HCI Lecture 4

Task analysis & design continued

Task design - elements to be designed

- system model/metaphor
- range of tasks to be performed
- language - actions & commands
- menus

Some methods for task analysis

- Hierarchical task analysis (HTA)
- Layered - Conceptual, semantic, syntactic, lexical analysis
- GOMS - Goals, Operators, Methods, Selection Rules
- Task-Action grammars
- Object-Action Interface model (Shneiderman)

A note on task analysis methods

- These methods are not interchangeable equivalents
- They do different things/are good at different things
- Sometimes used in combination
- May apply both in task analysis and task design

Hierarchical task analysis

- describe task in terms of a hierarchy of plans/operations
- breaks down plans & operations into more & more detail
- down to keystroke/mouse movements if required

Example: HTA for word processing - MS Word (1)

- top level
  - enter/load wp
  - enter text
  - format text
  - edit text
  - save file
  - print file
  - exit wp

Example: HTA for word processing (2)

- lower level – down the hierarchy from 'edit text'
  - delete blocks of text
    - select text block
      - move pointer to start of block
      - hold down button & move to end of block
    - press DELETE

Issues in using HTA

- *granularity* – levels and degree of detail
- can go down to keystroke/mouse movement level
- analysis may be either breadth-first or depth-first
- does not consider cognitive aspects of performing the task
  - *what the user must know/remember*

Task model v. technology model

- task analysis techniques can model
  - how **we** think of a task
  - how we decompose it into sub-tasks
  - how we order and prioritise the sub-tasks
- and
  - how s/w has designed a task
  - how it imposes order/priorities
  - how it **configures the user**

Design v evaluation

- in design, we model how the user understands the task
  - to build a system as close to the user as possible
- in evaluation, we model how the user understands the task
  - to compare user and technology models

Conceptual, semantic, syntactic, lexical approach

- Layered top-down approach – moves from a real-world conceptual analysis to system implementation
- Identifies concepts and functions required
- Considers how these will be expressed in the computer system's interface
- Identifies the detailed sequence of actions to be carried out to perform each function

Conceptual, semantic, syntactic, lexical example

- *conceptual* - user's mental model of the system (eg word processor – edit document)
- *semantic* – meaning/desired function (move paragraph)
- *syntactic* - how the semantic command is formed (Edit, Highlight, Cut, Paste)
- *lexical/interaction* - sequence of actions (point to Edit, Click, etc)

GOMS – Goals, Operators, Methods & Selection rules

- Also a top-down approach
- Can be used at any level of abstraction
- Useful for comparison of alternative systems/designs
- Based on the idea that we form goals (the things we want to achieve) and then form plans of action to carry them out

GOMS

- users form *Goals* (edit document)
- and sub-goals (insert word)
- achieved by using *methods* (sequence of actions)
- which consist of *operators* (keystrokes)
- *Selection Rules* - determine which course of action, when there is more than one way

GOMS example

- Goal *edit text*
  'The quick brown fox jumped over the the lazy dog.'
- Sub-goal *delete word* the
- Identify methods
  - Move to the word
    - Move to the word (mouse) **or**
    - Move to the word (cursor keys)
  - Delete
    - character by character, with Del, **or** Backspace
    - highlight word, delete word

GOMS example continued

- Identify selection rules
  - If hand is on mouse, use mouse to move to text, else use cursor/page keys
- assumes expert performance
- requires designer skill and intuition
- useful for comparison/evaluation of different interfaces for performance of the same task

Task-Action grammars

- aid in development of consistent interface
- map all tasks to action/command
- producing a complete set of tasks and the actions/commands to perform them (a sort of data dictionary)
- test for consistency/completeness

Task-action grammar example

- To delete/insert objects in word processing

delete character	*insert* character
delete word	*insert* word
delete paragraph	*insert* paragraph
remove heading	*create* heading

  - delete/insert character — consistent
  - remove/create heading — inconsistent with the rest

Object-Action Interface model (1)

- proposed by Shneiderman (1998)
- aimed at production of GUIs
- models real-world objects (task domain)
- decomposed to levels of detail
  - stock market statistics
  - information about one stock
  - today's price

OA Interface model (2)

- models actions users perform on objects
- decomposed
  - high-level check share price
  - medium-level load s/w, find the stock
  - low-level press sequence of keys
    click icons

Interface objects & actions

- interface metaphors/representations derived (icon design)
- screen representation of real-world object
- screen controls providing means of performing action

Analysis/design of task - conclusion

- There are no right answers (but there is better and worse software)
- There is no perfect methodology – only guidelines
  - Know & understand the task domain
  - Model your system on the task domain
  - Identify all the functions to be provided
  - Model in detail how you will provide them
  - Check for completeness and consistency

The 8 golden rules of Interface design (Shneiderman 1998)

- be consistent
- allow short-cuts
- give feed-back
- provide closure
- provide error prevention & error handling
- permit easy reversal
- support internal locus of control
- reduce short-term memory load

Next week

- interface design
- dialogues and interaction styles