CS 494
Object-Oriented Analysis & Design

Software Architecture and Design

Readings: Ambler, Chap. 7

(Sections 7.1-7.3 to start -- some of this is on detailed design.)

What is Design?

- Specification Is about *What*, and Design is the start of the *How*
- Inputs to the design process
  - Specification document, including models etc.
- Outputs of the design process
  - A design document that describes how the code will be written. Includes design models!
    - What subsystems, modules or components are used
    - How these integrate (i.e. work together)
  - Information allowing testing of the system

Design Goals

- Qualities Of A Good Design:
  - Correct, Complete, Changeable, Efficient, Simple
- Correctness:
  - It Should Lead To A Correct Implementation
- Completeness:
  - It Should Do Everything. Everything? It should follow the specifications.
- Changeable:
  - It Should Facilitate Change—Change Is Inevitable

Design Goals (cont'd)

- Efficiency
  - It Should Not Waste Resources. But:
  - Better A Working Slow Design Than A Fast Design That Does Not Work
- Simplicity
  - It Should Be As Understandable As Possible
- Important: A design should fully describe how coders will implement the system in the next phase
  - Designs are blue-prints for code construction

Levels of Design

- Three possible levels:
  - System Design, if appropriate
    - Part of Systems Engineering (see below)
  - High-level Software Design
    - Architecture, architectural design
  - Low-level Software Design
    - Detailed Design, Module Design
- Systems Engineering: Large combinations of hardware and software.
  - Decompose system into subsystems
  - Determine which subsystems are HW, which are SW
  - Software Engineering activities thus become a part of a larger activity.

High-level Design

- Goal: create a software system architecture, defining a system in terms of:
  - components;
  - interactions.
- Examples of components:
  - modules, classes
  - clients, servers
  - files, databases
  - layers

High-level Design (cont'd)

- **Interactions:**
  - Structural, behavioral
- **Examples of interactions:**
  - procedure calls
  - composition of objects
  - sharing variable access
  - More complex: communication protocols, broadcasting events, piped streams, etc.

Describing System Architecture

- Model components and how they interact
- Emphasis on component purpose and interaction
  - Not on internals of how components work (e.g. a black box approach)
- Modern techniques/ideas about architecture
  - Styles of architectures
    - E.g. book by Shaw and Garlan
    - Examples: pipe and filter; layers; repository-oriented; etc.
  - Frameworks
    - A partially completed design to which you add new components
    - Inversion of control

A View of Netscape's Architecture

- A copyrighted diagram of Netscape's architecture would appear here.
- I'm not allowed to put this on the Web.
- I've passed out copies in class. If you missed getting it in class, please contact me.

Comments on Netscape Arch. Model

- What do we see in this picture?
  - System boundary
  - Internal components
  - Relations: group together, linked
- What don't we see? What's missing?
  - Nature of relations: control? pass/share data?
  - Details of responsibilities of each component
  - Dynamic behavior (states, interactions and their order, events, etc.)
  - Threads
  - Where implemented (DLL, .EXE,…)

Architecture Views

- More than one aspect of software must be modeled and designed
- Many now use "Kructen's 4+1" model of these views
  - Logical view
  - Process view
  - Deployment view
  - Implementation view
- The "plus one" is: use-case view
  - Use cases show how the end-user interacts with the system.
  - So they affect and influence all other views

What Do Architecture Views Capture?

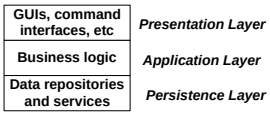
- Logical view (AKA design view)
  - Architecturally significant parts, such as layers, subsystems, components, etc.
- Process view
  - Processes or threads that make up the system
  - Are there parts of the system that run concurrently? Are locks or synchronization needed?
- Deployment view
  - Do parts of the system on separate hardware components?
- Implementation view
  - Source files, binaries, DLLs, SW components, etc.

Logical Architectural View

- We need a high-level logical view of system architecture and its components
  - Many think nothing in UML is particularly good for this. This is at a higher-level of abstraction than the level of classes.
  - Arch. Design Languages (ADLs) are an active research area
  - Often we draw a simple "box and line" diagram and explain it.
- As noted earlier, architectural styles may be useful. Examples:
  - pipe and filter; layered; client-server; black-board

Example: Layered Architectures

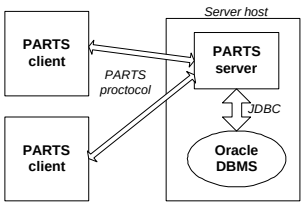
- A layer is one logical tier or stratum of the system
  - Each layer focuses on one functional subproblem
  - A layer uses the layer below it, and provides services to the layer above it.
- Three-tiered Architecture: A very common system organization



Three-Tiered Architecture

- Presentation Layer
  - User interface(s)
  - Classes from Java or MFC that provide views of data objects etc.
- Application Layer
  - Classes and objects derived from the domain (or business). Also controller classes (for business logic, etc.)
  - PARTS example: projects, problem reports, etc.
- Persistence Layer
  - How objects from the Application Layer are stored

PARTS Architectural Overview



PARTS and the Four Architectural Views

- Logical View: Client/Server
  - We must define protocol between clients and servers
  - Timeouts, state or state-less, etc??
- Process View
  - Is the server multi-threaded?
  - How are simultaneous updates handled? What are the rules? Can the DBMS enforce them all?
- Implementation View
  - Are parts of the executables stored in DLLs or JavaBeans?

PARTS and the Three-Tiered Model

- Lots of issues to be included!
- Presentation Layer
  - Screens, Windows, Style of Interactions
- Persistence Layer
  - Database used
  - Locking, backup, roll-back, security