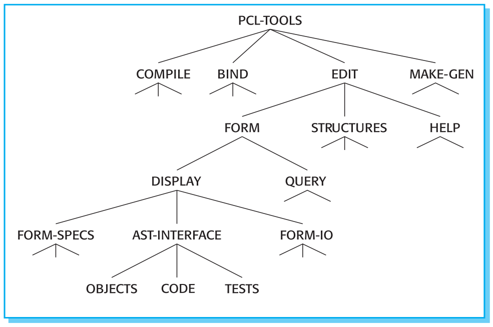
CM Plan (2)

- **Describes which tools to use**
  - Limitations
- **Defines the process of tool use**
- **Defines the CM database**
  - records configuration information.
- **May include information such as..**
  - the CM of external software
  - process auditing
  - etc…

Configuration item identification

- **Large projects → thousands of documents**
- **Documents follow the code (part of the configuration)**
- **Naming convention**
  - Each document needs a unique name
  - Related docs should 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



CM database implementation

- **Might be part of a SEE**
  - The CM database and documents → maintained on the same system
- **Might be integrated with other CASE tools**
- **Generally it is maintained separately**
  - Why? Cheaper and more flexible

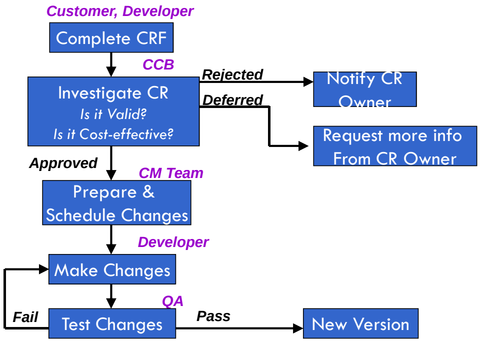
Software Changes Continually

- **Change requests:**
  - From users
  - From developers
  - From market forces
- **These changes need to be...**
  - Tracked
  - Managed
  - … cost-effectively!

The CM Process

- **Complete change request form (CRF)**
  - Formal document
- **Check if it is valid**
  - Is it really a fault or used incorrectly?
- **Cost-Assessment**
  - How much will this change cost?
  - Is it worth it?
- **If it is approved**
  - Make change
  - Test it
- **Create new version (when testing is complete)**

The Change Process



Customer, Developer

Complete CRF

CCB

Investigate CR
Is it Valid?
Is it Cost-effective?

Rejected → Notify CR Owner

Deferred → Request more info From CR Owner

Approved

CM Team

Prepare & Schedule Changes

Developer

Make Changes

QA

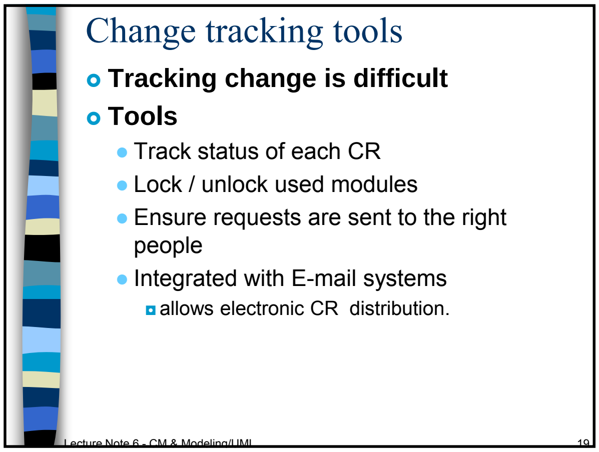
Test Changes

Fail

Pass → New Version

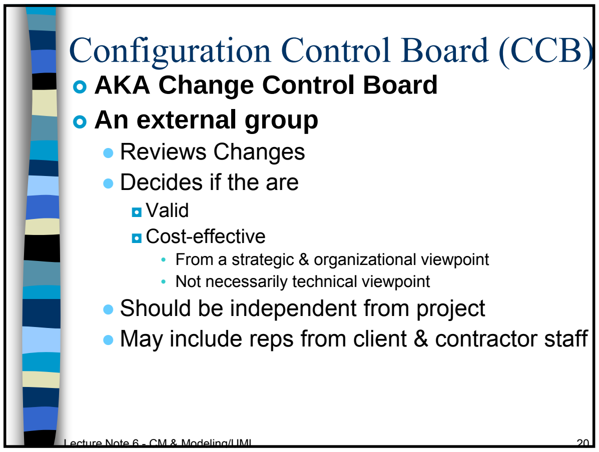
Change request form

- **Defined during CM Planning Process**
- **Records**
  - Change proposed
  - Who requested it
  - Why the change was suggested
  - Urgency of change
    - According to the requestor
- **It also records..**
  - Change evaluation
  - Impact analysis
  - Cost
  - Recommendations
    - to the System maintenance staff



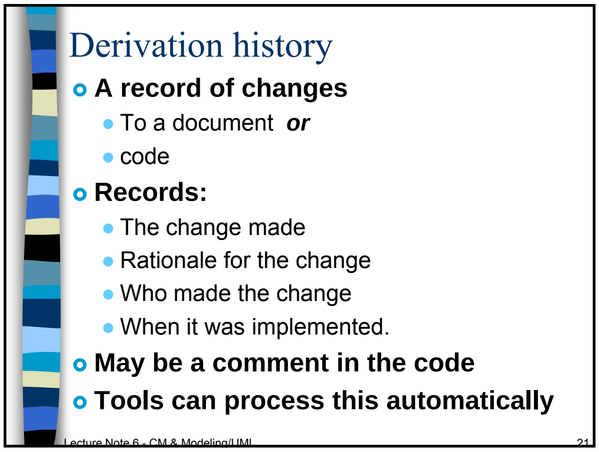
Change tracking tools

- **Tracking change is difficult**
- **Tools**
  - Track status of each CR
  - Lock / unlock used modules
  - Ensure requests are sent to the right people
  - Integrated with E-mail systems
    - allows electronic CR distribution.



Configuration Control Board (CCB)

- **AKA Change Control Board**
- **An external group**
  - Reviews Changes
  - Decides if the are
    - Valid
    - Cost-effective
      - From a strategic & organizational viewpoint
      - Not necessarily technical viewpoint
  - Should be independent from project
  - May include reps from client & contractor staff



Derivation history

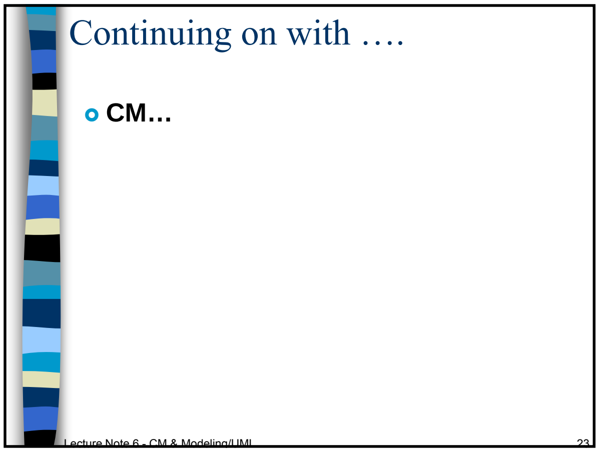
- **A record of changes**
  - To a document ***or***
  - code
- **Records:**
  - The change made
  - Rationale for the change
  - Who made the change
  - When it was implemented.
- **May be a comment in the code**
- **Tools can process this automatically**

Take a break!

- **Stretch, Relax**
- **Get some water, Use the restroom**
- **Make a phone call**
- **Enjoy some fresh air**
- **Chit chat**

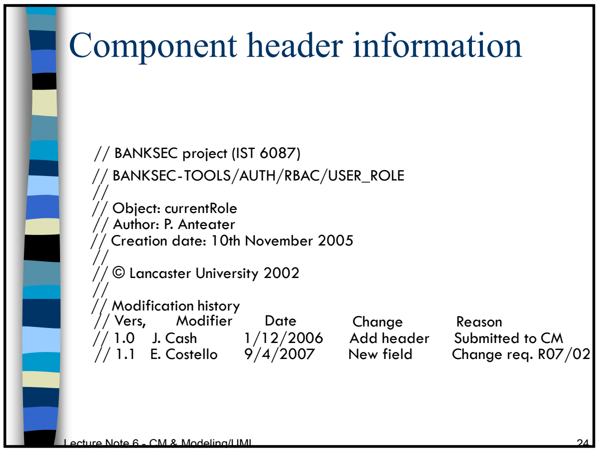
When we return…

- **CM continued**
- **Modeling**
  - OOAD
    - UML – Part 1



Continuing on with ….

- **CM…**



Component header information

```
// BANKSEC project (IST 6087)
// BANKSEC-TOOLS/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: P. Anteater
// Creation date: 10th November 2005
//
// © Lancaster University 2002
//
// Modification history
// Vers,     Modifier      Date        Change       Reason
// 1.0    J. Cash       1/12/2006    Add header    Submitted to CM
// 1.1    E. Costello   9/4/2007     New field     Change req. R07/02
```

Version and release management

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

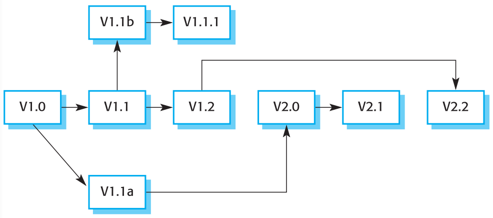
Version identification

- Versions should be identified in an unambiguous way
- 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.
- Derivation structure is a tree or a network
  - rather than a sequence
- CONS: Names are not meaningful
- A hierarchical naming scheme leads to fewer errors in version identification.

Version derivation structure



Attribute-based identification

- Use a combination of attributes to identify the version
  - Examples of attributes are Date, Creator, Programming Language, Customer, Status etc.
- More flexible than an explicit naming scheme
- Problem: it is difficult to keep the names unique
  - the set of attributes have to be chosen such that the versions can be uniquely identified.
- In practice, a version also needs an associated name for easy reference.

Attribute-based queries

- Pros: 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 + changes made
- Used for systems rather than components.
- Change set
  - describes changes made to implement the implementation
  - Then – change sets are applied in sequence
- in principle, a version of the system that incorporates an arbitrary set of changes may be created.

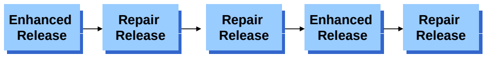
Release management

- Versions can stay internal → releases are external
- Releases must be…
  - Determined by Configuration Management Team
  - Must be Validated
  - Documentation must be updated
  - →This can be expensive

Serious faults can force a release

New Releases

- The more you change ➔ The more new faults introduced
  - System reliability may be impaired



Change Types

- **Corrective** – fix faults
- **Perfective** – improve non-functional behavior
- **Adaptive** – Change functionality
- Don't want to mix corrective with perfective or adaptive
  - Fix faults first!
  - Then change behavior
  - ➔ Too expensive to check if faults still apply

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.
- Should not assume that all previous releases have been accepted.

➔All files required for a release should be re-created