Lecture 6: Reusability, Client-Server Architecture

Prof. Jeff Turkstra
Announcements

- Homework 1
  - Due Today 11:59pm
- Code repository, project name, and product backlog
  - Due Friday, January 27
Lecture 06

- Basing development on reusable technology
- Client-server architecture
“If I have seen further it is by standing on ye sholders of giants.”
- Isaac Newton
Building on the experience of others

- Software engineers should avoid re-developing software
- ...and try to reuse:
  - Expertise
  - Standard designs and algorithms
  - Libraries
  - Frameworks
  - Complete applications
Reusability and reuse

- Reuse and design for **reusability** should be part of the culture of software development organizations

- But
  - Why take **extra time** to develop something that will benefit other projects?
  - What if management primarily rewards ‘visibility’?

- Software is often created without enough attention to **quality** or reuse
Vicious cycle

- Developers take **short cuts** to save time, sacrificing quality and reusability
- **Important to recognize that:**
  - This cycle costs **money**
  - **Investing** in reusable code is important
  - Attention to **quality** is essential
  - Employing reusable components often simplifies design
Frameworks

- A framework is reusable software that implements a generic solution to a generalized problem
  - Provides common facilities applicable to different applications
- Based on the principle that applications that do related things tend to have similar designs
- Frameworks promote reuse
- Intrinsically incomplete
  - **Slots**: certain classes or methods are missing
  - **Hooks**: optional functionality, allowance made for developer to provide it
- Developers use the services that the framework provides
  - **Application Program Interface (API)**
Object-oriented frameworks

- Framework is composed of a library of classes
  - API is defined by the set of all public methods
  - Some classes intentionally abstract

- For example
  - Payroll management
  - Frequent buyer clubs
  - University registration
  - E-commerce web site
Product line

- A *product line* (or *product family*) is a set of products built on a common technology base
  - Individual products have different features to satisfy different markets
  - Common software technology included in a framework
  - Each product produced by filling in desired hooks and slots
A distributed system is a system of discrete networked components
- Have concurrency
- Lack a global clock
- Can encounter independent failure of components
- Coordinate by passing messages

Components cooperate to create a system
Client-Server Architecture

- One kind of a distributed application or system
- Server
  - Program that provides a service for other programs
- Client
  - Program that accesses one or more servers to obtain services
- Communication Channel
  - Generally a computer network
  - Client must initially know the server, but not vice versa
Examples

- Email
- DNS
- World Wide Web
- /etc/services
Sequence

- Server starts running
  - Creates a socket
  - Binds the socket to an address
  - Waits for clients (listening)
- Client requests something from the server
  - Creates a socket
  - Attempts to connect to server
- Server accepts the connection
- Send and receive data (read and write)
...and

- Initialization
- Disconnection
- Termination
Client

- Initialize
- Initiate a connection to a server
- Interact with the user, sending messages to the server as necessary
  - Do: Respond to messages
  - Do: Handle server disconnection
- Respond to events triggered by the server
- Terminate
Advantages

- Work can be distributed among different machines
  - Load balancing
  - Failover
- Client can access server from a distance
- Client and server can be designed separately
  - Both can be simpler
- Data can be kept **centrally at the server**
- Or among many geographically **distributed** clients or servers
- Server may be accessed **simultaneously by multiple clients**
Thin vs. fat

- Thin-client
  - Client is small as possible
  - Most work done on server
  - Client easy to download

- Fat-client
  - As much work as possible delegated to clients
  - Server can handle more clients

- Usually somewhere in between
Protocols

- Protocols are to communications what programming languages are to computation
- Server and client are programmed to understand the protocol
Developing client-server applications

- Design the primary work to be performed by client and server
- Design how the work will be distributed
- Design the protocol
Broadly

- Initializing
- Handling connections
- Sending and receiving messages
- Terminating
Object Client-Server Framework (OCSF)
Using OCSF

- More at http://lloseng.com/
- Avoid modifying the three classes
- Create subclasses
- Call public methods
- Supply slot methods and override
- Provide hook methods