Usability analysis and inspection

Why and how?

Why?

- Iterative design
- Prototyping
- Measuring usability
- Objective/subjective feedback
- Quick and dirty
- Slow and clean
- With or without users
Evaluation

Questions that might arise during systems development:

- Will the switch board operator be able to answer more telephone calls per hour than before?
- Will less customers have problems withdrawing cash from the ATM?
- If a user enters a wrong number, what is the consequence and can we recover from it?
- Can we change the current menu structure without causing problems?
- How fast can the users read text from the screen?
- How long time will it take to find specific information?
- Decreasing the font size from 12 p to 10 p, what consequences will it have?
- How small can we make the buttons?
Evaluating usability

- These questions are about usability
- Usability can be measured through such questions
- Evaluations typically address whether the system meets the requirements, it seldom explains the reasons behind the problems.

Measuring usability

- No “silver bullet”
- Evaluation during the development, not afterwards — usability is not only a measure but a goal for developers and development.
- Usability builds on analysis and experiments.
- Inexperienced developers often ignore validation in preference of verification of a system.
**Costs for correcting errors**

Costs

![Graph showing the increase in costs over time since project start.]

**Analytical evaluation**

- A logical simulation of user behaviour
- E.g. GOMS, KLM, Cognitive Walkthrough
- Can be used to test things before they have been built.
- Saves time since we do not need to construct in advance.
- No user experiments necessary.
- Create a sequence of steps and measure the time per step.
Empirical evaluation

- Let users try out a prototype
- Simple and cheap tests.

Performance measurement

- Test users perform predefined tasks
- Define a set of usability goals
- Measure errors and times
- Laboratory
  + Results in hard numbers, easy to compare
  + Objective results
  - time consuming, expensive
  - skilled users for skilled performance
GOMS

Measuring speed of performance
Describes different possible ways (sequences) of solving a problem
- Goal - a certain objective to be achieved
- Operator - available actions
- Methods - Sequences of operators
- Selection rules - when choosing between methods

GOMS model

GOAL: PHOTOCOPY PAPER
GOAL: LOCATE-ARTICLE
GOAL: PHOTOCOPY-PAGE repeat until no more pages
  GOAL: ORIENT-PAGE
  OPEN-COVER
  SELECT-PAGE
  POSITION-PAGE
  CLOSE-COVER
GOAL: VERIFY-COPY
  LOCATE-OUT-TRAY
  EXAMINE-COPY
GOAL: COLLECT-COPY
  LOCATE-OUT-TRAY
  REMOVE-COPY (OBS! Yttre målet fullföljt!)
GOAL: RETRIEVE-JOURNAL
  OPEN-COVER
  REMOVE-JOURNAL
  CLOSE-COVER
Keystroke-Level Analysis

- Divide each task-performance into components
- assign execution times to each component
- e.g. press key - 0.08 s, point with mouse - 1.10 s
  + compare different methods
  - only on smaller parts of the work

---

**KLM model**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Anmärkning</th>
<th>Tid (i sekunder)</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>Tryck ned tangent</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>Gud maskinskrivare (90 ord/min.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Medelmässig maskinskrivare (40 ord/min.)</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td>Ilke-maskinskrivare</td>
<td>1.20</td>
</tr>
<tr>
<td>B</td>
<td>Musknappstryckning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nor eller upp dubbelklick</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.20</td>
</tr>
<tr>
<td>F</td>
<td>Peka med musen</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fästa lag</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Medelrörelse</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1log(D/S0.5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.10</td>
</tr>
<tr>
<td>H</td>
<td>Hand till eller från tangentbord</td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Mentalt förbereda sig</td>
<td>1.35</td>
</tr>
<tr>
<td>R</td>
<td>Mottag uttryck från systemet</td>
<td></td>
</tr>
</tbody>
</table>

Tabell X: tider för vissa operationer som en operatör utför enligt KLM-metoden (efter Card, Moran & Newell, 1983)
KLM example

- **Exempel.** Det finns två olika sätt att utföra kommandot ”Spara”. Antingen kan man välja Ctrl-S som en tangentkombination på tangentbordet, eller också kan man välja alternativet spara i arkivmenyn.

- Alternativ 1: Ctrl S – \( T_1 = M + H + K = 1,35 + 0,40 + 0,28 = 2,03 \) sek

- Alternativ 2: Menyalternativ – \( T_2 = M + H + P + B + P + B = 1,35 + 0,40 + 1,10 + 0,10 + 1,10 + 0,10 = 4,15 \) sek

Thinking aloud

- users verbalise their thoughts
- identify misconceptions of the system
  + direct feedback from the users
- not natural to think out loud
- difficult to verbalise their decisions
Questionnaires

- Subjective satisfaction
  + Can be distributed to many people
  - Users answer what they think they do, not what they actually do

Cognitive Walkthrough

- An evaluator walks through the interface
- Tries to act as a user
- Actions based on knowledge about users
  + Quick
  - Dirty
  - Does not involve users, difficult to capture domain specific problems
Cognitive walkthrough

Explanatory learning

- Constructed for novice users
- Suitable for walk-up-and-use-interfaces
- A successive walkthrough of tasks based on questions on different levels
  - Does the user see the search alternative?
  - Do the user select the right alternative?
  - Do the user understand the system response?

Card Sorting

- Physical paper cards with text and images
- The test person sorts the cards according to importance
- Good for sorting icons, concepts etc.
- Can promote understanding and relevance of concepts by users
- Requires real users for good results
Heuristic evaluation

- Uses sets of guidelines
- Inspection of the interface
- Document potential usability problems
  + Easy to apply
  + possible to use early in the design
  - Domain problems

Heuristic evaluation

- Jacob Nielsen, www.useit.com
- Measures usability
- Informal walkthrough
- Cheap and simple
- No users required, can be performed individually, requires minimal planning, can be used early in the process
- Concentrates on identifying errors in the design (good and bad)
**Heuristic evaluation method**

- A group of evaluators (more evaluators find more errors)
- 3-5 persons
- Design heuristics help in the inspection

---

**Heuristic evaluation**

1. Simple and natural dialogue
2. Speak the users language
3. Minimise Users Memory Load
4. Consistency
5. Feedback
6. Clearly marked exits
7. Shortcuts
8. Good error messages
9. Prevent errors
10. Help and documentation
Simple and Natural dialogue

- Obvious how to interact with the system
- Grouping of objects
- Enough information on the screen
- Overview and details
- Avoid confusing decoration

Speak the users language

- Domain language
- Avoid system or computer language
- Use metaphors familiar to the users
- Icons, pictures, etc., relevant for the domain
Minimise user memory load

- Enough information on the screen
- Recognition instead of recall
- Highlight important information
- Show status information
- Default values

Consistency

- Consistent layout
- Consistent interaction
- Consistent language
- Consistent functionality
Feedback

- Show waiting times
- Feedback for actions
- Work related feedback
- System failure

Clearly marked exits

- Show the way the user has “walked” through the system
- How to get back and forward
- Oriented in the information space
**Shortcuts**

- Use shortcuts to perform actions
- Type ahead
- Jumps to a desired location
- Reuse of interaction history

**Good error messages**

- Avoid obscure data code
- “An error No 13 has occurred”
- Use the users language
**Prevent error**

- Dialogues for actions that may lead to serious consequences
- Avoid modes
- Show status
- Support “undo” and “redo”

**Help and documentation**

- Manuals should not be necessary
- Give on-line help on the users initiative
- If no other solution is possible, give help at the initiative of the system
Nielsen’s new design heuristics

- **Visibility of system status**
  The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

- **Match between system and the real world**
  The system should speak the users’ language, with words, phrases and concepts familiar to the user rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logic order.

- **User control and freedom**
  Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

- **Consistency and standards**
  Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
Nielsen’s new design heuristics

- Error prevention
  Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

- Recognition rather than recall
  Make objects, actions and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

- Flexibility and efficiency of use
  Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

- Aesthetic and minimalist design
  Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
Nielsen’s new design heuristics

- Help users recognise, diagnose and recover from errors
  Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

- Help and documentation
  Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Heuristic evaluation

- Not suitable for repeated analysis (with the same person). Can be solved by rotating the inspection tasks
- Good results if the evaluators are skilled
- Good in combination with real user testing (the techniques complement one another)
Grading the errors

0 I do not agree that this is a usability problem at all
1 Cosmetic problem only
2 Minor usability problem: should be given low priority
3 Major usability problem: important to fix, so should be given high priority
4 Usability catastrophe: imperative to fix this before product can be released

How to use the results

- Analyse the results
- Identify solutions to the problems
- Create a new prototype
- Optimising between requirements
Formative evaluations

- Helps forming the system
- Iterative testing

Summative evaluation – tests the entire system in the end (when it is too late to do anything)

Formative evaluations

- Experience teaches you to build prototypes and to make concrete decisions
- Informal user tests with simple prototypes measures errors early and simple.
- Field tests with prototypes with a series of changes to measure difference in preference, error rate, speed, etc.
- Controlled experiments (all important parameters are under control)
Evaluation steps

- Identify usability aspects, specify special requirements on the product
- Produce a prototype
- Prepare experiments (Find users, select parameters to test, documentation method)
- Test and collect data
- Analysis
- Conclusions

Evaluate...

Which method should you use and why?

- Micro wave oven - which is the fastest way of heating your food, 1.10 or 1.11?
- System for phoning via your computer.
- Public service phone (e.g. Bank, RSV)
- Web site for finding courses (e.g. Asken)
- Digital wrist watch with one button