CS 520
Theory and Practice of Software Engineering
Fall 2021
User Interfaces
September 14, 2021
Recap: Simple Statistics App

https://github.com/LASER-UMASS/cs520-examples.git
Recap: Satisfied best practices
Recap: Violated best practices
Recap: Code reviewing

- Naming conventions
- Decomposition into packages, classes, and methods
- Type safety for methods (e.g., enum types instead of ints/strings)
- Pre- and post-conditions for methods (e.g., defensive programming techniques, run-time assertions)
Recap: Code reviewing (cont.)

- Version control system (e.g., git)
- Documentation (e.g., internal comments, javadoc comments, README files, build files)
- Test suite (e.g., junit)
User Interfaces (UIs)

- Usability
- UI design
- UI widget toolkits
- Prototyping
Usability and software design

- **Usability**: the effectiveness of users achieving tasks

- Usability and good UI design are closely related

- A bad UI can have serious results...
Florida, 2018 (Broward County)

Florida 2018 Undervote
The Share of Voters Who Cast Ballots for Both Senate and Governor

- 3.7% of Senate Ballots Blank
- 1% or Less of Senate Ballots Blank
- More Votes for Senate than Governor
Achieving usability

• User testing and field studies
  – having users try the product and gathering data
• Evaluations and reviews by UI experts (often from Human-Computer Interaction or HCI as well as Human Factors)
• Prototyping
  – Paper prototyping
  – Mock up tool prototyping
  – Code prototyping

• Good UI design focuses on the user
  not on the developer, not on the system environment
How do we avoid bad UI?

• Learn from bad UI designs (i.e. past mistakes)

• Apply good UI design principles (or rules)

• Build UI prototypes
How do I know whether my UI is good or bad?

• What are the ways in which UI quality can be quantified?
• What are some examples of software you use that have an especially good/bad UI?
• What do you think makes them good/bad?
Schneiderman's 8 Golden Rules

1. Strive for consistency.
2. Give shortcuts to the user.
3. Offer informative feedback.
4. Make each interaction with the user yield a result.
5. Offer simple error handling.
6. Permit easy undo of actions.
7. Let the user be in control.
8. Reduce short-term memory load on the user.

(from Designing the User Interface, by Ben Schneiderman of UMD, noted HCI and UI design expert)
UI design:
Sample 1 and Sample 2
UI design:
Sample 3 and Sample 4
UI design:
Sample 5 and Sample 6

Do you want to save the changes you made to '20200910Ui.pptx'?

Don't Save  Cancel  Save

Do you really want to delete the selected folder?

Please enter 'YES' to start the operation

OK  Cancel
UI design:
Sample 1 and Sample 2

good

bad
UI design:
Key issue with Sample 2
• Does not reduce short-term memory load on the user:
  – Large number of tabs
  – Large number of other components
  – Complicated layout
  – Many colors
UI design:
Sample 3 and Sample 4

bad

good
UI design: Key issues with Sample 3

• Does not strive for consistency:
  – Not using standard icons

• Does not reduce short-term memory load on the user:
  – Many colors
UI design:
Sample 5 and Sample 6

Do you want to save the changes you made to '20200910Ui.pptx'?

Don't Save  Cancel  Save

AK-Mail

Do you really want to delete the selected folder?

Please enter 'YES' to start the operation

OK  Cancel

good

bad
UI design:
Key issues with Sample 6

- Does not offer informative feedback
- Does not offer simple error handling
UI widget toolkits

• What are common components of such widget toolkits?
• What commonly used desktop applications provide widget toolkits?
• For UI design, how to select among widgets (or components)?
Common components of UI widget toolkits

• A frame or window
• A label
• A button
• A check box
• A radio button
• A text field
• A list
• A combo box
• A menu
• A dialog box
• Other…
Commonly used desktop applications that provide widget toolkits

• Programming language environment (e.g., Java Swing, Eclipse SWT)

Other applications?
Common desktop applications that provide widget toolkits

• Programming language environment (e.g., Java Swing, Eclipse SWT)
• Web browser (e.g., CSS/HTML/javascript)
• Operating system (e.g., GTK+ for Fedora, Aqua for Mac)
UI design –
Supporting user actions

• Use **buttons** for single independent actions that are relevant to the current screen.
  – Try to use button text with verb phrases such as "Save" or "Cancel", not generic: "OK", "Yes", "No"
  – use **Mnemonics** or Accelerators (Ctrl-S)

• Use **toolbars** for common actions.

• Use **menus** for infrequent actions that may be applicable to many or all screens.
  – *Users hate menus!* Try not to rely too much on menus. Provide another way to access the same functionality (toolbar, hotkey, etc.)
UI design –
Selecting among alternatives

• Use **check boxes** for independent on/off switches
• Use **radio buttons** for related choices, when only one choice can be activated at a time
UI design – Eliciting other user input

- Use **text fields** (usually with a label) when the user may type in anything they want.

- Use **lists** when there are many fixed choices (too many for radio buttons); *all* choices visible on screen at once.

- Use **combo boxes** when there are many fixed choices; don’t take up screen real estate by showing them all at once.

- Use a **slider** or **spinner** for a numeric value.
UI design –
Supporting multiple screens

• Use a **tabbed pane** when there are many screens that the user may want to switch between at any moment

• Use **dialog boxes** or **option panes** to present temporary screens or options
An example UI design

• Did the designer choose the right components? assume there are 20 collections and 3 ways to search
Feedback on the example UI design

- For eliciting user input, consider a combo box instead of a list.
- For selecting among YES/NO alternatives, consider a check box instead of a radio button.
- The button labels are unclear.
  - In particular, "Default" could be "Reset to Default"
What's the point of prototyping?

• Should I do it?
  – If so, when should I?

• Should I make my prototype on paper or digitally (either mock up or code)?
Prototyping

• **prototyping**: Creating a scaled-down or incomplete version of a system to demonstrate or test its aspects.

• Reasons to do prototyping:
  – aids UI design
  – provides basis for testing
  – team-building
  – allows interaction with user to ensure satisfaction
Some prototyping methods

1. Paper prototyping: a paper version of a UI
2. Mock up tool prototyping: a digital version of a UI
3. Code prototyping:
   • Written by hand
   • Automatically generated by UI builder
Some prototyping methods

1. **Paper prototyping**: a paper version of a UI
2. Mock up tool prototyping: a digital version of a UI
3. Code prototyping:
   - Written by hand
   - Automatically generated by UI builder
Creating a “paper” prototype

• gather materials
  – Paper, pencils/pens, scissors, ..
  – Whiteboard, markers, ...

• identify the screens in your UI
  – consider use cases, inputs and outputs to user

• think about how to get from one screen to next
  – this will help choose between tabs, dialogs, etc.
Example paper prototype screen

Page Setup

Margins

Paper Size

Paper Source

Layout

Paper Size:

- Letter (8.5 x 11 in)
- Width
- Height
- Orientation

Preview

Default...

OK  Cancel
Some advantages/disadvantages of paper prototyping

• Can be done by non-technical people
• More conducive to working in teams
• Much faster to create than mock up or code and can change faster
• More visual bandwidth (can see more at once)
• Feels less permanent or final
• Doesn’t have a real-world look & feel
Some prototyping methods

1. Paper prototyping: a paper version of a UI

2. **Mock up tool prototyping:** a digital version of a UI

3. Code prototyping:
   - Written by hand
   - Automatically generated
Mock up tool prototyping

• Create a UI design by dragging/dropping components from a UI widget toolkit (e.g., Infragistics Indigo Studio)

• Uses of a mock up tool:
  – A static image of the UI design
  – An animation of the UI design
Demonstration of Indigo Studio mock up for Page Setup

https://www.infragistics.com/
Some advantages/disadvantages of mock up tool prototyping

• Could be done by non-technical and/or technical people
• More conducive to working in teams
• Generally faster to create than code and can change faster than code
• Has a real-world look & feel
Some prototyping methods

1. Paper prototyping: a paper version of a UI
2. Mock up tool prototyping: a digital version of a UI
3. Code prototyping:
   • Written by hand
   • Automatically generated
1. Written by hand
   implement a quick version of your code

2. Automatically generated by UI builders (e.g., Visual Studio)
   draw a UI visually by dragging/dropping UI components on screen to automatically generate the code for those components
Some advantages/disadvantages of code prototyping

• Generally needs to be done by technical people
• Not as conducive to working in teams
• Usually more effort than paper or mock ups
• Has real-world look & feel
• Generated code is very difficult to understand
Commonly use the SUS (System Usability Survey)

<table>
<thead>
<tr>
<th>The System Usability Scale</th>
<th>Standard Version</th>
<th>Strongly disagree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I think that I would like to use this system.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>2</td>
<td>I found the system unnecessarily complex.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>3</td>
<td>I thought the system was easy to use.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>4</td>
<td>I think that I would need the support of a technical person to be able to use this system.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>5</td>
<td>I found the various functions in the system were well integrated.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>6</td>
<td>I thought there was too much inconsistency in this system.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>7</td>
<td>I would imagine that most people would learn to use this system very quickly.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>8</td>
<td>I found the system very cumbersome to use.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>9</td>
<td>I felt very confident using the system.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>10</td>
<td>I needed to learn a lot of things before I could get going with this system.</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
<td>○ ○ ○ ○ ○ ○ ○ ○</td>
</tr>
</tbody>
</table>

Final project description

- Each team of 5 will carry out one of the following projects:
  - MSR Mining Challenge
  - Replication Study (e.g., Automated Program Repair)
  - ML Development Toolkits (e.g., Weights & Biases)
  - EleNa: Elevation-based Navigation
- The key phases of the project are: topic selection, midpoint presentation, final presentation (and submission)
- More details available here:
  - [https://people.cs.umass.edu/~hconboy/class/2021Fall/CS520/finalProject.pdf](https://people.cs.umass.edu/~hconboy/class/2021Fall/CS520/finalProject.pdf)