Chapter Six: Design Rules and Implementation support

Contents

Design Rules

- Designing for maximum usability is the goal of interactive systems design.
- ~~Abstract principles offer a way of understanding usability in a more general sense, especially~~ if we
- can express them within some coherent catalog.
- Design rules in the form of standards and guidelines provide direction for design, in both general and more concrete terms, in order to enhance the interactive properties of the system.
- The essential characteristics of good design are often summarized through 'golden rules' or heuristics.
- Design patterns provide a potentially generative approach to capturing and reusing design knowledge.

Design rules

- Design rules (or usability rules) are rules that a designer can follow in order to increase the usability of the system/product e.g., principles, standards, guidelines.

- Designing for maximum usability
  - the goal of interaction design
- Principles of usability
  - general understanding
- Standards and guidelines
  - direction for design
- Design patterns
  - capture and reuse design knowledge

Types of design rules

principles

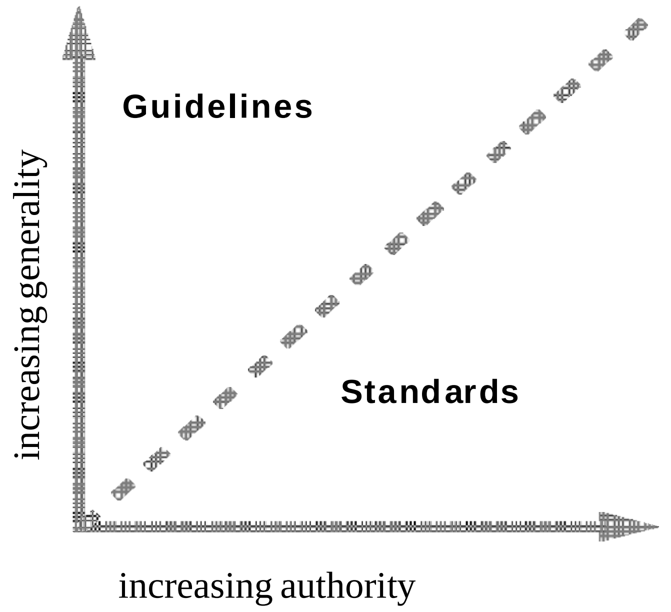
- abstract design rules
- low authority
- high generality
- (e.g. interface should be easy to navigate)

standards

- specific design rules
- high authority
- limited application
- (e.g. use color RGB)

guidelines

- lower authority
- more general application
- Can guide/advise on how achieve a principle
- (e.g. use color to highlight links)



Principles to support usability

Learnability

- the ease with which new users can begin effective interaction and achieve maximal performance

Flexibility

- the multiplicity of ways the user and system exchange information

Robustness

- the level of support provided the user in determining successful achievement and assessment of goal- directed behaviour

Standards

- set by national or international bodies to ensure compliance by a large community of designers
- standards require sound underlying theory and slowly changing technology
- hardware standards more common than software high authority and low level of detail
- ISO 9241 defines usability as effectiveness, efficiency and satisfaction with which users accomplish tasks

Guidelines

- more suggestive and general
- many textbooks and reports full of guidelines
- abstract guidelines (principles) applicable during early life cycle activities

  detailed guidelines (style guides) applicable during later life cycle activities
- understanding justification for guidelines aids in resolving conflicts

Golden rules and heuristics

“Broad brush” design rules Useful check list for good design

Better design using these than using nothing! Different collections e.g.

- Nielsen’s 10 Heuristics
- Shneiderman’s 8 Golden Rules
- Norman’s 7 Principles

Shneiderman’s 8 Golden Rules

1. Strive for consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialogs to yield closure
5. Offer error prevention and simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Norman’s 7 Principles

1. Use both knowledge in the world and knowledge in the head.
2. Simplify the structure of tasks.
3. Make things visible: bridge the gulfs of Execution and Evaluation.
4. Get the mappings right.
5. Exploit the power of constraints, both natural and artificial.
6. Design for error.
7. When all else fails, standardize.

HCI design patterns

An approach to reusing knowledge about successful design solutions

Originated in architecture: Alexander

A pattern is an invariant solution to a recurrent problem within a specific context.

Examples

- Light on Two Sides of Every Room (architecture)
- Go back to a safe place (HCI)

Patterns do not exist in isolation but are linked to other patterns in languages which enable complete designs to be generated

HCI design patterns (cont.)

Characteristics of patterns

- capture design practice not theory
- capture the essential common properties of good examples of design
- represent design knowledge at varying levels: social, organisational, conceptual, detailed
- embody values and can express what is humane in interface design
- are intuitive and readable and can therefore be used for communication between all stakeholders
- a pattern language should be generative and assist in the development of complete designs.

Implementation support

- Programming tools for interactive systems provide a means of effectively translating abstract designs and usability principles into an executable form. These tools provide different levels of services for the programmer.
- Windowing systems are a central environment for both the programmer and user of an interactive system, allowing a single workstation to support separate user-system threads of action simultaneously.
- Interaction toolkits abstract away from the physical separation of input and output devices, allowing the programmer to describe behaviors of objects at a level similar to how the user perceives them.
- User interface management systems are the final level of programming support tools, allowing the designer and programmer to control the relationship between the presentation objects of a toolkit with their functional semantics in the actual application.

Implementation support

programming tools

- levels of services for programmers

windowing systems

- core support for separate and simultaneous user-system activity

programming the application and control of dialogue interaction

toolkits

- bring programming closer to level of user perception

user interface management systems

- controls relationship between presentation and functionality

Introduction

How does HCI affect of the programmer?

Advances in coding have elevated programming

- ❑ hardware specific
- ❑ interaction-technique specific

Layers of development tools

- ❑ windowing systems
- ❑ interaction toolkits
- ❑ user interface management systems

Elements of windowing systems

Device independence

programming the abstract terminal device drivers

image models for output and (partially) input

- ❑ pixels
- ❑ PostScript (MacOS X, NextStep)
- ❑ Graphical Kernel System (GKS)
- ❑ Programmers' Hierarchical Interface to Graphics (PHIGS)

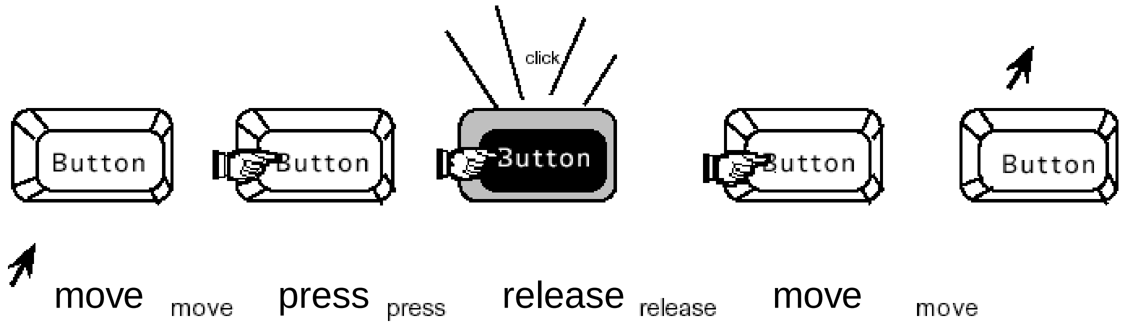
Resource sharing

- ❑ achieving simultaneity of user tasks
- ❑ window system supports independent processes isolation of individual applications

Using toolkits

Interaction objects

- input and output intrinsically linked



Toolkits provide this level of abstraction

- programming with interaction objects (or
- techniques, widgets, gadgets)
- promote consistency and generalizability
- through similar look and feel
- amenable to object-oriented programming

Interfaces in Java

Java toolkit – AWT (abstract windowing toolkit)

Java classes for buttons, menus, etc. Notification based;

- AWT 1.0 – need to subclass basic widgets
- AWT 1.1 and beyond -– callback objects

Swing toolkit

- built on top of AWT – higher level features
- uses MVC architecture (see later)

Chapter Seven and eight : Evaluation Techniques and user support

Contents

- ❑ Evaluation Techniques
- ❑ User support
  - ❖ Introduction
  - ❖ Requirements of user support
  - ❖ Approaches to user support
  - ❖ Designing user support systems

Evaluation

- ❑ tests usability and functionality of system
- ❑ occurs in laboratory, field and/or in collaboration with users
- ❑ evaluates both design and implementation
- ❑ should be considered at all stages in the design life cycle

Goals of Evaluation

- ❑ assess extent of system functionality
- ❑ assess effect of interface on user
- ❑ identify specific problems

Evaluating Designs

- ❑ Cognitive Walkthrough
- ❑ Heuristic Evaluation
- ❑ Review-based evaluation

Cognitive Walkthrough

- ❑ The cognitive walkthrough is a usability evaluation method in which one or more evaluators work through a series of tasks and ask a set of questions from the perspective of the user.
- ❑ The focus of the cognitive walkthrough is on understanding the system's learnability for new or infrequent users
- ❑ Evaluates design on how well it support user in learning task
- ❑ Usually performed by expert in cognitive psychology
- ❑ Experts walk through design to identify potential problem using psychological principle

The Four Questions to be Asked during a Cognitive walkthrough:

- ❑ Will the user try and achieve the right outcome?
- ❑ Will the user notice that the correct action is available to them?
- ❑ Will the user associate the correct action with the outcome they expect to achieve?
- ❑ If the correct action is performed; will the user see that progress is being made towards their intended outcome?

Heuristic Evaluation

- ❑ A heuristic evaluation is a usability inspection method for computer software that helps to identify usability problems in the user interface (UI) design.
- ❑ It specifically involves evaluators examining the interface and judging its compliance with recognized usability principles (the heuristics).
- ❑ Usability criterial (heuristics) are identified
- ❑ Design examined by experts to see if these are violated

Review-based evaluation

- ~~**Review-based evaluation** is an expert-based **evaluation** method that relies~~ on experimental results and empirical evidence from the literature (for instance from psychology, HCI, etc.) in order to support or refute parts of the user interface design.
- Results from the literature used to support or refute parts of design
- Model based evaluation

Evaluating Implementations

Requires an artefact:

- ❑ simulation, prototype, full implementation

Heuristics

Heuristics are Usability guidelines

- ❑ are rules that distill out the principles of effective user interfaces.

Plenty to choose from

- ❑ Nielsen’s 10 principles
- ❑ Tognazzini’s 16 principles
- ❑ Norman’s rules

Nielsen’s 10 heuristics

1. **Match the Real World (“Speak the user’s language”,)**
   - ❑ Use common words.
   - ❑ Allow aliases/synonyms in command languages

2. **Consistency and Standards**

Principle of Least Surprise

- ❑ Similar things should look and act similar
- ❑ Different things should look different

Other properties

Size, location, color, wording, ordering, …

Nielsen’s 10 heuristics

3. **Help and Documentation**
   - ❑ Users Prefer to spend time working toward their task goals, not learning about your system
   - ❑ But manuals and online help are vital (Usually when user is frustrated or in crisis)
   - ❑ Help should be Searchable, Context-sensitive, Task-oriented, Concrete, Short

4. **User Control and Freedom**
   - ❑ Provide undo
   - ❑ Long operations should be cancelable
   - ❑ All dialogs should have a cancel button

5. **Visibility of System Status**
   - ❑ Keep user informed of system state
   - ❑ Cursor change, Selection highlight, Status bar

Nielsen’s 10 heuristics

6. **Flexibility and Efficiency**
    - ❑ Provide easily-learned shortcuts for frequent operations

7. **Error Prevention**
    - ❑ Selection is less error-prone than typing
    - ❑ Disable illegal commands

8. **Recognition, Not Recall**
    - ❑ Use menus, not command languages
    - ❑ Use combo boxes, not textboxes
    - ❑ Use generic commands where possible (Open, Save, Copy)
    - ❑ All needed information should be visible

Nielsen's 10 heuristics

9. Error Reporting, Diagnosis, Recovery

Be precise; restate user's input

- ❑ Not "Cannot open file", but "Cannot open file named paper.doc"

Give constructive help

- ❑ why error occurred and how to fix it

Be polite

- ❑ Not "fatal error", not "illegal"

Hide technical details (stack trace) until requested

Nielsen's 10 heuristics

10. **Aesthetic and Minimalist Design**

"Less is More"

- ❑ Omit extraneous info, graphics, features

Good graphic design

- ❑ Few, well-chosen colors and fonts
- ❑ Group with whitespace
- ❑ Align controls sensibly

Use concise language

- ❑ Choose labels carefully

Hints for Better Heuristic Evaluation

Use multiple evaluators

- ❑ Different evaluators find different problems
- ❑ The more the better, but diminishing returns
- ❑ Nielsen recommends 3-5 evaluators

Alternate heuristic evaluation with user testing

- ❑ Each method finds different problems
- ❑ Heuristic evaluation is cheaper

Its OK for observer to help evaluator

- ❑ As long as the problem has already been noted
- ❑ This wouldn’t be OK in a user test

User Support

- Even if an interactive system is properly designed, the user will require varies assistance at various times, dependent on many factors:
  - their familiarity with the system,
  - the job they are trying to do, and so on.

There are four main types of assistance that users require:

1. quick reference
2. task-specific help
3. full explanation
4. tutorial

User support is provided by different types of support system

i. Help systems are problem oriented and specific

ii. Documentation is system oriented and generic

User Support Con…

Quick reference

- ❑ used as a reminder to the user for the details of tools

Example: find a particular command option and the syntax of the command

Task-specific help

- ❑ required how to apply the tool to his particular problem

Full explanation

- ❑ used when the more experienced or inquisitive (questioning) user may require a full explanation of a tool or command

Tutorial

- ❑ particularly aimed at new users of a tool and provides step-by-step instruction (perhaps by working through examples) of how to use the tool

Requirements of user support

What features will the help system have? These features are:

Availability

- access help at any time during his interaction with the system.

  Ideally, run concurrently with main application.

Accuracy and completeness

- provided should be accurate and complete.
- If provided proves not to match the actual behavior of the system the user will become disillusioned (disappointed) with the help facilities.

Consistency

- Online help should also be consistent with paper documentation in terms of content, terminology and style of presentation

Requirements of user support Con…

Robustness

- When the system is behaving unexpectedly or has failed altogether ,the help system should be correct error handling and predictable behavior.

Flexibility

- allow each user to interact with it in a way appropriate to his needs, experience and task.

Unobtrusiveness

- The help system should not prevent the user from continuing with normal work.
- The textual help system on a non-windowed interface may interrupt the user’s work.
- A possible solution to this is split-screen presentation.

Approaches to user support

A number of different approaches to providing help, each of which meets a particular need vary from simple captions to full adaptive help and tutoring systems.

The styles of help are given below:

Command assistance

The user requests help on a particular command and is presented with a help screen or manual page describing it.

e.g., UNIX man, DOS help, Windows Help.

It is good for quick reference.

Context sensitive help

used in menu-based systems to provide help on menu options

Example: The Microsoft Office tool-tips , Web page rollover.

Approaches to user support Con…

On-line tutorials

The user get how the application works by **experimenting with examples and demonstration of how to perform a task**

The user can **repeat parts of the tutoria**l if needed.

On-line documentation

paper documentation available on computer.

material available continually (assuming the machine is running!) in the same medium to a large number of users concurrently.

difficult to browse.

Documentation structured using **hypertext supports browsing**

Approaches to user support Con…

Wizards

A *wizard* is a **task-specific tool**.

allow the user actually to complete the tasks safely, quickly and accurately.

Example, the **Microsoft Word resumé.**

allow the user to move back a step as well as forward, will provide a progress indicator showing how much of the task is completed and how many steps remain, and will offer sufficient information to allow the user to answer the questions

Assistants

software tools that **monitor user behavior** and offer suggestions or hints when they recognize familiar sequences.

Example : **Eager**, a software agent. When it notices the user repeating a sequence of actions, a **cat icon appears**, suggesting the next action

Designing User Support Systems

There are a number of things which the designer should take into account.

1. User support is not an `add on’
   - should be designed integrally with the system.

2. Concentrate on content and context of help rather than technological issues.

3. Make decisions about how the help will be presented to the user and how this will be affected by implementation issues.

Designing User Support Systems Con…

Presentation issues

How is help requested?
command, button, function (on/off), separate application

How is help displayed?
new window, whole screen, split screen,
pop-up boxes, hint icons

Effective presentation requires
clear, familiar, consistent language
instructional rather than descriptive language
avoidance of blocks of text
clear indication of summary and example information

Designing User Support Systems Con…

Implementation issues

1. Will help be an operating system command, a meta-command or an application?
2. What physical constraint does the machine impose in terms of screen space, memory capacity and speed?
3. How the help data is to be structured: in a single file, a file hierarchy, a database? any structure should be flexible and extensible
4. Will users be able to browse through the system or only request help on one topic at a time?
5. Will the user make a hard copy of part of the help system to study later (manuals and documentation).
6. The designer should consider the authors of help material as well as its users. Even if the designer writes the initial help texts, these will be extended by other authors at different times.

End of the course!!!