Mental Models

*Mental Model*: The user’s understanding of how the system works.

- The mental model is in the user’s head and thus we can only indirectly see what the mental model is.
- Aim to design so that the user’s mental model matches what the system really does.
- User may have different mental models for system data, system functions, task domain.
- Avoid extremes of data-focused systems and step-by-step function focused systems.
Mental Models

- When users have a correct model of a system, they can find new ways to use the system or perform new tasks without explicit instruction.

- Users build mental models unconsciously, and usually cannot explain what their model is!

- Following the user’s present model of how a task should be performed may result in an inefficient new system.

- Thus, new mental models may be required.
  - In these cases it is vital that the interface indicates the required new model.

Developing a Mental Model

- Mental Model: The model in the user's mind! We cannot see it, and often the user is not aware of their own model.

- Cognitive Model: The observer's explanation of the user's mental model. Someone observes the user and describes their apparent mental model.

- Conceptual Model: The system designer's explanation of how the system works. Used for:
  - As a plan for designing the details of the system.
  - To explain to the user how the system works.

If the system is easy to understand the conceptual model and the cognitive model will be well matched.
Developing a Mental Model

Effects of model:
- What will happen if ...
- Expected functions
- What a message means

Model Life cycle:
- Fast development (hours)
- Gradual refinement (weeks)
- Proficiency (mapping dominates)
- Fading (non-intuitive models forgotten)

Life Cycle of Mental Models

- Fast development of basic model parts - within hours.
- Gradual refinement. Days, weeks or months. Radical changes of the mental model are hard.
- Proficiency - user forms a *mapping model* which maps user tasks to the system. Mental model is used only rarely for unusual tasks.
- Fading models. Non-intuitive mental models become wrong due to lack of use, user may forget or distort the mental model. Serious errors may occur in unusual situations.
• **Mental models for data**: The user sees some data on the screen or enters some data. When the system closes the window or moves to another one, the user does not assume the data disappear, but rather that it is stored somewhere. *(object permanence)*

• **Model models for system functions**: The user understands system functions in terms of stored data, and its relation to what is on the screen.

• **Mental models for the domain**: Users have mental models about the real world. A person with domain expertise knows about the objects and the data out there, and the procedures used for various tasks.
Mental models for handling a customer order

- User forms a model of what is stored in the system, and how it relates to other stored data, based on the display of data.
  - Model of files for order forms and customer forms.

- An understanding of a function, e.g. “Cancel” order.
  - Delete this order, but maintain the customer in the system.

Dialogues and Virtual Windows

- **Virtual Windows** - screens that the user should imagine being behind the physical screen, considered by the user to be permanent objects in the system.

- **Mapping Level** - how the user maps the task they want to carry out onto the functions that the system can carry out.

- **Cognitive Walkthrough** - examine each user task and plan necessary system functions to carry it out. Similar idea to object-orientated programming - consider the virtual windows as “objects”, then plan methods (functions) which need to be performed on these.
Bad Screen Design - Two Extremes

If the design does not take account of virtual windows in design, two unsatisfactory extremes can emerge.

- design too close to the database design.
- breaking task into multiple incremental steps which are too small.
Task: **Patient Monitoring**

Record temperature and pulse of a hospitalized patient at regular intervals.

Data stored in two separate database tables - poor design simply gives a separate window for each.

**Bad Screens - Database Extreme**

What is wrong with this?

- Window’s presented in this way don’t support the users’ tasks.
  - nurse needs to record both temperature and pulse at the same time, two screens makes this unnecessarily laborious.
  - needs to see lists of both features at the same time for diagnostic comparison, cannot swop between them and try to remember the details.

So neither the recording or diagnosis tasks are properly supported.
Task: **Work-hours Registration**

Record employee’s time on different projects and activities.

The user adds records and edits details through this screen.

In principle, it is adequate, but it is hard to get a visual overview of the data:

- no patterns
- no gestalts

Alternative views shows **patterns**. More information can now easily be shown, e.g. total no of hours per day and week for each activity and overall.

```
<table>
<thead>
<tr>
<th>Activity</th>
<th>Date</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>102 Lunch</td>
<td>23-09-02</td>
<td>0.5</td>
</tr>
<tr>
<td>715 Design DXP</td>
<td>23-09-02</td>
<td>4.0</td>
</tr>
<tr>
<td>812 Cust. meeting</td>
<td>23-09-02</td>
<td>3.0</td>
</tr>
<tr>
<td>102 Lunch</td>
<td>24-09-02</td>
<td>0.5</td>
</tr>
</tbody>
</table>
```

**Time registration**

**Employee:** MBH

**Week:** 39 **Year:** 02

```
<table>
<thead>
<tr>
<th>Activity</th>
<th>Mo</th>
<th>Tu</th>
<th>We</th>
<th>Th</th>
<th>Fr</th>
<th>Sa</th>
<th>Su</th>
<th>Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>102 Lunch</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>715 Design DXP</td>
<td>4.0</td>
<td>3.0</td>
<td>5.0</td>
<td></td>
<td>3.0</td>
<td>4.5</td>
<td></td>
<td>19.5</td>
</tr>
<tr>
<td>808 Review SPA</td>
<td></td>
<td>4.5</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.0</td>
</tr>
<tr>
<td>812 Cust. meeting</td>
<td>3.0</td>
<td></td>
<td>7.5</td>
<td>3.5</td>
<td></td>
<td></td>
<td></td>
<td>14.0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>7.5</td>
<td>8.0</td>
<td>8.0</td>
<td>7.5</td>
<td>7.0</td>
<td>4.5</td>
<td>0.0</td>
<td>42.5</td>
</tr>
</tbody>
</table>
```
Bad Screens - Step-by-step Extreme

- Tasks on top left.
- Separate webpage for each step.
- Figure outlines webpages for the first tasks - “create an order.”
Bad Screens - Step-by-step Extreme

- It assumes that the computer controls the input sequence completely.
- It is difficult to revise values if a mistake is made.
- Impossible to browse or overview the values entered.
- Specific example here, cannot check delivery details against current address.
- No overview of the number of steps ahead.
- TOO MANY PAGES!

Main problem here - the computer controls the user, not the other way round!

Successful Step-by-step Dialogues

ATM

- Very simple interface to support simple fixed tasks.
- Don’t extrapolate this design to more complex systems and assume that it will still work well.

Wizards, e.g. installation of Windows applications.

- can be frustrating! (see general problems of step-by-step data entry.)
- options can often be selected, often able to back track.
- only used occasionally, so user often needs guidance, idea is to make life simple for the novice user of the application.