HIC Lecture 5:

Task analysis

Hiroshi Shimodaira

Key points:
- Motivation: need to understand the task to be able to design for good interaction
- Define the scope and observe or generate representative interaction exemplars
- Organise/formalise the description
- Confirm validity
- Use at all stages of design process
Motivation

“assume knowing something about how users approach and carry out tasks will aid software designers when making design decisions which will ultimately affect computer system usefulness and usability” (Johnson & Johnson, 1991)
Motivation

- A ‘task’ can be defined as the activities required to obtain a particular goal in a particular domain
  - Could view as description of user’s procedural memory (or what should end up in their memory)

- Note that many design guidelines (lecture one) are implicitly task specific:
  - “Speak the user’s language” – so what is their language? What terms are used in the task domain, and how are they related?
  - “Be consistent” – with which other systems? Is the new design replacing an existing system, and will procedures transfer?
  - “Give appropriate feedback” – what is appropriate for this group of users executing this task? What information transfers to and from the user are necessary or desirable?
Motivation

Task analysis has a potential role at different stages of design:

- Supporting requirements analysis:
  - Functional specification of what system will do
  - Data requirements and data flow
  - Usability requirements
- Evaluating prototypes
- Performance testing
- User training

- Aim: to provide a ‘rigorous structured characterisation’ or ‘complete and explicit’ description of user activity
- Reality: more often a guiding framework

[Dix et al, p.195]
Stages of task analysis

- Defining purpose and scope
- Obtaining data
- Extracting activity lists

- Organising and describing task performance
  - Many methods: we will focus on just one (see Dix for alternatives)
- Confirming the validity of the description
  - If necessary, reiterating previous steps

- Using:
  - Note an important issue here is how the results are communicated:
    - In software engineering terminology?
    - In executable form?
    - With support tools?
Defining purpose and scope

- What stage(s) of the project need task analysis?
- What do you want to know and how are you going to use it?
- What is the scope?
  - A useful approach here is to identify the main components of the work system and the application domain.
  - Goals are desired future states of the application domain
  - The work system changes the application domain by doing tasks

![Diagram showing the relationship between System, Application Domain, Work System, Work, and Communicative Relationships]
Example of work system and task domain

Obtaining data

- Ideal: study real people doing real task in real context
  - E.g. video record and replay to get user commentary
  - If aim is to design new system, analysing existing systems helps
- Examine documentation, training programmes, etc. for descriptions of how to carry out tasks
- Interview stakeholders about the task (try to uncover real goals and procedural knowledge)
- Generate scenarios →

General problem:
- want to have concrete and complete descriptions, yet to avoid assuming a particular tool or system is being used for the task
Scenarios

- A **scenario** is an informal narrative description of a specific interaction, usually with a real-world setting.

- Scenarios can:
  - be elaborated down to low-level interactions
  - suggest desirable constraints such as response time, error behaviour

- May make use of **personas**, prototypical users
  - Based on composite/hypothetical user
  - Pose questions: “how would Betty react if. . . ”

- Non-essential details help things appear real, avoid designers falling into “if it was me” trap.
Betty, the Warehouse Manager:

“Betty is 37 years old, She has been Warehouse Manager for five years and worked for Simpkins Brothers Engineering for twelve years. She didn’t go to university, but has studied in her evenings for a business diploma. She has two children aged 15 and 7 and does not like to work late. She did part of an introductory in-house computer course some years ago, but it was interrupted when she was promoted and could no longer afford to take the time. Her vision is perfect, but her right-hand movement is slightly restricted following an industrial accident 3 years ago…”
Example Scenario

Personal Movie Player: Bluetooth Download

“Brian would like to see the new film “Moments of Significance” and wants to invite Alison, but he knows she doesn’t like “arty” films. He decides to take a look at it to see if she would like it and so connects to one of the movie sharing networks. He uses his work machine as it has a higher bandwidth connection, but feels a bit guilty. He knows he will be getting an illegal copy of the film, but decides it is OK as he is intending to go to the cinema to watch it. After it downloads to his machine he takes out his new personal movie player. He presses the menu button and on the small LCD screen he scrolls using the arrow keys to bluetooth connect and presses the select button. On his computer the movie download program now has an icon showing that it has recognised a compatible device and he drags the icon of the film over the icon for the player. . . .”
Scenarios

- Have become popular as a method in software design
  - See e.g. Carroll (2000)
- ‘Flexible and powerful’ because exploit human ability to envisage rich interactions with minimal information cues
- Sometimes described as an alternative to task analysis but:
  - Lacks comprehensiveness and structure
  - May conceal assumptions, make early commitment on design decisions that should be postponed
  - May lead members of design team to envisage different things, resulting in conflicting conceptual models
- Needs to be more rigorous to support design methods
  - Multiple scenarios to ensure coverage
  - Process in comparable way to real data, i.e., perform further analysis steps
Extract activity list

- For a representative sample of tasks and users, produce detailed list describing the accomplishment of a goal (‘interaction script’).
  - Each line is single main agent performing one main action that affects other agents or objects
  - May want in form of dialogue, e.g. user and computer

- Example:

<table>
<thead>
<tr>
<th>USER:</th>
<th>SYSTEM RESPONSE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Clicks button labelled “Start”</td>
<td>2. Menu pops up</td>
</tr>
<tr>
<td>3. Clicks “shut down”</td>
<td>4. Greys screen and presents dialogue box describing shut down and offering “OK” “cancel” and “Help” options</td>
</tr>
<tr>
<td>5. Clicks “OK”</td>
<td>5. Screen blanks and “Please wait” message shown</td>
</tr>
<tr>
<td></td>
<td>6. Computer turns off</td>
</tr>
</tbody>
</table>
Analysis

- Aim is to reduce the volume of information in the activity lists:
  - If possible, combine into one or a few more abstract descriptions (e.g. Use Cases)
  - Represent alternatives and options as branches
  - Decide if some user types/tasks should be excluded

- Approach may be top-down (start from overall goal and subdivide the steps) or bottom up (start from example actions and organise into appropriate hierarchy)
Hierarchical Task Analysis

- Best known method is Hierarchical Task Analysis (HTA)
- Start from overall goal, e.g. clean the house
- Break down into numbered subgoals, e.g.
  0. clean the house
     1. get the vacuum cleaner out
     2. clean the rooms
        2.1 clean the hall
        2.2 clean the living rooms
        2.3 clean the bedrooms
     3. empty the dust bag
     4. put vacuum cleaner and attachments away
- Describe order of subtasks, e.g.
  Plan 0: do 1, 2, 4 in order; when dust bag full, do 3
  Plan 2: do 2.1, 2.2, 2.3 in any order, as needed
Hierarchical Task Analysis

- Diagrammatic notation

- **Plan 0.**
  - do 1 at the same time, if the pot is full 2 then 3 – 4 after four or five minutes do 6

- **Plan 1.**
  - 1.1 – 1.2 – 1.3 when kettle boils 1.4

- **Sub-plans:**
  - 1.1. fill kettle
  - 1.2. put kettle on hob
  - 1.3. wait for kettle to boil
  - 1.4. turn off gas
Refining the HTA: check and improve the decomposition

Some heuristics are:

- **Check for paired actions**: where is “turn on gas”?
- **Restructure**: generate task “make pot”
- **Balance**: is “pour tea” simpler than “make pot”?
- **Generalise**: make one cup . . . or more
- **Sub-operations should be mutually exclusive and collectively exhaustive**
- **Consider possible alternatives**

N.B. Important that task decomposition is an iterative process that goes back to check decomposition with user
0. make cups of tea

plan 0.
do 1
at the same time, if the pot is full 2
then 3 – 4
after four or five minutes do 5

1. boil water
2. empty pot
3. make pot
4. wait 4 or 5 minutes
5. pour tea

Plan 1.
1.1 – 1.2 – 1.3 – 1.4
when kettle boils 1.5

Plan 3.
3.1 – 3.2 – 3.3

Plan 5.
5.1 → 5.2
empty NO for each guest 5.3

5.1. put milk in cup
5.2. fill cup with tea
5.3. do sugar

5.3.1. ask guest about sugar
5.3.2. add sugar to taste

5.3.1 – if wanted 5.3.2

1.1. fill kettle
1.2. put kettle on hob
1.3. turn on and light gas
1.4. wait for kettle to boil
1.5. turn off gas
Types of plan

- **sequence** 1.1 then 1.2 then 1.3
- **optional** if the pot is full 2
- **wait** when kettle boils, do 1.4
- **cycles** do 5.1 5.2 while there are still empty cups
- **parallel** do 1; at the same time ...
- **discretionary** do any of 1.3.1, 1.3.2 or 1.3.3 in any order

Most plans use several of these.

Waiting can be considered:

- a task — for “busy” waits, e.g. making tea
- part of the plan — end is the event, e.g. email reply received
Task decomposition

When is decomposition complete?
- When reach ‘actions’, i.e. task that user can execute without problem solving (but note this may differ for different users)
- Above ‘device specific’ implementation details (but note shape of task is often device dependent)
- Suggested heuristic is to stop when probability of error (p) multiplied by cost of error (c) is below threshold

- See also ‘ConcurTaskTrees’ (CTT) – expands HTA with more explicit temporal relationships, additional task attributes.
Using the task analysis

From the task description:
- Identify ‘critical’ steps, e.g. high attention load, need special knowledge
- Find possible points of failure, e.g. easily missed steps

For design:
- consider alternative task allocations
- taxonomies suggest menu layout
- object/action lists suggest interface objects
- task frequency guides default choices
- task sequences guide dialogue design
- task structure guides organisation of help menus/manuals,
- domain description clarifies learning requirements
Using the task analysis

For evaluation:

- Use task description to do walkthroughs with prototypes, checking for each step:
  - Is the user action motivated?
  - How will they know what to do?
  - How will they know they have done it?

- Example: part of proposed computer interface in catalog shop:

<table>
<thead>
<tr>
<th>Name</th>
<th>Catalog #</th>
<th>Trigger Invoice</th>
</tr>
</thead>
</table>

- ‘Enter name’ is not a standard step in the task of buying items for cash in a shop, but system needs some kind of identifier
- ‘Enter catalog number’ is motivated but error-prone: computer should at least confirm with item description
- ‘Trigger invoice’ is not obvious cue for required step of completing transaction
There are many different tools; can be useful to try several complementary approaches

Most methods still lack a ‘direct translation’ from the task analysis to the system design

- Recent interest in relating task analysis methods to UML, i.e. taking a user rather than system focus in creating use-cases, activity and sequence diagrams, class diagrams etc.
- See e.g. Nobrega et. al. (2005)
References


See also:
Dix et. al. Chapter 15