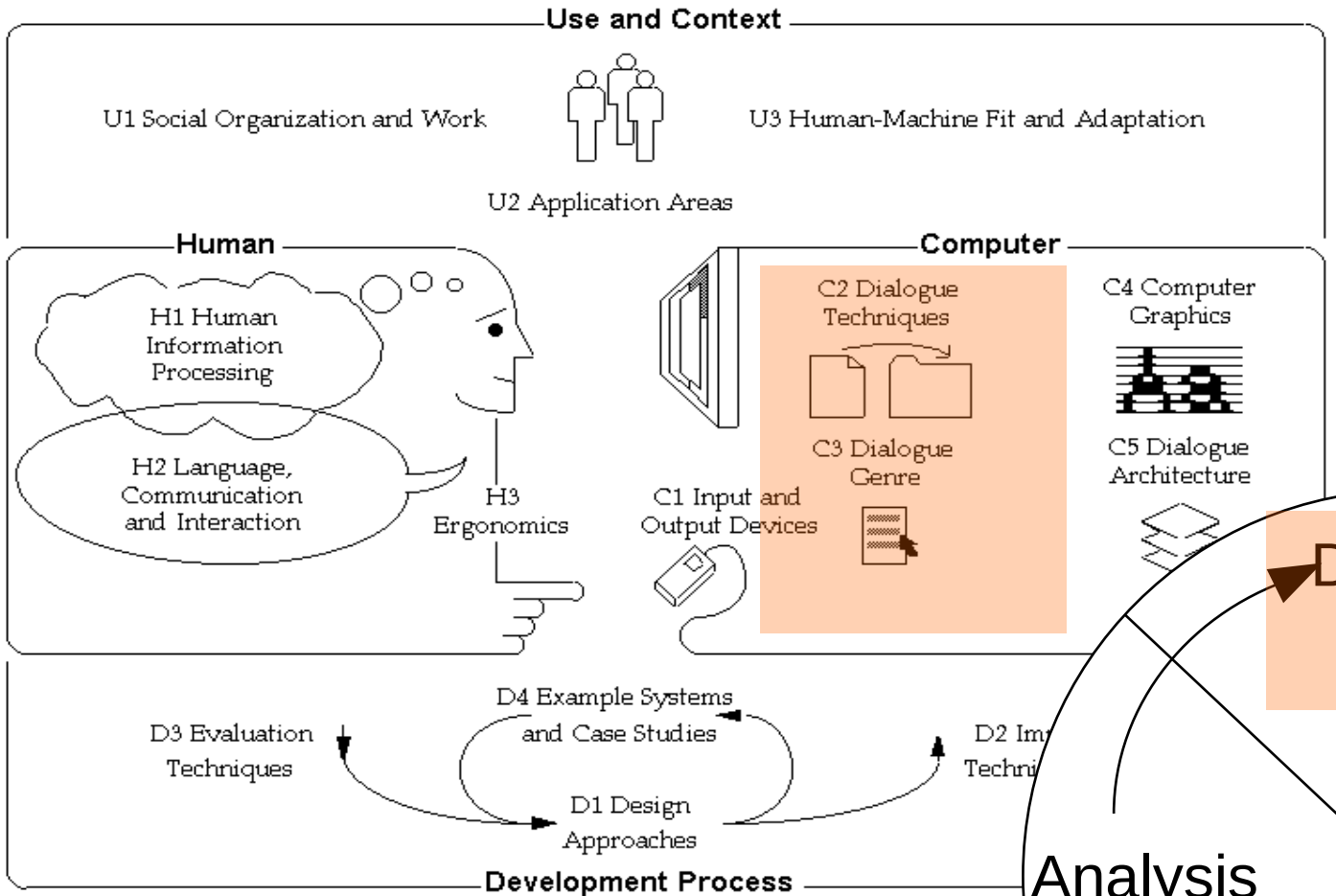
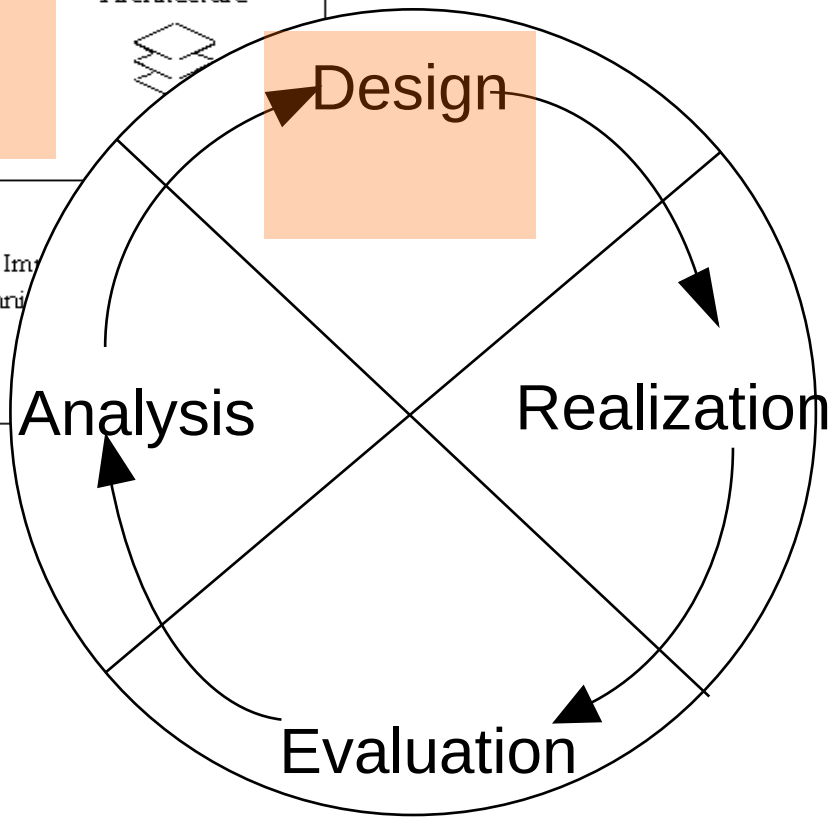
5 Designing Interactive Systems


Use and Context
U1 Social Organization and Work
U3 Human-Machine Fit and Adaptation
U2 Application Areas
Human
H1 Human Information Processing
H2 Language, Communication and Interaction
H3 Ergonomics
Computer
C1 Input and Output Devices
C2 Dialogue Techniques
C3 Dialogue Genre
C4 Computer Graphics
C5 Dialogue Architecture
D3 Evaluation Techniques
D4 Example Systems and Case Studies
D1 Design Approaches
Development Process


Design
Realization
Evaluation
Analysis

Requirements vs. Design

- Requirements (result of analysis phase)
  - Describe ***what*** the problem is
  - Is always very application / domain specific
  - Defines users, goals, tasks, context
  - Define the criteria for evaluating final solutions and intermediate design ideas
  - Limits the possible design options
- Design
  - Describes ***how*** the solution looks like and works
  - Has to conform to the requirements
  - Is a specific selection among many possible design options (design space)
  - Has to consider general design principles beyond the application domain
- Design follows the requirements
  - In general, requirements have to be known first
  - Sometimes, requirements have to be questioned during design!
    - » E.g. "*let's assume we have a much larger screen than on the phone now*"

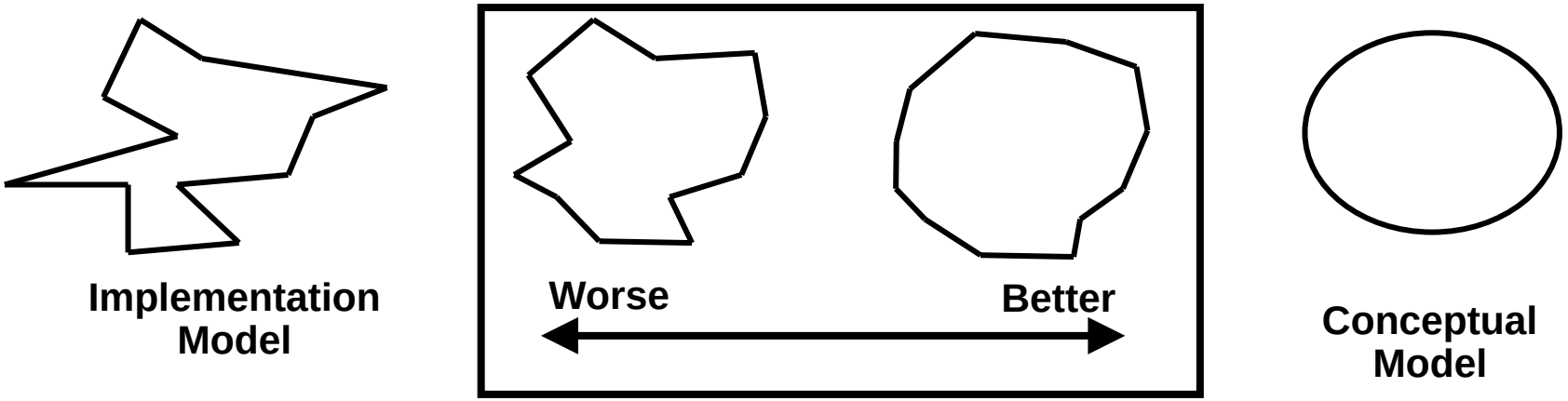
The Solution Space

- What technologies are available to create interactive electronic products?
  - Software
  - Hardware
  - Systems
- How can users communicate and interact with electronic products?
  - Input mechanisms
  - Options for output
- Approaches to Interaction
  - Immediate “real-time” interaction
    - » Variants thereof...
  - Batch / offline interaction

5 Designing Interactive Systems

From Requirements to a First Design

- Conceptual design
  - Transforming user requirements and needs into a conceptual model
- Key guidelines for conceptual design:
  - Separate real requirements from solution ideas
  - Keep an open mind but never forget the users and their context
  - Discuss ideas with other shareholders as much as possible
  - Use low-fidelity prototyping to get rapid feedback
  - Iterate, iterate, and iterate



Analysing the Problem Space

- Having a good understanding of the problem space can help to make informed decisions in the design space
  - Are there problems with an existing product?
  - Why do you think there are problems?
  - Why do you think your proposed ideas might be useful?
  - How would you see people using it with their current way of doing things?
  - How will it support people in their activities?
  - Will it really help them?

- Example:
  - What were the assumptions made by cell phone companies when developing WAP services?
  - Was it a solution looking for a problem?

WAP Example

- People want to be kept informed of up-to-date news wherever they are
  - reasonable
- People want to interact with information on the move
  - reasonable
- People are happy using a very small display and using an extremely restricted interface
  - not reasonable
- People will be happy doing things on a cell phone that they normally do on their PCs (e.g. surf the web, read email, shop, bet, play video games)
  - reasonable only for a very small group of users

First Steps in Formulating a Conceptual Model

- What will the users be doing when carrying out their tasks?
  - Interaction modes
  - Objects (data)
  - Activities (interaction styles)
- How will the system support these?
- What kind of interface metaphor, if any, will be appropriate?
- What kinds of interaction modes and styles to use?

Always keep in mind when making design decisions how the user will understand the underlying conceptual model

Good starting point: scenarios

The Software Engineering Way of Analyzing Scenarios (Ivar Jacobson 1999)

- Conceptual terms called “class embryos”, found in scenario texts

UML Icons:

  - *Boundary class:*
    Type and content of user interaction 
  - *Control class:*
    Processes, steps, and their order 
  - *Entity class:*
    Persistent objects 

- **Example:**

  ”Checking a booking request:
  1. Using the customer number, it is checked whether the customer is known.
  2. Its is checked whether this customer already has a booking for a seminar of the mentioned course type ..."


Booking request


Checking

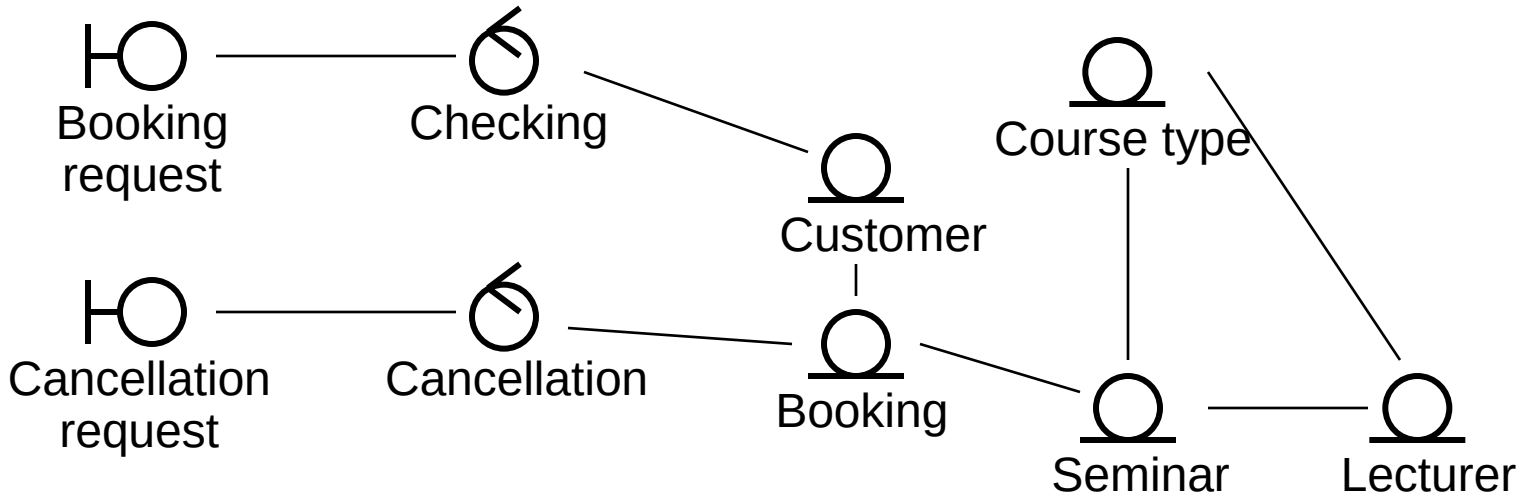

Customer


Booking


Seminar

Creating a Model by Analysing Scenarios

- Step by step analysis of all scenario texts
  - Integrate information into consistent model
  - Re-use all found conceptual terms
  - Create overview diagram (draft for class diagram)



The Interface Design Way of Analyzing Scenarios

- Step-by-step analysis of scenarios
  - Find interaction activities
    - » analyse interaction mode and style
  - Find interaction objects
- Rapidly map onto rough interface design
  - Which mixture of interaction styles?
  - Which concrete interfaces?
- Carry out early user prototyping
- Assess design decisions and possibly scenarios

- Two possible approaches (as in Software Engineering):
  - Focusing on activities
  - Focusing on objects

5 Designing Interactive Systems

Interaction Styles in Activity-Based Design

- Five main interaction modes with associated interaction styles:
  - **Giving instructions:** issuing commands
    - » command language, using keyboard and function keys
    - » menu system
  - **Conversing:** interacting with the system as if having a conversation
    - » using step-by-step windows
    - » using natural language (speech output / possibly speech recognition)
  - **Manipulating and navigating:** acting on objects
    - » using desktop icons
    - » using physical or virtual objects
  - **Exploring and browsing:** finding out and learning things
    - » web style, augmented reality
  - **Proactive computing:** computer acts based on assumed needs of the user
    - » automated filtering (e.g. spam filter)
    - » software agents

Interaction Mode 1: Giving Instructions

- Where users instruct the system and tell it what to do
  - e.g. tell the time, print a file, save a file
- Very common conceptual model, underlying a diversity of devices and systems
  - e.g. Unix shells, CAD, word processors, DVD player, vending machines
- Main benefit is that instructing supports quick and efficient interaction
  - Good for repetitive kinds of actions performed on multiple objects

Interaction Mode 2: Conversing

- Underlying model of having a conversation with another human
- Range from simple voice recognition menu-driven systems to more complex ‘natural language’ dialogues
- Examples include timetables, search engines, advice-giving systems, help systems
- Recently, much interest in having virtual agents at the interface, who converse with you, e.g. Microsoft’s Agents (e.g. Clippy)

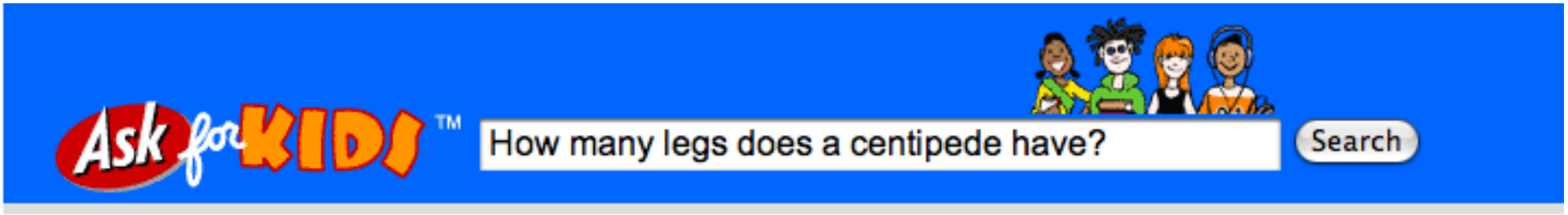
Pros and Cons of Conversational Model

- Allows users, especially novices and technophobes, to interact with the system in a way that is familiar
  - makes them feel comfortable, at ease and less scared

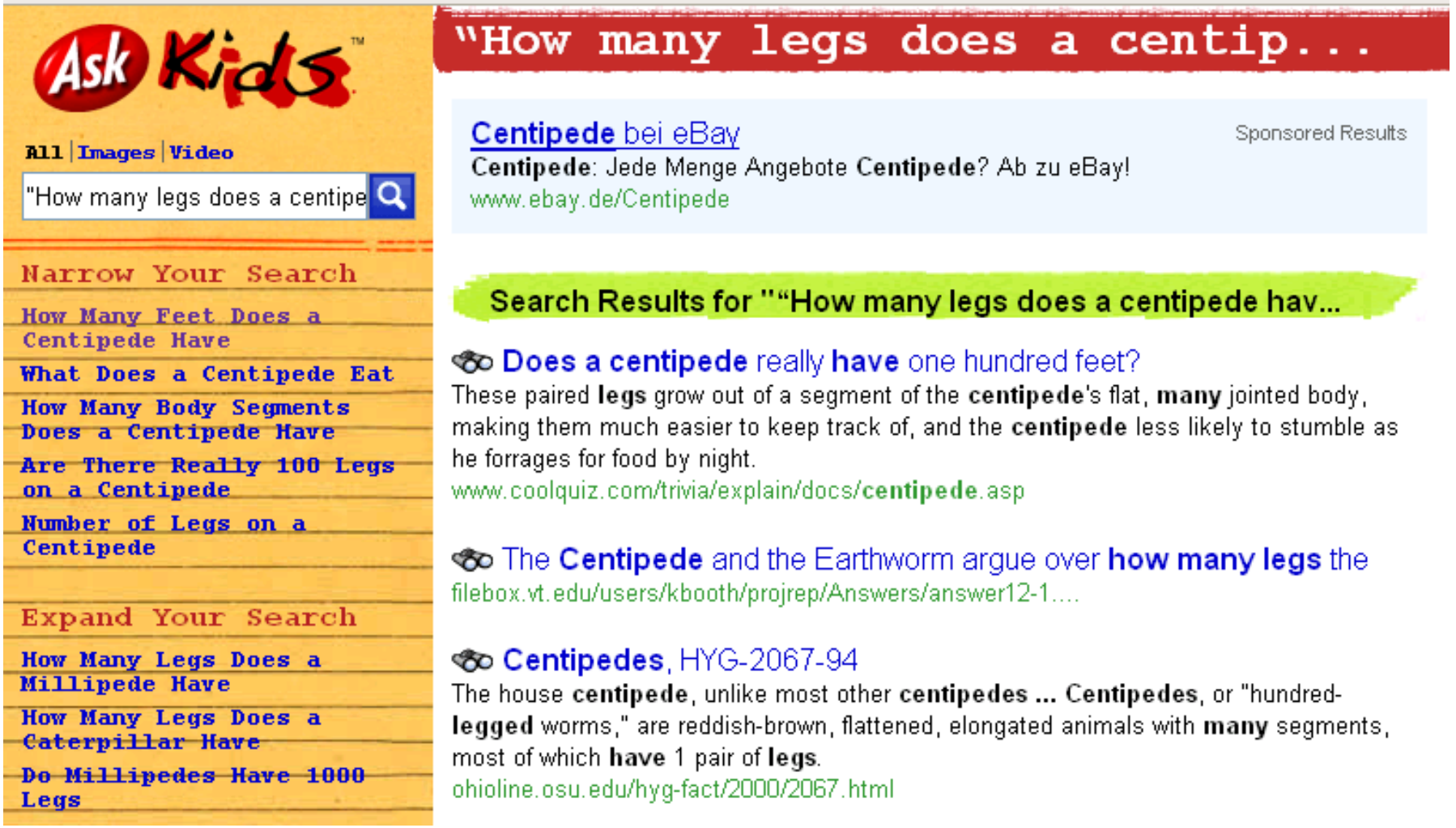
- Misunderstandings can arise when the system does not know how to parse what the user says
  - e.g. a child types into a search engine that uses natural language (http://www.ajkids.com/, http://www.ask.com/) the question:

    “How many legs does a centipede have?”

2006


Ask for KIDS™
How many legs does a centipede have?
Search
We know the answers to these questions:
Where can I see an image of the human
arm and leg muscles (cut views)
?
Go
Where can I find the free online arcade game
Centipede
?
Go
See results from other search engines:
10 matches by FactMonster
Lobster FAQ
Go
©2006 IAC Search & Media · Privacy Policy

2009


Ask Kids
All | Images | Video
"How many legs does a centipe
Narrow Your Search
How Many Feet Does a Centipede Have
What Does a Centipede Eat
How Many Body Segments Does a Centipede Have
Are There Really 100 Legs on a Centipede
Number of Legs on a Centipede
Expand Your Search
How Many Legs Does a Millipede Have
How Many Legs Does a Caterpillar Have
Do Millipedes Have 1000 Legs
"How many legs does a centip...
Centipede bei eBay
Sponsored Results
Centipede: Jede Menge Angebote Centipede? Ab zu eBay!
www.ebay.de/Centipede
Search Results for ""How many legs does a centipede hav...
Does a centipede really have one hundred feet?
These paired legs grow out of a segment of the centipede's flat, many jointed body, making them much easier to keep track of, and the centipede less likely to stumble as he forrages for food by night.
www.coolquiz.com/trivia/explain/docs/centipede.asp
The Centipede and the Earthworm argue over how many legs the
filebox.vt.edu/users/kbooth/projrep/Answers/answer12-1....
Centipedes, HYG-2067-94
The house centipede, unlike most other centipedes ... Centipedes, or "hundred-legged worms," are reddish-brown, flattened, elongated animals with many segments, most of which have 1 pair of legs.
ohioline.osu.edu/hyg-fact/2000/2067.html

Ask | when is easter 2009 | Search | Advanced Search | AskEraser | Settings

Web | Images | News | Videos | Maps | Q&A Beta | More

Showing 1-10 of 10,7
when is e

Answer **Easter Sunday** was Sunday, April 12, 2009 this year.

Search For: Gifts
Go To: Traditions · Boiling Eggs · Encyclopedia · Recipes · Crafts
Related Holidays: Good Friday · Lent

Source

Related Searches

Date of **Easter 2009**
When Is Easter Sunday **2009**
What Day **Is Easter** in **2009**
Calendar for January **2009**
When Is Spring Break **2009**
When Is Passover **2009**
Calendar Year **2009**
When Is Easter 2008
2009 Holidays
When Is Thanksgiving **2009**
How Is the Date of **Easter** Determined
Ash Wednesday

Sponsored Results

Holiday Are
Holiday Are - Buchen Sie zum garantierten Tiefpreis!
www.Holiday-Are.Reisen.de

Web | Images | Videos | Shopping | News | Maps | More | MSN | Windows Live

bing Beta

when is easter 2009

show all | only English | only from Germany

ALL RESULTS

ALL RESULTS

London **Easter** Holiday **2009** | Marc Tönsi
The main reason for our short trip to London had n
Olga is looking out for a room in the main capital c
www.marctv.de/blog/**2009**/04/13/london-**easter**-hol

Photo Gallery: A Somber **Easter** in L'Aqui
translate this page
Photo Gallery: A Somber **Easter** in L'Aquila. 04/13
region of Abruzzo ...
www.spiegel.de/fotostrecke/fotostrecke-41500.htm

What happens when the migration barriers
translate this page
... happens when the migration barriers for 10 new
Integration and Outlook for Temporary and Perman

Google | when is easter 2009 | Search | Advanced Search | Preferences

Web | Show options...

Easter — Date: (Western) April 12, **2009** (Orthodox) April 19, **2009**
According to http://www.holidays.net/**easter**/eadates.htm

When Is Easter 2009? - Date of **Easter 2009**
Easter, the greatest holiday in the Christian calendar, celebrates the Resurrection of Jesus Christ.
catholicism.about.com/od/.../f/**Easter2009**_Date.htm - Cached - Similar -

When is Easter 2009, 2010, 2011, 2012, 2013?
What day does **Easter** fall on in **2009**? **When is Easter** Sunday **2009**?
www.apples4theteacher.com/.../**easter**.../**when-is-easter**.html - Cached - Similar -

When is Easter 2009? - Calendar of **Easter 2009** Dates
When is Easter 2009? This **2009 Easter** calendar includes the dates for Western

Interaction Style 3: Manipulating and Navigating

- Involves dragging, selecting, opening, closing and zooming actions on virtual objects
- Exploits users' knowledge of how they move and manipulate in the physical world
- Examples
  - what you see is what you get (WYSIWYG)
  - the direct manipulation approach (DM)

- Shneiderman (1983) coined the term *Direct Manipulation* (DM), triggered by his fascination with computer games at the time
- Common model in the desktop world

Core principles of Direct Manipulation (DM)

- Continuous representation of objects and actions of interest
- Physical actions and button pressing instead of issuing commands with complex syntax
- Rapid reversible actions with immediate feedback on object of interest

Why are DM interfaces so enjoyable?

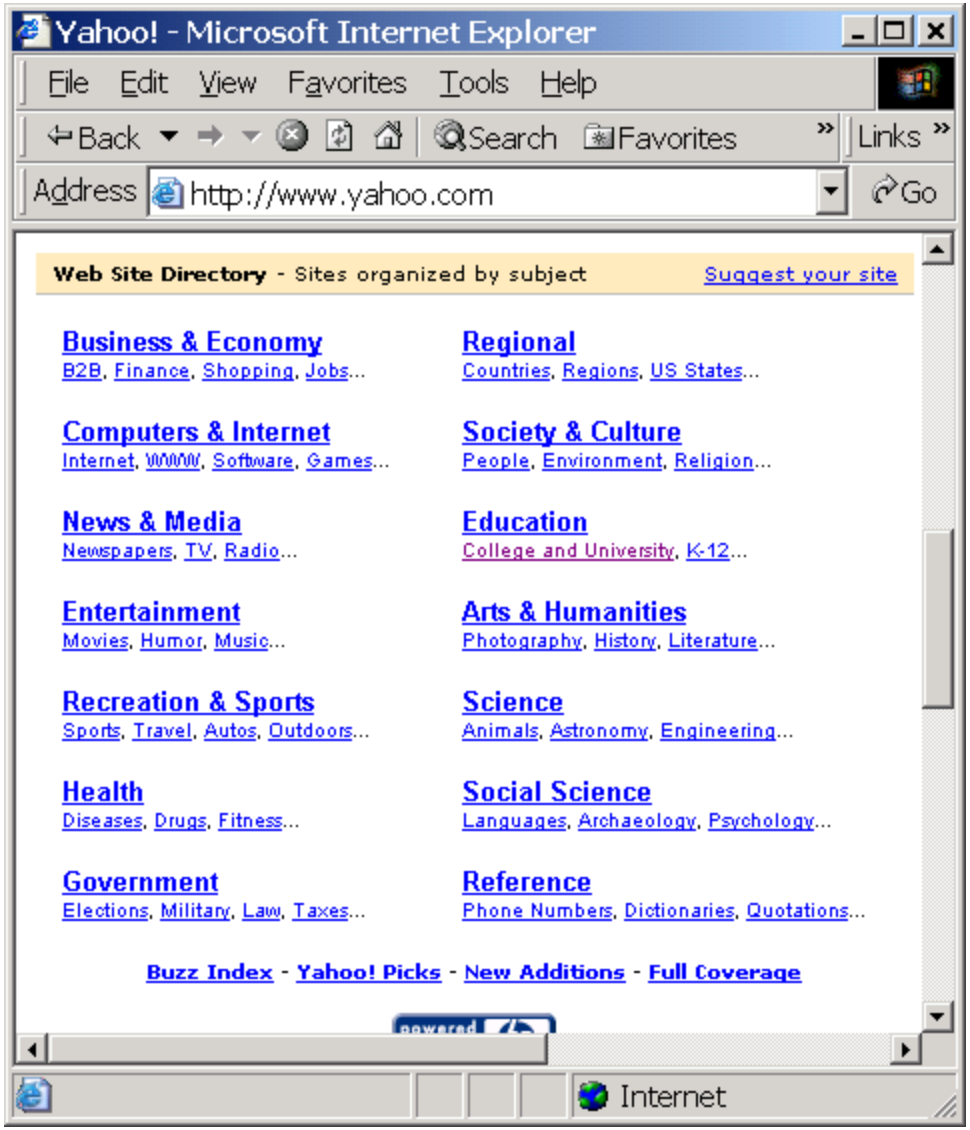
- Novices can learn the basic functionality quickly
- Experienced users can work extremely rapidly to carry out a wide range of tasks, even defining new functions
- Intermittent users can retain operational concepts over time
- Error messages are rarely needed
- Users can immediately see if their actions are furthering their goals and if not do something else
- Users experience less anxiety
- Users gain confidence and mastery and feel in control

What are the disadvantages with DM?

- Some people take the metaphor of direct manipulation too literally
  - Example: ejecting a volume in MacOS
- Not all tasks can be described by objects and not all actions can be done directly
- Some tasks are better achieved through delegating
  - e.g. spell checking
- Can waste extensive screen space
- Moving a mouse around the screen can be slower than pressing function keys to do the same actions

Interaction Style 4: Exploring and browsing

- Similar to how people browse information with existing media (e.g. newspapers, magazines, libraries)
- Information is structured to allow flexibility in the way user is able to search for information
  - e.g. multimedia, web



Interaction Style 5: Proactive Computing

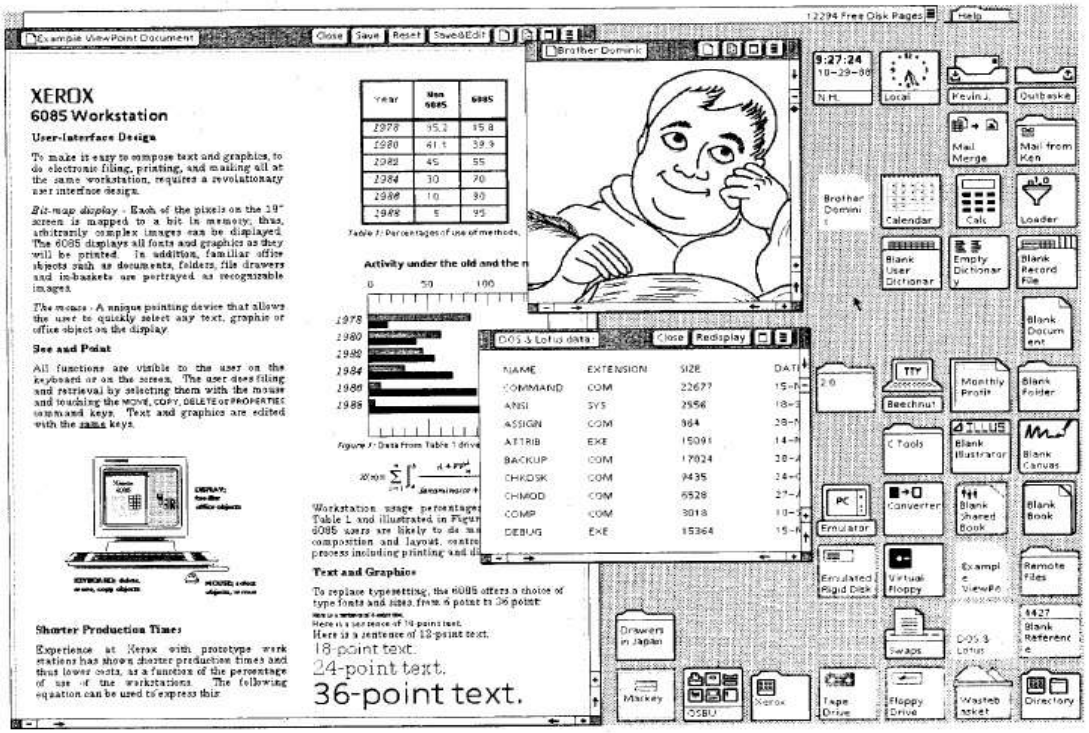
- The system tries to predict the future
- Range from simple sensing systems ...
  - Automatic sliding doors

- ... to methods from artificial intelligence
  - “You might be interested in” / “People who bought X also bought Y”
  - Sampling data, using, e.g. neural networks to find clusters / trends
- Advantages
  - Can speed up processes
  - Relieves the user from some tasks / load
- Problems
  - Can be error-prone
  - Can distract from the task

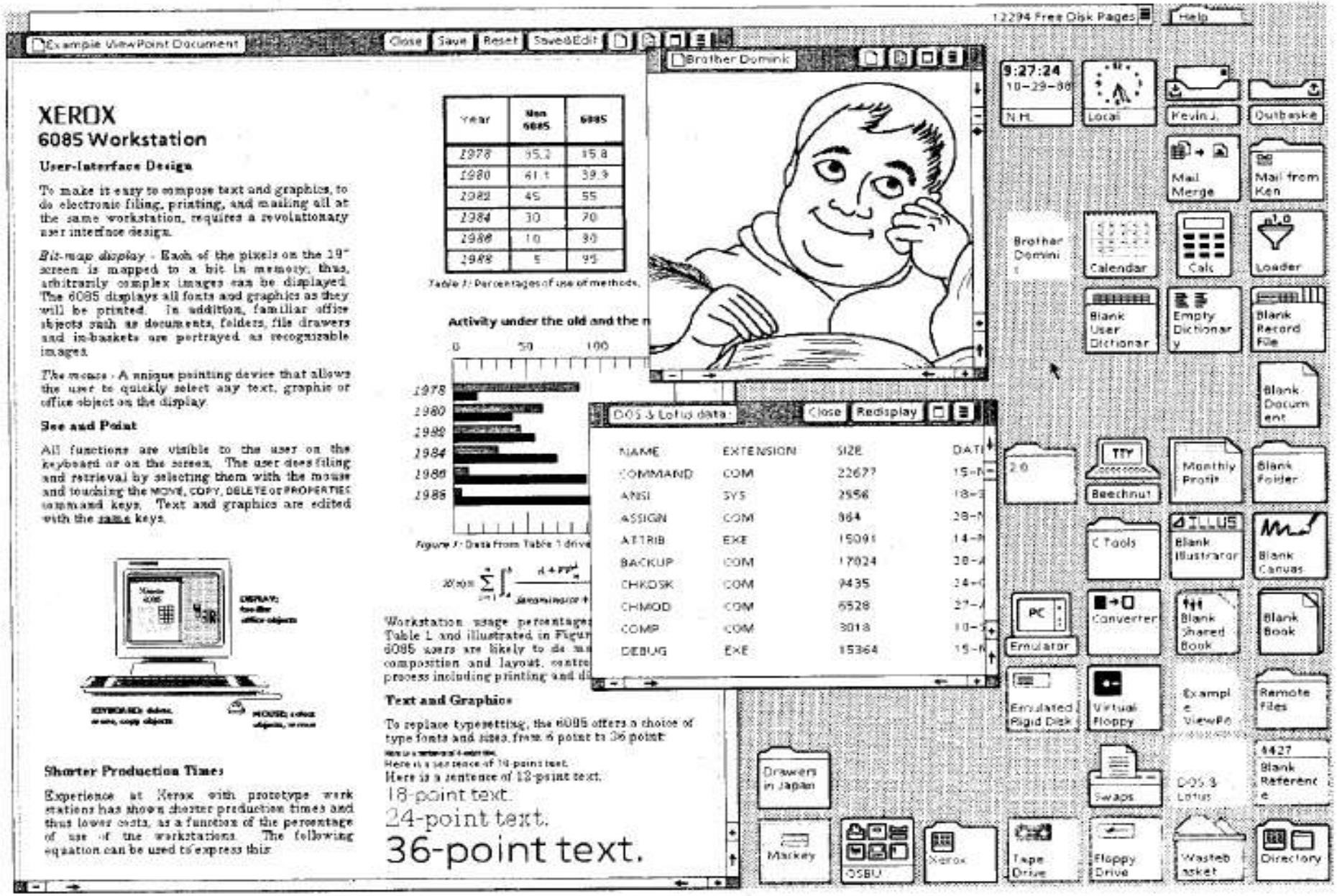
5 Designing Interactive Systems

Conceptual models based on objects

- Usually based on an analogy with something in the physical world
- Examples include books, tools, vehicles
- Classic: *Star* Interface based on office objects



Johnson et al (1989)



Johnson et al (1989)

Interface Metaphors

- "*A direct comparison between two or more seemingly unrelated subjects*"
  - Transfer of knowledge from one domain to another
- Interface designed to be similar to a physical entity but with own properties
  - e.g. desktop metaphor, web portals
- Can be based on activity, object or a combination of both
- Exploit user's familiar knowledge, helping them to understand 'the unfamiliar'

- Benefits:
  - Makes learning new systems easier
  - Helps users to understand the underlying conceptual model
  - Can be very innovative
  - Can lead to accessibility for a greater diversity of users

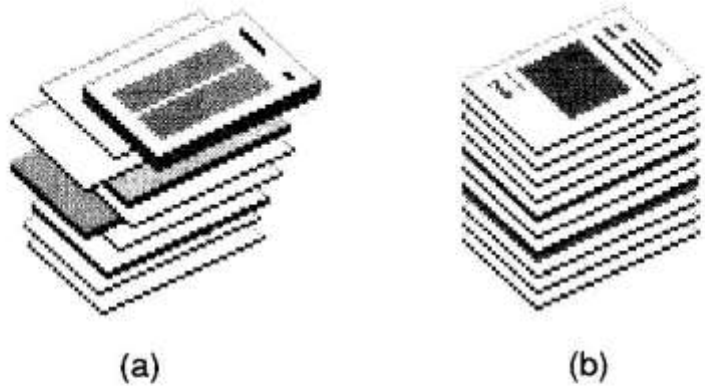
Problems with Interface Metaphors

- Sometimes break conventional and cultural rules
  - e.g. recycle bin placed on desktop
- Can constrain designers in the way they conceptualize a problem space
- Can conflict with design principles
- Forces users to only understand the system in terms of the metaphor
- Designers can inadvertently use bad existing designs and transfer the bad parts over
- Limits designers' imagination in coming up with new conceptual models

Example: Data Mountain
(Robertson, UIST'98, Microsoft)

Example „Pile“ metaphor

(Mander et al., CHI’92, Apple)



(a) (b)

Figure 1. Piles on the desktop. In general, piles can contain various media, such as folders and individual documents. The pile in (a) was created by the user, and is consequently disheveled in appearance. In addition, the system can create piles for the user, based on rules explicitly stated by the user or developed through user-system collaboration. These piles have a neat appearance, as shown in (b), to indicate that there is a script, or set of rules, behind them.

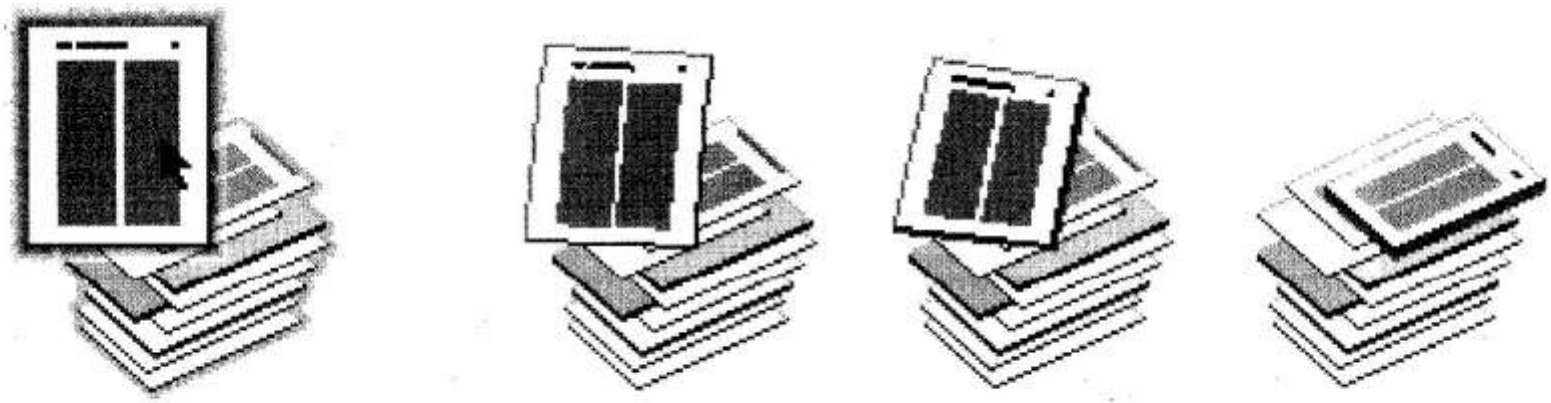
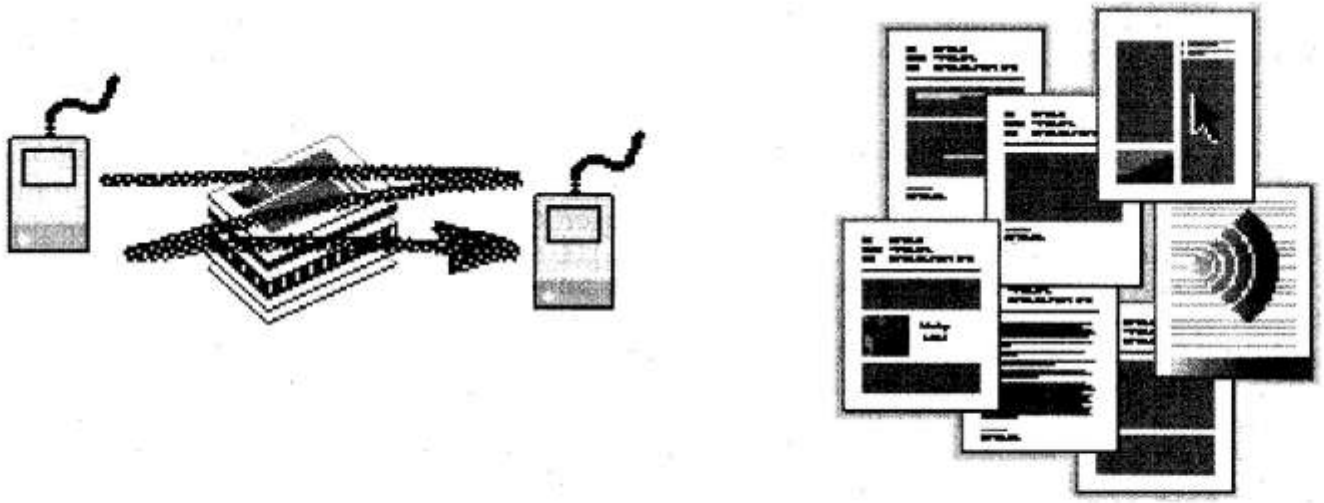
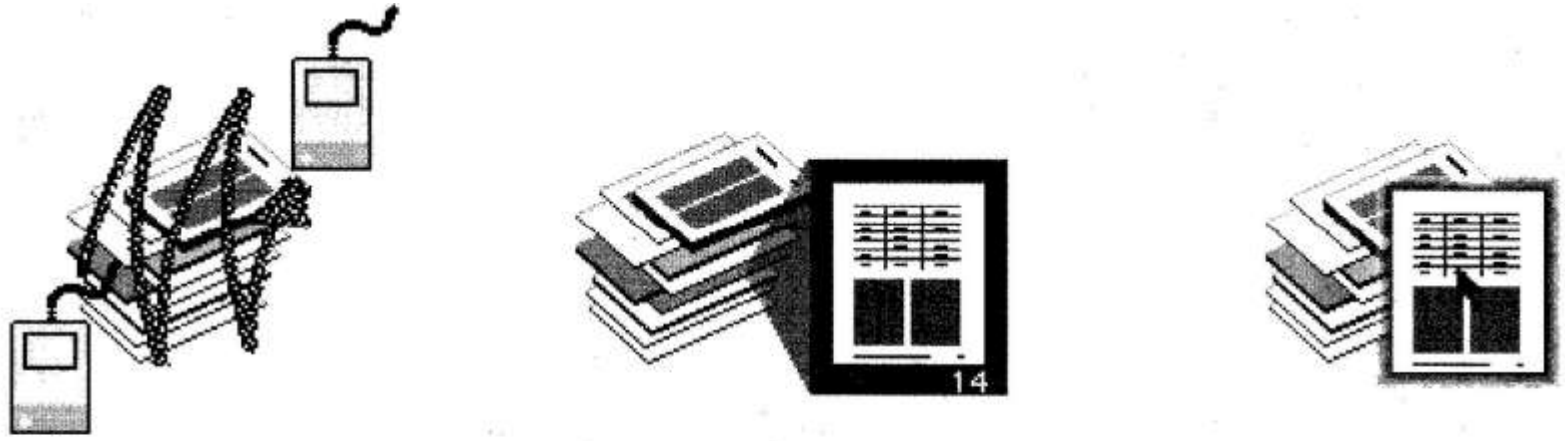


Figure 2. Adding a document to a pile. If a document is positioned over an existing pile, the pile highlights to show that it can accept the new document. When the mouse button is released the document 'drops' onto the pile.



(a) (b)

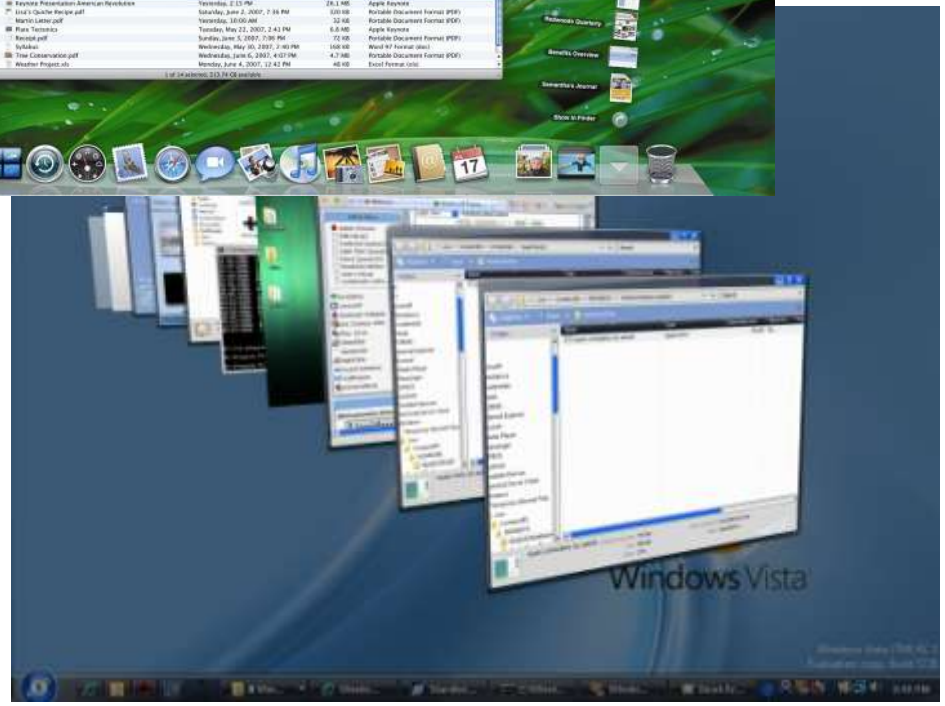
Figure 4. Browsing by spreading out a pile. Gesturing sideways with the mouse pointer, or with a finger in the case of a touch screen, causes the pile contents to spread out. Individual items can now be directly manipulated.



(a) (b) (c)

Figure 5. Browsing while maintaining the pile's structure. Gesturing vertically with the mouse pointer as shown in (a), or with a finger in the case of a touch screen, generates a 'viewing cone' (b) that contains a minature version of the first page of the item under the pointer. This viewing cone will follow the vertical position of the pointer; the miniature changes as the pointer moves over each item. The user can move through the pages of an item in the viewing cone by using the left and right cursor keys on the keyboard. When an item is visible in the viewing cone, it can be selected by clicking the mouse button. The item then appears next to the pile on the desktop, as shown in (c).

15 Years Later: “Flip 3D”, “Cover Flow”, “Stacks”



http://bumptop.com/

Which Conceptual Model is Best?

- **Direct manipulation** is good for ‘doing’ types of tasks, e.g. designing, drawing, flying, driving, sizing windows
- **Issuing instructions** is good for repetitive tasks, e.g. spell-checking, file management
- **Having a conversation** is good for children, computer-phobic, disabled users and specialised applications (e.g. phone services)
- **Exploring and browsing** is good if the task is explorative

- **Hybrid conceptual models** are often employed, where different ways of carrying out the same actions are supported at the interface
  - Toolbar, Menus and Keyboard short cut offer same function
  - Can replace *Expert-Mode* and *Novice-Mode* in the UI

5 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

Design Cycles & Prototyping

- Creating prototypes is important to get **early** feedback
  - From the project team (prototypes help to communicate)
  - From potential users
- Different types of prototypes
  - Low-fidelity prototypes (e.g. paper prototypes)
  - Hi-fidelity prototypes (e.g. implemented and semi-functional UI)
  - Fidelity is referring to detail
- Tools & Methods
  - Sketches & Storyboards
  - Paper prototyping
  - Using GUI-builders to prototype
  - Limited functionality simulations
  - Wizard of Oz

Minimize the time for design Iterations
“Make errors quickly!”

- Idea of rapid prototyping
- Enables the design team to evaluate more design options in detail
- If you go all the way before evaluating your design you risk a lot!
- Sketches and paper prototypes can be seen as a simulation of the real prototype

- Without paper prototyping:
  - Idea – sketch – implementation – evaluation

Slow Iteration

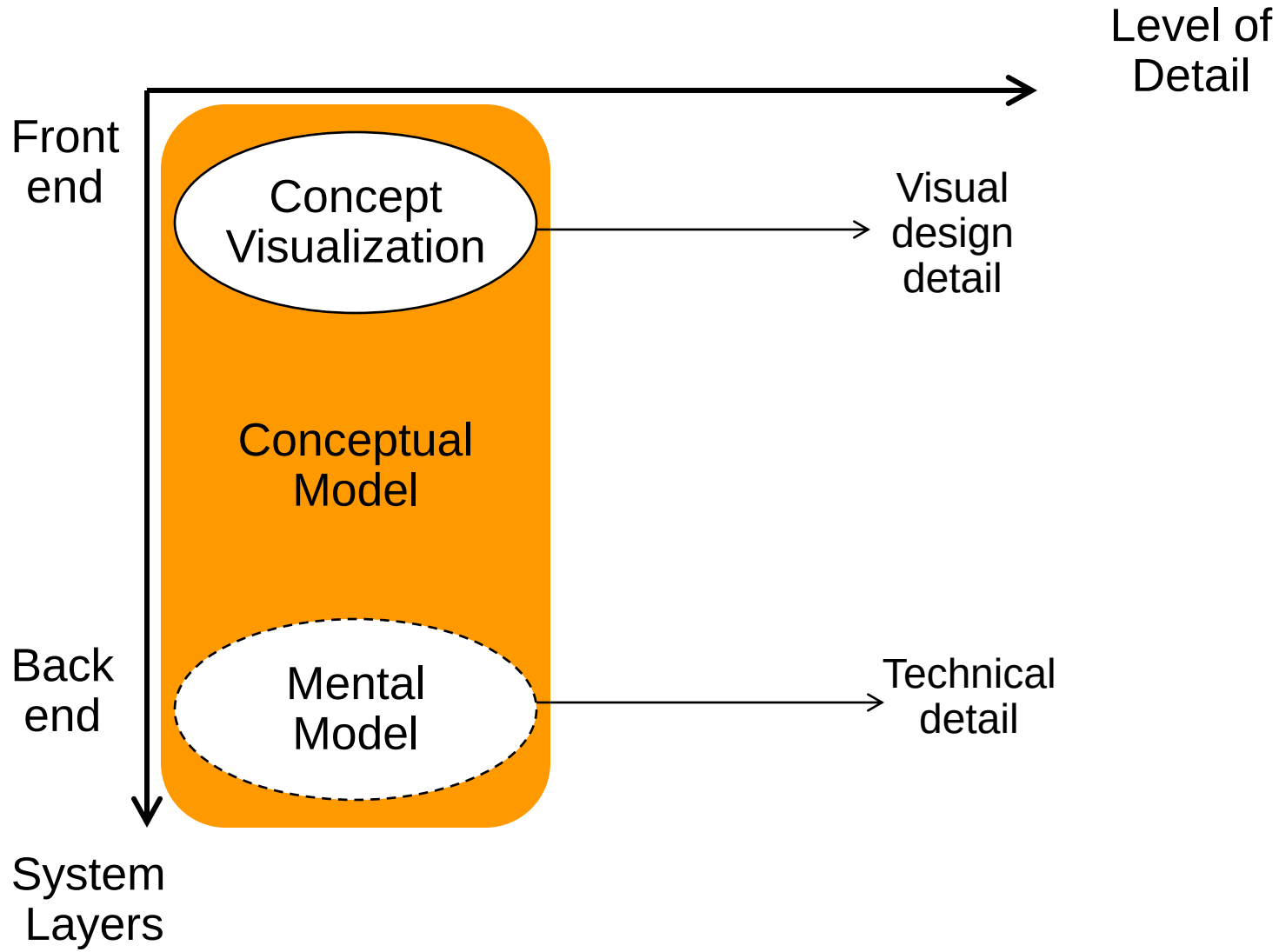
- With paper prototyping:
  - Idea – sketch/paper prototype – evaluation – implementation - evaluation

Quick Iteration **Slow Iteration**

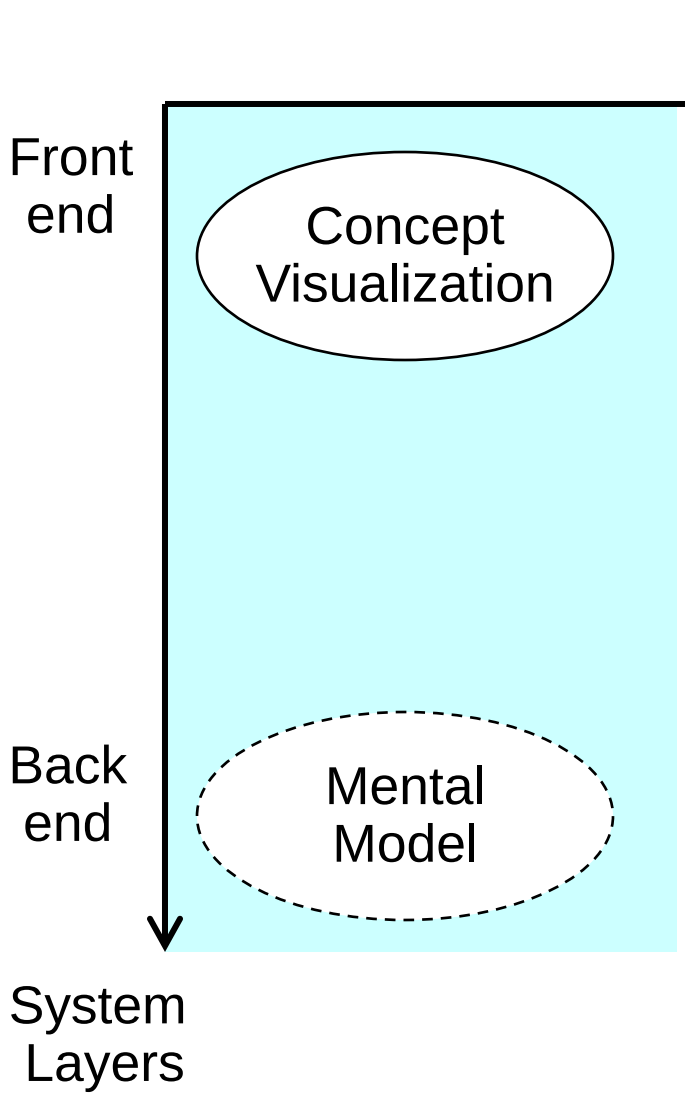
Testing prototypes to choose among alternatives

http://www.combimouse.com

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

Low-Fidelity Prototyping: Paper Prototypes

- Advantages of paper prototypes
  - Cheap and quick – results within hours!
  - Helps to find general problems and difficult issues
  - Make the mistakes on paper and make them before you do your architecture and the coding
  - Can save money by helping to get a better design (UI and system architecture) and a more structured code
  - Enables non-technical people to interact easily with the design team (no technology barrier for suggestions)
  - Provisional presentation invites observers to propose changes

- Get users involved!
  - To get the full potential of paper-prototypes these designs have to be tested with users
  - Specify usage scenarios
  - Prepare tasks that can be done with the prototype

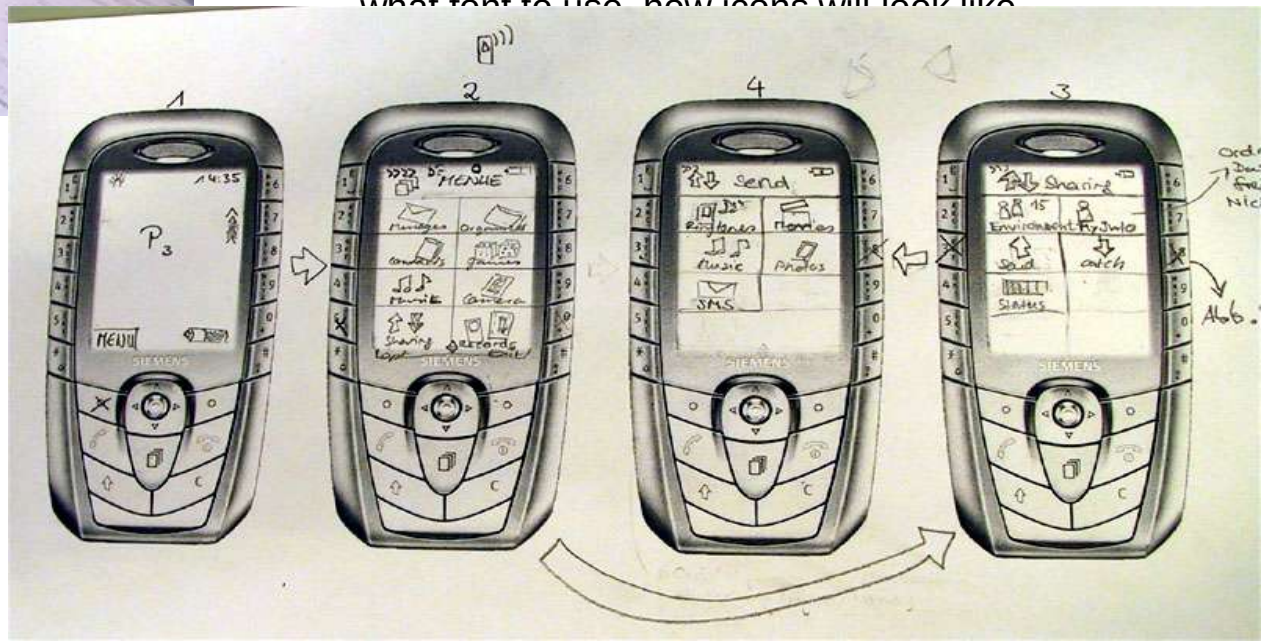
Paper Prototypes

- Specify the set of tasks that should be supported
- Create a paper prototype using office stationery
  - Screens, dialogs, menus, forms, …
  - Specify the interactive behavior
- Use the prototype
  - Give users a specific task and observe how they use the prototype
  - Ask users to “think aloud” – comment what they are doing
  - At least two people
    - » One is simulating the computer (e.g. changing screens)
    - » One is observing and recording
- Evaluate and document the findings
  - What did work – what did not work
  - Where did the user get stuck or chose alternative ways
  - Analyze comments from the user
- Iterate over the process (make a new version)

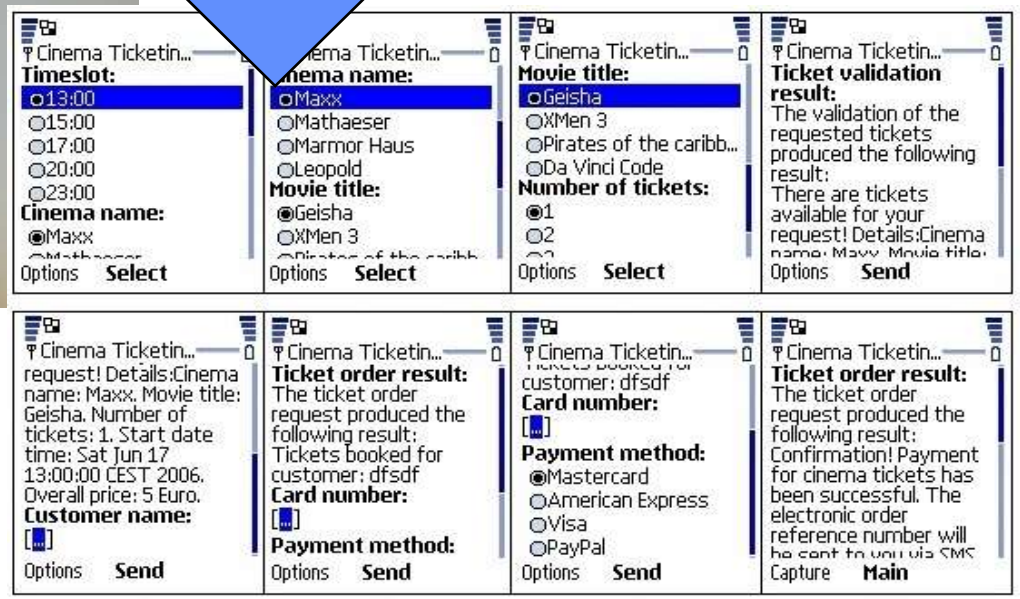
Paper Prototyping – Example I

Sketches & Storyboards

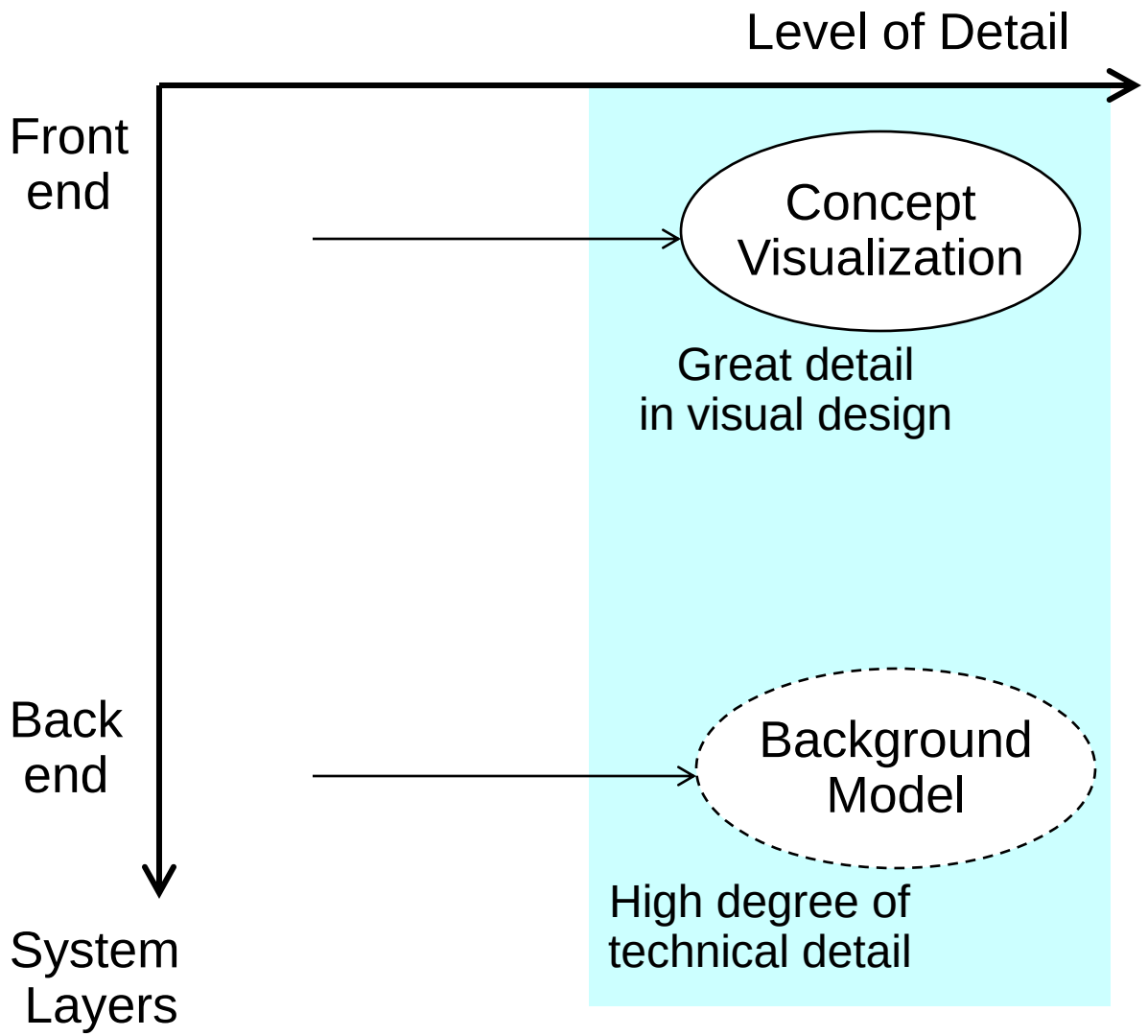
- Storyboards as for movies
  - A picture for each key scene
- Sketch out the application
  - Key screens
  - Main interaction
  - Important transitions
- Helps to communicate and validate ideas
  - Easy to try out different option, e.g. document base vs. application based
- Ignore details, e.g.
  - what font to use, how icons will look like



Paper Prototyping – Example II



High-Fidelity Prototypes



- HTML, Javascript
- Flash, Director
- GUI Builders

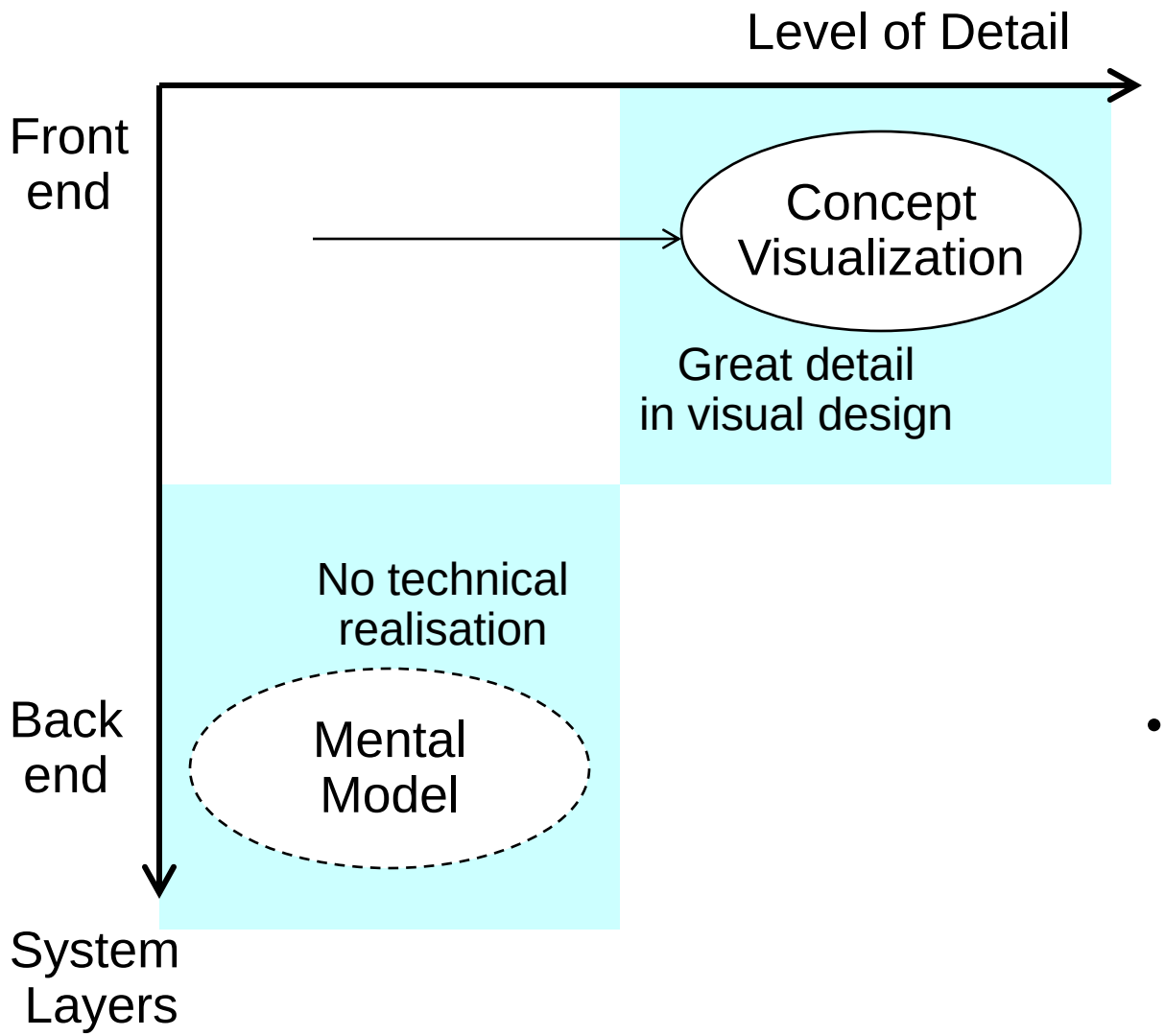
+ Realistic impression
+ Detailed user feedback
+ Timing, interaction

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

High-fidelity Prototypes

- Looks & feels like the final product to the user
  - Colors, screen layout, fonts, …
  - Text used
  - Response time and interactive behavior
- The functionality, however, is restricted
  - Only certain functions work
  - Functionality is targeted towards the tasks (e.g. a search query is predetermined)
  - Non-visible issues (e.g. security) are not regarded
- Can be used to predict task efficiency of the product
- Feedback often centered around the look & feel
- Standard technologies for implementation
  - HTML, JavaScript
  - Flash, Director, presentation programs
  - GUI Builder (e.g. Visual Basic, Delphi, NetBeans)

Cheap High-Fidelity Prototypes



- “Wizard of Oz”

Wizard-of-Oz Prototyping

- "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

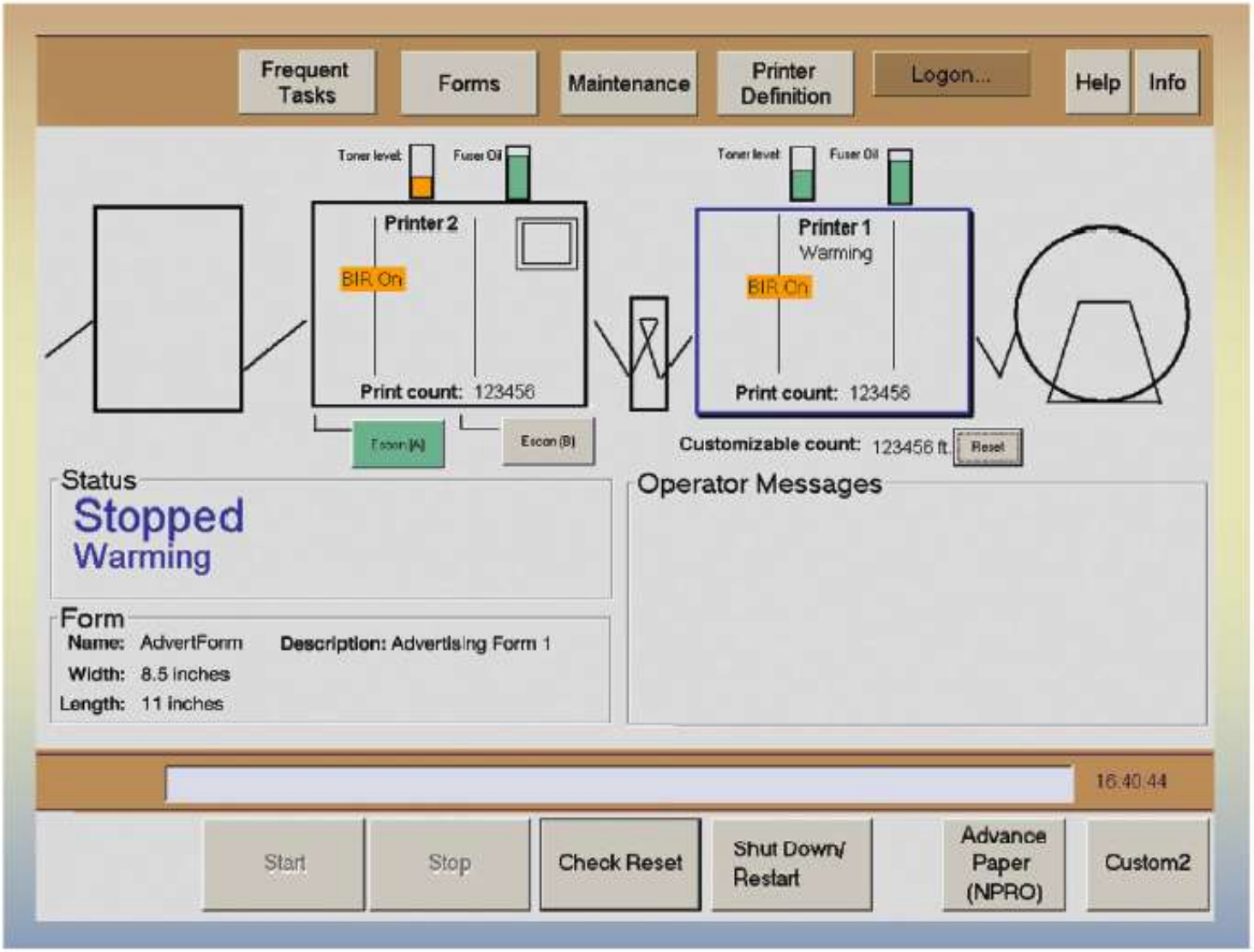
- Provides the user with the experience without extensive implementation effort for the prototype

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
- “Keep it ugly”

R. Van Buskirk and B. W. Moroney: Extending Prototyping,
IBM Systems Journal - Vol. 42, No. 4, 2003 - Ease of Use

“Keep it Ugly” - Example (1)

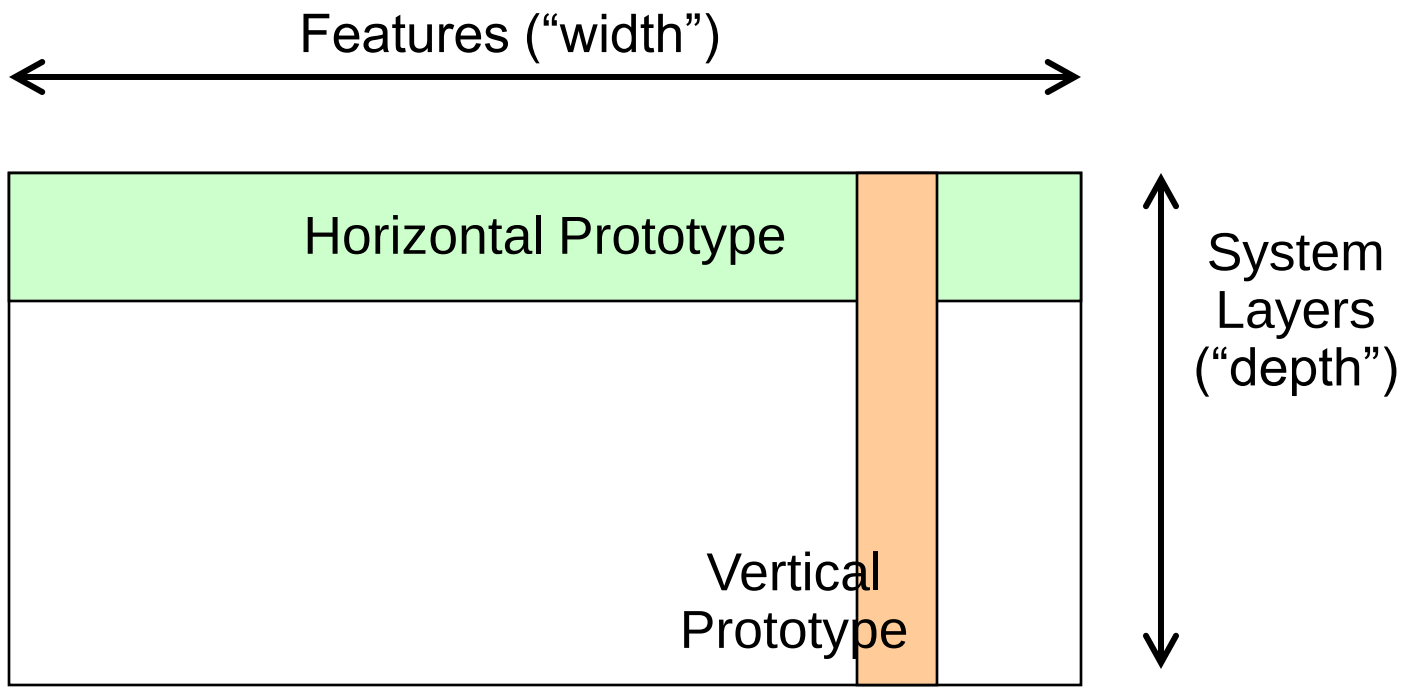


“Ugly” Version

“Keep it Ugly” - Example (2)

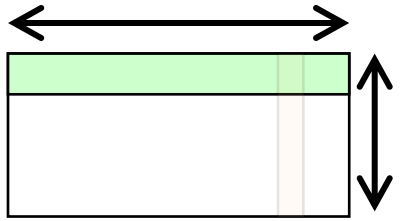
Polished Version

Horizontal and Vertical Prototyping



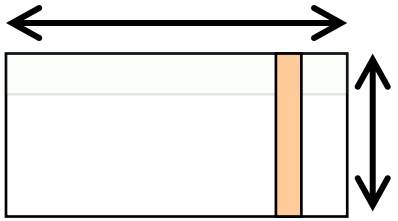
Please note: the meaning of the horizontal dimension is slightly different to previous drawings!

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

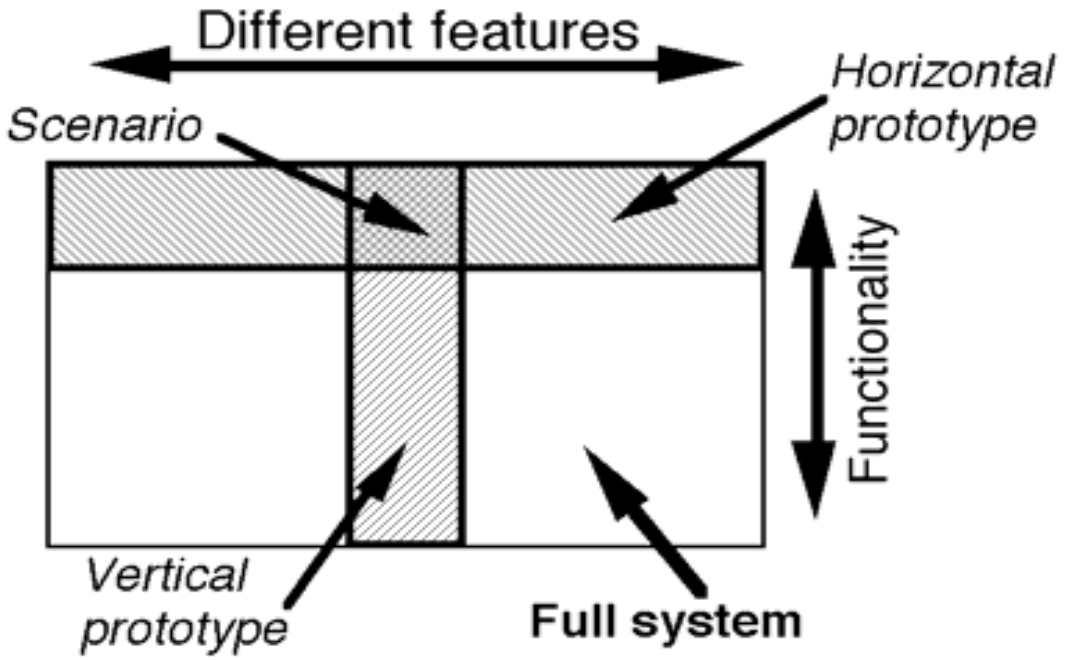
Vertical Prototyping



- 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 used 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 the usage of a specific function
- Example: a new input method 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)

Example: 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. 1987
J. D. Gould , S. J. Boies, S. Levy , J. T. Richards , J. Schoonard. ACM Comm. 30(9)
(http://www.research.ibm.com/compsci/spotlight/hci/p758-gould.pdf)

5 Designing Interactive Systems

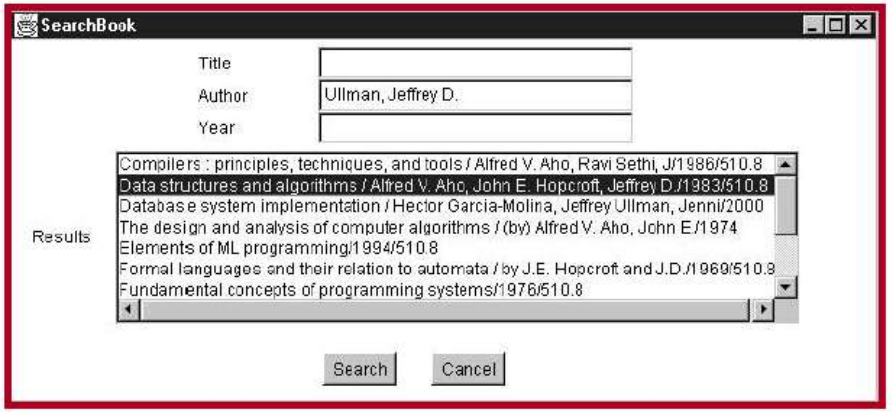
Interactive Systems – What can be described?

- 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
- …

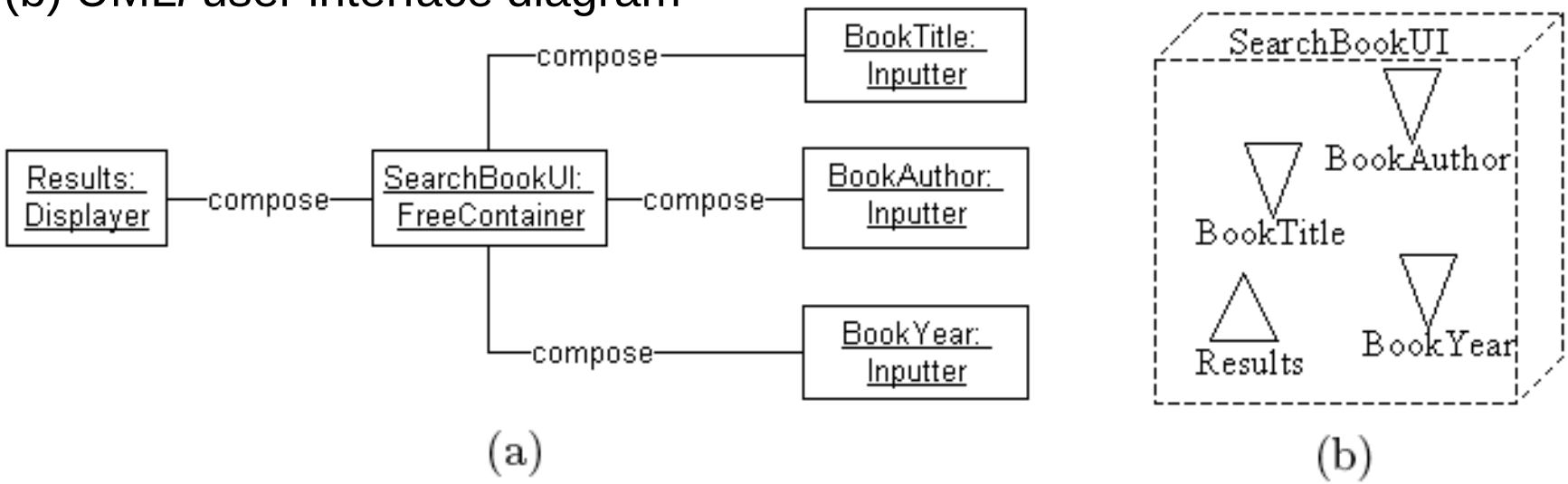
Interactive Systems – How to describe them?

- 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)



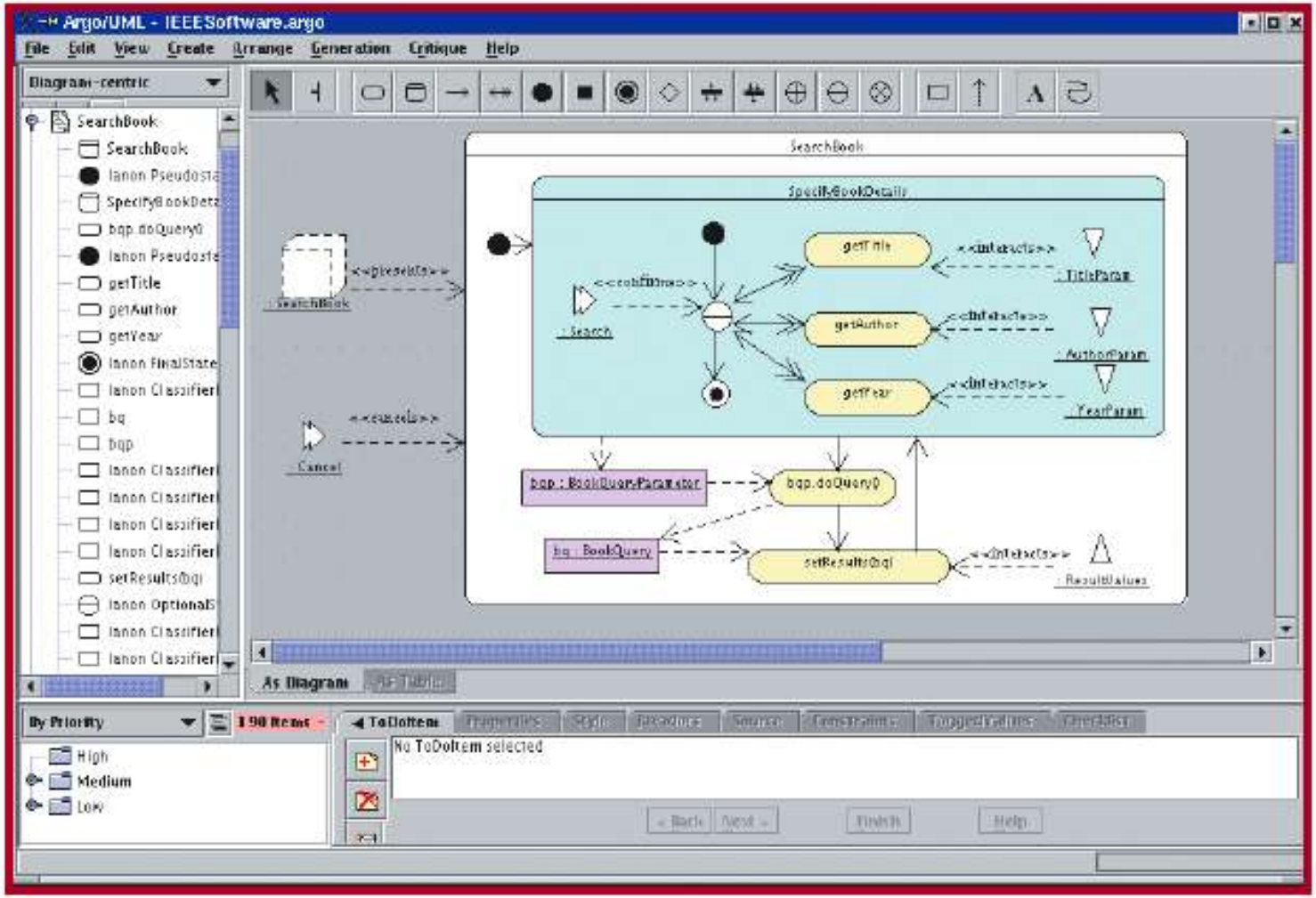
(b) UML*i* user interface diagram



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

UML*i* Example (2)

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

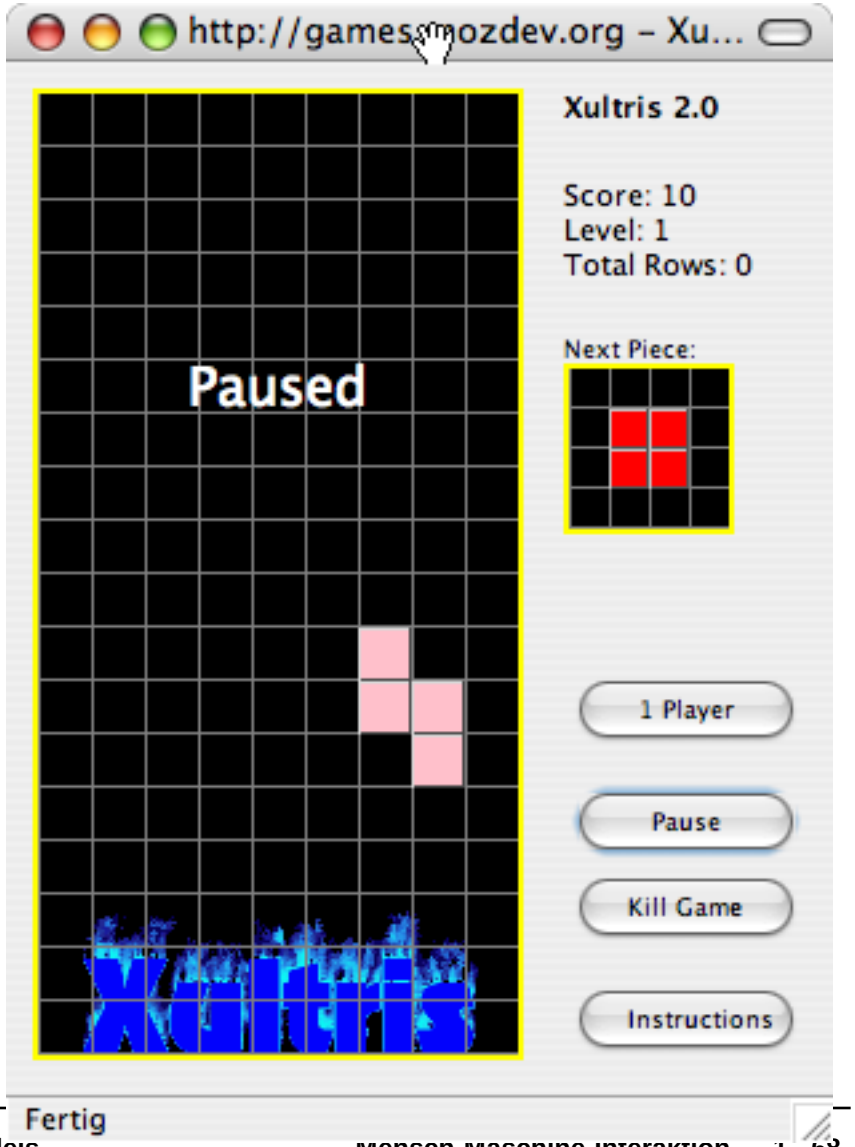


XUL Example

```
<?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/



5 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."

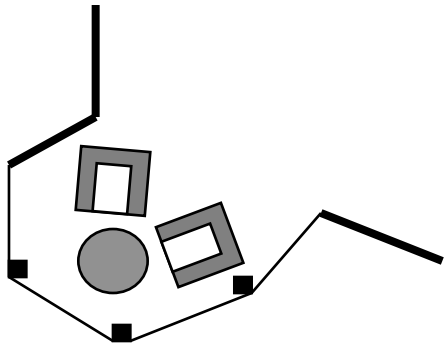
Christopher Alexander et al., A Pattern Language

- 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
    - » behavioral patterns
  - Language-dependent formulations (*idioms*)

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 diagram)
- Discussion
  - Advantages, disadvantages
  - Dependencies, constraints
  - Special cases
  - Known uses

Patterns as Knowledge Representation

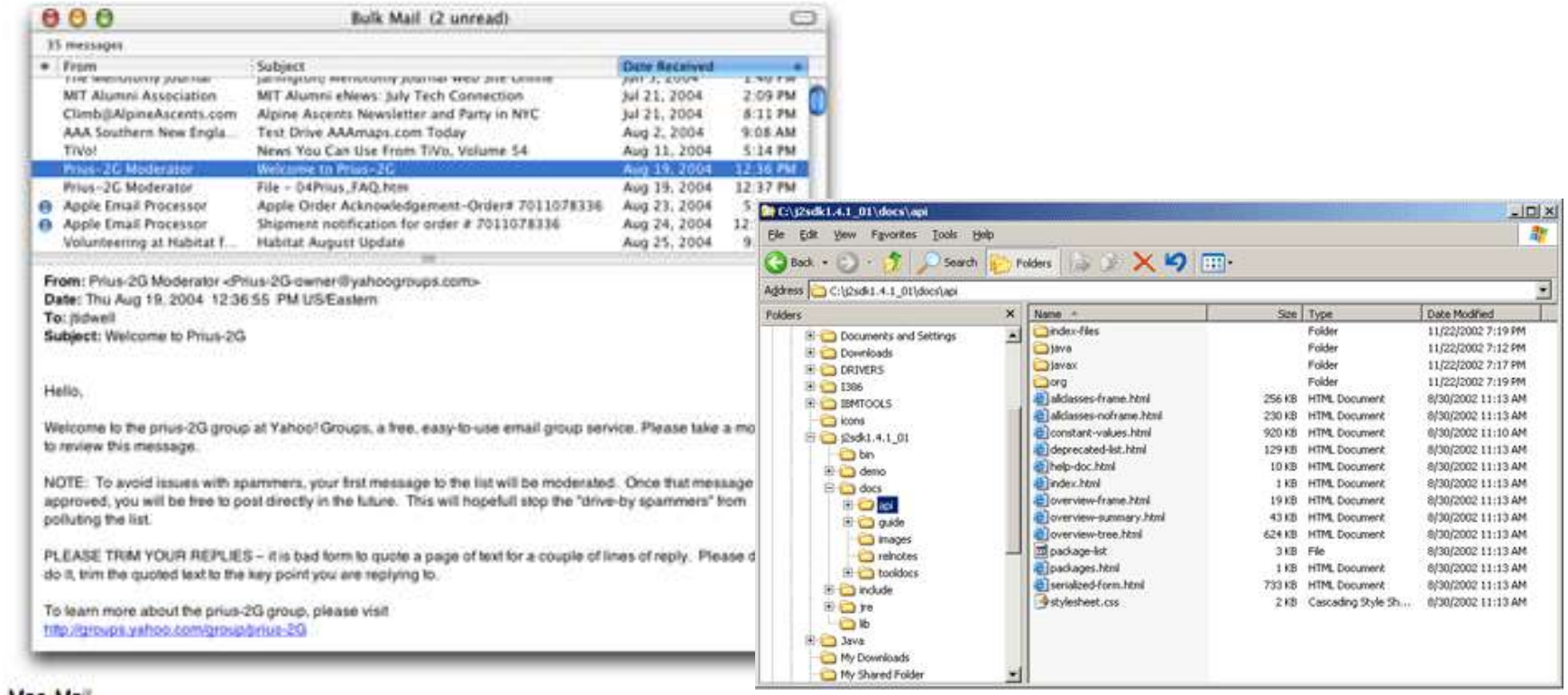
- Many facts and rules of HCI knowledge can be encoded as patterns
  - See e.g. Mahemoff/Johnston 1998
- Examples of pattern encodings of knowledge in HCI:
  - Task patterns
    - » e.g. “Open existing document”
  - User patterns
    - » e.g. “Expert user”, “novice”
  - User interface element patterns
    - » 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 http://www.designinginterfaces.com
- Martijn von Welie 2003:
  - Interaction design patterns for Web design, GUI design, Mobile UI design
  - See http://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)

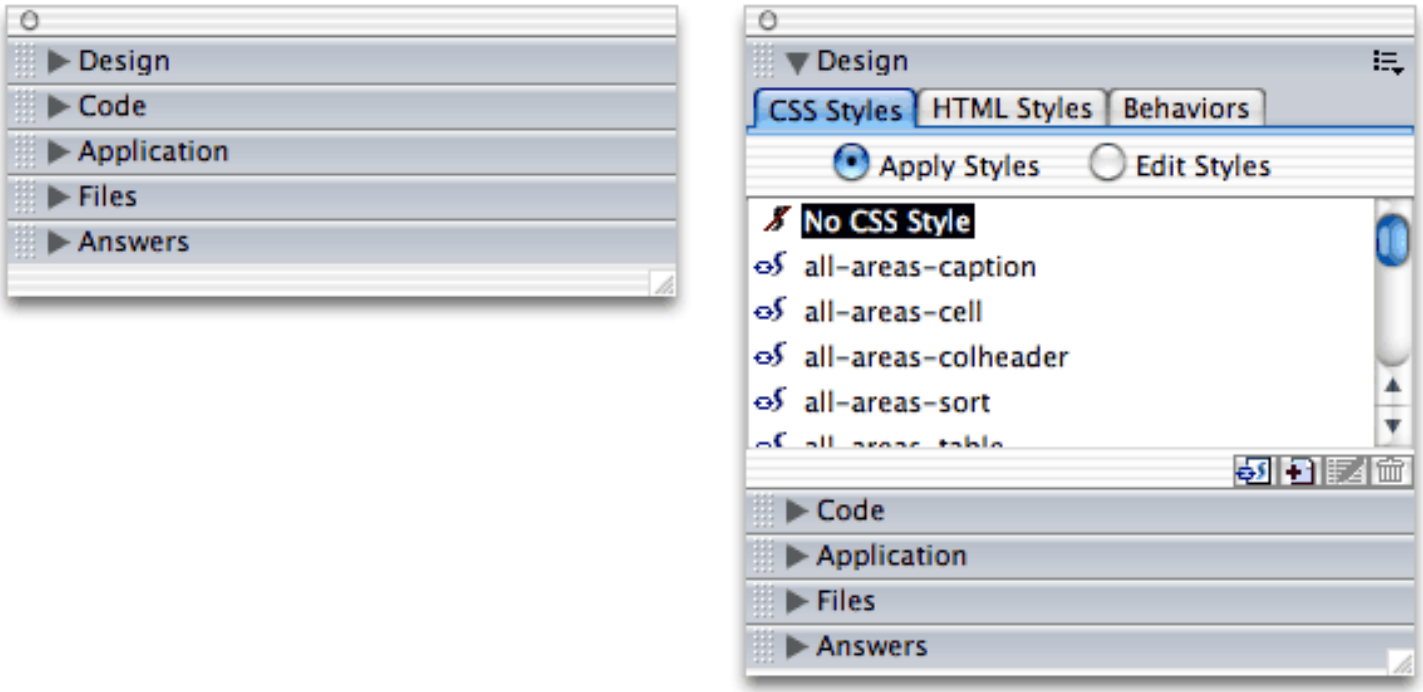
Global Navigation

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

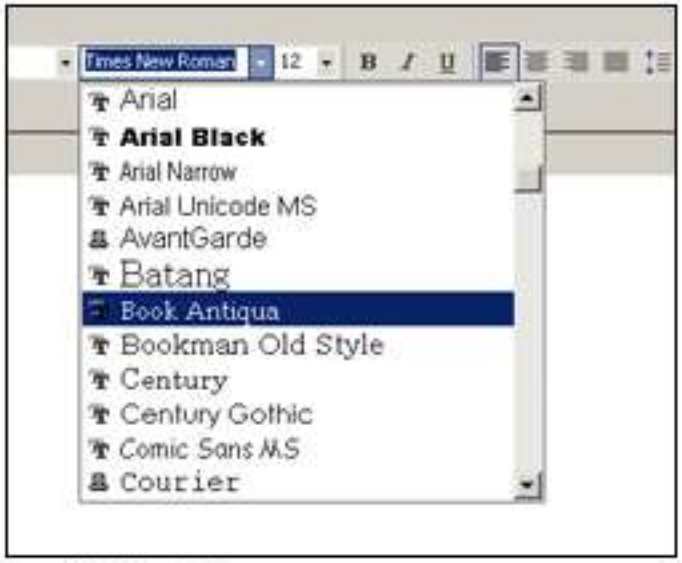


Dreamweaver MX

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

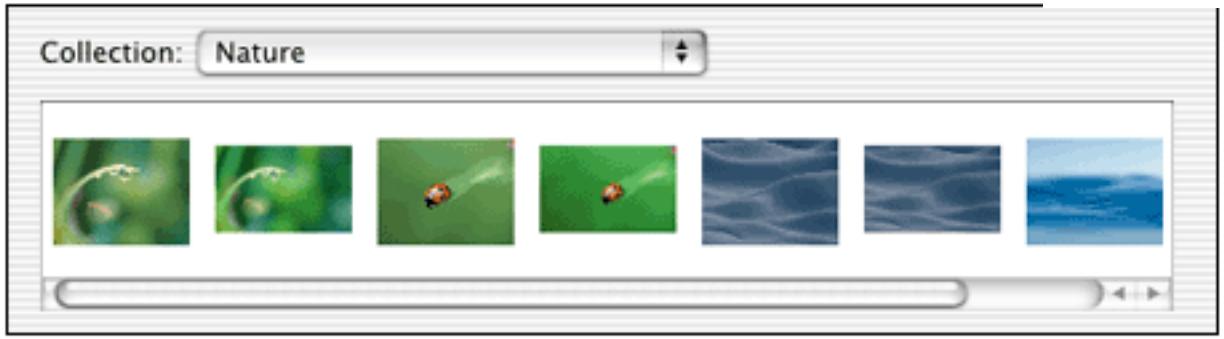
Use when: There's too much stuff to present on the page, but you want it all only one click away. The content is divisible into clearly-named sections, as with Titled Sections and Card Stack.

Example: Illustrated Choices (Tidwell)



From Word for Windows

Illustrated Choices



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.

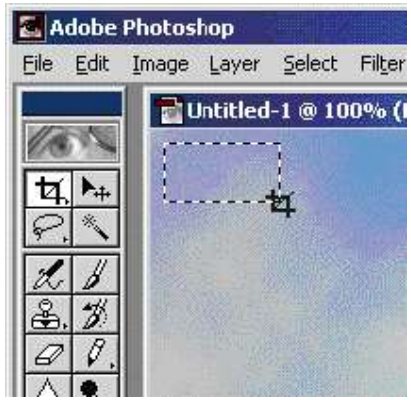
Principle Immediate Feedback (Feedback)

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

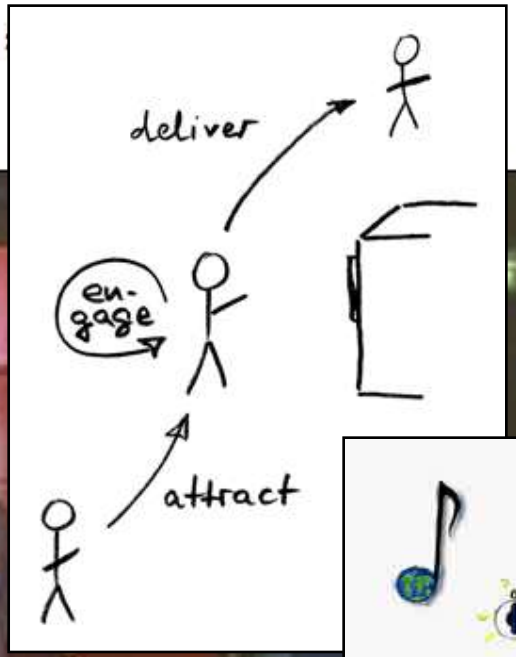
- Not every function may have an icon or shape.
- Completing a function may cause several intermediate states which may also need to be shown.
- The user needs immediate feedback on which function was selected, i.e. which mode/state the system is in.

Solution **Show the interface state in the cursor.**



Example: Attract–Engage–Deliver (Borchers)

H1 Attract-Engage-Deliver



Visitors in the Ars Electronica Center

References

- Alan Dix, Janet Finlay, Gregory Abowd and Russell Beale. (2003) Human Computer Interaction (3rd edition), Prentice Hall, ISBN 0130461091, 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* 42(4), 2003 (http://www.research.ibm.com/journal/sj/424/vanbuskirk.pdf, req. fee)
- M.J. Mahemoff and L.J. Johnston: Pattern Languages for Usability: An Investigation of Alternative Approaches. *APCHI'98*, 1998, p. 25-31. (http://mahemoff.com/paper/candidate/)
- Jennifer Tidwell: Designing Interfaces - Patterns for Effective Interaction Design, O'Reilly 2005, http://www.designinginterfaces.com
- Jan Borchers: A Pattern Approach to Interaction Design, Wiley 2001, http://www.hcipatterns.org/patterns/borchers/patternIndexHtml.html