Homework Discussion

- How do messages get from user to user?
- Should clients be “fat” or “thin?”
- What problems might we encounter?
Software Components for Client/Server Systems

- Client/Server applications are typically not seen as monolithic applications. They tend to be broken down into distinct subsystems:
  - User Interaction/Presentation Subsystem (client)
  - Application Subsystem (client or server)
  - Database Management Subsystem (server)
- “Middleware” consists of common software that exists on both the client and the server.
“Fat” and “Thin” Clients

- The application subsystem implements the requirements of the application.
- This can be done either on the server or the client.
- When the bulk of the implementation is done on the client, we call it a “Fat” client.
- When the build of the implementation is done on the server, we call the client a “Thin” client.
- Is our daytime client a “fat” or “thin” client?
Distribution of Software Components

- Distributed Presentation
  - Very Rudimentary
  - Server contains most logic, including screen layout
  - Special software is used to convert character-based screen information transmitted from the server into a GUI presentation
Distribution of Software Components

- Remote Presentation
  - An extension of the distributed presentation approach.
  - Clients have more logic to prepare the user presentation.

- Remote Data Management
  - Server gathers and formats information from other sources and feed that information to clients that then present the data.
Distribution of Software Components

• Distributed Databases
  – Data is spread across multiple servers and clients.
  – Clients must support data management software components as well as application and GUI components.
Guidelines for Distributing Application Subsystems

- The presentation/interaction subsystem is generally placed on the client
  - The availability of PC-based environments makes this approach cost effective
- If the database is to be shared by multiple users connected by the LAN, it is typically located on the server
- Static data that are used for reference should be allocated to the client
  - Places data closest to the users that require it
Middleware and Object Request Broker Architectures

- Clients and Servers must be capable of interacting with one another.
- An object request broker (ORB) is middleware that enables an object that resides on a client to send a message to a method that is encapsulated by an object that resides on a server.
- ORBs provide a layer of abstraction that make client/server communication appear as simple method calls.
Common Object Request Broker Architecture (CORBA)

- An Interface Definition Language (IDL) is used to define the interface between the client and the server (see example).

- The IDL is “compiled” to generate stubs for both the client and the server. These subs are responsible for marshalling and demarshalling parameters and return values across the network.

- At an interface level, the client/server application appears to be monolithic.
CORBA Implementation of a Daytime Client/Server

• See example

• How does this differ from our previous implementations?
Software Engineering of Client/Server Systems

• Almost any software engineering process can be used for Client/Server systems.

• Two are most commonly used:
  – An evolutionary paradigm that makes use of event-based and/or object-oriented software engineering.
  – Component-based software engineering that draws on a library of COTS and in-house software components.

• Client/Server systems are developed using the classic software engineering activities – analysis, design, construction, and testing.
Analysis Modeling Issues

• The requirements modeling activity for client/server systems differ little from the analysis modeling methods applied to more conventional computer architectures.

• Because an evolutionary approach to software engineering is applied for client/server systems, implementation decisions on the overall approach may be made during early analysis and design iterations (i.e. CORBA or simple network?)
Design for Client/Server Systems

- Design must be “customized” to accommodate the following issues:
  - Data and architectural design
    - Communicating process style – how do the client and server interact?
  - Behavioral modeling for event-driven systems
    - How are asynchronous event to be handled?
  - The user interaction/presentation design
  - An object-oriented view of design is often chosen.
Testing Issues

- The distributed nature of client/server systems pose a set of unique problems for software testers. The following are suggested areas of focus:
  - Client GUI considerations
  - Target environment and platform diversity considerations
  - Distributed database considerations
  - Distributed processing considerations
  - Nonrobust target environment
  - Nonlinear performance relationships
Overall Client/Server Testing Strategy

- Testing occurs at three different levels
  - Individual client applications are tested in a “disconnected” mode, the operation of the server and the underlying network are not considered.
  - The Client software and associated server applications are tested in concert, but network operations are not explicitly exercised.
  - The complete architecture, including network operation and performance, is tested.
Questions?