PV182
Human-Computer Interaction

Lecture 3

Mental Models

Fotis Liarokapis
liarokap@fi.muni.cz

01st October 2018

High level models of human-computer behaviour

Are there theories that describe how people interact with computers?
What is Shneiderman's syntactic/semantic model?
What is Norman's stages of human interaction?

Publish or perish …

- *Dilbert blog comment meltdown (thanks to AskTog)*:
- The blog allowed writing "draft". Drafts were only available to select readers. You could "publish" the draft. After Scott Adams "published" the draft, he didn't need the draft, so he deleted it. However, there was no difference between the draft and the published version in the software. Scott Adams deleted his own work…and 500 comments from users on it.

Human errors – Sliperies and Mistakes

- Slippery
  - ☺ – Understand system and goals
  - ☺ – Well formulated (chosen) action
  - ☹ – Incorrect (improper) action
- Mistake
  - ☹ – Not even the right goal !
- Repair ?
  - – Slippery – better UI design
  - – Mistake – better system´s understanding

Low-level models of human-computer behaviour

- Some low-level theories can be used to predict human performance
  - – Fitt's law
    - Time to select an item with a pointing device
  - – Keystroke level model
    - Sums up times for keystroking, pointing, homing, drawing, thinking and waiting

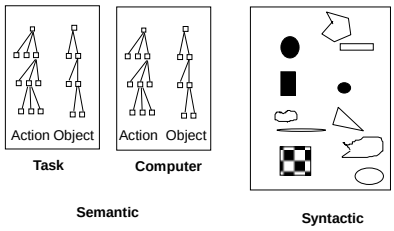
Not in this lecture, later …

High-level models of human-computer behaviour

- Developing Theories in HCI
  - – Must explain and predict human behavior in the human-computer system
  - – Must work in a wide variety of task situations
  - – Must work within broad spectrum of system designs and implementations
- General models that explain human behavior with machines
  - – Syntactic/semantic model (Shneiderman)
  - – Stages of interaction (Norman)
  - – All of psychology!

Syntactic/semantic model of user knowledge

- A high level model of interaction, developed by Ben Shneiderman

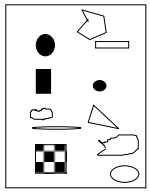


Syntactic Knowledge

- The rules or combinations of commands and signals
  - Seen as device-dependent details of how to use system
  - Examples:
    - backspace key — delete previous character
    - right mouse button — raise menu
    - grep < word> <file> — finding a word in a file
    - tab — moves to next field in a form
    - <cntl> X! — enlarges window by one line (gmacs)

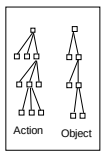
Syntactic Knowledge .

- User problems with syntactic knowledge
  - syntactic details differ between (and within!) systems
    - little consistency–> arbitrary
    - e.g. leaving mail reading in gmacs
      - "q" to quite mail system
      - "<cntl> x <cntl>c" to quit gmacs
      - "<cntl> d or logout" to quit Unix
  - hard to learn
    - acquired by rote memorization
    - repeated rehearsals to reach competency
    - must be frequently applied for retention over time
  - easily forgotten
    - expert/frequent users ok
    - novice/casual users troubled by syntactic irregularities



Semantic knowledge: Computer concepts

- The meaning behind computer concepts
- Usually follows a hierarchical structure
  - high level concepts decomposed to many low level concepts
  - objects
    - stored information as directories and files as name, length, creation date, owner,...
  - actions
    - saving a file, creating backups, verify access control, etc.
- How it works
  - people learn computer concepts by
    - meaningful learning
    - demonstrations
    - explanations of features
    - trial by error
    - model of concepts (abstract, concrete, analogical)
      - e.g. file hierarchies are like file/folder systems

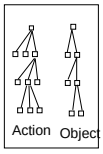


Semantic knowledge: Computer concepts .

- Properties of semantic knowledge (computer concepts)
  - Relatively stable in memory
    - High level concepts
    - Logical structure
    - Cognitive model produced
  - Usually transferable across computer systems
    - But not always!
- Problems
  - Many people now using computers are not computer scientists!
  - Must be trained in "computer literacy"
  - People prefer to concentrate on task, not on computer knowledge

Semantic knowledge: Task concepts

- The meaning behind the task concepts
  - Is independent of the computer
- Similar in mechanism to computer concepts
- Examples
  - How to write a business letter
    - Format concerns
    - Stylistic concerns
    - Paragraph structure, etc.
  - creating lecture notes



What Syntactic/Semantic Model reveals

- Mapping between three items is extremely important
  - Task semantics to computer semantics to computer syntax
    - task semantics: write letter
    - computer semantics: open a file, use editor, save it to disk
    - computer syntax: select menu items, key strokes for formatting,...
  - Bad mapping: using latex to write letter
    - aside from task semantics, must also know semantics/syntax of:
      - text editor
      - latex
      - Unix compiling and printing sequence (to typeset and print)
  - Relatively good mapping: trashcan to throw away files
    - must know mouse syntax of selecting and dragging
    - computer semantics almost analogous to task semantics

Guideline suggested by syntactic/semantic model

- Reduce the burden to the task-oriented user of learning a separate computer semantics and syntax
- Methods
  - computer semantics
    - metaphors allow computer artifacts to be represented as task artifacts
      - e.g. office workers: files/folders represent hierarchical directory/file systems
    - information hiding
      - don't force people to know computer concepts that are not relevant to their work

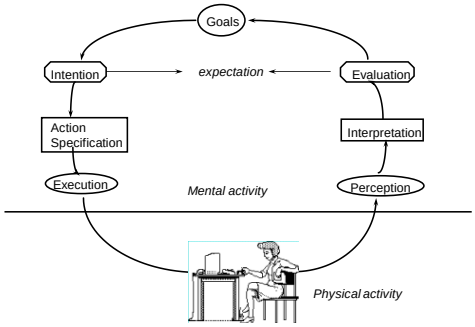
Guideline suggested by syntactic/semantic model .

- Reduce the burden to the task-oriented user of learning a separate computer semantics and syntax
- Methods
  - computer syntax
    - A little learning should go a long way...
    - Should be as understandable as possible (tied to semantics)
      - e.g. meaningful command names, icons, keyboard shortcuts
    - Should be as simple as possible and uniformly applicable
      - e.g., object selection with mouse: single click selects, double click activates
    - Generic commands
      - same command can be applied across different objects
    - Syntax should be consistent between systems!

The Four Stages of an Interaction

- Intention, Selection, Execution, Evaluation
  - a simplified version of Norman's 7 stages
- 1. Forming an intention
  - "What we want to happen"
  - internal mental characterization of a goal
  - may comprise goals and sub-goals (but rarely are they well planned)
  - similar to task semantics
    - e.g. "begin a letter to Aunt Harriet"
- 2. Selecting an action
  - review possible actions and select most appropriate
  - similar to mapping between task and compute semantics
    - e.g. "use the emacs editor to create a file harriet.letter"

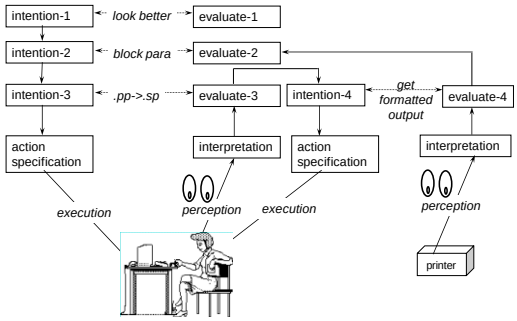
The stages of user activities when performing a task



The Four Stages of an Interaction .

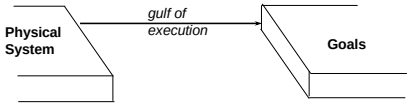
- 3. Executing the action:
  - carry out the action using the computer
  - similar to mapping between semantics and computer syntax
    - e.g. type "emacs –nw harriet.letter"
- 4. Evaluate the outcome
  - check the results of executing the action and compare it with the expectations
    - e.g. see if emacs editor is on the display and verify that buffer name is "harriet.letter"
  - requires perception, interpretation, and incremental evaluation

A typical task: making a business letter look better



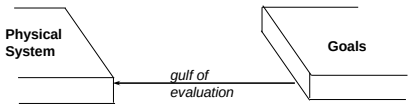
What the four stages model reveals

- The "Gulf of Execution"
  - do actions provided by system correspond to the intentions of the user?
  - Gulf: amount of effort exerted to transform intentions into selected and executed actions
  - A good system:
    - direct mappings between Intention and selections
    - e.g. printing a letter:
      - put document on printer icon
      - vs select print from menu
      - vs "latex letter.tex; lpr -Palw3 latex.dvi"
      - drawing a line: move mouse on graphical display vs "draw (x1, y1, x2, y2)"

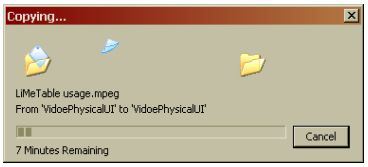


What the four stages model reveals .

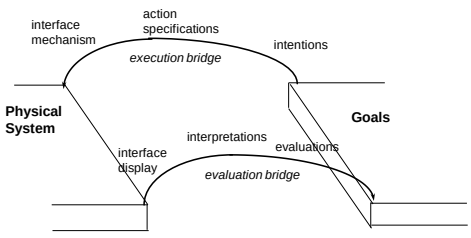
- The "Gulf of Evaluation"
  - can feedback be interpreted in terms of intentions and expectations?
  - Gulf: amount of effort exerted to interpret feedback
  - a good system: feedback easily interpreted as task expectations
    - e.g. graphical simulation of text page being printed
  - a bad system: no feedback or difficult to interpret feedback
    - e.g. Unix: "$", "bus error", "command not found"



Is it a good feedback ?



Bridging the Gulf of Execution and Evaluation



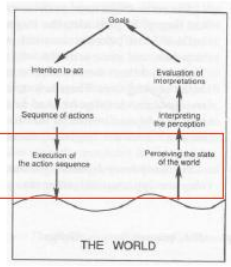
Design rules for HCI

Design Rules for HCI

- Many sets of rules have been proposed to encapsulate understanding and best practice
  - Operate at various levels
- principles
  - abstract design rules
  - "an interface should be easy to navigate"
- guidelines
  - advice on how to achieve principle
  - may conflict; understanding theory helps resolve
  - "use colour to highlight links"
- standards
  - specific rules, measurable
  - "MondoDesktop links are RGB #1010D0"

Errors

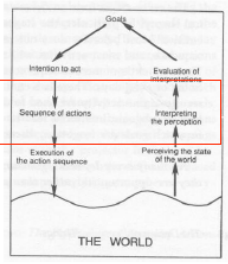
Low-level: slips of execution, misperceptions



Errors

Mid-level: problems in translation to machine I/O

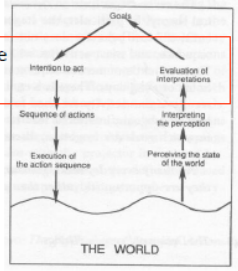
- Cannot produce input that the machine understands or produce input that it 'misinterprets' e.g due to mode
- Cannot understand or misinterpret the output from the machine e.g. due to mindset



Errors

High-level: unable to conceive or recognise goal satisfaction

- Cannot determine correct thing to do, or make wrong choice
- Uncertainty about outcome or mistaken belief that have made progress towards the goal



HCI Ergonomy – adapted evaluation

- **ISO 9241**, *Ergonomics of Human System Interaction*, adopts traditional usability categories with specific measures, e.g.:

Usability objective	Effectiveness measures	Efficiency measures	Satisfaction measures
Suitability for the task	Percentage of goals achieved	Time to complete a task	Rating scale for satisfaction
Appropriate for trained users	Number of power features used	Efficiency relative to expert user	Rating scale for ease of learning
Learnability	Percentage of functions learned	Time to learn criterion	Rating scale for ease of learning
Error tolerance	Percentage of errors corrected successfully	Time spent on correcting errors	Rating scale for error handling

Design rules - Schneiderman

Shneiderman's 8 Golden Rules (1987):

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

Design rules - Norman

Norman's 7 Principles (1988):

1. Use both knowledge in the world and knowledge in the head.
2. Simplify the structure of tasks.
3. Make things visible.
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.

Design rules - Nielsen

Nielsen's 10 Usability Heuristics (1994):

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Help users recognize, diagnose and recover from errors
6. Error prevention
7. Recognition rather than recall
8. Flexibility and efficiency of use
9. Aesthetic and minimalist design
10. Help and documentation

Source of rules and recommendations ?

- Many seem like common sense - but often violated
  - Home exercise: pick one everyday object and one piece of software and assess with respect to these rules
- Some are grounded in our understanding of how humans perceive, think and learn (c.f. next lectures)
- Some are the result of empirical study (e.g. Nielsen's heuristics are based on factor analysis of 249 usability problems)
- Some are derived from particular characterisations of the nature of human action (e.g. Norman's principles are closely related to his theory of action)
- Some are collections of experience (e.g. Shneiderman's rules)
- Some can be directly related to computational complexity
- In this course we will study the background and justification of these rules and elaborate on how they can be applied in specific contexts to design and assess human computer interaction.

Usability principles categories - Dix

Dix groups these and related principles as follows:

- **Learnability**
  - the ease with which new users can begin effective interaction and achieve maximal performance (e.g. familiarity, generalisability, predictability)
- **Flexibility**
  - the multiplicity of ways the user and system exchange information (e.g. customisability, substitutability, user control)
- **Robustness**
  - the level of support provided to the user in determining successful achievement and assessment of goal-directed behaviour (e.g. observability, recoverability)

Using four stages to ask design questions

- How easily can a user
  - Determine the function of the system?
  - Tell what actions are possible?
  - Determine mapping from intention to selection?
  - Perform the action?
  - Tell what state the system is in?
  - Determining mapping from system state to interpretation?
  - Tell if system is in the desired state?

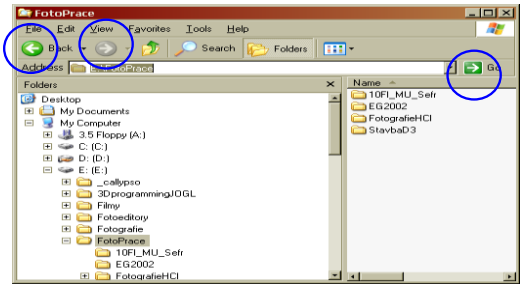
Using four stages to ask design questions

- Questions similar to principles of good design:
  - visibility
    - can see state of application and alternatives for actions
  - good conceptual model
    - consistency in presentations of operations and results
    - coherent system image
  - good mappings
    - relations between
      - actions and results
      - controls and their effects
      - system state and what is visible
  - feedback
    - full and continuous feedback about results of actions
- Principle of transparency
  - "the user is able to apply intellect directly to the task;
  - the tool itself seems to disappear"

You Know Now

- Several high level theories exist that describe how people interact with computers
- Shneiderman's syntactic/semantic model
  - A user's mapping between computer syntax, computer semantics, and task semantics
  - Problems identified when the user's mapping is poor
- Norman's stages of human interaction
  - Intention, selection, execution, evaluation
  - Problems identified as gulfs of execution and evaluation

Meaning of these buttons?



Old vs. new concept

Scott Jenson: Default thinking

A bit of wisdom …

- Bertrand Russell
  - "The trouble with the world is that the stupid are cocksure and the intelligent are full of doubt."
- Daniel Boorstin
  - "The greatest obstacle to discovery is not ignorance — it is the illusion of knowledge."

Once upon a time …

- Analog Microwave:
  - Turn the timer to 30 seconds
  - Turn the power to low
  - Put in the food
  - Close the door
  - Press Start

Current Way of Living

- Digital Timer Microwave:
  - Tap the 10-second button three times to get 30 seconds
  - Tap the Power/Level button four times (to cycle down from 100 percent power to 70 percent power to 50 percent power to 30 percent power)
  - Put in the food
  - Close the door
  - Press Start

Mental Model and Interface

https://developer.apple.com/

Questions

Acknowledgements

- Special Thanks to Prof. Jiri Sochor