15-413

15-413 Software Engineering Introduction

Bernd Bruegge
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213

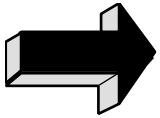
25 August 1998

Software Engineering

- Software systems are complex
  - **Impossible to understand by a single person**
  - **Many projects are never finished: "vaporware"**
  - **The problem is arbitrary complexity**
- 1968 Definition:
  - **Software Engineering means the construction of *quality software* with a *limited budget* and a *given deadline***
- Our definition:
  - **Software Engineering means the construction of quality software with a limited budget and a given deadline in the context of constant *change***
- Emphasis is on both, on software and on engineering

15-413 Software Engineering at CMU

- A Single Semester Course
  - **Lectures: Theoretical foundations and background**
  - **Project: Learn how to apply them in practice**
  - **Lectures and Project work are interleaved**



A Single Project Course

- **Everybody is working on the same project**

- Cheating Rule for 15-413
  - **You cheat if you do not acknowledge the contribution made by others.**

Outline of Today's Class

- Introduction
  - **Objectives of Course**
- Project
  - **PAID System**
  - **Top Level Design**
- Syllabus
  - **Introduction of People**
  - **Administrative Matters**
  - **Course Schedule**
- What is Software Engineering?
  - **Problem Solving using Decomposition, Abstraction, Hierarchy, Modeling**

Objectives of this course

- Acquire technical knowledge
  - **Understand difference between program and software product**
  - **Be able to reconstruct the analysis and design of an existing software system**
  - **Be able to design and implement a subsystem that will be part of a larger system**
- Acquire managerial knowledge
  - **produce a high quality software system within budget & time**
  - **while dealing with complexity and change**

Emphasis is on team-work

- Participate in collaborative design
- Work as a member of a project team, assuming various roles
- Create and follow a project and test plan
- Create the full range of documents associated with a software product
- Complete a project on time

How can we accomplish this?

- Course Project
  - **PAID: Platform for Active Information Dissemination**
- The 4 R's:
  - **Real Problem: Provide constantly changing information to 6000 dealers in 118 countries in 18 languages**
  - **Real Client: Helmuth Ritzer, Daimler Benz Corporation**
  - **Real Data: Databases provided by Daimler Benz**
  - **Real Deadline: 10. December 1998**

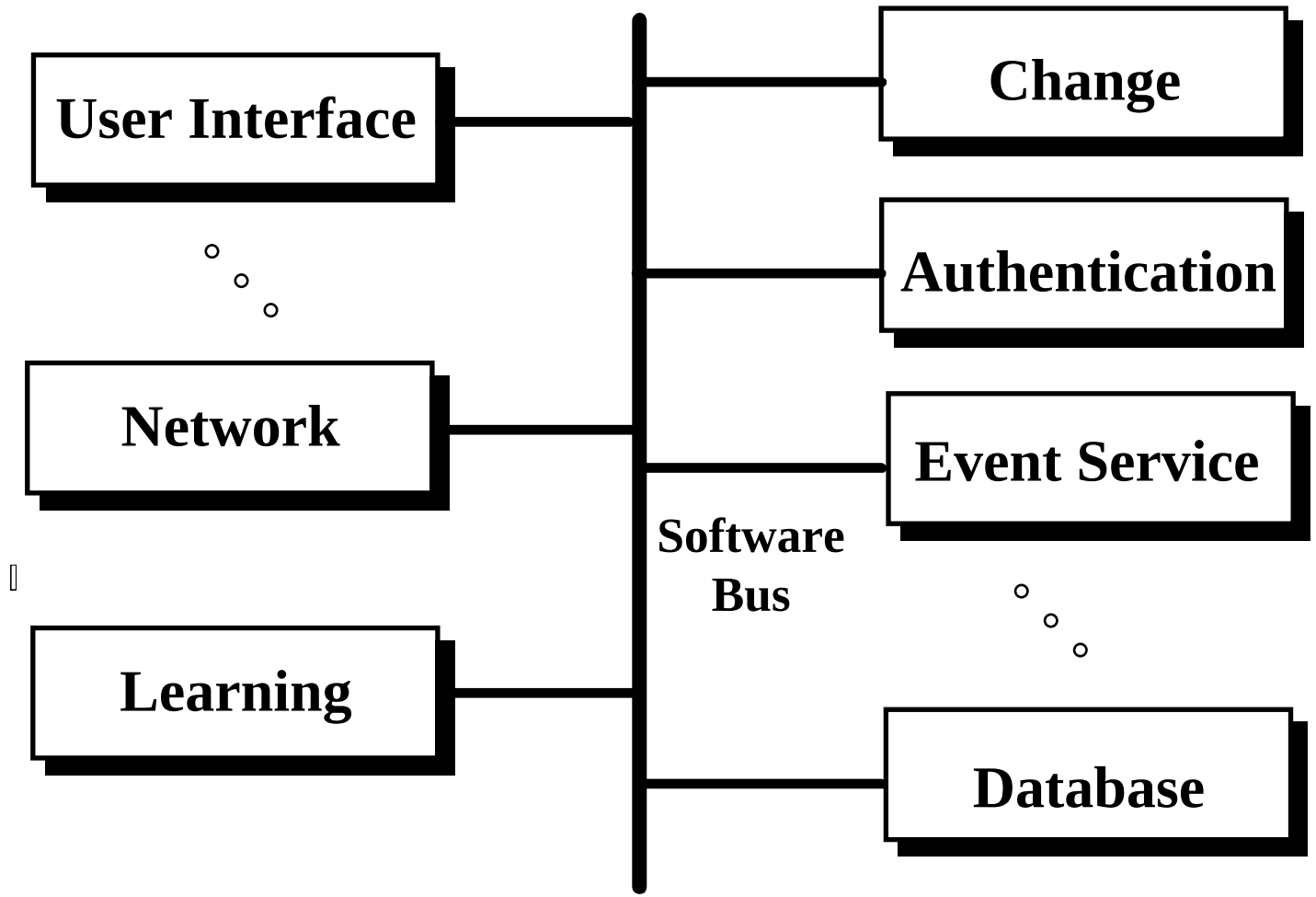
Assumptions and Requirements for this Class

- Assumption:
  - **You are proficient in a programming language (Java preferred), but have no experience in analysis or design of a system**
  - **You have access to a Web Browser**
    - **Course Homepage: http://sierra.se.cs.cmu.edu/PAID/default.html**
- Requirements:
  - **You have taken one of the required courses (Compiler Construction, Operating Systems or Artificial Intelligence)**
- or
  - **You have practical experience with maintaining or developing a large software system**

Project Goals

- Creation of a software architecture for intelligent dissemination of constantly changing information using the web
- Demonstration of a conceptual prototype to a worldwide audience

Basic Software Architecture for PAID Project



PAID Subsystems

- ***User Interface:*** Provides user interface for all the computers used in the PAID system (Palmpilot, PC, ...)
- ***Network subsystem:*** Provides an adaptive, selective multicast mechanism for sending information to subscribers
- ***Learning subsystem:*** Monitors the behavior of the user accessing the net and provides this information to the network subsystem

PAID Subsystems ctd

- ***Authentication subsystem:*** Provides smart-card based access to the PAID system for different types of users
- ***Change subsystem:*** Detects change in the databases
- ***Database subsystem:*** Provides replication and caching of information and access to Daimler Benz databases
- ***Event Service:*** Provides an subscriber/publisher model and event notification protocol

Project Management

- Coaches:
  - **Bernd Bruegge(Authentication)**
  - **Elizabeth Bigelow (Architecture)**
  - **Elaine Hyder and Jack Moffett (User Interface)**
  - **Robin Loh (Network)**
  - **Eric Stein (Learning)**
  - **Keith Arner (Event Service)**
  - **Swati Gupa (Database)**
  - **Joyce Johnstone (Documentation)**
  - **? (Change)**
- Infrastructure
  - **Eric Stein (Lotus Notes)**
  - **Joyce Johnstone (Web)**
  - **Guenter Teubner (Tutorials)**

Subsystems and Teams

- PAID will be developed in a team-based approach
- Each subsystem in the software architecture will be mapped on a team
- You will be member of one or more teams
  - **Development teams**
  - **Crossfunctional teams (Architecture, Documentation)**
- You can give us your team preferences after the client has presented the problem statement on Thursday August 27.
  - **Deadline: Friday 12 noon.**
- Team selection is done by project management and will be announced on Tuesday, September 1.

Problem Statement

- Customer Presentation:
  - **Thursday, 27 August, 9:00-10:20 AM**
- Online Version:
  - **http://sierra.se.cs.cmu.edu/PAID/PS.html**
  - **Available: Wednesday, 26 August, 10:00 AM**

Electronic Communication

- Web Page: http://sierra.se.cs.cmu.edu/PAID/default.html
- Course bboards:
  - ***Announce:*** **For course announcements**
  - ***Discuss:*** **For discussion of topics relevant for everyone**
  - ***Help*****: "24 hour help desk"**
  - ***Client:*** **Communication with the client**
- Team Bboards (after the teams are announced):
  - **Discussion of issues relating to the subsystem developed by the team.**
- Daily access to these bboards is required.
- Access to bboards is restricted to registered students and students on the waiting list:
  - **User Name: Firstname Lastname**
  - **Password: last 4 digits of social security number**

Project Milestones

- External Milestones
  - **Aug 27: Client Presentation**
  - **Nov 5: System Design Review**
  - **Nov 24: Implementation Review**
  - **Dec 10: Client Acceptance Test**

- Internal Milestones
  - **Sep 1: Announcement of Teams**
  - **Oct 22 & 27: Analysis Review**
  - **Dec 8: Dry run of Client Acceptance Test**

Client Acceptance Milestone

- The PAID system must be successfully demonstrated on Dec 10, 1996 (“15-413 Final”)
- The acceptance criteria are established in a dialog with the Daimler Benz client during the requirements analysis phase
- The PAID system will be delivered with the following artifacts on a CD-ROM
  - **Requirements Analysis Document**
  - **System Design Document**
  - **Object Design Document**
  - **Test Manual**
  - **Source Code Depot**

Administrative matters: Textbooks

- Textbook (Required)
  - **Bernd Bruegge and Allen Dutoit, Model-based Software Engineering: A Project-oriented Approach, Prentice Hall.**
  - **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns, Addison-Wesley, 1996, ISBN 0-201-63361-2**
- Other Recommended Readings
  - **Ivar Jacobson, M. Christerson, P. Jonsson, G. Övergaard, "Object-Oriented Software Engineering" , Addison Wesley, 1992**
  - **Grady Booch, "Object-Oriented Design with Applications", Benjamin Cummings, 1991.**
  - **James Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice Hall, 1991**

Readings

- Additional readings in syllabus
- Check Readings on the 15-413 home page
- Readings due on day of class
  - **Readings not in the textbooks will be made available a week before the lecture**
- Reading for Thursday: Problem Statement
  - **Available on the 15-413 Home page by Wednesday 10am.**

Grading

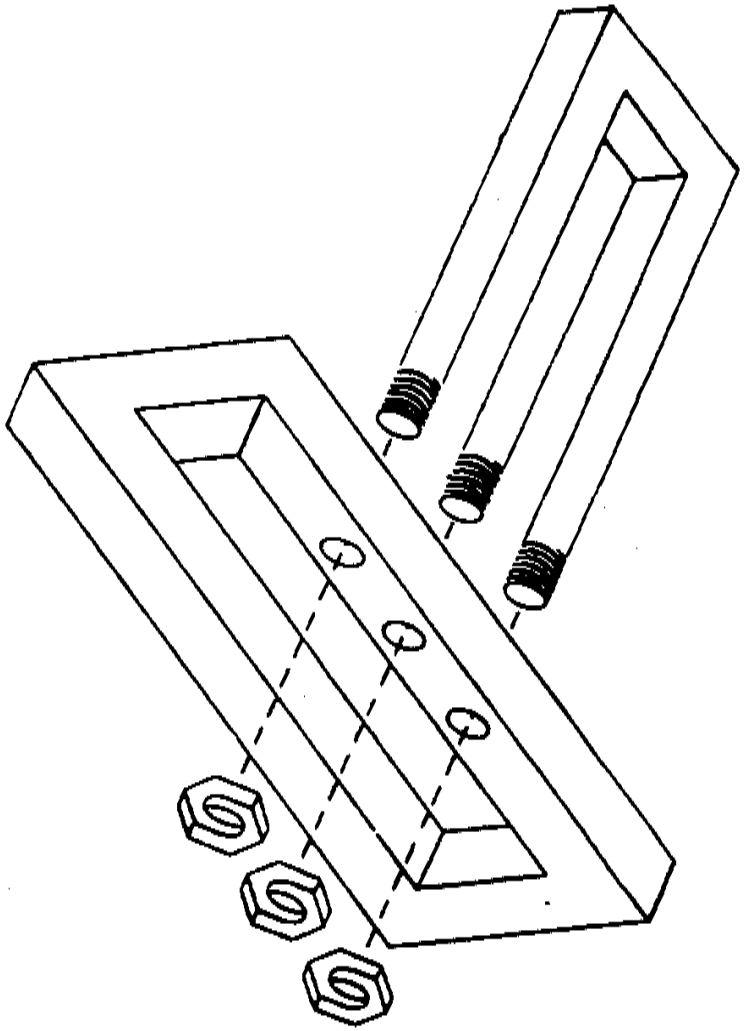
- Project
  - **Process and associated deliverables: 35 points**
  - **Communication: 10 points**
  - **System integration and system delivery: 20 points**
- Lectures
  - **4 homeworks : 5 points for each of 4 homeworks**
  - **Quizzes: 15 points**

- **Standards**
  - **A: 90+**
  - **B: 75-89+**
  - **C: 56-74 (including at least 20 points from lectures and 40 points from project)**
  - **D: 40-55, or 56-74 with wrong proportion of lecture and project points**
  - **R: less than 40**

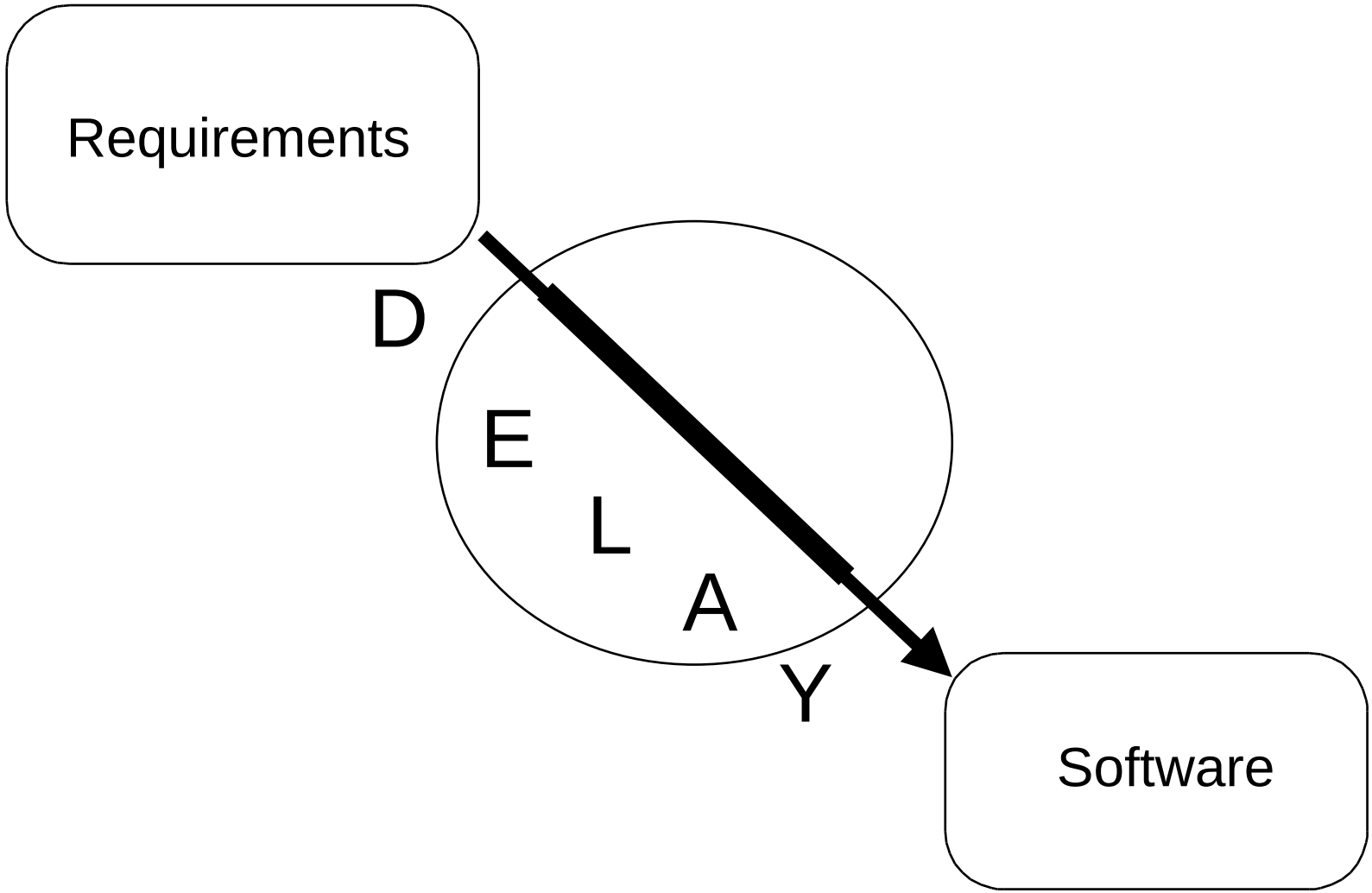
Tools to be used in the Course

- Java-based CASE tool for UML: Together-J
- Electronic communication: Domino (Web-based version of Lotus Notes)
- Java Development Environment
- User interface Development:
  - **Director**
  - **Palmpilot Development Kit**
- Wordprocessing:
  - **Any editor capable of producing HTML**
- Presentations:
  - **Powerpoint 4.0**
  - **HTML**
- Configuration management:
  - **CVS**
- Document distribution:
  - **Web and CD-ROM**
- CD-R Authoring software

Can you develop this?



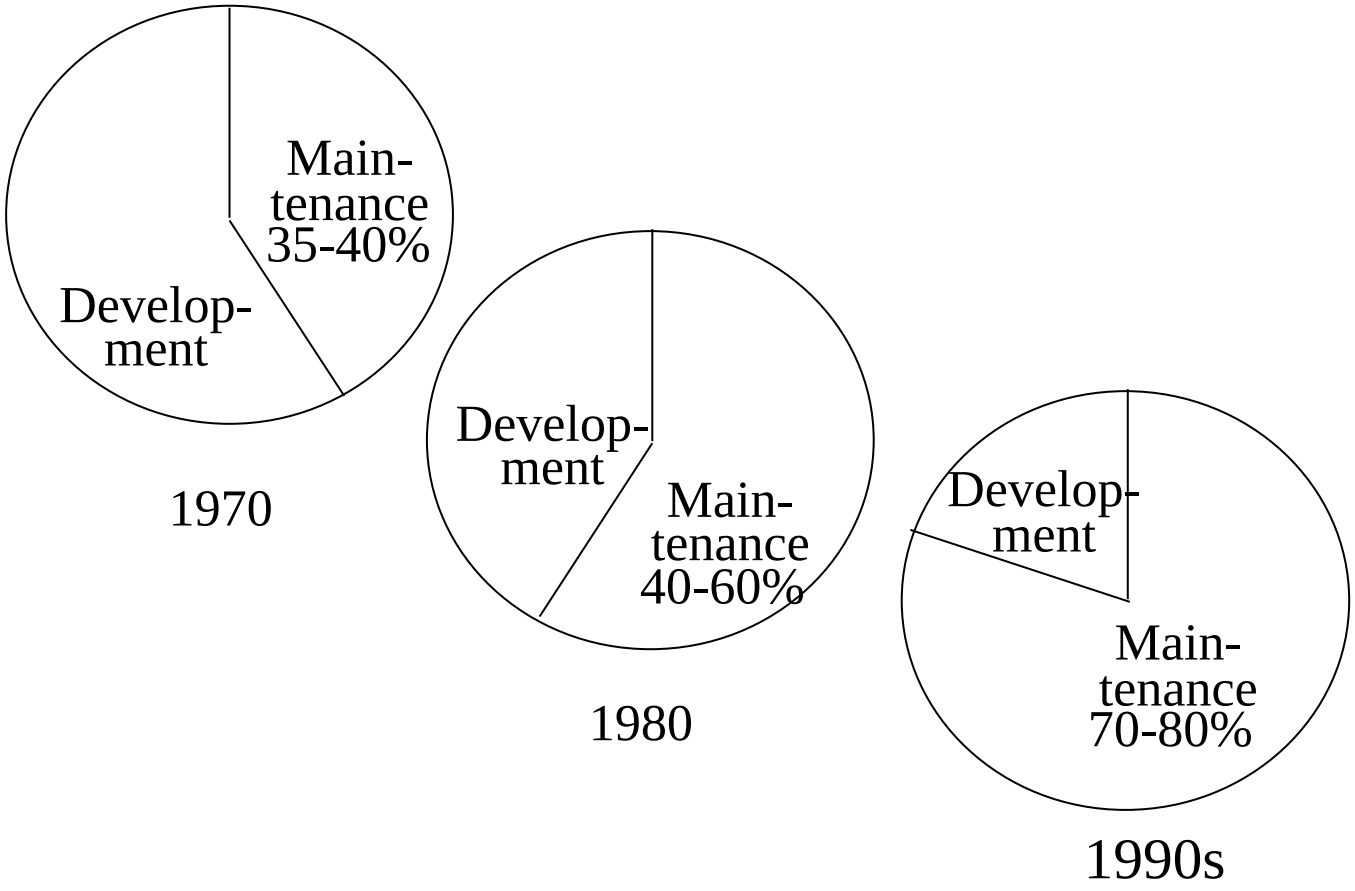
Limititations of Non-engineered Software



Today's Software

- Quality of today's software:
  - **The average software product released on the market is not errorfree.**
- Software maintenance
  - **Now represents over 70% of the cost**

Software Maintenance Cost 8/25/98



Software Engineering: A Problem Solving Activity

- ***Analysis:***
  - **Understand the nature of the problem (Process of breaking a problem into pieces)**
- ***Synthesis:***
  - **Putting the pieces together into a large structure**
- To solve problems we use:
  - ***Technique (Method):*** **Formal procedure for producing results using some well-defined notation**
  - ***Methodology:*** **Collection of techniques applied across the software development lifecycle and unified by some general philosphical approach**
  - ***Tool:*** **Instrument to accomplish a technique**

Software Engineering: Definition

- Techniques, Methodologies and Tools that help with the production of
  - ***high quality software* system**
  - **with a given *budget***
  - **before a given *deadline***
  - **in the context of *change*.**

Engineering a System

- Produce a quality system at a low cost in time while everything is changing
- What does this mean?
  - **Different people have a different idea of quality**
  - **Should I consider development cost only? Or maintenance cost as well?**
  - **How can I develop within time when requirements are changing 5 minutes before delivery**
  - **How do I manage complexity? Change?**
- Software Engineering is not a science but still an art
  - **Key to success: Good communication between all the people involved in the development and in the use of the system**

Producing Software: Our Track Record

- Early 1980: Automated income tax processing system (Sperry Corporation)
- ***Quality:*** System proved inadequate to the workload
- ***Cost:*** Nearly twice as much as expected
  - **The IRS needed an extra 90 Million Dolar for enhancing the 103 Million Dollar of Sperry Equipment.**
- ***Time:*** Because of the problems experienced, IRS missed a deadline and was forced to pay
  - **40.2 Million US Dollars in interest**
  - **22.3 million US Dollars in overtime wages**

Space Shuttle Software

- **Cost:** $10 Billion, millions of $$ more than planned
- **Time:** 3 years late
- **Quality:** First launch of Columbia was cancelled because of synchronization problem with the Shuttle's 5 onboard computers.
  - **Error was traced back to a change made 2 years earlier when a programmer changed a delay factor in an interrupt handler from 50 to 80 milliseconds.**
  - **The likelihood of the error was small enough, that the error caused no harm during thousands of hours of testing.**
- Substantial errors still exist. Astronauts are still supplied with a book of known software problems "Program Notes and Waivers".

Why are Software Systems so Complex?

- The problem domain is difficult. In general you are not expected to be an expert in the domain.
- Software offers extreme flexibility
- Software systems are discrete systems
  - **Continous systems have no hidden surprises**
  - **Discrete systems have!**
- The development process is very difficult to manage in the context of change

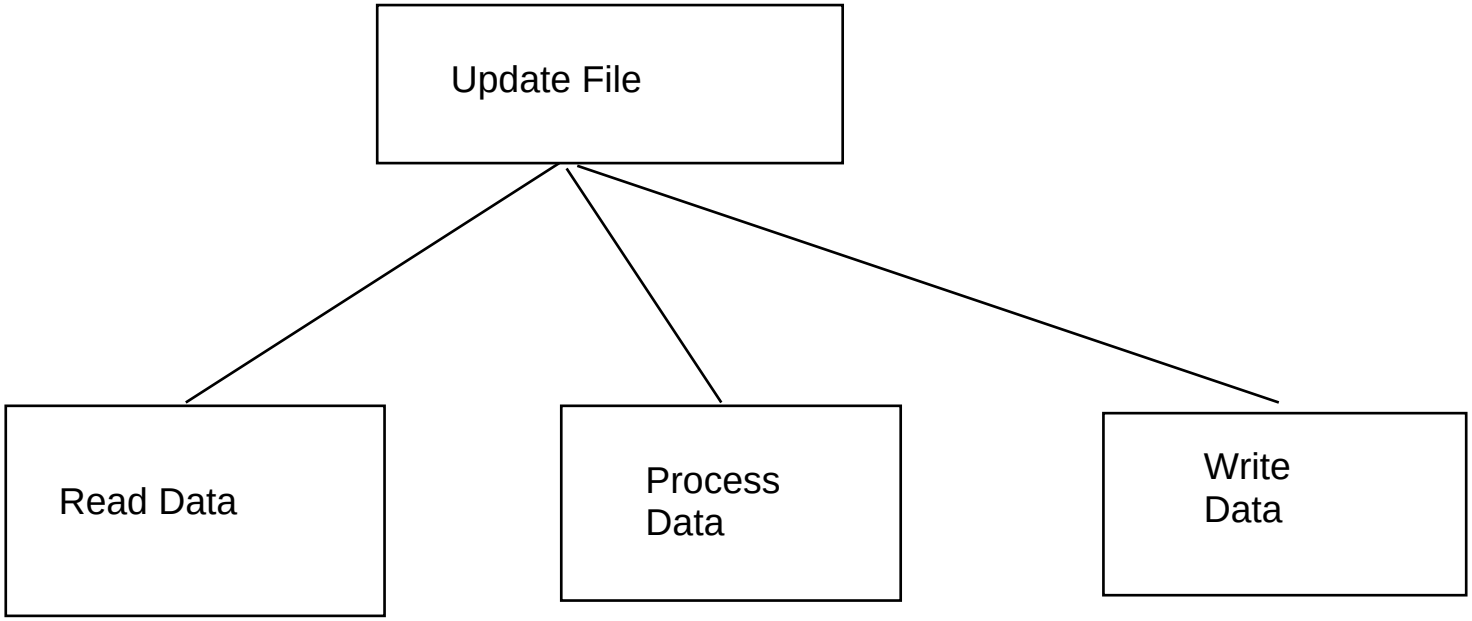
Ways to deal with Complexity

- Decomposition
- Abstraction
- Hierarchy
- Modeling

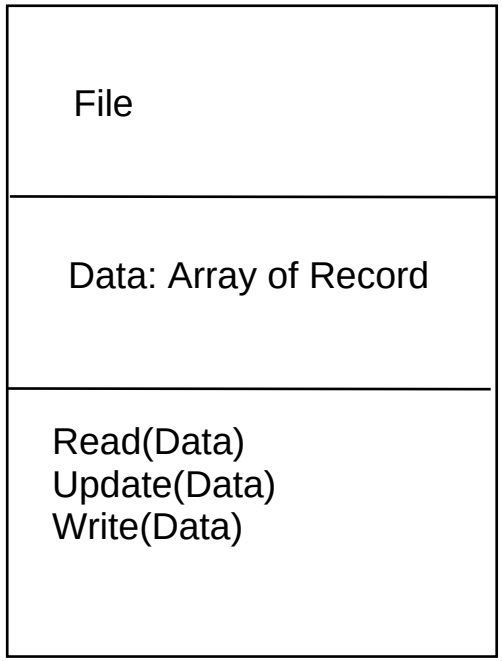
Decomposition

- Also called modularization
- Decomposition is best technique to master complexity: divide and conquer
- Functional decomposition
  - **Each module in the system is a major processing step (function)**
  - **Example of functional decomposition**
- Object-oriented decomposition
  - **Decompose the system according to key abstractions (classes) in the problem domain**
  - **Example of object-oriented decomposition**
- Which decomposition is the right one?

Example of Functional Decomposition



Example of Object-oriented Decomposition



Abstraction

- Some of our limitations to deal with complexity are due to our short term memory
  - **The 7 +- 2 phenomen**
- Abstraction allows us ignore unessential details
- Chunking: Group collection of objects
  - **Patterns**

Patterns are used by many people

- Chess Master:
  - **Openings**
  - **Middle games**
  - **End games**
- Writer
  - **Tragically Flawed Hero (Macbeth, Hamlet)**
  - **Romantic Novel**
  - **User Manual**
- Architect
  - **Office Building**
  - **Private Home**
- Software Engineer
  - ***Composite Pattern:* A collection of objects needs to be treated like a single object**
  - ***Adapter Pattern (Wrapper):* Interface to an existing system**
  - ***Bridge Pattern:* Interface to an existing system, but allow it to be extensible**
  - **....**

Hierarchy

- Hierarchy brings order into chunks obtained by abstraction and decomposition

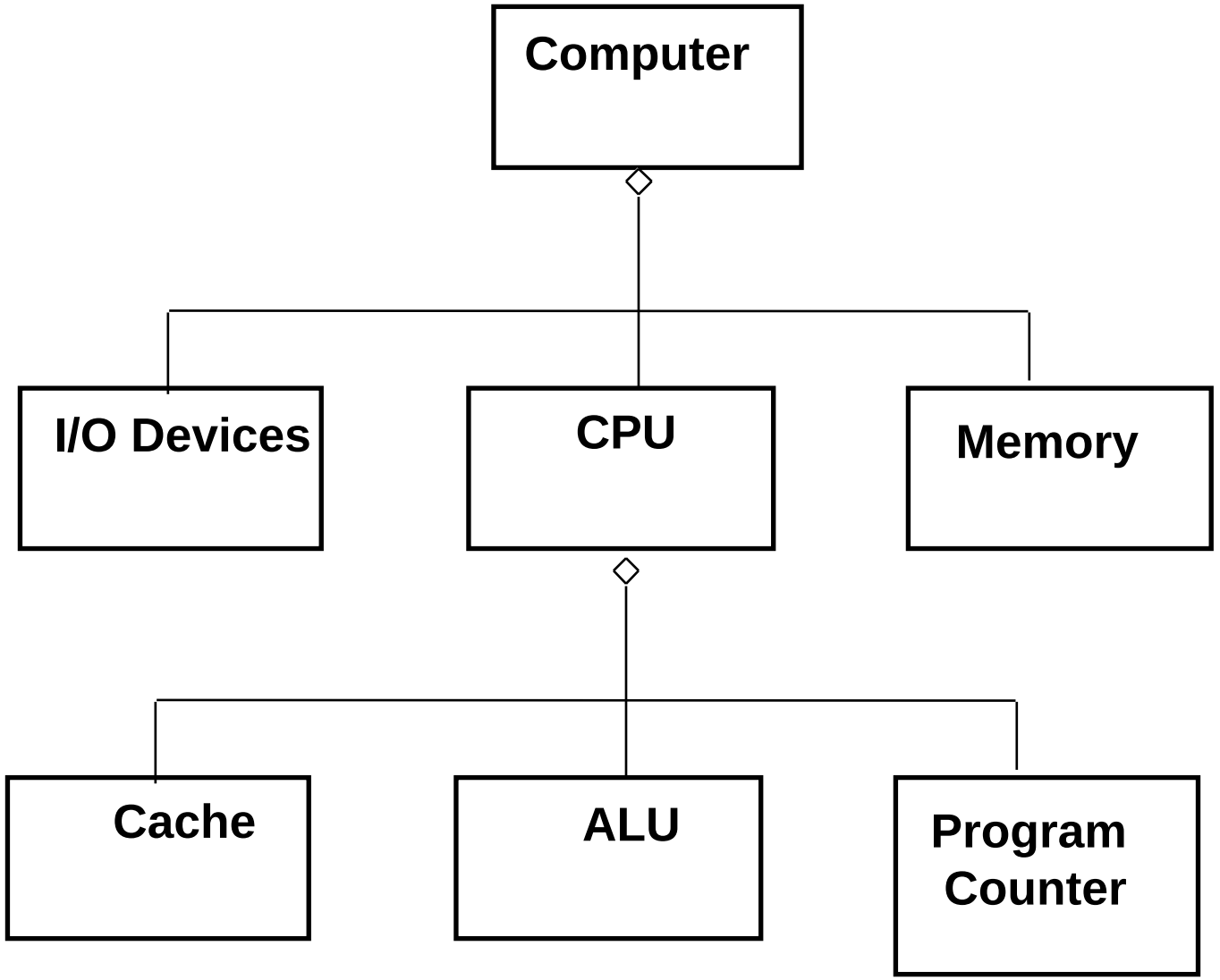
- Two very important relations between objects
  - **"Part of" hierarchy**
  - **"Is-kind-of" hierarchy**

Example of a “part of” Hierarchy

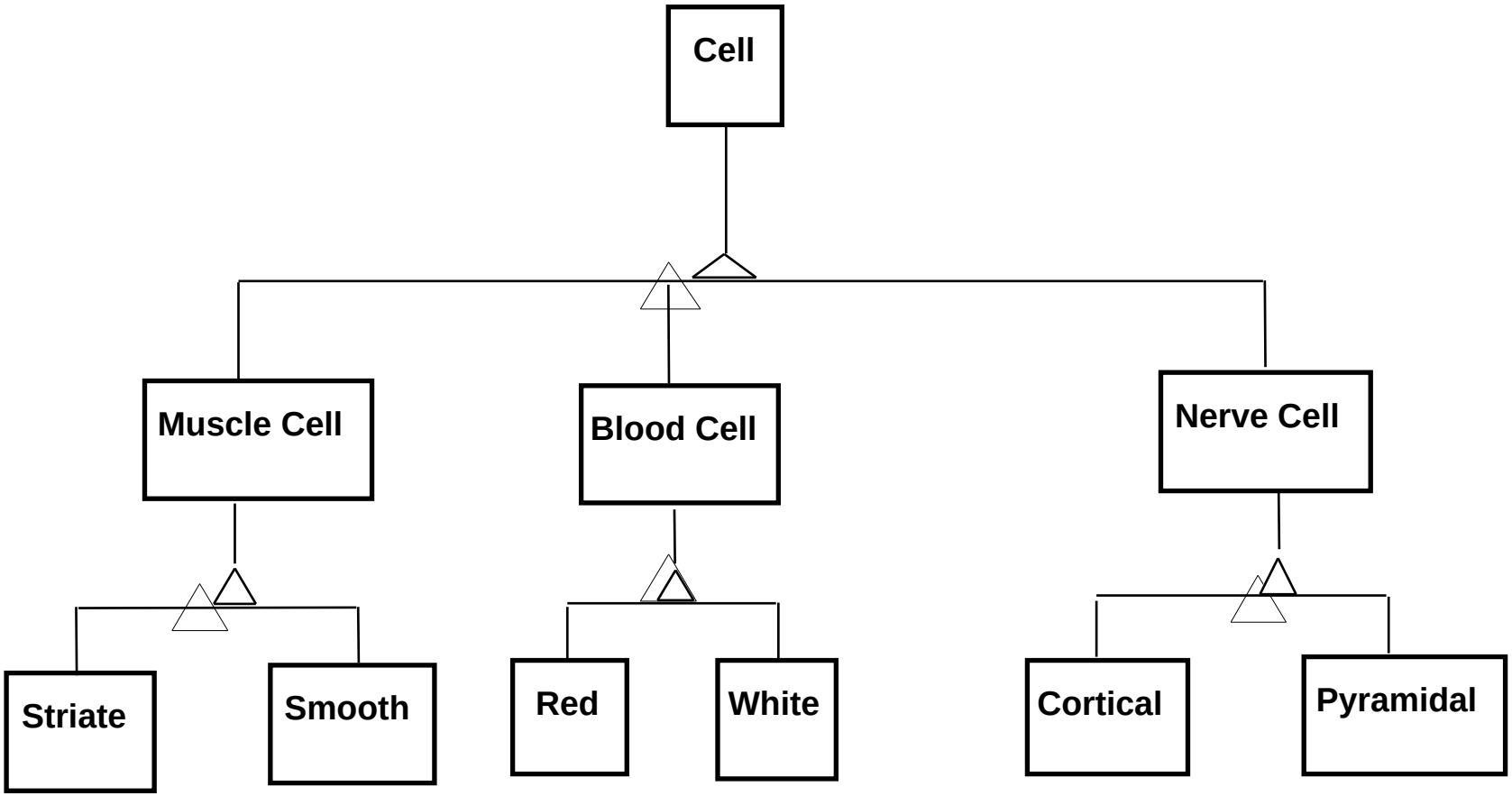
- Personal Computer
- Elements: CPU, I/O Devices, Memory
- Each of these elements can be further decomposed.
  - **CPU: Cache, ALU, program counter**
- ….

- "Part of" hierarchy describes a set of parts that logically form a whole

Part of Hierarchy



Example of a "Is-Kind-of" Hierarchy



Modeling

- Model building allows to apply decomposition, abstraction and hierarchy to software development

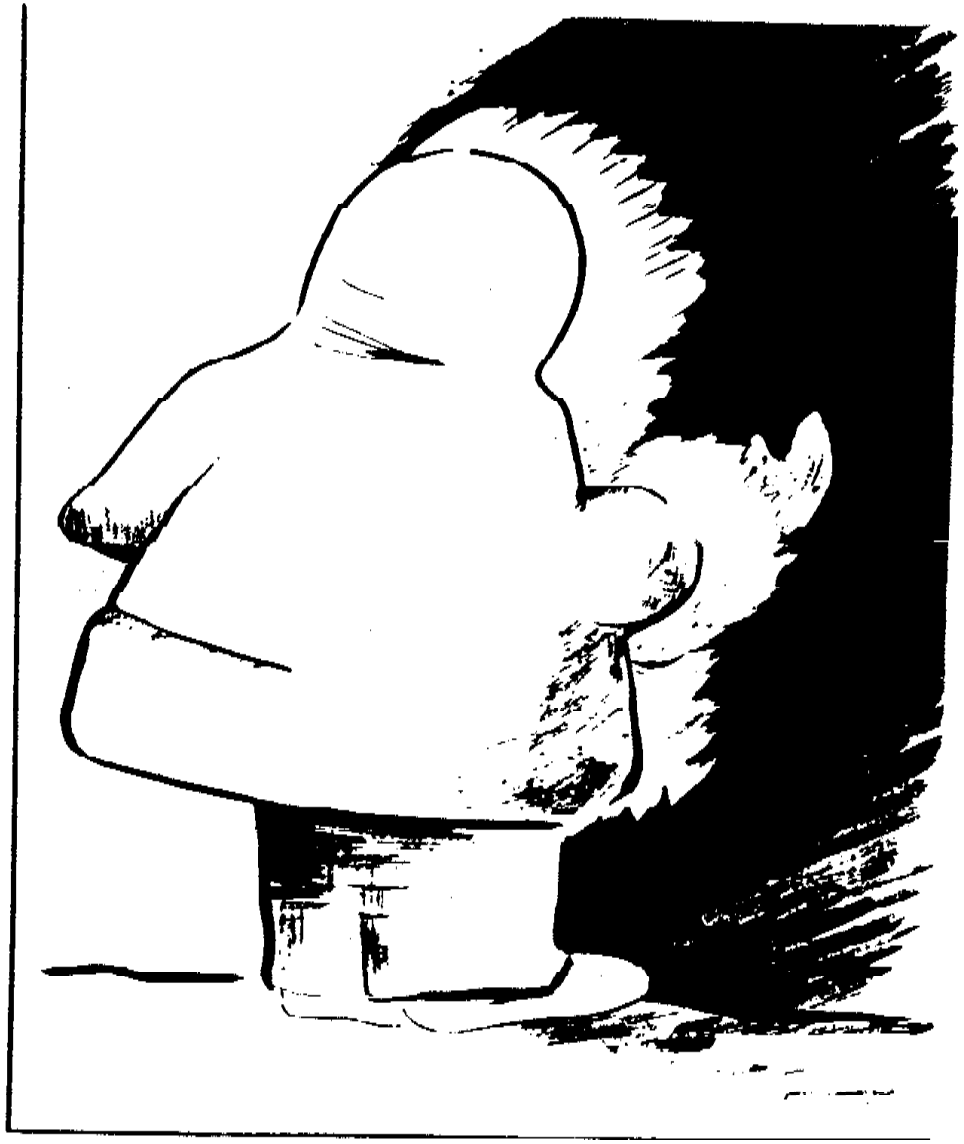
- ***Model***:
  - **Describes a specific aspect of the system under consideration. To express a complex system, a single model is not enough**

- Terminology related to Models:
  - ***Notation:*** **Language to express each model (We will use the UML notation)**
  - ***Process:*** **Guidelines for orderly construction of the models**
  - ***Product:*** **A system artifact or description of the model**

Important Modeling Activities

- Two activities:
  - **Object identification**
  - **Relationship identification**
- The problem of object identification
  - **Find objects**
  - **Define their Attributes**
  - **Define their Operations (Methods)**

What is this?



Summary

- Software engineering is a problem solving activity that addresses addresses complexity and change
- Ways to deal with complexity
  - **Abstraction**
  - **Decomposition**
    - **Functional decomposition**
    - **Object-oriented decomposition**
  - **Hierarchy**
  - **Modeling**
- Ways to deal with change
  - **Evolutionary approach to development**

What do you have to do right now?

- Access the PAID homepage
  - **http://sierra.se.cs.cmu.edu/PAID/default.html**

- Wait for the problem statement to appear on the home page (Wednesday 10am) and read it before Thursday, 9am:
  - **http://sierra.se.cs.cmu.edu/courseDocs/PS/probStmt.html**

If you need help

- Questions about Passwords, Logging into the BBoards, Accessing the Home page:
  - Eric Stein (es5f+@andrew.cmu.edu)
  - Joyce Johnstone (jdarej@cs.cmu.edu)
  - Building D 154
- Questions about the Course
  - Bernd Bruegge (WeH 4123): August 26, between 2:00PM-4:00PM
  - Elizabeth Bigelow (WeH 4110):
  - Send mail to bruegge@cs.cmu.edu or ebigelow@cs.cmu.edu
- Post your question on the Help bboard
  - Read the Help Bboard as well