CS 580 Client-Server Programming
Spring Semester, 2009
Doc 5 Some GUI
9 Feb, 2009

Copyright ©, All rights reserved. 2009 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.
References


Domain-Driven Design, Eric Evans, 2004, Addison-Wesley

Patterns of Enterprise Application Architecture, Martin Fowler, 2003, Pearson Education

Recommended Reading – Designing GUIs

User Interface Design for Programmers, Joel Spolsky, 2001

There is a printed longer version of the book. The on-line version is free and will get you started.

Design of Everyday Things, Donald Norman, 1990

This is an excellent book, is entertaining and only costs $12 new at Amazon. Anyone that designs or builds anything has to read this book.

These books do not cover the mechanics building a GUI. They do not cover which fonts and colors to use. They just get you started thinking about the really important questions related to GUI design.
Software Structure with UI & Database

Tiers – parts of program run on different machines

Layers – parts of program run on same machine

Layers

Presentation – Display of Data

Domain – Logic related to purpose of the application

Data Source – Communication with data source
Keep the presentation & domain layers separate

GUI code should not contain domain logic

In simple cases different methods in one class may handle presentation and domain logic

A single method does either presentation or domain logic

Can you test the domain logic with automated unit testing
public class TimeDateClient{
    private static final char CARRIAGE_RETURN = (char) 13;
    private static final char LINE_FEED = (char) 10;
    String server;
    int serverPort;

    public TimeDateClient(String serverNameOrIP, int port){
        server = serverNameOrIP;
        serverPort = port;}

    public String date() throws IOException  {  return sendMessage("date");}

    public String time() throws IOException{ return sendMessage("time"); }

    public String sendMessage(String message) throws IOException{
        Socket serverConnection = new Socket(server, serverPort);
        writeMessage(message, serverConnection);
        byte[] result = readMessage(serverConnection);
        serverConnection.close();
        return new String(result);
    }
}
private byte[] readMessage(Socket serverConnection) throws IOException {
    UpToFilterInputStream in = new UpToFilterInputStream(
        new BufferedInputStream(
            serverConnection.getInputStream()));
    byte[] result = in.upTo(LINE_FEED);
    return result;
}

private void writeMessage(String message, Socket serverConnection) throws IOException {
    OutputStream out = new BufferedOutputStream(
        serverConnection.getOutputStream());
    out.write((message + CARRIAGE_RETURN).getBytes());
    out.flush();
}
Now add a GUI that uses Domain layer
Smart UI Pattern

“the separation of UI and domain is so often attempted and so seldom accomplished that its negation deserves a discussion”
Eric Evans, Domain-Driven Design

The Pattern
Put all business logic into user interface

Divide the application into different small user interfaces

Use relational databases as back end

Use automated UI building and visual programming tools
Smart UI Pattern

Advantages

High and immediate productivity for simple applications
Little training need by the developer
Short development time for small modules

Disadvantages

No reuse – code gets duplicated
Integration of applications difficult
Very difficult to add new functionality to existing application
Difficult to build complex applications
GUI Clients & Servers

GUI Clients
   Used to get work done
   Good when they help people get work done

Users do not care about the client-server protocol

Don’t expose the user to the client-server protocol

Build the GUI to help people accomplish a task, not just to perform the client-server protocol
Example: Time Date Client

Protocol has two messages
  Date
  Time

Client has one button to get time & date
Interface Design When You Don't Know How
Basic Rule for Good Visual Design

Hire a graphic/GUI designer
Basic Rule for Almost Everything Else

Painstakingly follow established standards
What Makes a Good GUI?

A user interface is well-designed when the program behaves exactly how the user thought it would.

Deleting a file on a Mac

Move it to the trash can!

How do you unmount floppies & external hard drives?

Move it to the trash can

But users think this will delete or erase it
Mental Models & Users

Users don’t read the manual

More text on the screen - fewer people will read it

Users have a mental model of how your program works

Make sure your program fits their mental model

Users think the trash can deletes things
The Process of Creating a GUI

Plan ahead
Use bite-sized chunks
Abandon the waterfall life cycle in favor of iterative design
Conduct user testing early and often
Focus on the users' needs and involve them in the process
Come up with good, testable usability goals
Hire a graphic designer
Why iterative design?
But we don't have time to redo the GUI!
Prototype the GUI?
Problems with Software Prototypes

Software Prototypes take too long to build and change

Testers tend to comment on "fit and finish" issues

Developers resist changes

Managers ship the prototype

Software Prototypes set false expectations

Single bug in a software prototype can halt a test
Use Paper Prototypes
Paper Prototype Kit

White paper
   Unlined
   Heavy enough to endure repeated testing and revisions

5-by-3-inch cards
   Use as construction material
   Taking notes

Adhesives
   Tape: clear, colored, double-backed, etc.
   Glue sticks
   Post-It glue
   White correction paper

Markers
   Colored pens and pencils
   Highlighters
   Fine and thick markers

Sticky note pads
Acetate sheets
Scissors,
X-acto knives,
straightedges,
Band-Aids
White-out
Build A Paper Prototype

Do it fast

Construct Models, not illustrations
Test your Prototype with Users
Preparing for a Test

Select your users

Know the demographics of your users

Testers should represent the whole range of users

Prepare test scenarios

Practice
Conducting a Test

Greeter
- Welcomes users
- Puts users at ease

Facilitator
- Runs the test
- Gives users instruction
- Encourage the user to express their thoughts

Computer
- Runs the computer

Observer
- Takes notes
The Test

Video tape the test

Before starting the test, explain the process to the user

Debrief the user after the test

Evaluate the Results