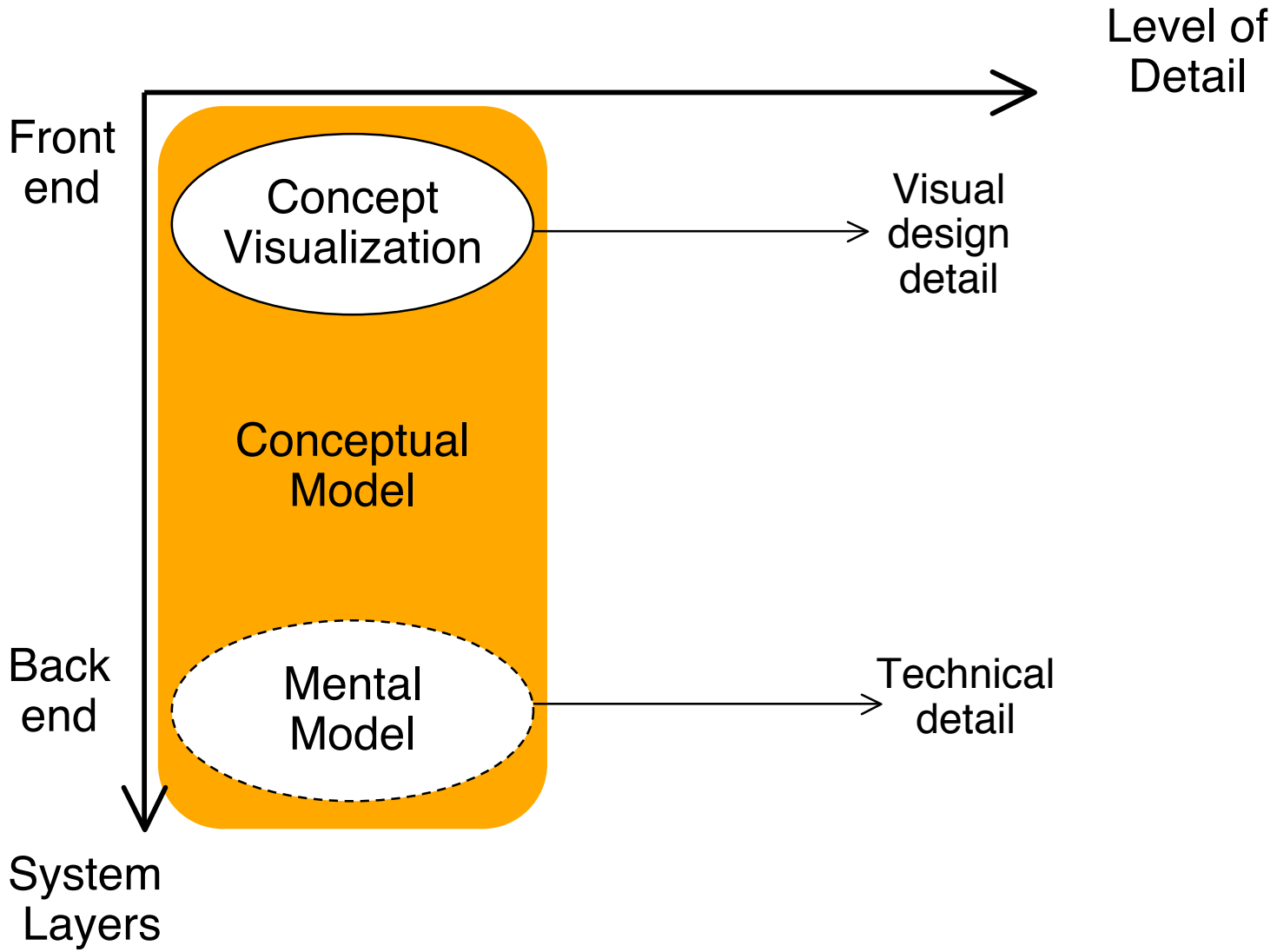
6 Designing Interactive Systems

Purposes of Prototypes

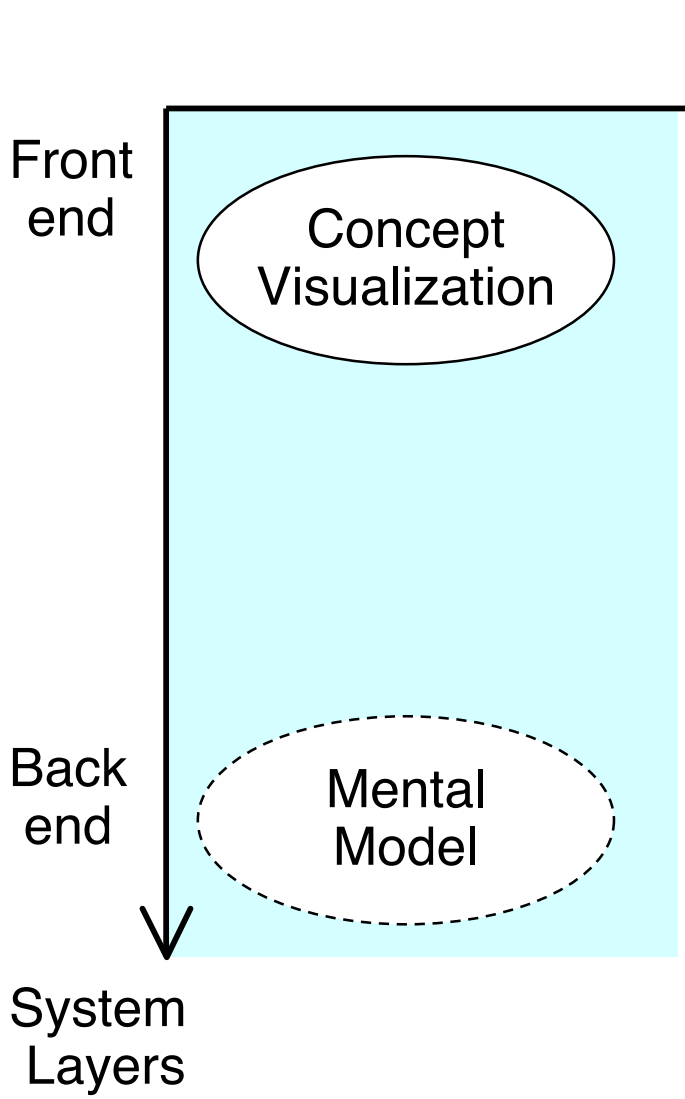
- Usability testing
  - How the product will fit into the users' lives
- Validation of customer requirements
- "Living" design specification
- Information development
  - Which data needs to be recorded?
- Marketing support
  - Convincing upper level management

http://www.research.ibm.com/journal/sj/424/vanbuskirk.pdf

Concept and Details



Low-Fidelity Prototypes

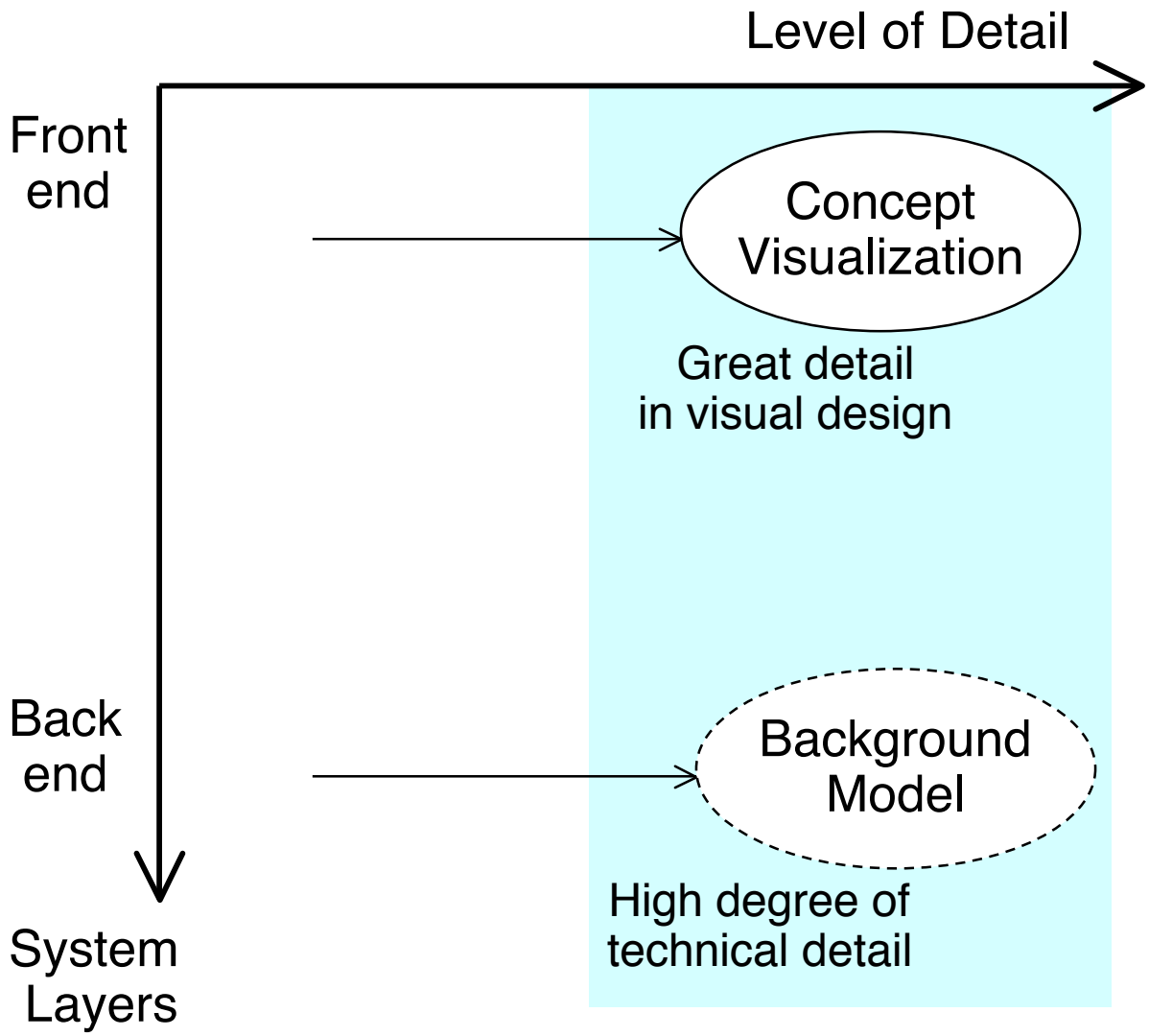


- Paper Prototypes
- Storyboards & sketches
- Concept videos

+ Cheap, good for basic concepts
+ Early in development
+ No technology barrier

– May alienate users
– Often limited coverage of system features

High-Fidelity Prototypes

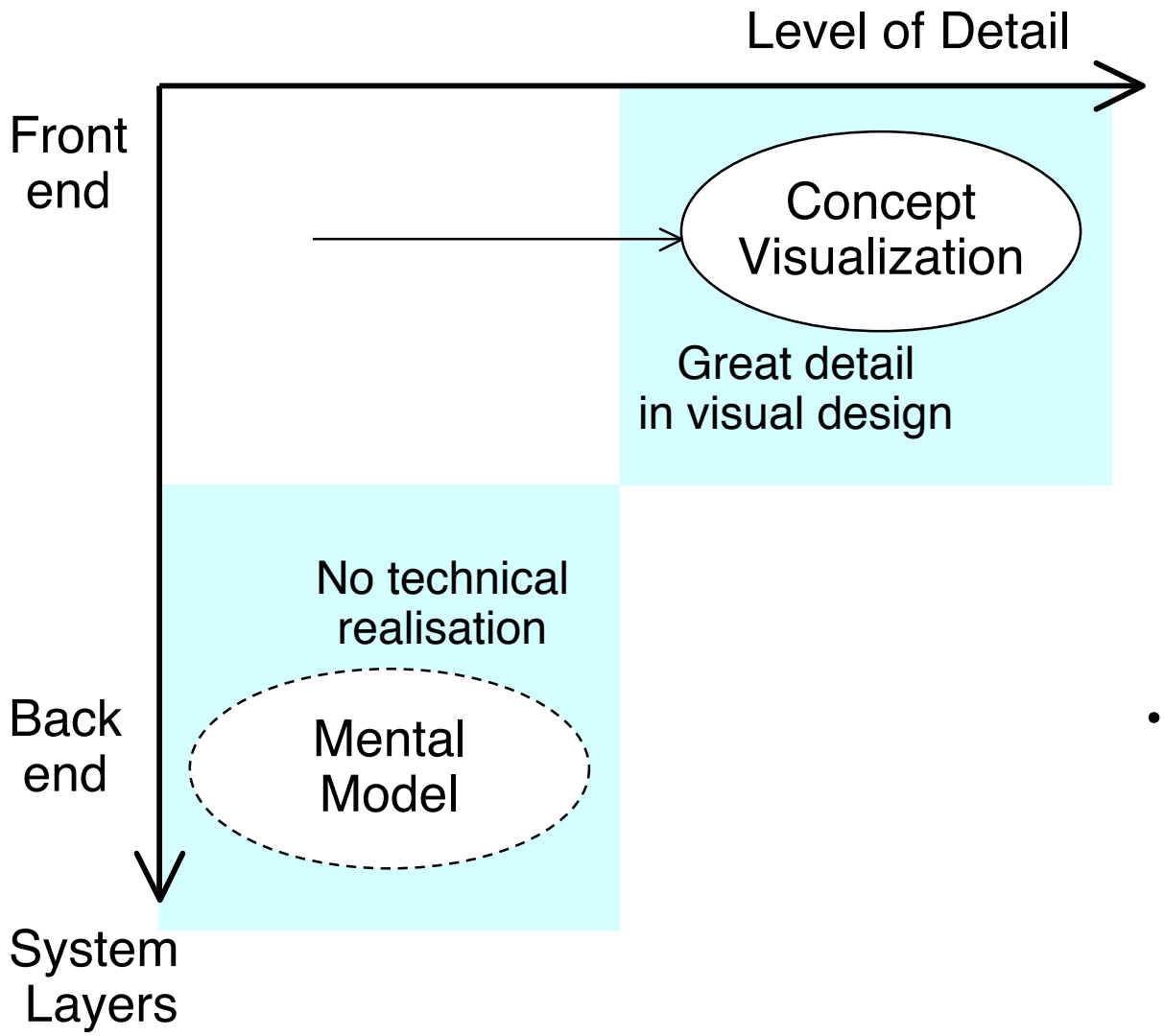


- HTML, Javascript
- Flash, Director
- GUI Builders

+ Realistic impression
+ Detailed user feedback
+ Timing, interaction

– Expensive
– Functionality needs to be restricted
– May limit creativity of test users

Cheap High-Fidelity Prototypes



- “Wizard of Oz”

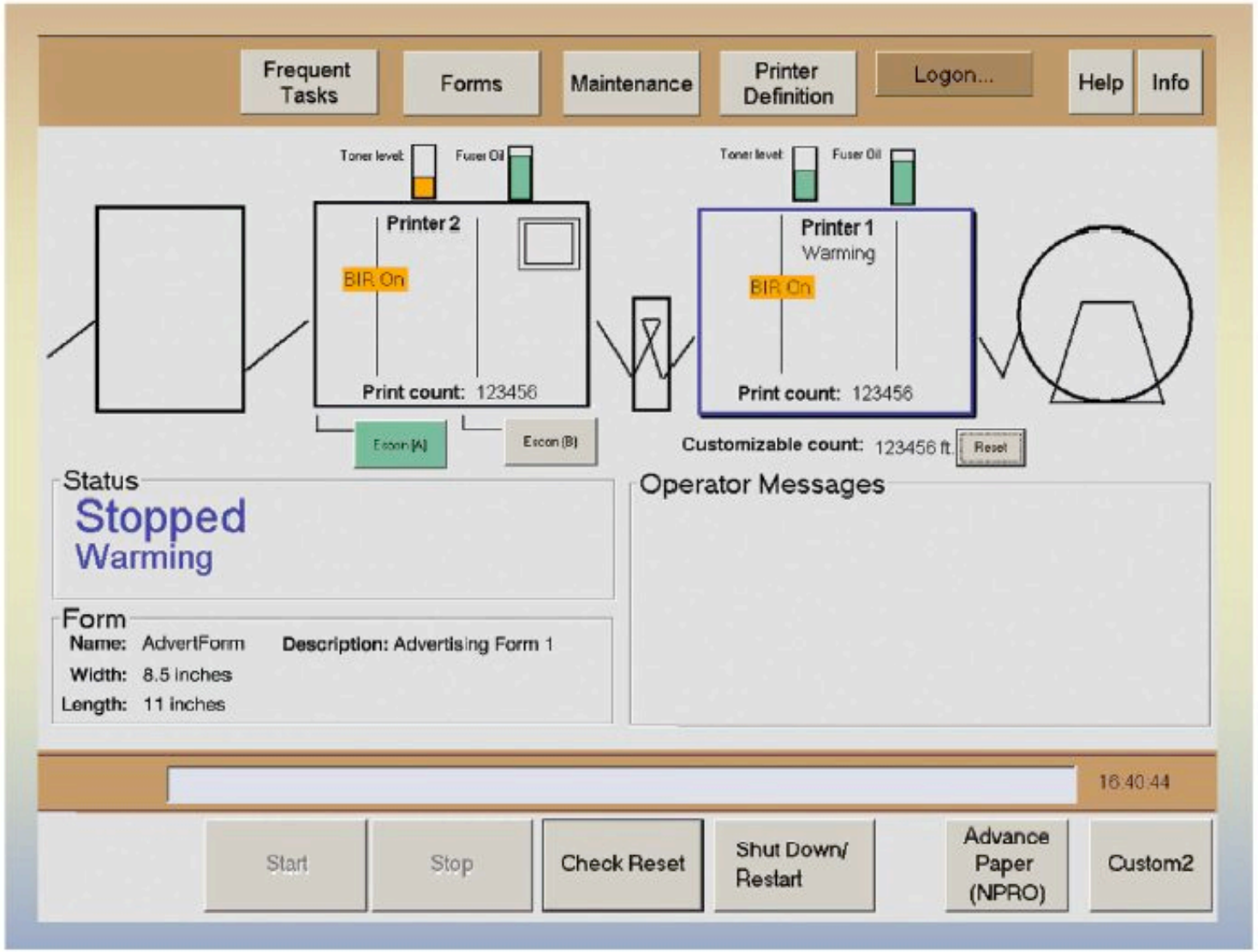
Wizard-of-Oz

- "The man behind the curtain"
  - Children's book 1900, movie 1939
- Do not implement the hard parts in the prototype – just let a human control the system's reaction
- Typical areas
  - Speech recognition
  - Speech synthesis
  - Annotation
  - Reasoning
  - Visual Perception
- Provides the user with the experience without extensive implementation effort for the prototype

Visual Design and User Feedback

- Highly realistic, aesthetically pleasing interface prototype
  - Often leads to restricted scope of evaluation comments
  - Users do not question the basic concept anymore
  - Feedback concentrates on details of visual design, interaction details
- Realistic but aesthetically less convincing prototype
  - May help users to question the concept
  - Can easily be improved to better visual designs
- R. Van Buskirk and B. W. Moroney: Extending Prototyping, *IBM Systems Journal* - Vol. 42, No. 4, 2003 - Ease of Use
  - “Keep it ugly”

“Keep it Ugly” - Example (1)

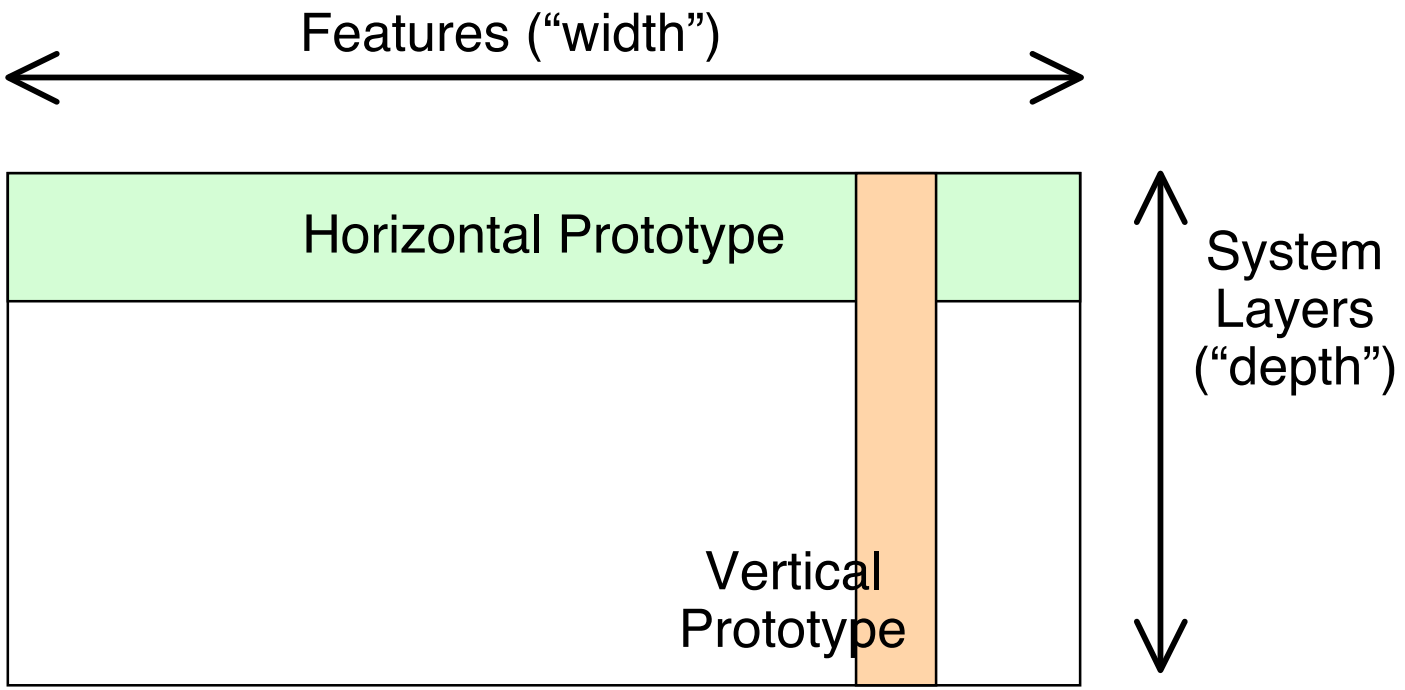


“Ugly” Version

“Keep it Ugly” - Example (2)

Polished Version

Horizontal and Vertical Prototyping



Please note the meaning of the horizontal dimension is slightly different to previous drawings!

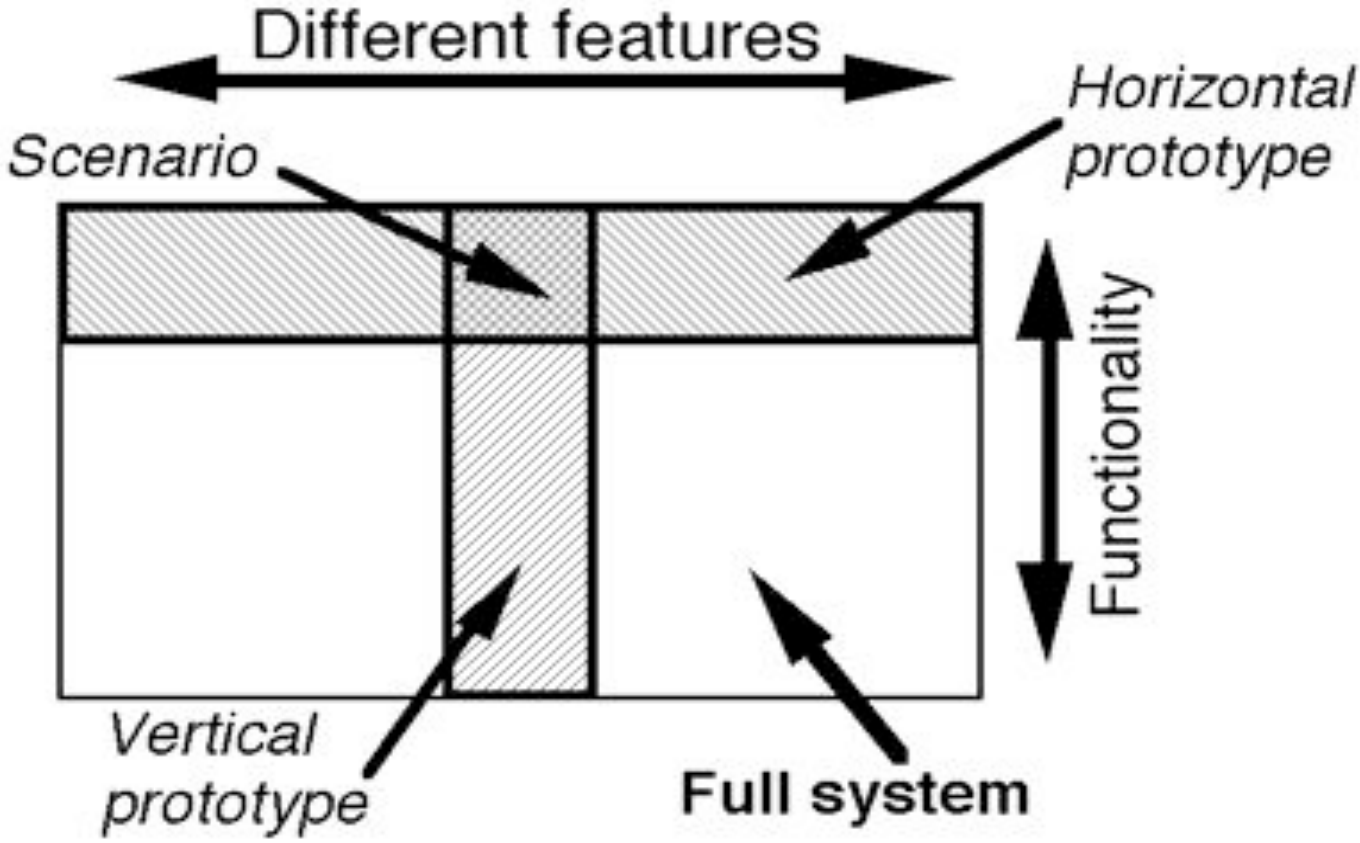
Horizontal Prototyping

- Demonstrate the feature spectrum of a product
- Allows the user to navigate the system
- The actual functions are not implemented
- Helps to evaluate/test
  - Navigation (e.g. finding a specific function or feature)
  - Overall user interface concept
  - Feature placement
  - Accessibility
  - User preferences
- Applicable in low-fidelity prototyping and high-fidelity prototyping
- Used in early design stages
  - To determine the set of features to include
  - To decide on the user interface concept
- Example: overall usage of a mobile phone

Vertical Prototyping

- Demonstrate a selected feature of a product
- Allows the user only to use this specific function
- The details of the function/feature are shown/implemented
- Helps to evaluate/test
  - The optimal design for a particular function
  - Optimize the usability of this function
  - User performance for this particular function
- Mainly use in high-fidelity prototyping but can be applicable to low-fidelity prototyping
- Used in early design stages
  - To compare different designs for a specific function
- Used in later design stages
  - To optimize usage of a function
- Example: a new input methods for writing SMS on a mobile phone

Scenarios as Cheap Prototyping Strategy



Scenario as intersection of horizontal and vertical prototyping

http://www.useit.com/papers/guerrilla_hci.html (Jakob Nielsen 1994)

1984 Olympic Message System
A human centered approach

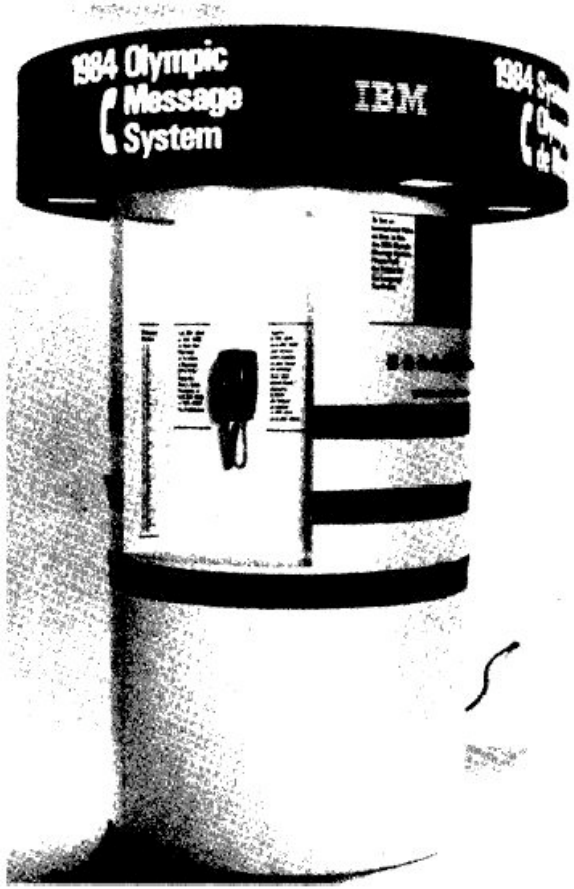
- A public system to allow athletes at the Olympic Games to send and receive recorded voice messages (between athletes, to coaches, and to people around the world)

- Challenges
  - New technology
  - Had to work – delays were not acceptable (Olympic Games are only 4 weeks long)
  - Short development time

- Design Principles
  - Early focus on users and tasks
  - Empirical measurements
  - Iterative design

  → Looks obvious – but it is not!

- … it worked! But why?



1984 Olympic Message System Methods

- Scenarios instead of a list of functions
- Early prototypes & simulation (manual transcription and reading)
- Early demonstration to potential users (all groups)
- Iterative design (about 200 iterations on the user guide)
- An insider in the design team (ex-Olympian from Ghana)
- On site inspections (where is the system going to be deployed)
- Interviews and tests with potential users
- Full size kiosk prototype (initially non-functional) at a public space in the company to get comments
- Prototype tests within the company (with 100 and with 2800 people)
- “Free coffee and doughnuts” for lucky test users
- Try-to-destroy-it test with computer science students
- Pre-Olympic field trail

The 1984 Olympic Message System: a test of behavioral principles of system design John D. Gould , Stephen J. Boies , Stephen Levy , John T. Richards , Jim Schoonard Communications of the ACM September 1987 Volume 30 Issue 9 (http://www.research.ibm.com/compsci/spotlight/hci/p758-gould.pdf)

6 Designing Interactive Systems

Interactive Systems
What can be described?

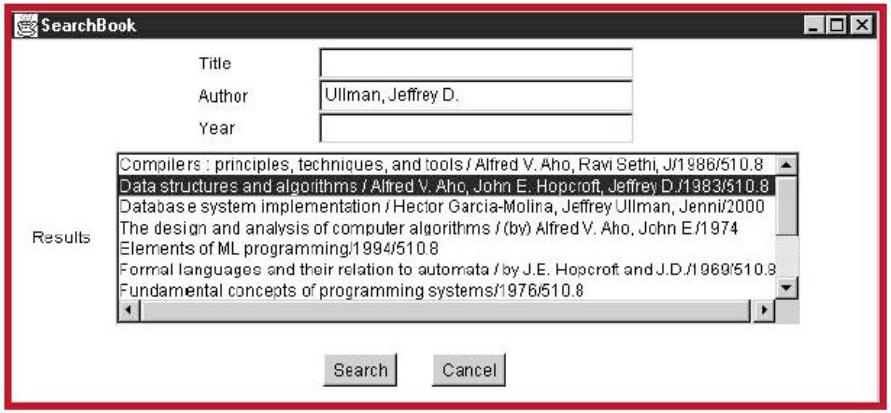
- System functionality with regard to interaction
- Overall interaction concepts (metaphors, styles)
- Layout of key screens, sketches
- Layout of user interface elements (e.g. buttons, icons)
- Navigation and interaction details
- Interactive behavior of a system
- Platform requirements
- Functional assertions (e.g. login will take on average 7 seconds, average time per case is 2 minutes)
- User groups
- …

Interactive Systems
How to describe them?

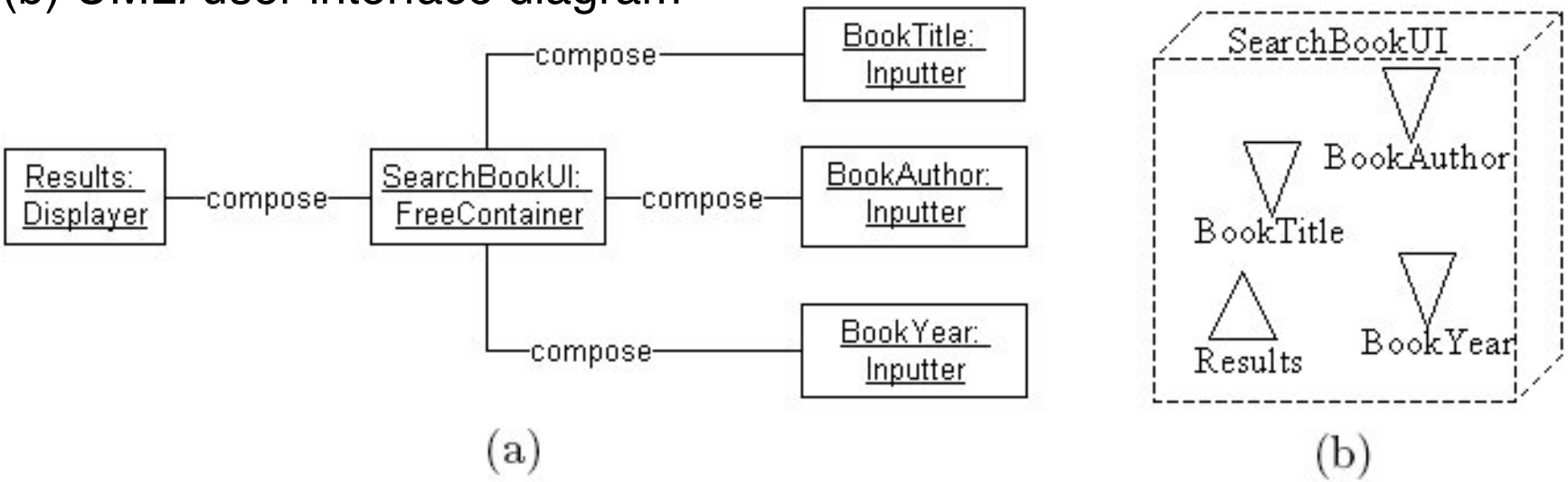
- Informal
  - System descriptions in plain text
  - Scenarios and use cases
  - Sketches and designs
  - Task-action-mappings
- Semi-formal
  - Task-action-grammar
  - Abstract UI description languages, e.g. UML based
    - » Examples: UMLi, CTT?
- Formal
  - E.g. Z, state machines
- Implementation languages
  - XML based languages
    - » e.g. XUL (= XML User Interface Language), Microsoft XAML
  - Can be used to generate a concrete UI for the target platform

UML*i* Example (1)

- P. de Silva/N. Paton: User Interface Modeling in UML*i*, *IEEE Software* 20(4) 2003, pp. 62-69

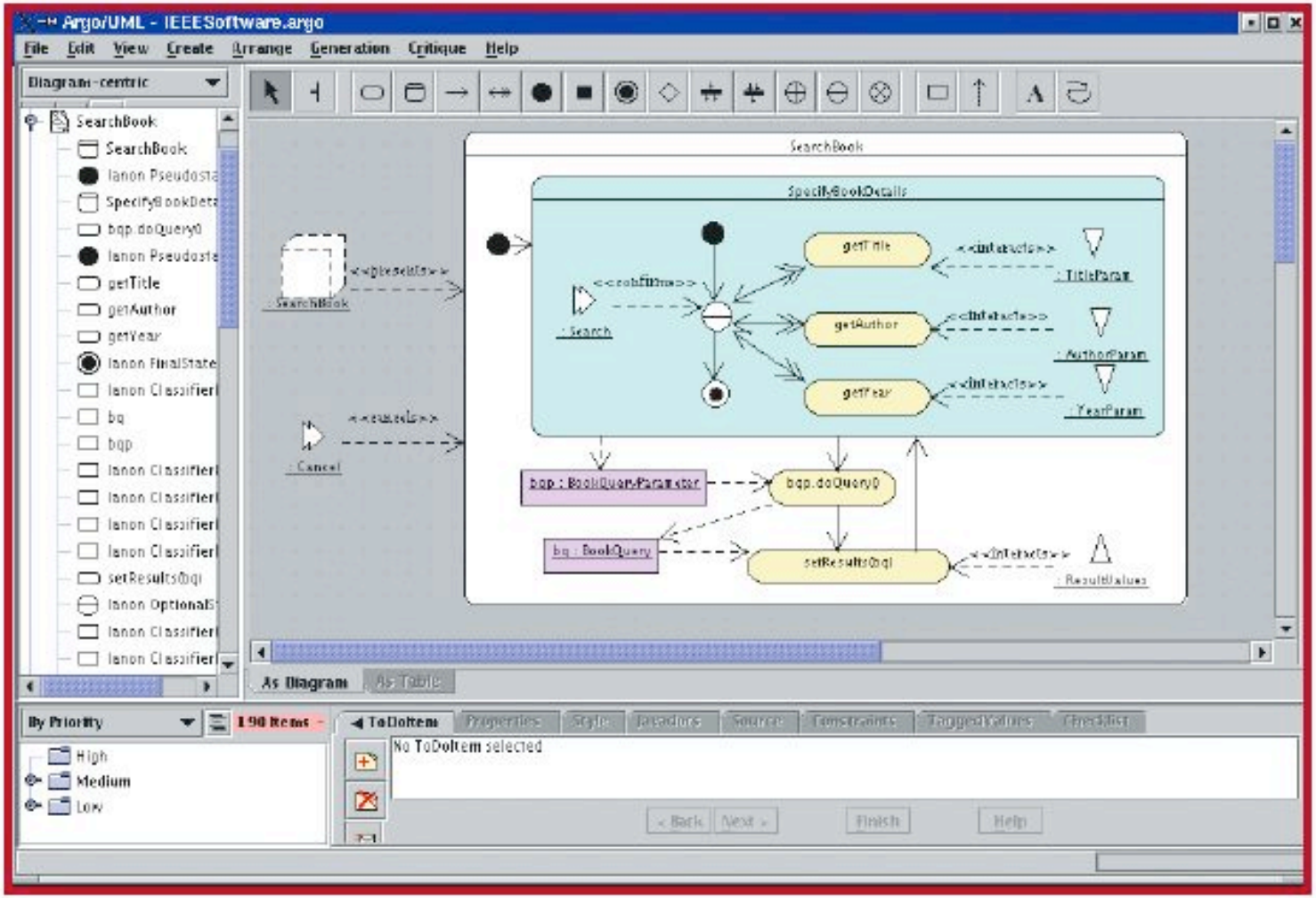


(b) UML*i* user interface diagram



UML*i* Example (2)

- Diagram types for static structure and dynamic behaviour
- Tool support based on UML CASE tools

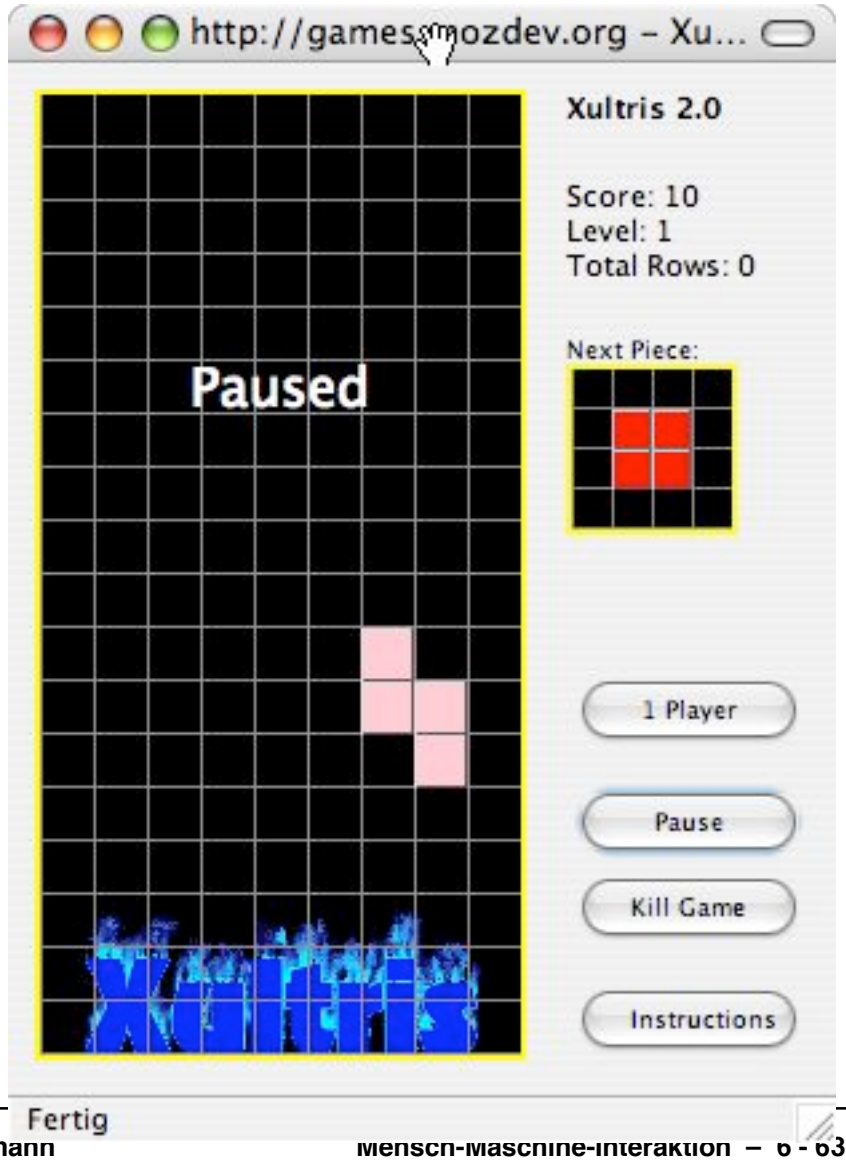


XUL Example

```
<?xml version="1.0"?>
<!-- Sample XUL file -->
<window xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
<box align="center">
  <button label="hello xFly" onclick="alert('Hello World');" />
</box>
</window>
```

XUL: Platform Independent Interfaces

- Full UI programming environment based only on XML and JavaScript
- Example: http://games.mozdev.org/xultris/



6 Designing Interactive Systems

Design Patterns

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

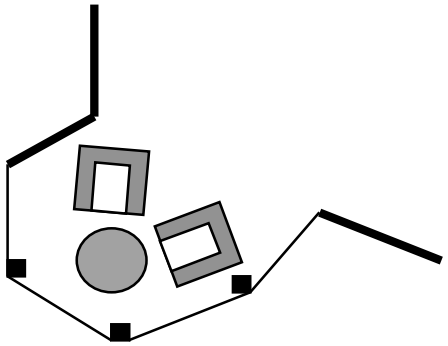
Christopher Alexander et al., A Pattern Language

- Design patterns
  - Originated from architecture (Christopher Alexander 1977)
  - Were made popular for software design issues by Gamma/Helm/Johnson/Vlissides ("Gang of Four") 1995
- Patterns are never "invented"
  - Patterns are retrieved from working solutions for problems by generalization
  - Often product of a community, using online repositories for patterns
- Principle is applicable to HCI issues as well
  - In fact, Alexander's patterns focused on usability!

Window Place: A “True” Architectural Pattern

- Based on:
  Christopher Alexander et al., A Pattern Language, 1977
  (as referred to by Buschmann et al. 1996)

- **Problem:** If a room does not have a window, which offers itself as a “place”, users cannot decide between comfortable sitting and the attractions of light and view.
- **Solution:**
  One window of any living room shall be a “window place”.
- **Structure:**



What is a Design Pattern for Software?

- **Definition** A *pattern* is a schematic solution for a class of related problems.
- Patterns appear on various levels:
  - Analysis patterns
  - Architectural patterns
  - Design patterns
    - » Structural patterns
    - » Creational patterns
    - » Behavioural patterns
  - Language-dependent formulations (*idioms*)

- Literature:

  E. Gamma et al., Design Patterns (dt. Entwurfsmuster), Addison-Wesley 1995

  M. Grand, Patterns in Java - Volume 1, Wiley 1998

Description of a Software Design Pattern

- Name
- Problem
  - Motivation
  - Application area
- Solution
  - Structure (class diagram)
  - Pieces (usually names of classes, associations or operations):
    - » “Role names”, i.e. place holders for parts of application
    - » Fixed parts of implementation
  - Object interaction (e.g. Sequence diagra)
- Discussion
  - Advantages, disadvantages
  - Dependencies, constraints
  - Special cases
  - Known uses

Patterns as Knowledge Representation

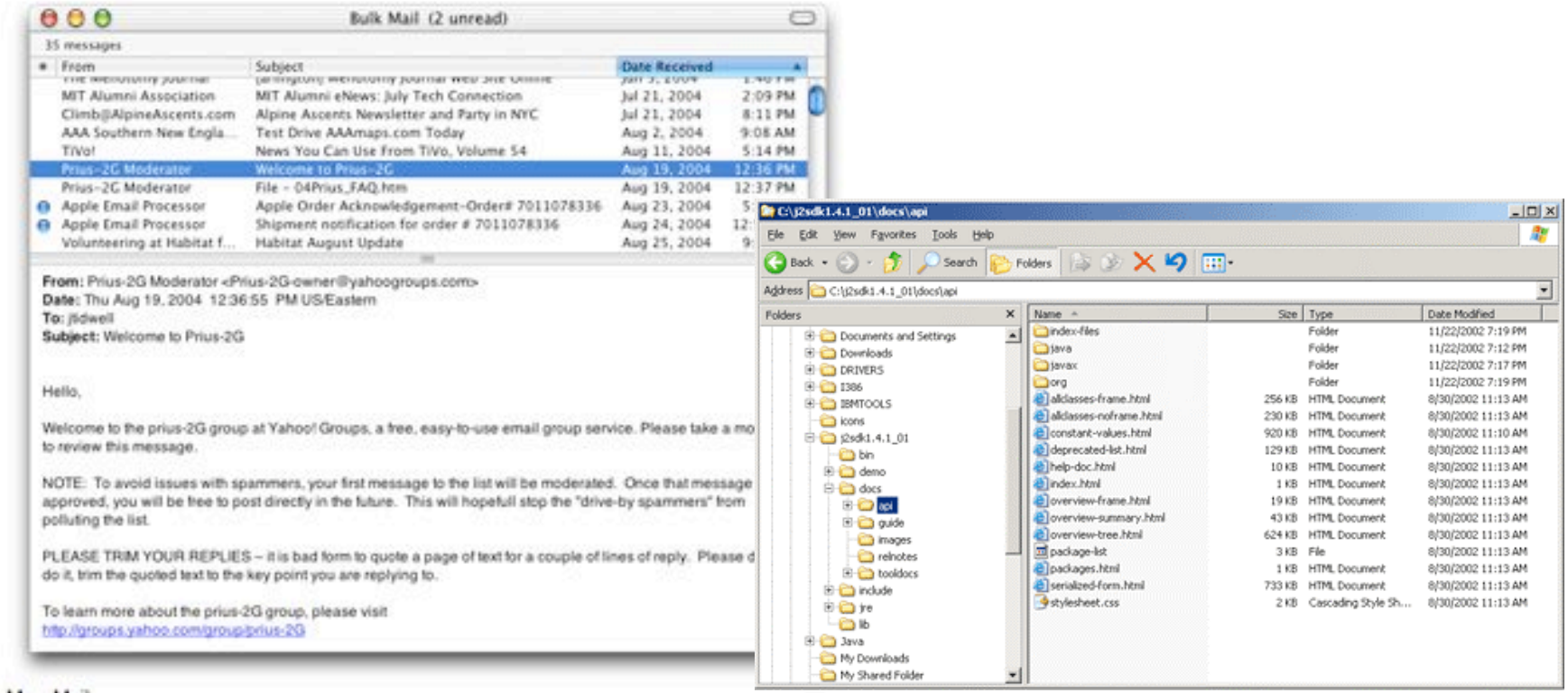
- Many facts and rules of HCI knowledge can be encoded as patterns
  - See e.g. Mahemoff/Johnston 1998
- Examples of pattern encodings of knowledge in HCI:
  - Task patterns
    - » E.g. “Open existing document”
  - User patterns
    - » E.g. “Expert user”, “novice”
  - User interface element patterns
    - » E.g. “Scrollbar”
  - User interface arrangement patterns
    - » E.g. “Show status”
  - Interaction style patterns
    - » E.g. “Instructions”, “Conversion”, “Exploration”
  - Organisational patterns
    - » E.g. “Online repository”

Concrete Interface Design Patterns

- Jennifer Tidwell 1999, 2005:
  - Catalogue of very concrete user interface design solutions
    - » Organizing the content, Getting around, Organizing the page, Commands and Actions, Showing complex data, Getting input from users, Builders and Editors, Making it look good
  - See designinginterfaces.com
- Martijn von Welie 2003:
  - Interaction design patterns for Web design, GUI design, Mobile UI design
  - See www.welie.com/patterns
- Jan Borchers 2001:
  - Design patterns for interactive museum exhibits
  - See http://www.hcipatterns.org/patterns/borchers/patternIndexHtml.html

Example: Two-Panel Selector (Tidwell)

Two-Panel Selector

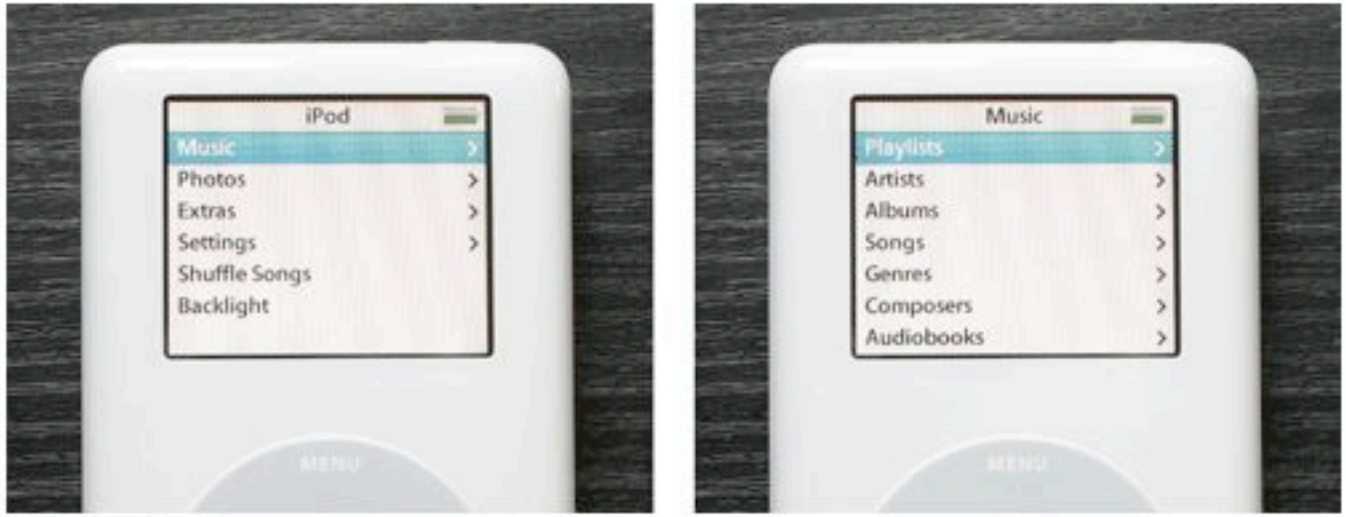


Mac Mail

What: Put two side-by-side panels on the interface. In the first, show a set of items that the user can select at will; in the other, show the content of the selected item.

Example: One-Window Drilldown (Tidwell) (1)

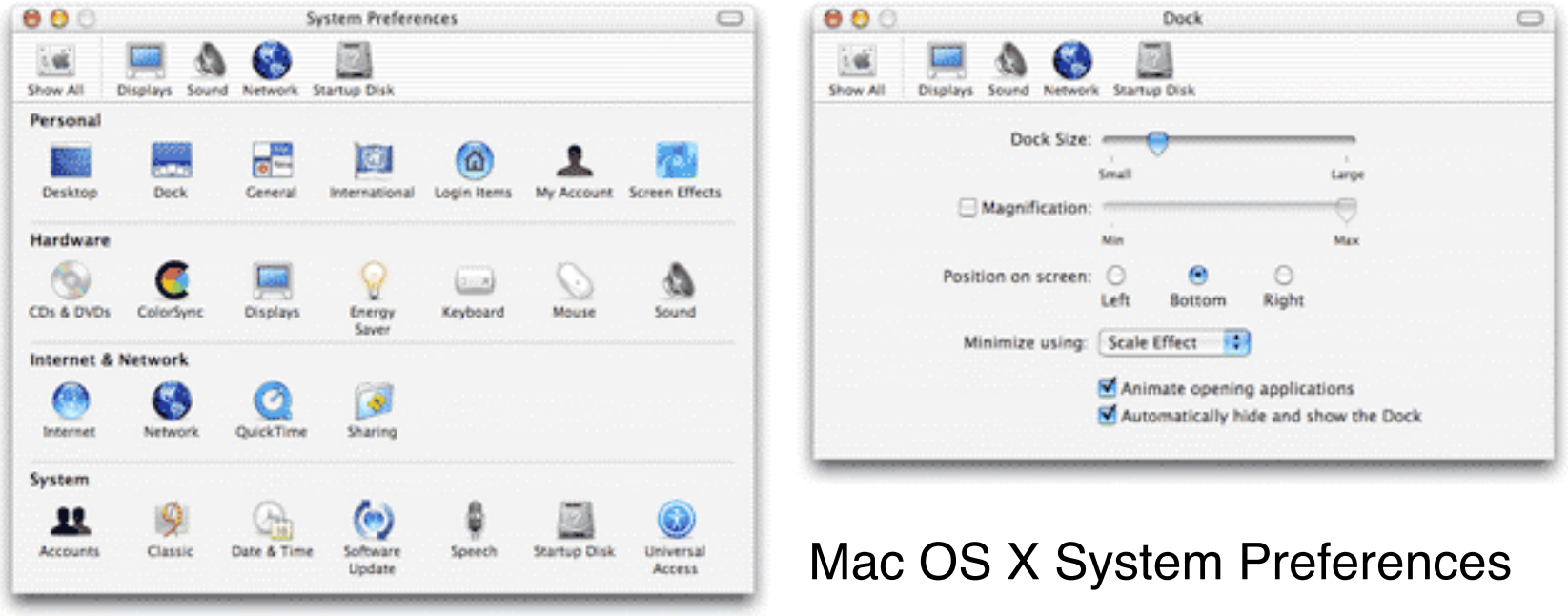
One-Window Drilldown



Two iPod menus

What: Show each of the application's pages within a single window. As a user drills down through a menu of options, or into an object's details, replace the window contents completely with the new page.

Example: One-Window Drilldown (Tidwell) (2)



Mac OS X System Preferences

- When to use One-Window Drilldown:
  - Good for restricted display space
  - Good for infrequent usage
- When to use Two-Panel Selector:
  - Good for frequent usage and frequent navigation in content
  - Relieves short term memory of user, remains still simple

Example: Extras on Demand (Tidwell)

Extras On Demand

The color dialog box in Windows 2000

What: Show the most important content up front, but hide the rest. Let the user reach it via a single, simple gesture.

Use when: There's too much stuff to be shown on the page, but some of it isn't very important. You'd rather have a simpler UI, but you have to put all this content somewhere.

Example: Global Navigation (Tidwell)

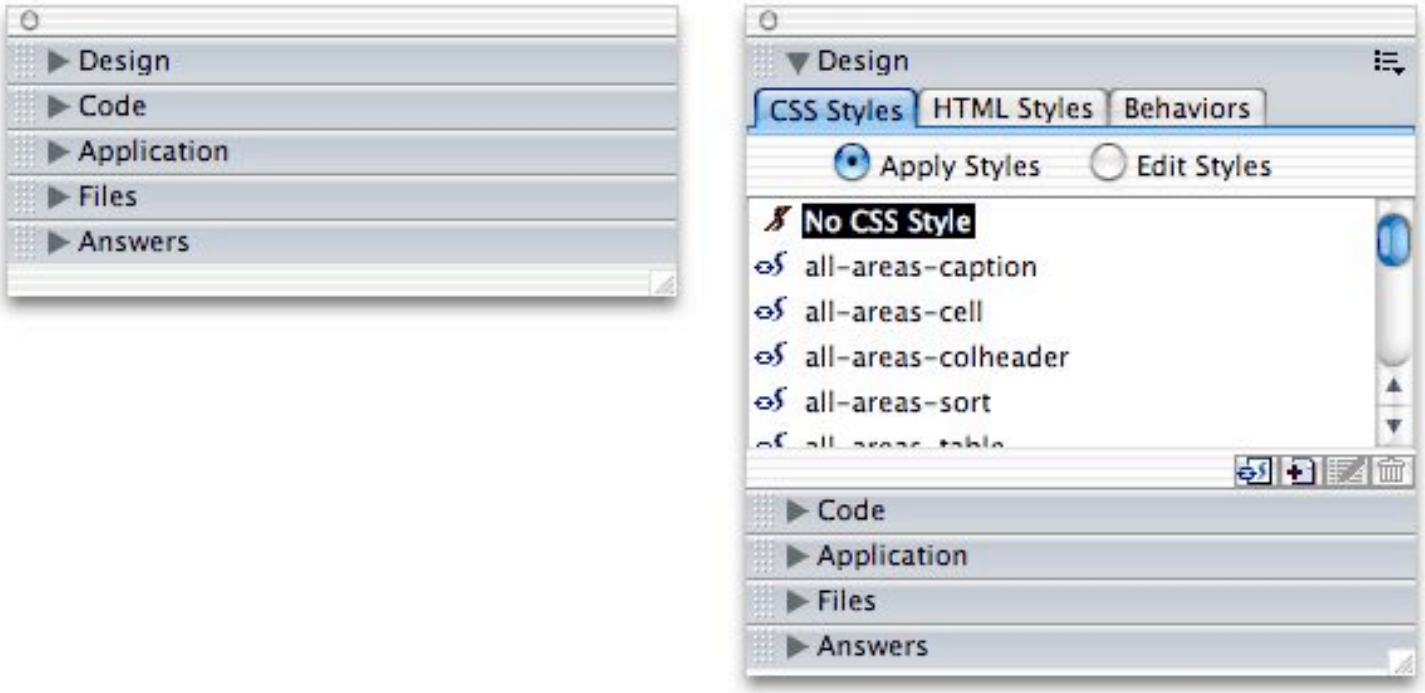
Global Navigation

From Microsoft Money

What: Using a small section of every page, show a consistent set of links or buttons that take the user to key sections of the site or application.

Example: Closable Panels (Tidwell)

Closable Panels



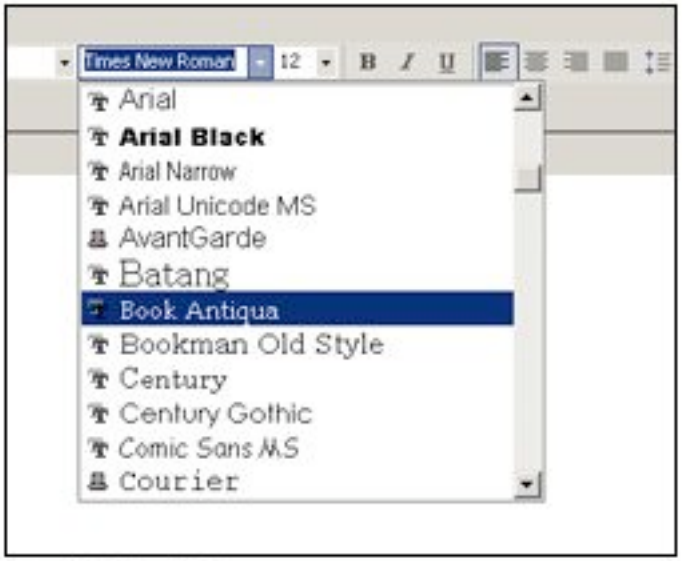
Dreamweaver MX

What: Put sections of content onto separate panels, and let the user open and close each of them separately from the others.

Use when: There's too much stuff to present on the page, but you want it all only one click away. The content is divisible into clearly-named sections, as with Titled Sections and Card Stack.

Example: Illustrated Choices (Tidwell)

Illustrated Choices



From Word for Windows

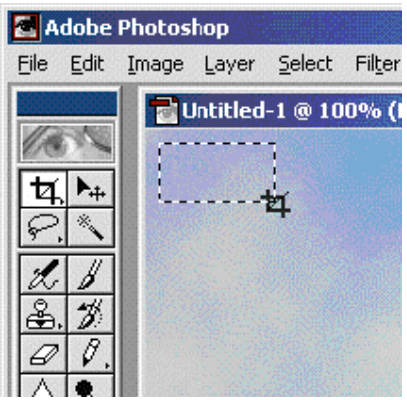
Mac OS X System Properties

What: Use pictures instead of words (or in addition to them) to show available choices.

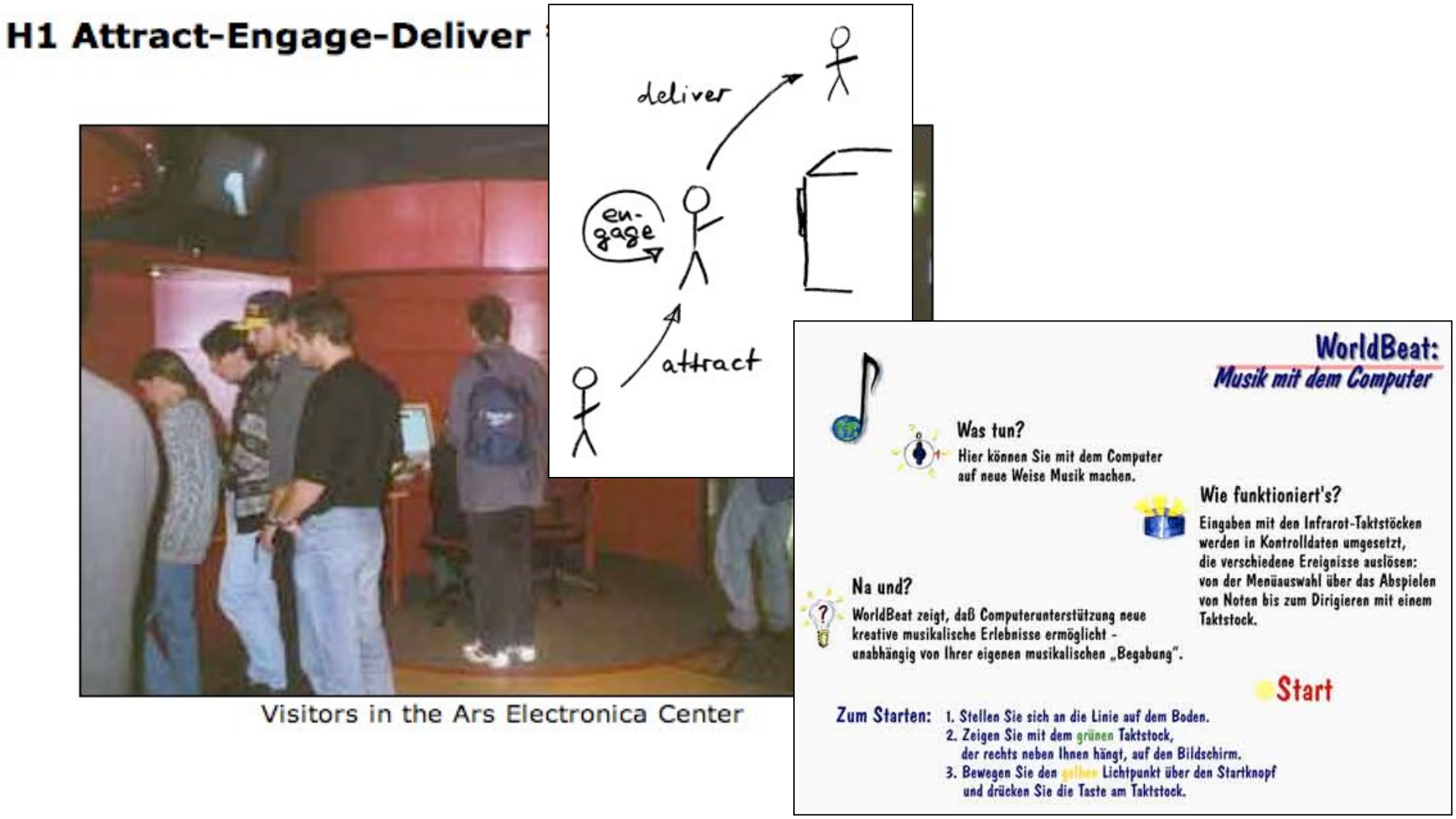
Example: Mode Cursor (Welie)

Mode Cursor

Author	Martijn van Welie
Problem	The user is creating or modifying an object and needs to know which edit function is selected.
Principle	Immediate Feedback (Feedback)
Context	In many direct manipulation applications the users first selects a tool/function, thus entering a special mode/state, and then works on an object. Since such applications usually offer many functions to create or modify objects.
Forces	• Not every function may have an icon or shape. • Completing a function may cause several intermediate states which may also need to be shown. • The user needs immediate feedback on which function was selected, i.e. which mode/state the system is in.
Solution	**Show the interface state in the cursor.**



Example: Attract–Engage–Deliver (Borchers)



Visitors in the Ars Electronica Center

References

- Alan Dix et al, chapters 6 and 7
- Jakob Nielsen: Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier, 1994

  (http://www.useit.com/papers/guerrilla_hci.html)
- R. Van Buskirk and B. W. Moroney: Extending Prototyping, *IBM Systems Journal* - Vol. 42, No. 4, 2003 - Ease of Use

  (http://www.research.ibm.com/journal/sj/424/vanbuskirk.pdf)
- M.J. Mahemoff and L.J. Johnston: Pattern Languages for Usability: An Investigation of Alternative Approaches. *APCHI 98 Conferenne Proceedings*, Los Alamitos 1998, p. 25-31.

  (http://mahemoff.com/paper/candidate/)
- Jennifer Tidwell: Designing Interfaces - Patterns for Effective Interaction Design, O'Reilly 2005
- Jan Borchers: A Pattern Approach to Interaction Design, Wiley 2001