ICS 52: Introduction to Software Engineering

Fall Quarter 2001

Professor Richard N. Taylor

Lecture Notes: CM, Management, and Evolution

Many slides taken from Ian Sommerville’s text…

http://www.ics.uci.edu/~taylor/ics52_fq01/syllabus.html
A “Survival Fare” of Topics

◆ Configuration Management
◆ Maintenance and Evolution
◆ Project Management
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
System families

- Initial system
- DEC version
- Unix version
- PC version
- VMS version
- Workstation version
- Mainframe version
Configuration Hierarchy (for 1 family member)
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
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 derivation structure
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
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

Make-oids
Component dependencies

comp

scan.o  syn.o  sem.o  cgen.o

scan.c  syn.c  sem.c  cgen.c

defs.h
Types of maintenance

- Maintenance to repair software faults
  - Changing a system to correct deficiencies in the way meets its requirements

- Maintenance to adapt software to a different operating environment
  - Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation

- Maintenance to add to or modify the system’s functionality
  - Modifying the system to satisfy new requirements
Distribution of maintenance effort

- Fault repair (17%)
- Software adaptation (18%)
- Functionality addition or modification (65%)
Management of Software Engineering

- Planning
  - Objectives
  - Necessary resources
  - How to acquire resources
  - How to achieve goals
- Organizing
  - From small group structure to large organizations
- Staffing: the key resource in software development
- Directing
  - Ensure continuing understanding and buy-in
- Controlling
  - Measure performance and take corrective action when necessary
Project Control: Task-based

- Work Breakdown Structures
  - Hierarchical statement of the tasks to be performed
    » a subset of a statement of the process which will be followed
- “Off-line” management schemes
  - Gantt charts
    » Bar charts where length of bar proportional to the length of time planned for the activity
    » Can be used as a statement of schedule
    » Useful for analysis of resource deployment (e.g. maximum number of engineers needed at any one time)
  - PERT charts
    » A network of activities showing dependencies (precedence relationships
    » Exposes critical path
    » Shows maximal possible parallelism in project execution
Gantt Chart Example

Figure 8.2  Gantt chart for a simple compiler project.
PERT Chart Example

![PERT Chart](image)

**Figure 8.4** PERT chart for a simple compiler project. Activities on the critical path are shown in bold.

Source: Ghezzi, et.al., pg. 438