CS 580 Client-Server Programming
Spring Semester, 2005
Doc 5 Some GUI
Contents

References

Rettig, Marc. Interface Design When You Don't Know How, Communications of the ACM, Jan. 1992, Vol. 35, No. 1, pp. 29-34

Rettig, Marc. Prototyping for Tiny Fingers, Communications of the ACM, April. 1994, Vol. 37, No. 4, pp. 21-27

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

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

User Interface Design for Programmers, Joel Spolsky, 2001
http://www.joelonsoftware.com/navLinks/fog0000000247.html

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

Rule One

- 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

Time Date Domain Layer

```
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);
      }
```

Time Date Domain Layer

```
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();
    }
```

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

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

Handling Domain Logic[1]
Transaction Script

Each procedure (method) handles one request from the UI

Each database transaction handled by one procedure

Advantages

- Simple to implement
- Easy to follow how things get done

Disadvantages

- No reuse – code gets duplicated
- Very hard to handle complex operations
- Does not scale well

[1] See Fowler 2003

Table Module

Each class represents one table in database

A request on the database is done through its table module

Advantages

- Easily maps to the database
- Reflects the database structure – once you know the database schema, you know the table module logic

Disadvantages

- Difficult to handle complex domain logic
- Database schema is reflected in application structure

Domain Model

Use objects to model the domain

Develop application with OO design

Advantages

Scales well with complexity

Disadvantages

Overly complex for simple solutions
Need to map object model to database schema (structure)

GUI Clients & Servers

GUI Clients

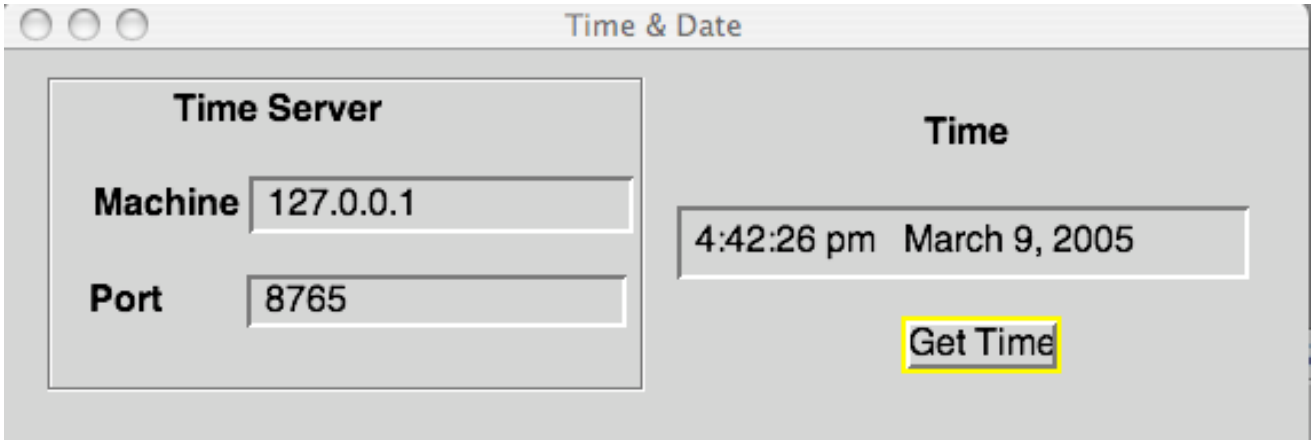
- Used to get work done
- Good when they help people get work done

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

All major interfaces have published detailed standards for user interface design

- Windows
- MacOS X
- Java
- GNOME
- KDE

What Makes a Good GUI?

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

Example - 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

The more text you put on the screen the 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

User Interface Standards Manual

Produce standards manual for the project

Manual should address the following:

- Lexical Structure

  What symbols are there?

- Syntactic Structure

  How do they relate to each other?

- Semantics

  How do they relate to the things they represent?

- Pragmatics

  How do they relate to the user?

The Process

- 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

Planning

Set a short time limit (4 weeks) on the planning process

Planning Documents

Document One

- Defines design goals
- Sets the direction for the work
- Identifies open problems that need to be researched

Document Two

- Project plan for next four months
- Describe series of tasks, each no longer than two weeks

Document Three

- Plan for user testing

Document Four

- Definition of Iterative Process

Prototyping
Problems with Software Prototypes

Software Prototypes take too long to build and change

Testers tend to comment on "fit and finish" issues

- Get comments on selection of colors, fonts and buttons
- Developers spend time on colors, fonts and buttons

Developers resist changes

Managers resist change

Software Prototypes set false expectations

Single bug in a software prototype can halt a test

Lo-fi or Paper Prototypes
The 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

Set a short deadline

- The prototype will be done by 4:00 this afternoon
- Tomorrow at 9:30 we will demo the prototype
- Just get a first pass on all aspects of the prototype
- It will be "wrong" so don't spend weeks on it

Construct Models, not illustrations

- Build the parts of your prototype so they can be used
- The parts need to move around
- User will use the model, must see the changes
- On early models don't be picky - get ideas down
- Later models can look "professional"

Preparing for a Test

Select your users

- Know the demographics of your users
  - Educational background
  - Knowledge of computers
  - Typical tasks involved in their job
- Can use "surrogate users"
  - People with same demographics of your users
  - Undergrads are cheap available labor
- May wish to avoid actual customers, employees, friends, and family
- Testers should represent the whole range of users
- Do at least one in-house test with surrogates and one field test with typical end users

Prepare test scenarios

Practice

Conducting a Test

It takes four people to get the most out of a test

Greeter

- Welcomes users
- Puts users at ease
- Users often worry about:
  - Flunking the test
  - Co-workers finding out how they did
  - Answering the question correctly
- Have users fill out forms - experience profile

Facilitator

- Runs the test
- Gives users instruction
  - Gives user hand written tasks to perform
- Encourage the user to express their thoughts
  - Elicit the user's thought during the tests
- Makes sure test runs on time

Computer

- Runs the computer
- Moves the model in response to users "actions"

Observer

- Takes notes
- One observation per index card

The Test

Video tape the test

Before starting the test, explain the process to the user

Debrief the user after the test

During the test, don't:

- Give user hints
- Laugh, gape, gasp, or say "a-ha"
- Don't display any reaction to the user's actions

Evaluate the Results

Go through all the notes taken during the tests

Use post-it notes to put comments on components of prototype that need change