Application UI Design Styles

The structure of how the functionality of the application is expressed using UI structure, relationships and widgets.
Application Design UI Styles

- SDI
- MDI
- Workbook
- Project
- Wizard
- Dialog
SDI vs. MDI

- **SDI - Single Document Interface**
  - Each document has its own executing copy of the application.
  - Most applications are SDI now or support switching between SDI and MDI styles.

- **MDI – Multiple Document Interface**
  - Originally championed by Windows for most apps but now out of favor. (Shared menus, shared code in memory saves memory and made multitasking easier to implemented)
  - Had an advantage on mem and real estate constrained systems.
  - More expert oriented software may use MDI
SDI

- Each instance running has its own set of **menus and tools**. To switch between the documents you use the **taskbar** to switch executing instances or via the **Window** menu.
- A single primary window for each document / object
- Target – Novice and Practitioner
- Ex: Word, Excel, PowerPoint
Switching windows in SDI

• Via the windows menu

• Via the task bar/system tray
SDI

• G:
  – Data Centered
  – Less Confusing than MDI
  – Single View (at any given time) of a given object e.g. work processor,
  – Targets Novices, Practitioners

• B:
  – Complex workflow type apps
  – Time dependent/oriented operations
  – Multiple views of the same object e.g. IDEs, HTML editor (Browser View, Text View,..), Video Editors (Clips, Time Line for Clips, ...)

Copyright (c) 2003 - Poelman & Associates, Inc. unless otherwise copyrighted. All rights reserved.
MDI – two windows showing different views of the same object
MDI Example – different objects
MDI

- A parent window with children (undocked and floating) inside the parent window frame.
- NetBeans can be configured to be SDI or MDI.
- **Child** windows share the menus of the **parent** window.
- Used to present multiple views of the same object. Ex: java code and swing visual editor of NetBeans
- Used to allow comparing two different objects (docs) and/or moving data between them via drag and drop (or cut/paste)
MDI

• G:
  – Multiple views of the same object i.e. code & GUI/visual editor in netbeans
  – Managing multiple windows open at same time because they are contained within the parent app window
  – Easy drag and drop of objects within the app

• B:
  – Powerful but confusing to novices (and some experts)
  – Constrained real estate situations
  – Can be confusing what window the menu is affecting
  – No real advantage in resources used by app over SDI
Even SDI is some MDI
PowerPoint acting as a SDI app

Application “Flavors” (Types)

- Sovereign
- Transient
- Parasitic
- Daemonic
- Kiosk
PowerPoint acting as a MDI app
Workbooks

• Multiple windows in the same application but accessed via **tabs** like a workbook
• Child windows don’t exist
• Used to manage a set of views of an object e.g. options for an application
• Quick navigation
• Can be used shows order of windows by order of tabs
• Can’t present simultaneous views
Workbook example
Workbook (Tabbed) Dialog

Options

- View
- General
- Edit
- Print
- Save
- Security
- Spelling and Style
- Asian

File encryption settings for this document
- Password to open: [Blank]
- Advanced...

File sharing settings for this document
- Password to modify: [Blank]
- Digital Signatures...

Privacy options
- Remove personal information from this file on save

Macro security
- Adjust the security level for opening files that might contain macro viruses, and specify the names of trusted macro developers.

Options

- View
- General
- Edit
- Print
- Save
- Security
- Spelling and Style
- Asian

Spelling
- Check spelling as you type
- Hide all spelling errors
- Always suggest corrections
- Ignore words in UPPERCASE
- Ignore words with numbers

Style
- Check style
- Style Options...
Workbook

• Selected (and visible Tab) is the one that the menus are acting on if document or smaller focus of action.

• Can be used as part of a SDI application – ex: Excel
Workbook

• G:
  – Setting many Options/States/Configurations in a single dialog
  – Managing multiple different open document/object instances in a single application and supporting easy switching between

• B:
  – Can’t compare two open instance on same screen (unlike a MDI)
  – Targeted at Experts and Gurus
Project

- A single main window (parent or controller) that coordinates and controls multiple child / peer windows. These windows are not constrained within the parent window but may be anywhere on the desktop / maximized / minimized / overlapped.
- Maybe no controller window (peers) in some cases.
- An extension of MDI where child windows are NOT contained within the parent.
- Multiple cooperating “tool” windows or different views of the same object or collections of objects and views of each object in the collection.
- Target – Expert, Guru users
- Ex: Video Editing Software, Music Players, IDE and Testing Environments
Example Project Application
Netbeans as a Project Style
Project

• G:
  – Complex application
  – Workflow intensive apps
  – Tool based applications
  – Flexibility to close unused windows/tools
  – Multiple monitor systems with lots of real estate
  – Flexibility to arrange the child windows as user chooses by dragging around on the desktop

• B:
  – Managing too many windows – can’t find the window underneath a different window
Wizard

• A set of dialogs that are arranged in a constrained sequence of operations.
• Usually a set of **Previous** and **Next** buttons on the dialog with a **Finish** button on the final dialog that **commits** the previous actions.
• Target - Novices
Wizard
Wizard

- **G:**
  - Q&A type setup operations
  - Limiting the range of operations that a user can do
  - Novices
  - Rarely executed operations (not daily or weekly type operations)
  - Simplified and linear/hierarchical workflows
  - Preventing users creating a set of states that are NOT sensible
  - Predictable result of setting a certain state without requiring setting it and then querying for the result

- **B:**
  - Experts and Gurus
  - Non-linear/non-hierarchical workflows
  - Non-Q&A types of operations
Dialog

• Top level window is a Dialog Box (no menu bar or very simple menu)
• Pops up and does some limited set of operations
• Often configuration of the OS or Machine
• Usually can’t be maximized e.g. calculator
• Often simple tools
Dialog

- Ex: Desktop Properties, Calculator, Magnifier, Computer Properties, Volume Control
Dialog

• G:
  – Simple apps
  – Simple desktop tools
  – Configuring hardware / OS features
  – Displaying settings
  – Transient apps

• B:
  – Complex applications
  – Sovereign apps
Application Design UI Styles

- SDI
- MDI
- Workbook
- Project
- Wizard
- Dialog

*Often an application is a combination of multiple design styles that the style is appropriate to the users goals and constraints*
Application UI “Execution”

Flavors

How the users will use the application in general
Sovereign

- The “King / Queen” application
- Open and utilized by the user for long periods of time
- Main focus of users activities
- Ex: Worprocessor, Video Editor, IDE, …
Transient

• Usage is transitory
• May be executed, utilized and then closed or minimized.
• Does use all the screen real estate, usually
• May “hang around” but in a screen real estate minimal mode e.g. thin floating tool bar of a MP3 player
• Ex: Tools, MP3 Player, IM Chat app
Parasitic

• Runs in response to some other application
• Comes in response to an event in a different application and then is dismissed
• Could also be run on its own
• May take up most of the screen real estate temporarily
• Ex: Scanner applet, printer controller / spooler applet
Daemonic

- Runs in the background without requiring ANY UI be visible
- UI may be “summoned” when needed but application runs without the UI visible
- Often, no taskbar entry
- Often, a service in the OS e.g. firewall software
- Ex: Network card adapter, firewall, OS services, HTTP server
Kiosk

- An application that runs full screen without allowing the regular users to terminate the application, run applications, control the OS
- Single purpose for the machine that the software is running on – dedicated machine
- Targets novices and maybe practitioners
- Ex: ATM, Pay Telephone, Directions Kiosk, Subway Card Machine, Airport Self Check-in, Brochureware Website!
Application UI Process
To effectively design for the user you must:

• Understand the users abilities
• Understand the users goals
• Understand the tasks the user does to achieve those goals
• Understand the frequency of expected use by the user
• What are the users previous experience and training
• What are the users expectations
• Involve the user at various stages in the design process
User’s Abilities

• Their experience with computer interfaces
• Novice, Practitioner, Expert or Guru in the business domain the software will serve
• With the use of any UI frequently enough the user will become more adept at the UI and need different features that they did as a beginner with the UI.
Users

• Understand who the users are
  – Type – Novice, Practitioner, Expert, Guru
  – Frequency of use – continual, frequent, occasional, once, ?
  – Tolerance of a learning curve – none, a little, expected

• Understand user’s goals

• Understand what tasks achieve those goals
Determine the application flavor

• Sovereign, Transient, Parasitic, Daemonic or Kiosk (S, T, P, D, or K)

• A single application may support multiple execution flavors
  – ex: a sovereign application like word that can run as a transient app to enable printing a document from windows explorer

• Focus on one flavor at a time. Possibly have multiple separate teams that target the needs of one flavor. Later reconcile them together.
The Mental Model

• Determine what a users mental model **SHOULD** be.
• Keep that in mind while designing the UI mockups and prototypes
• The implementation model should be delayed until later in the design, avoid having the implementation dictate the UI (and the mental model of the User)
The Model

- Different user types may have different mental models and different goals.
- Different **aspects** of the UI may target different user **types** i.e. rich menus for novices, toolbars for experts and a command line feature for gurus.
- Don’t try to build a single UI for all possible users. Target the **80%** of all users (or those that will pay the most for the software, maybe).
Sovereign Application Example

Wordprocessor
Design aspects

• Define the User Types (Abstracted from Concrete Users)
  – N, P, E or G? (for Wordproc P and E)

• How will the users generally want to use the app i.e. app flavors desired?
  – Define likely App Flavor for users
  – S,T, P, D, or K? (For wordproc S or K)

• Based on workflows and user experience level/tolerance of complexity – determine likely app UI design
  – SDI, MDI, Project, Workbook, Dialog, Wizard? (For wordproc SDI, MDI, Workbook)

• Revisit these decisions later, see if they still make sense.

• Pick most likely of each NOT all possibilities, at once.
Concrete Goals

• Review the high level business goals/activities/tasks (verbs)
  – Be Famous Author
    • Write Novel
    • Get Novel Published
    • Do Book Tour
    • Get on Oprah
  – Send Out Newsletter to Customers
  – Get PhD
    • Take Classes
    • Do Research
    • Write Dissertation
    • Defend Dissertation
    • Graduate from NYU
  – Write Mom a Letter
    • Write Letter
    • Print Letter
    • Mail Letter
Abstract Goals

• Usually a phrase with high level business actions and entities

• Ex:
  – Print things
  – Receive things
  – Send things
Abstracted Terms

• Analyze and Abstract these a little, see what you get.
  – Documents
  – Diagrams
  – Typefaces, Fonts and Styles
  – Manage Lifecycle – Create, Edit, Print, Save, Update
  – Send and Receive via Email
Abstract User Goals

• **Manage Documents** (manage the lifecycle of a document)
  – Read a Document
  – Create a Document
  – Save a Document
  – Print a Document
  – Change an Existing Document (edit)

• User term might be “write a document”
Tasks

• A **Process** is done to achieve a goal.
• A process is **one or more** tasks done to achieve a goal.
• A task consists of one or more **activities** done in a defined **sequence** with possibility of **conditional paths** between activities and **looping back** allowed.
• A successfully completed task should accomplish a part of a user goal

• **Ex Tasks:**
  – Write Novel
  – Write Thesis
  – Read Email
  – Write and Send Email
Activities

• An action (verb) on some business entity or part of a business entity
• Usually expressed as a verb phrase.
• There may be conditional paths that connect together activities.
• Output of one activity is often the input to the next activity.
• May be part of one or more tasks to achieve a goal.
High Level Activities

• Often maps to the **high level** items on an applications **menus** or **buttons** that spawn dialogs

• Ex high level activities:
  – Create a blank document
  – Open an existing document
  – Enter text into a document
  – Enter images into a document
  – Enter charts into a document
  – Combine parts of one document with another
  – Proof the document for spelling and grammar
  – Format and Print an existing document
Lower Level Activities

- Lower level activities often change the **state** of a business entity
- Often maps to radio buttons, check boxes, sliders, drop downs, toggle buttons, button cons, tool palettes, mode changing buttons, popup property menus
- Ex low level activities:
  - Enter and manipulate text position - type
  - Editing text – cut, copy, paste, move, delete
  - Formatting text - change text size, color, typeface, style, …
  - Change layout - of the document paragraphs, tabs, spacing, …
  - Formatting the document – landscape, portrait, paper size
Business Entities and Subentities

• Generally the target of the actions of an application
• Nouns / noun phrases
• May be containers of other instances of the same object or smaller grained objects

• Ex:
  – Novel contains TOC, preface, chapters, index
  – Folder that contains folders and/or files
  – Doc that contains pages, pages contain paragraphs, paragraphs contain sentences, Sentences contain words, spaces and punctuation. Words are one or more characters and the display attributes of those characteristics.
Parts of a document

• Doc
  – 0,1 or more pages
  – Page
    • Paragraphs
      – Sentences
        » Words
        Characters
    • Page Number
    • Header
    • Footer

• Each part of a document (Doc, Page, Para, …) has the following aspects
  – Character Code (what letter, symbol, or punctuation)
  – Type face and size
  – Typeface Style – normal, italics, Bold, Underline, Super, Sub,
  – Foreground Color
  – Background Color
Sketch the **object** hierarchy

- Review the business entities – select the appropriate ones
  - Docs
  - Pages
  - Paragraphs
  - Words
  - Characters
Sketch the **action** hierarchy

- **Document Related Stuff**
  - Create
  - Delete
  - Edit
  - Print
  - Save
  - Rename
  - Email
- **Select Part of Document**
- **Change Part Of Document**
- **Change the view of the document**
- **Help with the application**
Sovereign Apps

• Own the desktop
• Expert users (and novices)
• Support multiple tasks generally
SDI / MDI / Workbooks

- Powerful and complex set of tasks
- Not time sensitive / knowledgeable
- Multiple open instances
- Multiple views of a object
- Drag and drop between instances?
- Workflow needs?
Basic menu design
What would novices do?

• Ex:
  – Create Docs
  – Edit Docs – enter text, change font
  – Print Docs

• Place on highest level menus
What would be on the menus?
(based on novices needs)

- **Document**
  - Create (New)
  - Edit (Open)
  - Print
  - Email
- **Select (Edit)**
  - All
  - Paragraph
  - Sentence
  - Word
  - Character
  - Header
  - Footer
- **Manipulate**
  - Cut (to clipboard)
  - Copy (to clipboard)
  - Paste (clip board)
  - Move (??)
  - Print
- **Format (Selected Area)**
  - Typeface & Size
  - Style
  - FG Color
  - BG Color
- **View**
  - Zoom Current
  - Select Which to View
  - Show Multiple Docs
- **Help**
  - Help On
  - About the application
What would practitioners do?

• All that a novice would
• Other activities:
  – Lots for extra formating stuff
  – fonts, typefaces, colors, BG, header, footer,
    page margins, superscript, subscript, strikethrough, …
• Place on sub menus and toolbars
What buttons should be exposed for the practitioners?

- New
- Print
- Format Text
- Cut & Paste
- Spell check & Grammar check
- Create a chart
- Draw objects
What would experts do?

• Ex:
  – Cut, Copy, Paste
  – Save frequently
  – Search for text
  – Insert Tables
  – Insert Hyperlinks
  – Look at help
  – Undo

• Make accelerator and Mnemonic keys
• Make available through context sensitive, right-click pop up menus
What accelerator keys should be defined for experts?

• Ctrl Keys
  – Ctrl + N – New Doc
  – Ctrl + S – Save
  – Ctrl + O – Open
  – Ctrl + P – Print
  – Ctrl + X – cut
  – Ctrl + V – paste (move)
  – Ctrl + C – copy
  – Ctrl + Z – undo
  – Ctrl + F – Find
  – Ctrl + H – find and replace

• Top level menus as Alt keys
  – Alt + D – Doc
  – Alt + S – Select
  – Alt + M – Manipulate
  – Alt + F – Format
  – Alt + V – View

• Function Keys
  – F1 – Help
  – F7 – Spell Check
An Example Wordprocessor
An Example Wordprocessor 2
Usability = User + Ability to use

What you see may be vastly different than what you want.
What is usability

• A gauge of how well a UI (and the application behind it) support the needs of the user.
  – Intuitiveness / predictability
  – Ease of use
  – Power of use
  – Managing complexity
Productivity and Return on Investment

We measured users' task performance for sixteen common employee tasks across the fourteen intranets. As one might expect, usability varied widely, and some designs supported much faster performance than others.

When salaries and overhead costs were taken into account, we calculated that a company with one of the least usable designs in our study would spend $3,042 per employee annually to cover time spent on the sixteen tasks we measured.

In contrast, the average company would spend $2,069 per employee per year, and a company that was among the best in usability would spend $1,563.

The total annual cost of intranet use can be estimated by multiplying by the number of employees. For companies with 10,000 intranet users, the annual costs are as follows:

- High usability (among the best 25%): $15.6 million
- Average usability: $20.7 million
- Low usability (among the worst 25%): $30.4 million

Clearly, the biggest productivity gains are found by moving a company with bad intranet usability to a design with average usability. But the gains for improving from average to good usability are significant as well.

Intranets typically support mission-critical applications and other specialized tasks that differ between companies and cannot be included in a cross-company comparison. We assume that these company-specific tasks account for the same amount of intranet usage as the sixteen general tasks we studied. Thus, the full costs are likely twice as big as the ones listed in the table.
Usability and Senior Citizens

Usability Metrics Twice as High for Non-Seniors

In the quantitative study, we asked users in both groups to perform the same four tasks:

- Fact-finding
- Buying an item
- Retrieving information
- Comparing and contrasting

The following table shows the measurements of four usability attributes averaged across the four tasks.

<table>
<thead>
<tr>
<th></th>
<th>Seniors (65+ Years)</th>
<th>Control Group (21-55)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate (task completed correctly)</td>
<td>52.9%</td>
<td>78.2%</td>
</tr>
<tr>
<td>Time on Task (min:sec)</td>
<td>12:33</td>
<td>7:14</td>
</tr>
<tr>
<td>Errors (erroneous actions per task)</td>
<td>4.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Subjective Rating (scale: 1=low, 7=high)</td>
<td>3.7</td>
<td>4.6</td>
</tr>
<tr>
<td>Overall Usability (normalized geo. mean)</td>
<td>100%</td>
<td>222%</td>
</tr>
</tbody>
</table>

The differences between seniors and the control group are all highly significant.
Usability and disabled web users

© http://www.useit.com/alertbox/20011111.html

The control group’s success rate was 78%, which is considerably higher than the success rates we’ve found in most of our other studies. Usually, success rates in Web usability studies range from 40% to 60%, so the current average of 78% for the control group probably indicates that the test tasks were a little bit easier than those in our other studies. At the same time, the tasks were not overly easy, as many members of the control group did not complete tasks, and the average task performance time was more than seven minutes.

The following table compares the control group’s performance with an average of the two user groups with disabilities. In this table, we consider the usability measured for users with disabilities to be 100%; the usability experienced by the control group has been normalized relative to that level. Bigger numbers indicate better usability (higher success, faster task completion, fewer errors, and greater satisfaction).

<table>
<thead>
<tr>
<th>Relative score for users without disabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate</td>
</tr>
<tr>
<td>Task Performance</td>
</tr>
<tr>
<td>Error Avoidance</td>
</tr>
<tr>
<td>Subjective Rating</td>
</tr>
<tr>
<td>Overall Usability (geometric mean)</td>
</tr>
</tbody>
</table>

As the table shows, the Web’s current usability is about three times better for users without disabilities than it is for users with disabilities. This is a huge difference; the numbers are much bigger than we typically see in usability.
A few guidelines
© http://www.useit.com/alertbox/20001112.html

Designs to Avoid

Drop-down menus do have their advantages. First, they conserve screen space. They also prevent users from entering erroneous data, since they only show legal choices. Finally, because they are a standard widget (even if an unpleasant one), users know how to deal with a drop-down menu when they encounter it.

Despite these advantages, Web usability would increase if designers used drop-downs less often. To that end, here are some examples of designs to avoid:

- **Interacting menus**, wherein the options in one menu change when users select something in another menu on the same page. Users get very confused when options come and go, and it is often hard to make a desired option visible when it depends on a selection in a different widget.
- **Very long menus** that require scrolling make it impossible for users to see all their choices in one glance. It’s often better to present such long lists of options as a regular HTML list of traditional hypertext links.
- **Menus of state abbreviations**, such as for U.S. mailing addresses. It is much faster for users to simply type, say, “NY,” than to select a state from a scrolling drop-down menu. Free-form input into fields with restricted options does require data validation on the backend, but from a usability perspective it’s often the best way to go. (This is guideline #178 for e-commerce usability because of the many errors we observed in check-out forms with state drop-downs.)
- **Menus of data well known to users**, such as the month and year of their birth. Such information is often hardwired into users’ fingers, and having to select such options from a menu breaks the standard paradigm for entering information and can even create more work for users, as the following example shows.
• In one study – 56% e-commerce users failed because of site design to buy anything.
How do you test for usability?

• Rate users and their relative qualities
• Watch different level / type users using the software
• Ask users to “think out loud”
• Time performance for certain tasks across multiple users.
• Record what they do during the timed trials.
• Often “usability” really means ‘learnability” of your software.
Frequency of Expected Use

• How often will the user be using the UI?
  – Kiosk in a mall – most users infrequent
  – Word processor – if you’re a writer probably quite frequently, if you are a graphic artist less frequently.
  – NetBeans – if you take more courses than this that use NetBeans, you will likely become an expert even though now you are likely a novice.
  – Website – many info sites you will not use very frequently, others you may use everyday.
Involve the User

- Observation - Watch users doing what they do.
- Interviews - Ask them about what they do and why they do it.
- Usability Testing - Have them use the UI, not just tell you about what they might do with it.
- Repeat these steps during the process several times, especially the usability testing.
Users

• Group actual users into stereotypes i.e. abstract user types ex:
  – Manager
  – Associate Sales Representative
  – Senior Sales Rep
  – System Administrator
Myths About Users
The “average user”

- There is no average user.
- Users change over time and use as to what type of user they are.
  - Use the UI more you become more expert
  - As you age your capabilities change – colors, reaction time, hearing, dexterity, mental acuity
  - Users expectations change as they see other UIs.
Designers Intuition

• Designers aren’t good models of real users
• Designers may organize information in ways that other users wouldn’t. They have different cognitive styles