Models & Metaphors

Jörg Cassens

Medieninformatik II
Contextual Design of Interactive Systems
SoSe 2016

Universität Hildesheim
medieninformatik
IMAI – Institut für Mathematik und Angewandte Informatik

1 Models

Models in Human-Computer Interaction

- Model of a system describes how it works
  - its constituent parts and how they work together to do what the system does
- We are here concerned with three models:
  - The **system model** (sometimes called implementation model) is how the system actually works.
  - The **interface model** (or manifest model) is the model that the system presents to the user.
  - The **user model** (or conceptual model) is how the user thinks the system works.
- There are more models
  - The model the developers have about how they think the user model is like
  - The model the system has about the user (inscribed, in terms of Actor Network Theory)

System Model

- Pixel editing *vs.* structured graphics
  - pixel based as in Gimp, Photoshop
  - vector based as in Inkscape, Visio
- Text file as single string *vs.* list of lines
  - End of line as a normal character, like in Emacs
  - End of line as a special character, like in vi
- Asset Management system
  - List of assets in a flat file
  - Records of assets in a database

Interface Model

- Help system
  - as a binder
  - as a paper clip
- Discussion spaces
  - as a forum
    - threaded
    - linear
  - as a wiki
- On a more general level: learning management systems
  - among peers (kitchen party)
  - with leaders (lectures)

Interface models can be built on different levels (activity design, information design, interaction design) and should connect to existing knowledge (metaphors)

Interface Model Hides System Model

- The interface model should be:
  - Simple
  - Appropriate: reflect user's model of the task (learned from task analysis)
  - Well-communicated
  - Usable
    - Efficient
    - Effective
    - Satisfying
- Implementation model does not have to be exposed
  - Text editor can store list of lines, expose a continuous text
  - A cell phone is not a wired phone, still it acts like one – no need to show things like handover between base stations
  - The interface model should closely reflect the user's model of the actual task

User Model

- Electricity as water
  - Electricity moves through wires like water through pipes
  - Plug it in, use the gadget (water the flowers)
- Thermostat as a valve
  - A thermostat needs to be opened fully to get as much heat (water) out as quickly as possible

User Models may be Wrong

- Sometimes harmless
  - Electricity as water
    - The power cord is no water hose, electrons don't move fast because of "electrical friction"
- Sometimes misleading
  - Thermostat as a valve
    - What is the fastest way to heat up a room? Fully opening the thermostat, like a valve?
    - A thermostat is basically an on/off switch: full power till warm enough

- Ignorant user?
  - The heater in a car actually much more like a valve
  - A dimmer is working as a thermostat: set it to the desired lighting conditions
  - Problem: the thermostat does not effectively communicate its model to the user, in particular, there is not enough feedback

Interaction Styles

- We look at the following Interaction Styles
  - Command language/command line
  - Menus & forms
  - Direct manipulation
    - Touch and Mouse
- Also interesting, but outside the scope today
  - Other forms of graphical interaction
  - 3D-Gestures
  - Natural Language Interfaces
  - Explicit *vs.* implicit interaction
  - Behavioural Interfaces

Command Line Interface

- User types in commands in an artificial language
  - Unix shell (`ls -l *.java`)
  - Search engine query language (`AND, OR`)
  - SQL (`SELECT FROM Book WHERE price > 100.`)
  - SPARQL (`SELECT ?name ?email WHERE {?person a foaf:Person. ?person foaf:name ?name. ?person foaf:mbox ?email.}`)
- Command syntax is important
- Powerful tool with a steep learning curve – find all .tex files that mention the word foo in a given subtree and replace those occurrences with bar
- When designing a command language, the key problem is the syntax
- Task analysis drives the choice of commands, the names you give them, the parameters they have, and the syntax for fitting them together

Menus & Forms

- User is prompted to choose from menus and fill in forms
  - web sites "before Web 2.0"
  - dialog boxes
- Navigation structure is important
  - Wizard: linear sequence of forms
- The navigation structure is the important design problem for menu/form interfaces
- Task analysis tells you what choices need to be available, where they should be placed in a menu tree, and what data types or possible responses need to be available in a form

Direct Manipulation

- User interacts with visual representation of data objects (based on Shneiderman, Designing the User Interface, 2004):
  - Continuous visual representation
    - Verbal or iconic
  - Physical actions or labeled button presses
    - most direct kind of action, analog to real world interaction
    - not everything can be easily mapped – convert a text to bold – so "command actions" are allowed
  - Rapid, incremental, reversible, immediately visible effects
    - within 100ms (why?)
    - drag a bit, see the change
    - physical or logical

Direct Manipulation II

- Examples
  - Files and folders on a desktop
  - Scrollbar
  - Dragging to resize a rectangle
  - Selecting text
- Visual representation and physical interaction are important
- It is powerful since it exploits perceptual and motor skills of the human user
- Some say it depends less on linguistic skills than command or menu/form interfaces
  - Only partly true and for a limited understanding of language

Touch *vs.* Mouse

- While the underlying metaphor does still work, differences between mouse and touch need to be considered
- For touch-based devices, we need to look at
  - Size of elements
    - "Even bigger"
  - Interaction option
    - There is no mouseover
  - New "natural" (cultural?) patterns
    - Swipe, pinch to zoom
- We still have objects to interact with, what about
  - 3D-Gestures
  - Speech Interfaces
  - Implicit Interaction
- Direct Manipulation has served us well, but we need to move on

Comparison of Interaction Styles

- Knowledge in the head *vs.* world
  - CLI needs practice, training, references, manuals
  - M&F put much more information into the world
  - DM has information from affordances and constraints of metaphor
- Error messages regarding the interaction itself
  - DM rarely needs them – try to drag a scroll bar too far
- Efficiency
  - CLI good for experts
  - M&F demand good shortcuts
  - DM if appropriate for task, but mis-using can be labor intensive
- User experience
  - CLI best for experts
  - M&F, DM better for novices, infrequent users

Comparison of Interaction Styles II

- Synchrony
  - CLI synchronous, M&F (user types, system does)
  - DM asynchronous, user can point anywhere, do anything
- Programming difficulty
  - CLI are easy, parsing rigid texts well understood
  - M&F, DM with substantial toolkit support
- Accessibility
  - CLI, M&F easier since both are text based
  - DM much harder

2 Direct Manipulation Cues

Direct Manipulation Cues

- What is the language in which a system communicates its model to the user?
- What cues rely the users on to learn the model – the parts that make up the interface, and how they work together?
- Donald Norman, The Design of Everyday Things (1988), identifies a number of cues
  - Affordances
  - Constraints
  - Natural mapping
  - Visibility
  - Feedback
- Since DM interfaces intend to be a visual metaphor, we look at how these apply to UI

Affordances

- *Perceived* and *actual* properties of a thing that determine how the thing could be used
  - Chair is for sitting
  - Knob is for turning
  - Button is for pushing
  - Listbox is for selection
  - Scrollbar is for continuous scrolling or panning
- Perceived *vs.* actual
  - A paper-mache chair still has a perceived affordance for sitting
  - A pole has no perceived affordance for sitting, but you can sit on it (albeit uncomfortably)
- The DM UI should agree on perceived and actual affordances

Constraints I

- Graphical screen layout relies greatly on conventional interpretations of the symbols and placement
- Different types of constraints:
  - *Physical* constraints are closely related to real affordances
    - it is not possible to move the cursor outside the screen
    - Restricting the cursor to exist only in screen locations where its position is meaningful
  - *Logical* constraints use reasoning to determine the alternatives
    - If we ask the user to click on five locations and only four are immediately visible, the person knows, logically, that there is one location off the screen
    - It is how the user knows to scroll down and see the rest of the page
    - Logical constraints go hand-in-hand with a good conceptual model.

Constraints II

- Different types of constraints (contd):
  - *Cultural* constraints are conventions shared by a cultural group
    - That the graphic on the right-hand side of a display is a "scroll bar" and that one should move the cursor to it, hold down a mouse button, and "drag" it downward in order to see objects located below the current visible set is a cultural, learned convention
    - The choice of action is arbitrary: there is nothing inherent in the devices or design that requires the system to act in this way
    - "Arbitrary" does not mean that any random depiction would do equally well: the current choice is an intelligent fit to human cognition, but there are alternative methods that work equally well.

Natural Mapping

- *Physical arrangement* of controls should match *arrangement of function*
- Best mapping is direct, but natural mappings do not have to be direct
  - Light switches
    - If the switches are arranged in the same fashion as the lights, it is much easier to learn which switch controls which light
  - Stove burners
    - Most stoves have four plates in a square and four controls in a row
  - Car turn signals
    - Up and down instead of left and right, but synchronous to turning wheel
  - DJ audio mixer
    - between turntable
- What is a direct mapping anyway?
  - Rudder of a boat *vs.* steering wheel of a car

Visibility

- *Relevant parts* of system should be *visible*
- If the user cannot see an important control, they would have to
  - guess that it exists, and
  - guess where it is
- Not usually a problem in the real world
  - Look at a bike or a pair of scissors
  - Hiding often takes effort (hidden doors)
  - Design can come in the way
- But takes extra effort in computer interfaces
  - Mouse clicks can be interpreted in arbitrary ways

Feedback

- *Feedback:* what the system does when you perform an action
- When the user successfully makes a part work, it should appear to respond
- Actions should have immediate, visible effects
  - Push buttons depress and release
  - Scrollbars move
  - Drag & drop following the cursor
- Kinds of feedback
  - Visual – see above
  - Audio – clicks made by keyboard (or, artificially, touch screens)
  - Haptic – vibrating touch screens, force feedback 3D-mouse

3 Errors

Anticipating Errors

- Users will make errors
- It is important to take possible errors into account, when designing the system
- Usability guidelines
  - Nielsen: Usability Heuristics
    - Avoid errors
    - Constructive feedback
  - Shneiderman: Golden rules
    - Avoid errors
    - Easy undo
- There are different kind of errors
  - Knowing them makes it easier to recognize the problem

Modeling Human Error

- Description error
- Capture error
- Mode error

cc by-nc-sa freekz0r at flickr

Description Error

- Intended action is replaced by another action with many features in common
- The user intends to do one action, but accidentally substitutes the other
  - Pouring orange juice into your cereal
  - Putting the wrong lid on a bowl
  - Throwing shirt into waste paper instead of hamper
- *Mitigation:* Avoid actions with very similar descriptions
  - Long rows of identical switches
  - Adjacent menu items that look similar

Capture Error

- A sequence of actions is replaced by another sequence that starts the same way
- The user starts executing one sequence of actions, but then veers off into another (often more familiar) sequence
  - Leave your house and find yourself walking to school instead of where you meant to go
  - Vi `:wq` command
- Picture for this: you have developed a mental groove from executing the same sequence of actions repeatedly, and this groove tends to capture other sequences that start the same way
- *Mitigation:* Avoid habitual action sequences with common prefixes

Mode Error

- Modes: states in which actions have different meanings
  - Vi's insert mode *vs.* command mode
  - Caps lock
  - Drawing palette
- Mode errors occur when the user tries to invoke an action that doesn't have the desired effect in the current mode
- *Mitigation:* Avoid modes ☺

Avoiding Mode Errors

- Eliminate modes
- Visibility of mode
  - much harder problem for mode status than it is for affordances
  - the user isn't actively looking for the mode, like they might actively look for a control
  - mode status indicators must be visible in the user's locus of attention → caps lock light does not work well
- Spring-loaded mode
  - the user has to do something active to stay in the alternate mode, essentially eliminating the chance that they'll forget what mode they're in
  - Shift key, drag-and-drop

Avoiding Mode Errors II

- Temporary modes
  - in many graphics programs, when you select a drawing object like a rectangle or line from the palette, that drawing mode is active only for one mouse gesture
  - afterwards, the mode automatically reverts to ordinary pointer selection
- Disjoint action sets in different modes
  - mode errors may still occur, when the user invokes an action in the wrong mode, but the action can simply be ignored

4 Metaphors

Metaphors

- Another way to address the model problem
- Advantage: borrowing a conceptual model the user already has experience with.
- Can convey a lot of knowledge about the interface model all at once
- Examples
  - Desktop
  - Trashcan
- Each of these metaphors carries along with it a lot of knowledge about the parts
  - purposes
  - interactions
- The user can draw on these to make guesses about how the interface will work.

Levels

- Metaphors can be used on different levels
  - A metaphor for how the system works (Activity design in Scenario-Based Development)
    - A discussion forum like a lecture or like the kitchen at a party
  - A metaphor for how information is displayed (Information Design)
    - Free space left on hard disk as a partially full bar
  - A metaphor for the interaction offers (Interaction Design)
    - Dragging a file into the waste paper basket

Dangers of Metaphors I

- Hard to find
  - Particularly with real-world objects
  - Basic rule for metaphors is: use it if you have one, but don't stretch for one if you don't
- Deceptive
  - Leading users to infer behavior that your interface doesn't provide
  - Looks like a book, but can I write in the margins?
- Constraining
  - Strict adherence to the desktop metaphor wouldn't scale, because documents would always be full-size like they are in the real world.

Dangers of Metaphors II

- Breaking the metaphor
  - Your interface is presumably more capable than the real-world object, so at some point you have to break the metaphor
  - Nobody would use a word processor if really behaved like a typewriter
  - Features like automatic word-wrapping break the typewriter metaphor, by creating a distinction between hard carriage returns and soft returns
- Use of a metaphor doesn't excuse bad communication of the model
  - If it looks like a book, but you don't show how to open it, the metaphor does not help

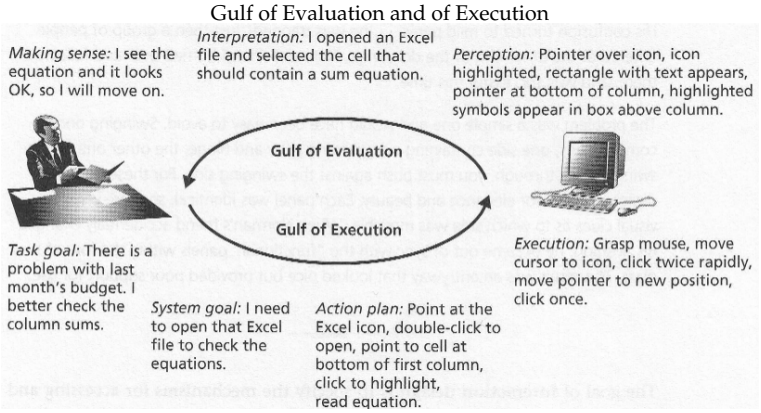
5 Norman's Gulfs

Stages of Interaction

- There are lots of places where interaction between human and machine can go wrong
  - Perception
  - Cognition
  - Action
- Stages of action proposed by David Norman (1986)
- Two gaps
  - **Gulf of Evaluation:** the "cognitive distance" between what is displayed and the user's mental representation
  - **Gulf of Execution:** distance between the user's goals and the procedures and actions provided to pursue this goals

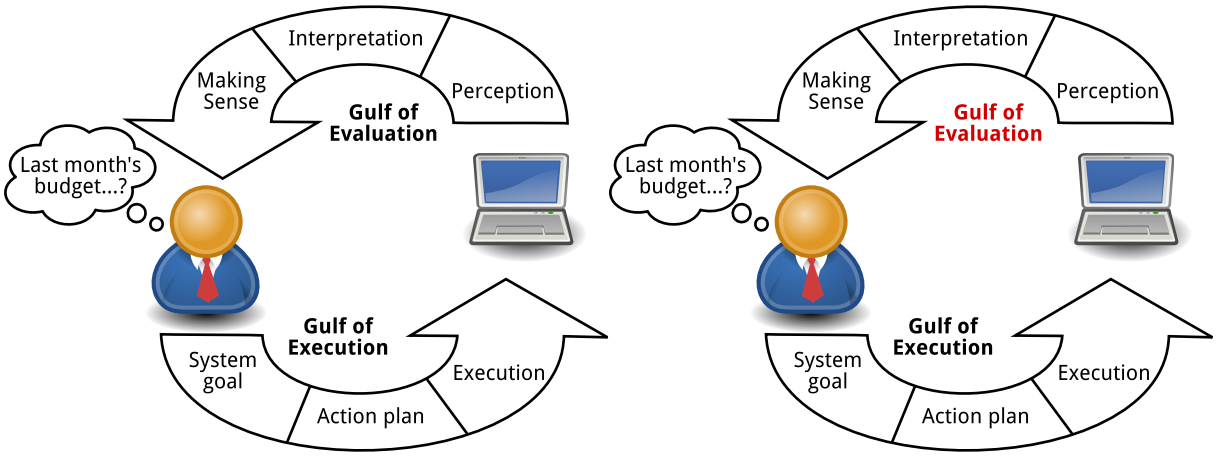
Norman's Gulfs

Gulf of Evaluation and of Execution



Source: Rosson & Carroll, 2002

Gulf of Evaluation



Information-Design

- The **objects** and **actions** possible in a system are **represented** and **arranged** in a way that facilitates **perception** and **understanding**
- Includes the design of
  - Application screens
  - Web pages
  - Menus
  - Dialogs
  - Icons
- Other modalities
  - Sound
    - Speech synthesis
  - Tactile
    - Force feedback game controls
  - Visual
    - 3D-displays (geowall)
- Addresses the Gulf of Evaluation

Perception

- Guiding viewers to see the structure in an information display
- Gestalt principles
  - Similarity
  - Closure
  - Area
  - Symmetry
  - Continuity
  - Proximity
- Organization

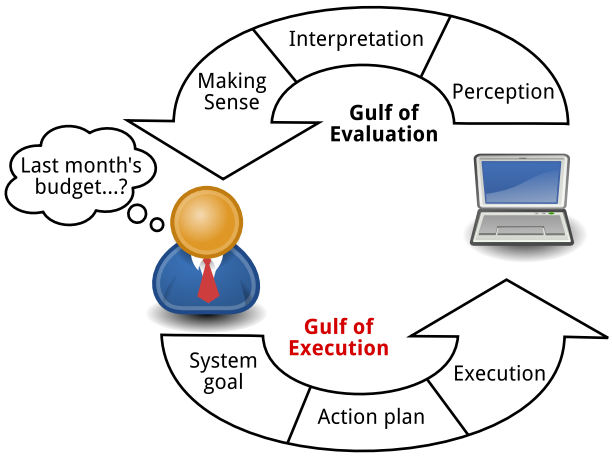
Interpretation

- During interpretation, the content is recognized as input fields for data, choices for presentation, etc.
- Key concepts:
  - Familiarity
    - Connect to existing knowledge
  - Realism and abstraction
    - Realistic – easy to recognize, but maybe too particular?
    - Abstract – harder to recognize, but maybe more general?
  - Recognizing Affordances
    - Show the user what can be done and where

Making Sense

- Relating the information to what they currently understand about their task
- Evaluating whether and how it addresses their active goals and interests
- Consistency
  - Visual design program: fonts, logo, colors
- Visual metaphors
  - Is it map? Does it work as a map?
- Information models
  - hierarchies, maps
- Dynamic display
  - redisplay or animation

Gulf of Execution



Interaction-Design

- Goal: specify the mechanisms for accessing and manipulating task information
- **Information design** focuses on determining which task objects and actions to show and how to represent them
- **Interaction design** tries to make sure that people can **do the right things at the right time**
- Broad scope:
  - Selecting and opening a spreadsheet
  - Pressing and holding a mouse button while dragging it
  - Specifying a range of cells
- Addresses the Gulf of Execution

Task & System Goal

- Task goal
  - the task the user really wants to achieve
- System goal
  - translate the real world goal into a system goal
  - UI-Models/Interaction style
  - opportunistic goals

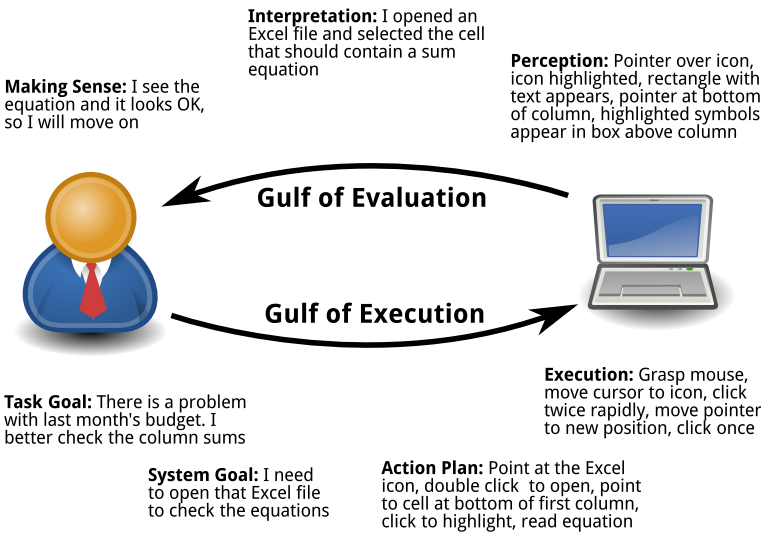
Action plan

- steps needed to achieve a system goal comprise an action plan
- task analysis → idealized action plan
- Other key concepts:
  - mental models
  - making actions obvious

Execution

- final phase: execution of plan steps
- articulatory directness: mapping of physical movement with a device to a task's input requirements
  - Mouse, keyboard, trackball, joystick
- Feedback and undo
- optimizing performance
  - efficient interaction
  - sane defaults

Gulf of Evaluation



Helpdesk

☞ Medieval Helpdesk