CIS 422/522
Software Methodologies
Final Overview

A. Hornof – Fall, 2000

The motivation for the class

- "Software is coming onto center stage as the key empowering technology of the Information Age. Without software, one cannot enter or exit the information superhighway. Without software, a Cray supercomputer is a rather expensive loveseat and a Powerbook laptop is a rather expensive paperweight. If their software is bungled, our cars, radiation therapy machines, bank accounts--and we ourselves--can get into big trouble."

Barry Boehm, in *Software Engineering*, edited by Dorfman and Thayer, 1997, IEEE Press.

The point of the class

- Learn how to build software systems that are adaptable, robust, reliable, and usable.
- Learn how to work effectively on a team.
- Learn and apply structured approaches for analyzing systems requirements, specifying software design, and managing the development process.

From the Group Dynamics lecture...

What is a Great Team?

- Diverse Skills
  - People skills, communication and writing skills, design skills, implementation skills and knowledge
- Coherence
  - Ability to build and maintain a shared vision
  - Shared expectations
- Mutual Respect and Responsibility
  - You don't *have* to like each other, but you *need* to trust and respect each other — and to earn your teammates trust and respect
  - Conflict can be healthy and productive, but it must be carefully managed.

Group Dynamics Question

- (22.5) Question asked earlier in the term: Explain why keeping all members of a group informed about progress and technical decisions in a project can improve group cohesiveness.
- Answer: There is more buy-in, better motivation, everyone sees the big picture, you can be convinced that others are working.
- New question: What are some specific examples of how communication helped in your projects?

Project Management Question

- What is a specific example, from one of the projects, of how you could see the benefits of project management? The management included assigning roles and responsibilities, devising a work breakdown with milestones, setting up monitoring and reporting, and analyzing risks.

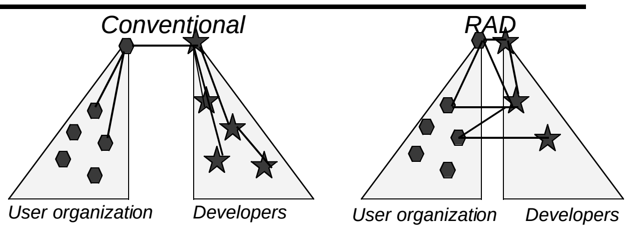
From the Software Processes lecture...

What are the activities involved?

- General activities are requirements engineering, design, implementation, testing, and evolution.
  - Requirements engineering develops a requirements specification
  - Design transforms the requirements specification into a clear design specification that can be handed to programmers
  - Implementation transforms the design specification into an executable program
  - Validation checks to make sure that the system meets its specifications and the users' needs
  - Evolution modifies the system after it is in use

From the Software Processes lecture...

RAD communication structure



- Peer-to-peer communication between users and developers
- Intense user involvement (and commitment) in negotiating requirements and testing prototypes
- Emphasis on rapid delivery and change, not on preserving information for a longer period. Hence, reduced paper documentation
  - Active, intense user participation among fixed personnel (including user representatives) reduces need for documents as orientation and communication.

From the lecture on

Requirements Engineering Processes

- The processes used for requirements engineering vary widely depending on the application domain, the people involved and the organisation developing the requirements
- However, there are a number of generic activities common to all processes
  - Requirements elicitation
  - Requirements analysis
  - Requirements validation
  - Requirements management

From the Requirements Engineering lecture...

ATM viewpoints

- Bank customers
- Representatives of other banks
- Hardware and software maintenance engineers
- Marketing department
- Bank managers and counter staff
- Database administrators and security staff
- Communications engineers
- Personnel department

From the lecture on

Software Requirements

- **Requirements** set out what the system should do and define constraints on its operation and implementation
- **User requirements** are high-level statements of what the system should do, from the user's perspective
- **System requirements** communicate the system services and constraints
  - **Functional requirements** set out the services the system should provide
  - **Non-functional requirements** constrain the system being developed or the development process
- A **software requirements document** or specification is an agreed statement of the system requirements

From the Software Requirements lecture...

Detailed user requirement

3.5.1 Adding nodes to a design

3.5.1.1 **The editor shall provide a facility for users to add nodes of a specified type to their design.**

3.5.1.2 The sequence of actions to add a node should be as follows:

1. The user should select the type of node to be added.
2. The user should move the cursor to the approximate node position in the diagram and indicate that the node symbol should be added at that point.
3. The user should then drag the node symbol to its final position.

Rationale: The user is the best person to decide where to position a node on the diagram. This approach gives the user direct control over node type selection and positioning.

Specification: ECLIPSE/WS/Tools/DE/FS. Section 3.5.1

From the Software Architecture lecture...

High-level activities in architectural design

- Design the high-level system structure.
  - Decompose the system into the principal subsystems, and identify the communication and data flow that would be necessary among the subsystems.
- Design the high-level control model.
  - Establish a model of the control relationships among the various subsystems.
- Further decompose each subsystem.
  - Perform modular decomposition, and design the architecture of each subsystem.

From the OO System Development lecture...

An object-oriented design process

- Figure out the use of the system
- Design the system architecture
- Identify the principal system objects
- Develop interaction and state-transition models
- Specify object interfaces

- (but not necessarily in that order)

From the OO System Development lecture...

Start with the problem statement

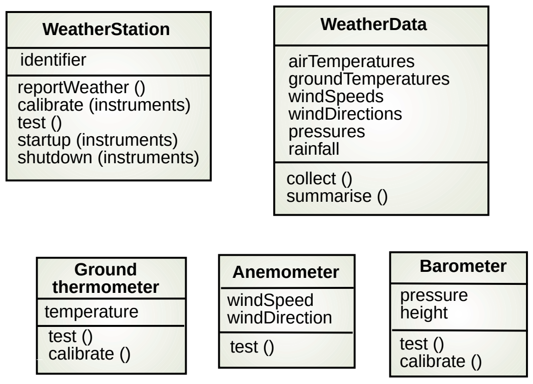
Weather Station

A weather station is a package of software controlled instruments which collects data, performs some data processing and transmits this data for further processing. The instruments include air and ground thermometers, an anemometer, a wind vane, a barometer and a rain gauge. Data is collected every five minutes.

When a command is issued to transmit the weather data, the weather station processes and summarises the collected data. The summarised data is transmitted to the mapping computer when a request is received.

From the OO System Development lecture...

Design the object classes



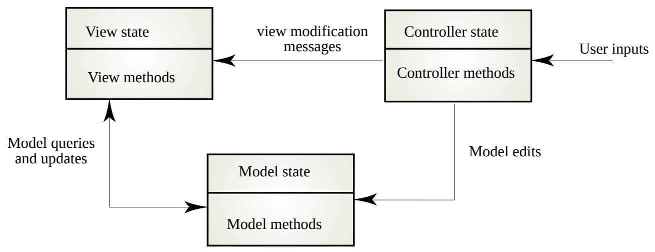
From the User Interface Design lecture...

User-system interaction

- The question that must be answered before the UI design can start: What is the user's task?
- Two problems must be addressed in interactive systems design
  - How should information from the user be provided to the computer system?
  - How should information from the computer system be presented to the user?
- These problems persist regardless of the interface style.

From the User Interface Design lecture...

Model-view-controller

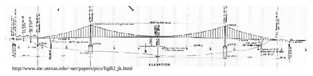


- MVC design cleanly accomodates different interfaces to the same data
- View and controller are often combined into a single UI delegate

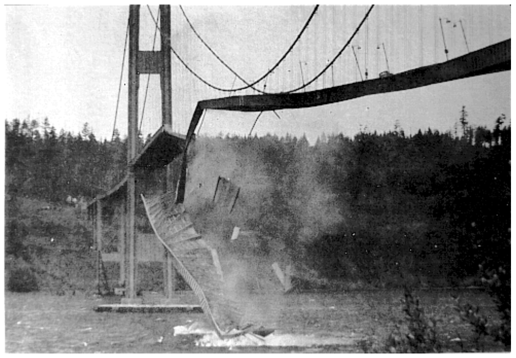
The point of the class

- Learn how to build software systems that are adaptable, robust, reliable, and usable.
- Learn how to work effectively on a team.
- Learn and apply structured approaches for analyzing systems requirements, specifying software design, and managing the development process.

Don’t Build This

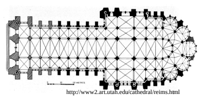
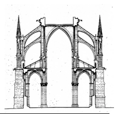


http://www.me.utexas.edu/~uer/papers/pics/figB2_jk.html

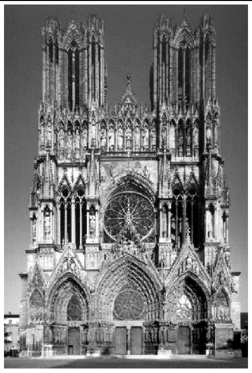


The Tacoma Narrows Bridge collapsed in 1940 because it was poorly designed. (UPI Photo)

Build This

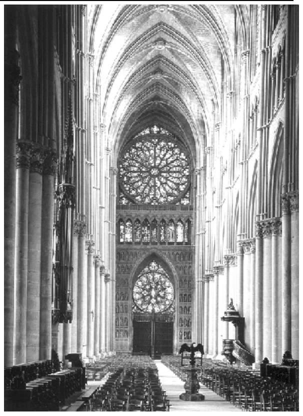



http://www2.art.utah.edu/cathedral/reims.html



http://vrc.ucr.edu/tutorials/gothic/reims/fi/00000000.htm

Reims Cathedral (13th century)



http://vrc.ucr.edu/tutorials/gothic/reims/reims.html