



# In-Class Presentations/Demos

- ✦ Next three class periods
  - ✦ March 1, 6, and 8
- ✦ Note room and time changes
  - ✦ See handout

*Come and learn about the other projects!*

# Philosophy

- ✦ Keep it simple!
- ✦ ~User-Level
- ✦ Pull diagrams from delivered documents
- ✦ Suggest view graphs (“foils”) made from photocopies of delivered documents
  - ✦ Keep prep time/effort to a minimum
  - ✦ Keep class presentation short & sweet

# Suggested Presentation Outline

- ☀ Intro (PROD/ADMIN) {2 minutes}
  - Intro team members and roles
  - Show web site
  - “Two-sentence” project description
- ☀ Technical (DIR) {5 minutes}
  - User-level requirements
  - High-level design
- ☀ Demonstrate prototype (QA) {5 minutes}
  - At least high-level with stubs
- ☀ Schedule Status (LIB) {2 minutes}

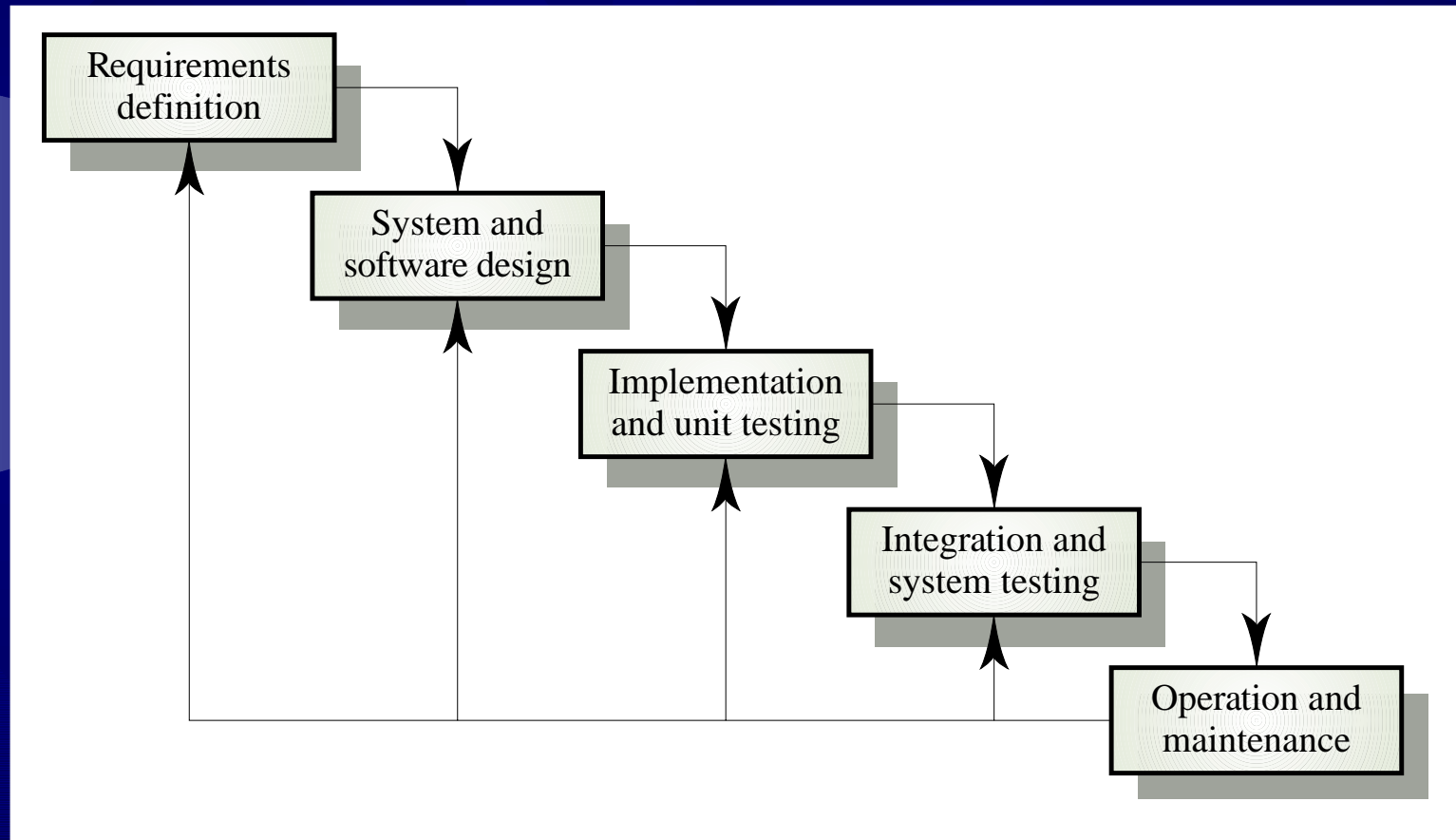
# Design Document Summary

- ✦ Overall fine—more variance
- ✦ Writing the document
  - ✦ Figure out “how” versus “what” (requirements)
    - Plan/design the actual system architecture
    - Sub-system and module boundaries, data structures, messages/member functions, etc.
- ✦ The document
  - ✦ Blueprint/plans for the actual coding
  - ✦ Starting point for Implementation Manual
- ✦ Requirements traceability matrix serves as a checklist for individual developers



# Configuration Management

# Review the Software Lifecycle





# NASA Jet Propulsion Labs, Software Management Plans

✦ (show/see actual documents)



# Configuration Management

“Configuration management (CM) is the development and application of standards and procedures for managing an evolving system product.”

[Sommerville, Software Engineering, 6th Ed.]

# Why?

- ★ New versions of software systems
  - For different machines/OS
  - Offering different functionality
  - Tailored for particular user requirements
- ★ Evolving software systems
  - System change is a team activity
  - CM aims to control the costs and effort involved in making changes to a system
- ★ **Quality assurance/control**

# When and How?

- ✦ Used throughout the process
  - ✦ Source code management
  - ✦ Build/delivery management
  - ✦ Document management
- ✦ Achieved with a combination of
  - ✦ **procedures** (forms, review boards)
  - ✦ **CASE tools**

# CM Standards

- ☀ CM procedures should be based on a set of standards applied within an organization
- ☀ Standards should define how items are identified, how changes are controlled and how new versions are managed
  - Change Request Forms
  - Change Control/Review Boards
- ☀ May be (based on) external
  - E.g., IEEE, ISO 9000, etc.



# Planning/Developing a Standard

☀ **All products** may have to be managed

- Requirements Specifications
- Design Documents
- Program modules
- Compiler/IDE/make configurations
- Test data and results
- User manuals

☀ **Thousands** of distinct documents are generated for large software systems

# CM Plan Will...

- ✦ Define **types** of documents to be managed and define a **naming scheme**
- ✦ Specify **who takes responsibility** for the CM procedures and creation of baselines
- ✦ Defines **policies for change control** and version management
- ✦ Defines the CM **documentation/records** that must be maintained

# CM Plan Will...

- ✦ Describe CASE **tools** to assist the CM process, and any limitations on their use
- ✦ Define the **process** of tool use
- ✦ Define the **CM database** used to record configuration information
- ✦ (May) include information such as the CM of external software, process auditing, compiler versions, etc.



# Change Management

- ✦ Software systems are subject to continual **change requests (CR)**
  - From users
  - From developers
  - From market forces
- ✦ Manage changes, ensure they are implemented in the most cost-effective and safe way

# Change Request Forms

- ✦ Definition of **change request** (CR) form is part of the CM planning process
- ✦ Initial information
  - Description of change required
  - Person/group suggesting change
  - Reason for change
  - Urgency of change
  - Related impacts
- ✦ Change board resolution/certification
- ✦ (see JPL example)

# Typical Process

Request change by completing a change request form

Analyze change request

**f** 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

# 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

# System Building

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

# 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 CD-ROM or as downloadable installation files from the web

# CASE Tools

- ✦ CM processes are **standardized** and involve applying **pre-defined procedures**
- ✦ **Large amounts of data** must be managed
- ✦ CASE tool support for CM is **essential**
- ✦ Mature CM tools are range from stand-alone tools to integrated workbenches

# Version Management

- ★ 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 (**user!**)
- ★ Independent development
  - One or more versions at a time may be checked out for change. (Parallel development.)



# Practical Examples

- ✦ RCS and CVS
- ✦ (see class “Notes” web page for today)