Design Principles

Guidelines, Principles, and Theories

This material has been developed by Georgia Tech HCI faculty, and continues to evolve. Contributors include Gregory Abowd, Jim Foley, Diane Gromala, Elizabeth Mynatt, Jeff Pierce, Colin Potts, Chris Shaw, John Stasko, and Bruce Walker. Modified by Valerie Summet, 2011. Permission is granted to use with acknowledgement for non-profit purposes. Last revision: January 2011.
Outline

• Introduction/Context
• Guidelines / Principles (Schneidermann, et al.)
  – 8 Golden Rules
• Guidelines / Principles (Dix, et al.)
  – Learnability
  – Flexibility
  – Robustness
• Styleguides (Friday)
What are Design Principles?

• Rules of thumb to help with the design process

• AKA Design Guidelines, Design Rules
Design Principles - Overview

• Become obvious to user of poorly-designed UIs
• Are easy to ignore
• Apply at multiple levels of design
• Are neither complete nor orthogonal
  – Can all be “broken”, often to satisfy another rule
• Have underpinnings in psychology or experience or common sense
Many Sets of Design Principles

- Shneiderman, *Designing the User Interface*
- Dix, Finlay, Abowd, Beale, *Human-Computer Interaction*
- Foley et al, *Computer Graphics: Principles and Practice*
- And many more - including in styleguides, discussed later
Schneiderman, et al.

• 8 “Golden Rules” of Interface Design
  – 2.3.4
  – 2.3.5
  – 2.3.6
  – pgs. 70-78

• As you read, compare/contrast to class material from Dix, et al.
8 Golden Rules of Interface Design

• Strive for consistency
• Cater to universal usability / plasticity
• Offer informative feedback
• Design actions to yield closure
• Prevent errors
• Permit easy reversal of actions
• Support internal locus of control
• Reduce short-term memory load
Dix, et al.

- Guidelines / Principles
  - Learnability
    - Support for learning for users of all levels
  - Flexibility
    - Support for multiple ways of doing tasks
  - Robustness
    - Support for recovery
“Every designer wants to build a high-quality interactive system that is admired by colleagues, celebrated by users, circulated widely, and imitated frequently.”

(Shneiderman, 1992, p.7)

…and anything goes!…
Concepts, Principles, Guidelines

• No “cookbooks”
• No simple, universal checklists
• There are many concepts, principles, and guidelines
• **Understand** the higher level **principles** that apply across situations, display types, etc.
• **Implement** the standards and **guidelines**

  …a few details…
Levels of Consideration

- Meta-display level
  - Apply to the whole system, across media & across displays
  - Focus on this in Basic Layout Stage

- Display Layout
  - Apply to groups of elements in a display
  - Focus on this in Prototyping and Redesign

- Element level
  - Details about specific parts of a display
  - Colors, sound attributes, symbols
UI Design Principles

• 8 Golden Rules of Interface Design

• Categories
  – Learnability
    • Support for learning for users of all levels
  – Flexibility
    • Support for multiple ways of doing tasks
  – Robustness
    • Support for recovery

• Always think about exceptions, suitability
1. Learnability Principles

• Ease with which new users can begin effective interaction and achieve maximal performance
  – Predictability
  – Synthesizability
  – Familiarity
  – Generalizability
  – Consistency
1.1 Predictability

• I think that this action will do…. 

• Operation visibility
  Can see avail actions
  – e.g. menus vs. command shell
  – Grayed menu items provide context

• Mental models can help
Mental Models - Aid
Predictability

• Mental models are not always right
• Two Classes:
  – Functional model
    • Stimulus - response
    • “Press the accelerator once, then turn the key”
    • At surface or superficial level
  – Structural (or Operational) model
    • Deeper sense of why it happens, not just what happens
    • “Press the accelerator to engage the automatic choke on a carburetor”
Another example - directions...

• Functional model
  – Give turn-by-turn directions:
    • “East on Ponce, go past the Avondale MARTA station, then go left at the third set of lights”
Another example - directions...

- Structural model
  - Provide higher-level understanding
  - “Using map, go to the Dekalb Farmer's Market”
  - Can augment with “East on Ponce, turn left just after Harry’s in a Hurry, then go to the third set of lights”
1.2 Synthesizability

• Support for user in assessing the effect of past operations on current system state

  – Moving a file in UNIX shell vs. Mac/Windows
  – Is same feedback needed for all users, all apps?

• “Gulf of Interpretation” - Don Norman

• Understanding what just happened

Can a user figure out what caused this error?
1.3 Familiarity

• Does UI task leverage existing real-world or domain knowledge?
  – Really relevant to first impressions
  – Use of metaphors
    • Potential pitfalls
  – Are there limitations on familiarity?
Metaphors at the UI - What

• *Metaphor* - Application of name or descriptive term to another object which is not literally applicable
  – Use: Natural transfer - apply existing knowledge to new, abstract tasks
    • “Checkout” for online shopping
    • Desktop metaphor
    • Magnifying glass
  – Problem: May introduce incorrect mental model
    • Macintosh - dragging CD to trash can to eject
1.4 Generalizability

• Can knowledge of one system/UI be extended to other similar ones?
  – Example: cut & paste in different applications

  – Does knowledge of one aspect of a UI apply to rest of the UI?

  – Aid: UI Developers guidelines
1.5 Consistency

- Section 2.4.3 (p 85-86)
- Likeness in behavior between similar tasks/operations/situations/terminology
  - Interaction sequences
    - Quicken on Mac – Option-P prints check, not current document
  - Output
    - Dialogue box always has a close button
  - Screen layout
    - Menu items always in same place - leverage “muscle memory”
- Is this always desirable for all systems, all users?
Consistency (cont’d)

• Avoid special cases and special rules
• Supports generalization by user, avoids frustration
• For command line systems - consistent syntax
• For GUIs - consistent presentation
(In)Consistency Example

• For a graphics/drawing program that uses a CSO (Currently-Selected Object)
  – Consistent:
    • Create a new primitive, it becomes the CSO
  – Inconsistent:
    • Duplicate a primitive, the old primitive remains as CSO
(In)Consistency Example - Macintosh

Drag a file icon to:  
Folder on same physical disk  
Folder on another physical disk  
Different disk  
Trash can

Result:  
File is moved to folder  
File is copied there  
File is copied there  
File is discarded
Design Principles

Principles, Guidelines, Theories

This material has been developed by Georgia Tech HCI faculty, and continues to evolve. Contributors include Gregory Abowd, Jim Foley, Diane Gromala, Elizabeth Mynatt, Jeff Pierce, Colin Potts, Chris Shaw, John Stasko, and Bruce Walker. Modified by Valerie Summet, 2011. Permission is granted to use with acknowledgement for non-profit purposes. Last revision: January 2011.
Sum up: Learnability

• Ease with which new users can begin effective interaction and achieve maximal performance
  – Predictability
  – Synthesizability
  – Familiarity
  – Generalizability
  – Consistency

• Questions?
2. Flexibility Principles

• Multiplicity of ways that users and system exchange information
  – “Dialog” Initiative
  – Multithreading
  – Task migratability
  – Substitutivity
  – Customizability

• Book calls this “plasticity”
2.1 “Dialog” Initiative

- Interaction between system and user is a “dialog”
  - Turn-taking amongst participants (user & system)
  - Who has initiative/control of dialog?
- Not hampering the user by placing constraints on how dialog is done
  - User pre-emptive
    - User initiates actions
    - More flexible, generally more desirable
  - System pre-emptive
    - System does all prompts, user responds
    - Sometimes necessary
2.2 Multithreading

- Allowing user to perform more than one task at a time

- Two types
  - Concurrent
    - Input goes to multiple tasks simultaneously
    - Ex: many gaming systems
  - Interleaved
    - Many tasks, but input goes to one at a time

- User bad, system good
2.3 Task Migratability

• Ability to move performance of task to the entity (user or system) that can do it better
  – Ex: Spell-checking, safety controls in plant

  – For what kinds of tasks should the user be in control?
    • See Box 2.3, p. 74
2.4 Substitutivity

• Flexibility in details of operations
  – Allow user to choose suitable interaction methods
  – Allow different ways to
    • perform actions, specify data, configure
  – Allow different ways of presenting output
    • to suit task & user
Substitutivity Ex - file viewer
2.4 Substitutivity

• Drafting & page layout systems
  – Indicate positions with cursor
  or
  – By typing in coordinates
• Point at spreadsheet cell vs enter name
• Give temperature via slider or by typing
• Other examples???
2.5 Customizability

- Ability of user to modify interface
  - By user
    - Is this a good thing?
  - By system
    - Is this a good thing?
Customizability Ex - Toolbars in Powerpoint

• Pros and cons of using?
Sum Up: Flexibility Principles

• Multiplicity of ways that users and system exchange information
  – “Dialog” Initiative
  – Multithreading
  – Task migratability
  – Substitutivity
  – Customizability

• Questions?
3. Robustness Principles

• Supporting user in determining successful achievement and assessment of goals
  – Observability
  – Recoverability
  – Responsiveness
  – Task Conformance
3.1 Observability

• Can user determine internal state of system from what she perceives?
  – Browsability
    • Explore current state (without changing it)
    • Reduces memory load
    • But don’t overwhelm user with information either
  – Reachability
    • Navigate through observable states
  – Persistence
    • How long does observable state persist?
  – Observability also aids learnability
3.1 Observability - Role of Feedback

- Feedback helps create observability
- Feedback taxonomy (generally don’t need all of these)
  - “I understand what you have asked me to do”
  - “I am doing what you have asked me to do”
    - “And it will take me this much longer”
    - “And here are other things to do while we wait” (busy interval as opposed to idle interval)
    - “And here are some intermediate results to keep you happy until I am done
  - “All done, what’s next?”
3.1 Observability - Forest (Big Picture) + Trees (Details)

- Represents overall drawing
- Represents portion of drawing seen in work area
- Overall display area
- Menus, feedback, tools, etc.
- Work area, showing part of a larger drawing
3.1 Observability – Acrobat Reader

Acrobat Reader with ToC to give context

Forest is the bookmarks, tree is the single page
3.1 Observability - How did I Get Here?

- Example: Cascading menus
3.1 Observability - How Did I Get Here?

- Ex: Google searches
- Tree showing history of queries

- Presidents elected before age 50
  - Democratic
    - from East of Mississippi
    - served two terms in office
    - graduate degree
    - from farm family
3.1 Observability - Where can I go?

Lots of “next steps” presented for user.
3.1 Observability - Memory Load

• Use as a tool to decrease user memory load

• Also to avoid mis-steps
  – Gray-out disabled menu items
  – Indicate type of input (alpha, numbers) expected
3.2 Recoverability

- Ability to take corrective action upon recognizing error
  - Forward recovery
    - Ability to fix when we can’t undo
  - Backward recovery
    - Undo previous error(s)
  - Abort operation underway
    - Only makes sense if is a slow operation

- Encourages experimentation & exploration (hence learnability) by reducing cost of making mistakes

- See 2.3.5 (p. 71-73)
3.2 Do Not Set the User Up

• Make it hard for the user to make errors
  – Instead of allowing them to make error and then saying “tsk, tsk”
  – Remember the DreamHost case study?
• Gray out disabled menu items
• Ask for confirmation of major actions
3.2 Prevent Errors

• Don’t let the user do something that will lead to an error message
3.2 Help User Avoid Errors

• For command language applications, create matching “end” token when “begin” token is typed
  - ( → )
  - <title> → </title>
  - /* → */
  - “ → ”
3.3 Responsiveness

- Users perception of rate of communication with system (not always right)
  - “JND”: Just Noticeable Difference
- Response time: Time for system to respond in some way to user action(s)
  - Ex: Mars Rovers - 3-20 minutes
- Response OK if matches user expectations
- Once user enjoys fast response, is hard to go back to slower response
  - Dial-up vs DSL or Cable Modem
3.3 Responsiveness

- Response to motor actions
  - Keyboarding, mouse movement - less than 100 msec.
  - Rich Human Factors literature on this
- Consistency is important - experimental result
  - Users preferred longer but more consistent response time
  - Times that differed 10% to 20% were seen as same
- Sometimes argued that too fast is not good
  - Makes user feel like they need to do something quickly to keep up with computer?
3.4 Task Conformance

• Does system support all tasks user wishes to perform in expected ways?
  – Task completeness
    • Can system do all tasks of interest?
  – Task adequacy
    • Can user understand how to do tasks?
  – Does it allow user to define new tasks?
    • Extensibility
Using the Principles

• In doing design and implementation of your project, revisit this list
• Assess your design against these usability principles
Sum Up: Robustness Principles

• Supporting user in determining successful achievement and assessment of goals
  – Observability
  – Recoverability
  – Responsiveness
  – Task Conformance

• Questions?
Styleguides

“We want principles, not only developed ... but applied”
- Horace Mann, 1867

- Codify many design principles for a particular look and feel
  - Mac OS, Windows, iOS, Palm, Blackberry
- Developed in concert with software toolkits, but go beyond toolkit
- See Sections 2.2 (p 57-62), 3.3.2 (p 104-107)
Typical TOC - MAC OS X

Introduction to the Apple Human Interface Guidelines
  What Are the Mac OS X Human Interface Guidelines?
  Who Should Read This Document?
  Organization of This Document
  Conventions Used in This Document
  See Also

Part I: Fundamentals
Human Interface Design
  Human Interface Design Principles
  Keep Your Users in Mind
The Development Process
  Design Decisions
  Managing Complexity
  Extending the Interface
  Involving Users in the Design Process

Part II: The Macintosh Experience
First Impressions
  Packaging
  Installation
  General Installer Guidelines
  Setup Assistants
Mac OS X Environment
  The Finder
  The Dock
  The File System
  Multiple Users
  Remote Log In
  Assistive Technologies
  Networking
  Application Services
  Displays
  The Always-On Environment

Using Existing Technologies
  Providing User Assistance
  Internationalizing Your Application
  Storing Passwords
  Printing
  Choosing Colors
  Setting Fonts and Typography Characteristics
  Selecting Attributes Associated With People
  Speech Technologies

Part III: The Aqua Interface
User Input
  The Mouse and Other Pointing Devices
  The Keyboard
  Selecting
  Editing Text
Drag and Drop
  Drag and Drop Overview
  Drag and Drop Semantics
  Selection Feedback
  Drag Feedback
  Destination Feedback
  Drop Feedback
  Clippings
Text
  Fonts
  Style
Icons
  Icon Genres and Families
  Icon Perspectives and Materials
  Conveying an Emotional Quality in Icons
  Suggested Process for Creating Aqua Icons
  Tips for Designing Aqua Icons
Cursors
  Standard Cursors
  Designing Your Own Cursors
Drag and Drop Overview

Ideally, users should be able to drag any content from any window to any other window that accepts the content’s type. If the source and destination are not visible at the same time, the user can create a clipping by dragging data to a Finder window; the clipping can then be dragged into another application window at another time.

Drag and drop should be considered an ease-of-use technique. Except in cases where drag and drop is so intrinsic to an application that no suitable alternative methods exist—dragging icons in the Finder, for example—there should always be another method for accomplishing a drag-and-drop task.

The basic steps of the drag-and-drop interaction model parallel a copy-and-paste sequence in which you select an item, choose Copy from the Edit menu, specify a destination, and then choose Paste. However, drag and drop is a distinct technique in itself and does not use the Clipboard. Users can take advantage of both the Clipboard and drag and drop without side effects from each other.

A drag-and-drop operation should provide immediate feedback at the significant points: when the data is selected, during the drag, when an appropriate destination is reached, and when the data is dropped. The data that is pasted should be target-specific. For example, if a user drags an Address Book entry to the “To” text field in Mail, only the email address is pasted, not all of the person’s address information.

You should implement Undo for any drag-and-drop operation you enable in your application. If you implement a drag-and-drop operation that is not undoable, display a confirmation dialog before implementing the drop. A confirmation dialog appears, for example, when the user attempts to drop an icon into a write-only drop box on a shared volume, because the user does not have privileges to open the drop box and undo the action.
Styleguides

• Collection of Styleguides and Guidelines
  – From experiencedyanmics.com
  – Not exhaustive but comprehensive
What Makes a Good Design?

• Functionality
• Speed & efficiency
• Reliability, security, data integrity
• Standardization, consistency
• **USABILITY !**
A Philosophy

The human user of any system is the focus of the design process.
Planning and implementation is done with the user in mind.
The system is made to fit the user, not the other way around.
Good Design Does NOT Mean…

✗ NOT just applying checklists and guidelines

✗ NOT using oneself as the model user

✗ NOT just common sense

✗ Knowing how to design a fire alarm so it will be heard over background noise is not something we all know.
Good Design Does Means...

✔ Systems are built for **humans**; must be designed for the user
✔ Recognize individual differences; appreciate design implications of these **human factors**
✔ Recognize the design of things, procedures, etc., influences human behavior and well-being
✔ Emphasize empirical data & evaluation
✔ Rely on the scientific method
✔ Things, procedures, environments, and people do not exist in isolation