Predictive models

Predictive models

- Evaluate how people will interact with interfaces.
- Measures the user performance without real testing it (Comparative analysis of applications and devices).
- Useful when it is not possible to execute user tests.

GOMS

- Goals, operators, methods and selection rules.
- Modelling techniques to analyse the complexity of interactive systems.
- Developed by Card et al., 1983.
- Used by software designers to model user behaviour.
- Models the user cognitive process when interacting with the application.
- Hierarchically decompose the problem in sub-objectives.

GOMS

- The user behaviour is modelled in terms of:
  - **Goals**: State what the user wants to achieve (ex: find a website). Hierarchical decomposition in sub-goals.
  - **Operators**: elementary perceptual, motor or cognitive actions that must be executed to achieve the goal (ex: press the button “X”; read dialog box, double-click-mouse).
  - **Methods**: procedures that describes how to accomplish goals. Consist in the exact sequence of steps required to achieve the goal (ex: drag the mouse to the keywords text entry field, enter the text, press the button “search”). There may be several possible methods to achieve the same goal.
  - **Selection rules**: are used to determine which method should be used when there are several possible methods (ex: press button “search” with the mouse or press “enter”). The selection rule determines which method should be used in a certain situation. In general, they are “if-statements”.

GOMS

- These models can be used for different purposes:
  - Verify functionalities:
    - Verify that exists a method to guarantee the achievement of all of the user goals.
  - Preview execution times:
    - Preview the time required for a user to execute a certain tasks and achieve a certain goal (experienced user do not make errors). Ex: compare several design solutions.
  - Help systems:
    - Being explicit representations of the experienced user' s behaviour, can be used in the design of help systems and user guides to assist users in achieving goals.

GOMS

```
GOAL: CLOSE-WINDOW
.   [select GOAL: USE-MENU-METHOD
               .   MOVE-MOUSE-TO-FILE-MENU
               .   PULL-DOWN-FILE-MENU
               .   CLICK-OVER-CLOSE-OPTION
               GOAL: USE-CTRL-W-METHOD
               .   PRESS-CONTROL-W-KEYS]

For a particular user:

   Rule 1: Select USE-MENU-METHOD unless another
           rule applies
   Rule 2: If the application is GAME,
           select CTRL-W-METHOD
```

GOMS

- Basic model is appropriate to make qualitative previews about the tasks where users make few errors.
- Associates times and distributions of time to each operator.
- Depending on the analysis details, several variations of the GOMS model can be used.

GOMS

- There are four different versions of :
  - CMN-GOMS
  - KLM
  - NGOMSL
  - CPM-GOMS
- All techniques consider the coverage of the functionality of a system and provide estimates of task performance time.
- Details on John and Kieras, 1996.

Keystroke level model

- Proposed by Card et al. (1980).
- Simplified version of GOMS
- Complementary with goal hierarchies

- Provides numeric previews about user’s performance.

- Compares the time required to complete a tasks using the different possible methods.

- Card et al., 1983 analysed empirical studies and obtained average execution times for the most common physical actions (ex: press button) and cognitive processes (ex: decide what to do, system response time).

Keystroke level model

- K - key stroking
- P - pointing
- H - homing
- B - button pressing
- D – drawing line (mouse)
- M - mental preparation
- R – system response
- Times are empirically determined (T=Task).

  (See Dix et al., Human-Computer Interaction (second edition). Prentice Hall Europe, London, 1998, p. 248.)

- T_execute = T_K + T_P + T_H + T_B + T_D + T_M + T_R

Keystroke level model

- Execution time for a task is estimated through the sequence of operators that compose the method, adding the times associated with each operator.
- Good preview – error 20%
- Used for micro-interaction.

Keystroke level model

- Example: substitute a wrong character

1. move hand to mouse	H [rato]
2. position mouse after wrong character	PB [left]
3. return to keyboard	H [teclado]
4. delete character	MK [delete]
5. Insert correct character	K [caracter]
6. reposition insertion point	H [rato] MPB [left]

$T_{exe} = 2t_K + 2t_B + 2t_P + 3t_H + 2t_M$

Keystroke level model

- Find&replace a

4 letter word in Word:

Description	Operation	Time (s)
Reach the mouse	H [mouse]	0,40
Move pointer to ”Replace” button	PPP [menu item]	3*1,10
Click on ”Replace”	B [mouse]	0,20
Home on keyboard	H [keyboard]	0,40
Insert word to be replaced	M4K [word]	2,47
Reach mouse	H [mouse]	0,40
Move pointer to correct field	P [field]	1,10
Click on field	B [mouse]	0,20
Home on keyboard	H [keyboard]	0,40
Insert new word	M4K [word]	2,47
Reach the mouse	H [mouse]	0,40
Move pointer on “Replace-all”	P [Replace all]	1,10
Click “Replace all” button	B [mouse]	0,20
	Total	13,04

CMN-GOMS

- Original GOMS model proposed by Card et al., 1983.
- Predicts:
  - operator sequences
  - task execution times.

- CMN-GOMS builds on KLM by adding subgoals and selection rules.

- Expand the goals hierarchy until the desired detail level is achieved.

- Serial execution of tasks.

CMN-GOMS

```
GOAL: DELETE-FILE .
          GOAL: SELECT-FILE . .
                    [select: GOAL: KEYBOARD-TAB-METHOD . .
                          GOAL: MOUSE-METHOD] . .
                    VERIFY-SELECTION .
          GOAL: ISSUE-DELETE-COMMAND . .
                    [select*: GOAL: KEYBOARD-DELETE-METHOD . . .
                                        PRESS-DELETE . . .
                                        GOAL: CONFIRM-DELETE . .
                          GOAL: DROP-DOWN-MENU-METHOD . . .
                                        MOVE-MOUSE-OVER-FILE-ICON . . .
                                        CLICK-RIGHT-MOUSE-BUTTON . . .
                                        LOCATE-DELETE-COMMAND . . .
                                        MOVE-MOUSE-TO-DELETE-COMMAND . . .
                                        CLICK-LEFT-MOUSE-BUTTON . . .
                                        GOAL: CONFIRM-DELETE . .
                          GOAL: DRAG-AND-DROP-METHOD . . .
                                        MOVE-MOUSE-OVER-FILE-ICON . . .
                                        PRESS-LEFT-MOUSE-BUTTON . . .
                                        LOCATE-RECYCLING-BIN . . .
                                        MOVE-MOUSE-TO-RECYCLING-BIN . . .
                                        RELEASE-LEFT-MOUSE-BUTTON]
```

*Selection rule for GOAL: ISSUE-DELETE-COMMAND If hands are on keyboard, use KEYBOARD-DELETE-METHOD, else if Recycle bin is visible, use DRAG-AND-DROP-METHOD, else use DROP-DOWN-MENU-METHOD

CMN-GOMS

```
GOAL: MOVE-TEXT .
    GOAL: CUT-TEXT . .
          GOAL: HIGHLIGHT-TEXT . . .
                    [select**: GOAL: HIGHLIGHT-WORD . . . .
                                                    MOVE-CURSOR-TO-WORD . . . .
                                                    DOUBLE-CLICK-MOUSE-BUTTON . . . .
                                                    VERIFY-HIGHLIGHT . . .
                         GOAL: HIGHLIGHT-ARBITRARY-TEXT . . . .
                                                    MOVE-CURSOR-TO-BEGINNING          1.10 . . . .
                                                    CLICK-MOUSE-BUTTON                0.20 . . . .
                                                    MOVE-CURSOR-TO-END                1.10 . . . .
                                                    SHIFT-CLICK-MOUSE-BUTTON          0.48 . . . .
                                                    VERIFY-HIGHLIGHT]                 1.35 . .
          GOAL: ISSUE-CUT-COMMAND . . .
                    MOVE-CURSOR-TO-EDIT-MENU                                          1.10 . . .
                    PRESS-MOUSE-BUTTON                                                0.10 . . .
                    MOVE-CURSOR-TO-CUT-ITEM                                           1.10 . . .
                    VERIFY-HIGHLIGHT                                                  1.35 . . .
                    RELEASE-MOUSE-BUTTON                                              0.10 .
    GOAL: PASTE-TEXT . .
          GOAL: POSITION-CURSOR-AT-INSERTION-POINT . .
                    MOVE-CURSOR-TO-INSERTION-POIONT                                   1.10 . .
                    CLICK-MOUSE-BUTTON                                                0.20 . .
                    VERIFY-POSITION                                                   1.35 . .
                    GOAL: ISSUE-PASTE-COMMAND . . .
                              MOVE-CURSOR-TO-EDIT-MENU                                1.10 . . .
                              PRESS-MOUSE-BUTTON                                      0.10 . . .
                              MOVE-MOUSE-TO-PASTE-ITEM                                1.10 . . .
                              VERIFY-HIGHLIGHT                                        1.35 . . .
                              RELEASE-MOUSE-BUTTON                                    0.10

                                                       TOTAL TIME PREDICTED (SEC) 14.38
```

NGOMSL

- Natural GOMS Language – natural language notation to represent the model.
- Based on the Cognitive Complexity Theory (CCT).
- Considers learning time estimative.
- Task descriptions can become extensive.

CPM-GOMS

- Allow to represent parallel tasks (assumes that perceptual, cognitive and motor operators can be performed in parallel).
- Uses PERT diagrams to represent operators and operator' s dependencies.
- Based on the Model Human Processor.

GOMS - Advantages

- Significant influence in HCI theory: GOMS models continue to be applied to the evaluation of software systems, and GOMS remains an active area of scientific research.
- Helps to find usability problems.
- Saves time and resources.
- Easy to build a simple model.
- Predictive: it can be used to predict the time it will take a user to perform the tasks under analysis, as long as the developer can come up with time estimates for the operators involved in each model.
- Descriptive: it is a representation of the way a user performs tasks on a system. The methods, sub-goals and selection rules provide the designer with a description of the process
- Prescriptive: it can serve as a guide for developing training programs and help systems.

GOMS - Limitations

- Previews are only valid for experienced users, who do not make errors.
- Do not considers differences between users (statistical average for operator' s execution time).
- Do not consider the social impact nor the user satisfaction.

Evaluation

Evaluation

- 3 main goals:
  - System’ s functionalities
    - Available functionalities according to the analysis
  - Interface impact on the user
    - Learnability, usability, user’ s attitude
  - Identification of system's specific problems

Evaluation

- Design: possibly without directly involving the user.
- Implementation: studies the real use of the system.
- Some techniques are applied in both cases.

Heuristic evaluation

- Created by J. Nielsen, who made several studies to evaluate its efficiency (favourable cost/benefits rate).

- Executed by an expert.

- Based on analysis and judgement.

- Process:
  - Evaluator exhaustively inspects the interface
  - Compare the interface against heuristics
  - Elaborate a list of usability problems
    - Explains and justify each problem according to heuristics.

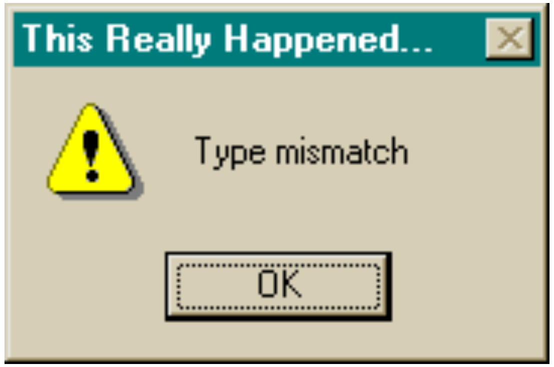
Usability guidelines (Heuristics)

- A lot to choose from
  - Nielsen principles

    (https://www.nngroup.com/articles/ten-usability-heuristics/)
  - Tognazzini principles (http://www.asktog.com/basics/firstPrinciples.html)
  - Norman: Design of Everyday Things
  - Mac, Windows, ... guidelines
- Help to select design alternatives
- Help to identify problems in interfaces (heuristic evaluation).

Usability heuristics – J. Nielsen

1. Match the real world
   - “Speak the user language”
   - Use common words
   - Don't put limits on the user defined names
   - Allow synonyms in command languages
   - Metaphor are useful, but...take care.



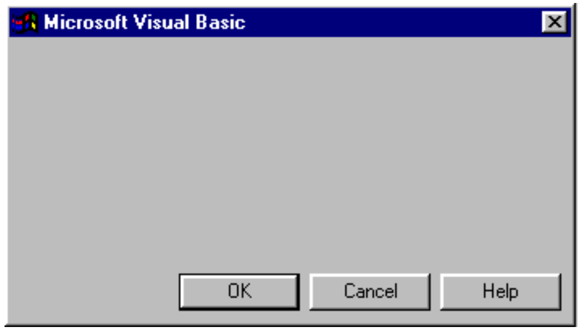
This Really Happened...

Type mismatch

Usability heuristics – J. Nielsen

2. Consistency & Standards

- “Principle of least surprise”
  - Similar things should look and act similar
  - Different things should look different
- Wording, colour, position, size, ordering
- Follow platform standards.



Usability heuristics – J. Nielsen

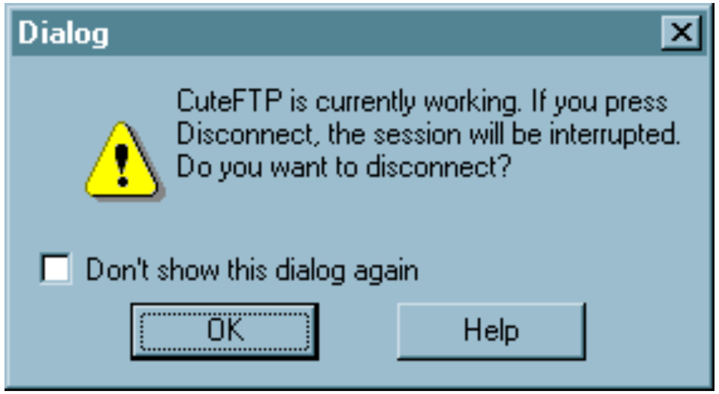
3. Help & Documentation
   - In general, users don't read user guides

   ... except when they have no other choice

   - But user guides and on-line help are essential
   - Help should be:
     - Contextual
     - Searchable
     - Task-oriented
     - Concrete
     - Concise

Usability heuristics – J. Nielsen

4. User control & Freedom
   - "Clearly Marked Exits"
   - Users should not be trapped by the interface
   - Provide "Undo" to support exploration
   - Allow users to cancel long operations
   - Option "cancel" in dialogue boxes.



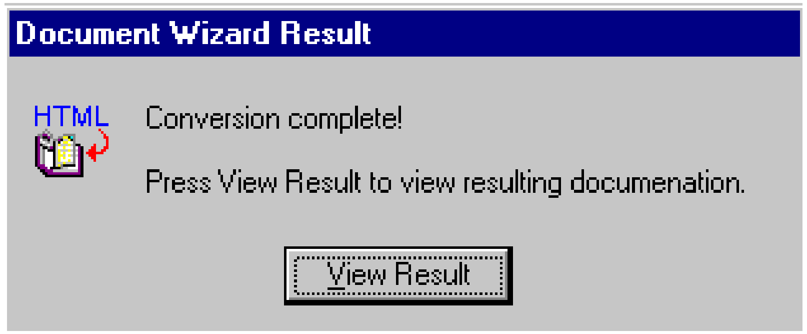
CuteFTP is currently working. If you press Disconnect, the session will be interrupted. Do you want to disconnect?

☐ Don't show this dialog again

Usability heuristics – J. Nielsen

5. Visibility of system status

- “Feedback”
- Keep the user informed about the system state
  - Selection highlight
  - Cursor change
  - Status bar, progress bar
  - Don't over do it
- Response time
  - < 0,1s: seems instantaneous
  - 0,1-1s: user notices, but there is no need for feedback
  - 1-5s: display busy cursor
  - > 5s: display progress bar



Document Wizard Result

Conversion complete!

Press View Result to view resulting documenation.

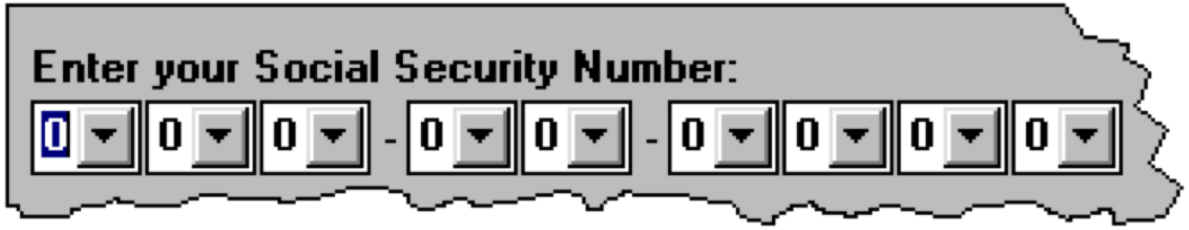
Usability heuristics – J. Nielsen

6. Flexibility & Efficiency of use
   - Shortcuts for frequent operations
     - Keyboard accelerators
     - Command abbreviations
     - Bookmarks
     - History
   - Macro for repetitive actions

Usability heuristics – J. Nielsen

7. Error prevention

- Don’t give users the opportunity to make errors
- Selection is less error-prone than typing
- Disable illegal operations
- Avoid modes.

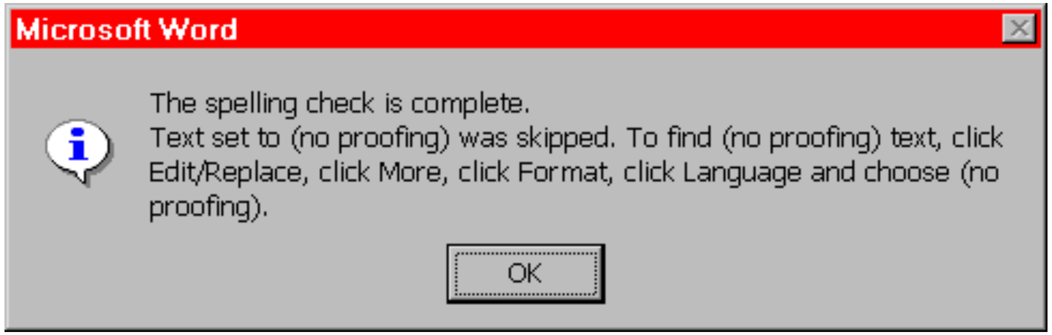


Enter your Social Security Number:

Usability heuristics – J. Nielsen

8. Recognition, Not Recall

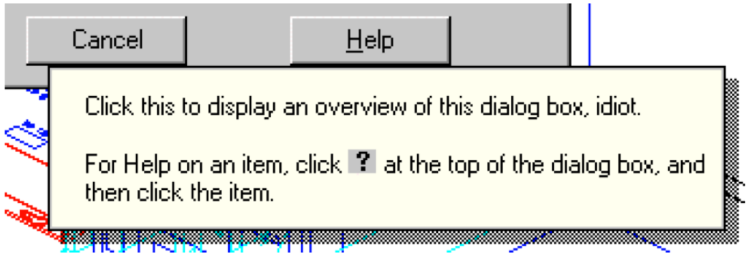
- "Minimize Memory Load"
- Norman: "Knowledge in the head" vs "knowledge in the world"
  - Menus vs commands
  - Combo boxes vs textboxes
- All needed information should be visible.



The spelling check is complete.
Text set to (no proofing) was skipped. To find (no proofing) text, click Edit/Replace, click More, click Format, click Language and choose (no proofing).

Usability heuristics – J. Nielsen

9. Error reporting, diagnosis, and Recovery
   - Good error messages should:
     - Be precise: “Can't open file” vs “Can't open file nnn.doc”
     - Be constructive: why error occurred and how to fix it
     - Be polite and non-blaming
     - Hide technical details until requested



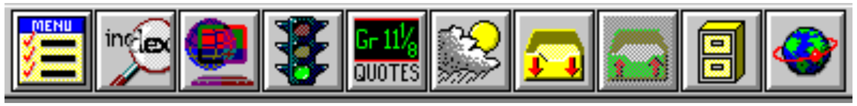
Click this to display an overview of this dialog box, idiot.

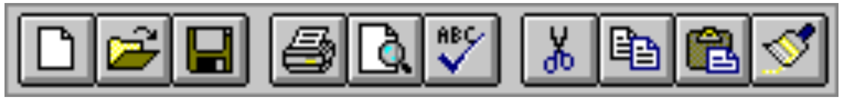
For Help on an item, click ? at the top of the dialog box, and then click the item.

Usability heuristics – J. Nielsen

10. Aesthetic and Minimalist Design

- "Simplicity"
- "Less is more"
  - omit all that is superfluous
- Good graphic design
  - few, well-chosen colours and fonts
  - group with white spaces
  - alignment controls
- Use concise language
  - Choose labels carefully



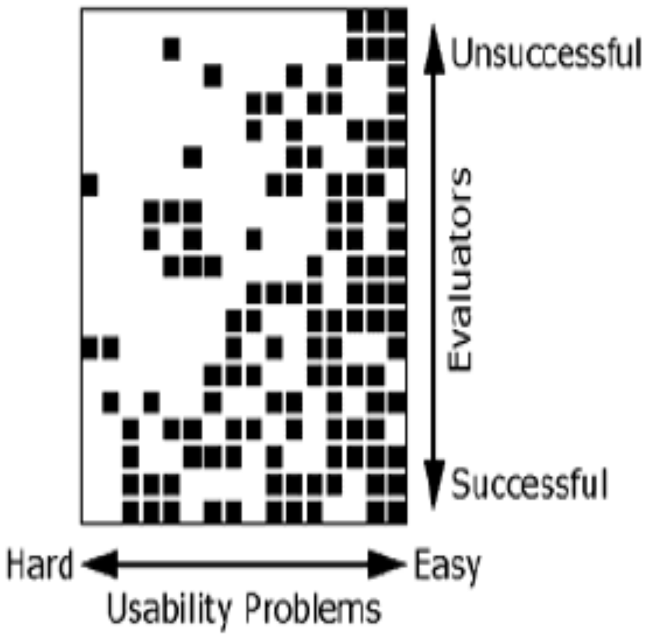


Heuristic evaluation

- Heuristic evaluation ≠ user tests
- The evaluator is not an user.
- Analogy: code inspection vs testing
- Allows to discover problems not detect in the user tests (ex: font inconsistency).
- Applies to both sketches and functional prototypes.

Heuristic evaluation

- Use several evaluators
  - Different evaluators find different problems
  - Each new evaluator finds few new problems
  - Nielsen recommends 3-5 evaluators
  - Good evaluators find simple and complex problems



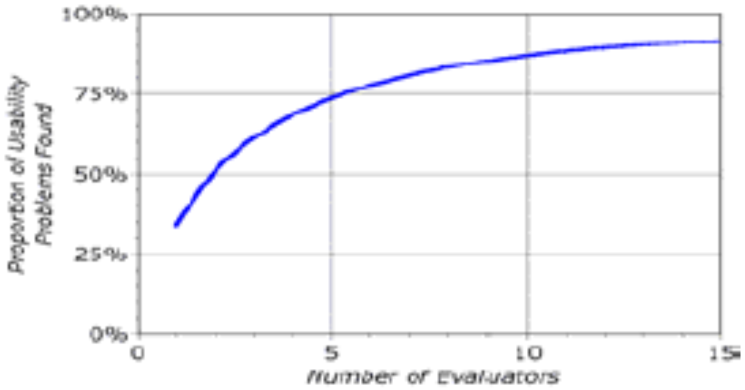
Heuristic evaluation

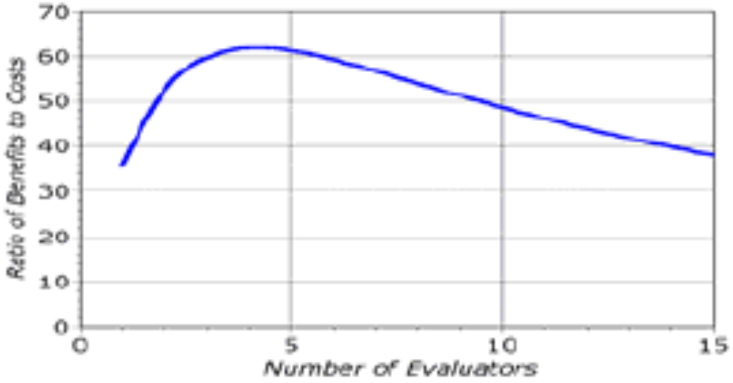
- One evaluator – unreliable results
  - – detects 35% of the usability problems
- 5 evaluators
  - – detect 75% of the usability problems
- Why not more? 10? 20?
  - – Expensive
  - – Each new evaluator finds few new problems

Nielsen, 1993

Heuristic evaluation

- Cost/benefit ratio





Nielsen, 1993

Heuristic evaluation

- Formal process
  1. Training
     - Meeting for design team and evaluators
     - Application, target users, scenarios,...
  2. Evaluation
     - Evaluators work separately
     - Produce a written report or oral comments recorded by the observer
     - Identify the problems, but not their severity
  3. Severity classification
     - All problems identified by all of the evaluators are compiled in a list
     - Evaluators classify each one of the problems
     - Calculate the average of the evaluators ratings.
  4. Discussion of results
     - Design and evaluators team
     - Brainstorm → solutions

Heuristic evaluation

- Severity
  - Factors:
    - Frequency (common or rare)
    - Impact (easy or difficult to overcome)
    - Persistency (how often to overcome?)
  - Scale:
    1. Cosmetic: correction is not mandatory
    2. Minor: correct, low priority
    3. Major: correct, high priority
    4. Catastrophic: correction is essential

Heuristic evaluation

- Justify each problem with a heuristic.
  - “Allows to put out-of-stock items in the shopping cart” – “Error prevention”
  - “I do not like the font.” – subjective
- List all the problems found
  - The same interface element may have several problems.
- Inspect the interface twice
  - Once to get a general view – feel of the system
  - Again to focus on particular interface elements.
- Go beyond the Nielsen’s 10 principles
  - Affordances, visibility, Fitt’s law, colour principles...
  - But the 10 heuristics are easy to compare against

Heuristic evaluation report

- Be polite
  - Not: “the menu organization is a mess”
  - Better: “menus are not organized by function”
- Be specific
  - Not: “text is unreadable”
  - Better: “text is too small and as poor contrast”
- Also include positive comments.

Heuristic evaluation report

- Should include (for the practical work)):

Nº	Problem	Heuristic	Description	Severity	Solution	Screenshot

Heuristic evaluation

- Applies to:
  - Sketches
  - Paper prototypes
  - Unstable prototypes

- Pros e contras

  + Cheaper

  + Quick

  + Identifies a lot of problems: minor and mayor.

  - More difficult to identify missing elements on a sketch.

  - More difficult to identify problems related to the problem domain.

Heuristic evaluation

- Alternate heuristic evaluation with user tests
  - Identify different problems
  - Heuristic evaluation – less expensive

- The observer can help the evaluator
    - As long as the problems are identified.

Tog’s Principles

1. Aesthetics
2. Anticipation
3. Autonomy
4. Colour
5. Consistency
6. Defaults
7. Discoverability
8. Efficiency of the User
9. Explorable Interfaces
10. *Fitt’s Law*
11. Human-Interface Objects
12. Latency Reduction
13. Learnability
14. Metaphors
15. Protect the User's Work
16. Readability
17. Simplicity
18. Track State
19. Visible Navigation

Tog's Principles

1. Aesthetics
   - Aesthetics should never trump usability.
2. Anticipation
   - Anticipate the user' s needs.
3. Autonomy
   - Give control to the user.
4. Colour
   - Color blindness. Color as vital interface element
5. Consistency
   - Mainly, consistency with the user expectations.

Tog’s Principles

6. Defaults
    - Easy to change, selected

7. Discoverability
    - If the user cannot find, it does not exists

8. Efficiency of the User
    - Focus on the user productivity, not the computer’ s
    - Quicker? Warm water in the microwave for 1 m e 10s or 1m e11s?
    - Keyword should come first on buttons labels and menus.

9. Explorable Interfaces
    - “Give users well marked roads and landmarks, then let them shift to four-wheels drive”
    - Reversible actions – “Undo”

Tog’s Principles

10. Fitt’ s Law
    - Big buttons are faster

11. Human-Interface Objects
    - Ex: directories, files, recycle bin
    - Consistent, stable, self-meaningful

12. Latency Reduction
    - Multi-tasking
    - Visual and audio feedback from buttons in 50 ms
    - ½ -2s display hourglass, animated hourglass
    - Progress bar, sign operations end (beep)
    - Messages indicating the system’ s actions
    - Trap multiple clicks of the same button or object

Tog’s Principles

13. Learnability

- Ideal: no learning curve

14. Metaphors

- Chose metaphor that allow an instantaneous understanding of the conceptual model.
- Create images in the user’ s mind.

15. Protect the User's Work

- Make sure the users never lose their work

16. Readability

- Contrast. Black and white.
- Font size: visual deficiencies, elderly people.
- Contrast. Black text on white background.
- Font size: visual deficiencies, elderly people.

Tog’s Principles

17. Simplicity

- Avoid the ”illusion of simplicity”

18. Track State

- Where was the user in the last session?
- Cookies

19. Visible navigation

- Make navigation visible
- Reduce navigation needs
- Clear and natural navigation

Cognitive Walkthrough

- Focus on learnability: users prefer to learn while exploring.
- Code walkthrough in software engineering.
- Evaluators execute a sequence of actions to achieve a goal, searching potential usability problems.

Cognitive Walkthrough

For a walkthrough, you need:

1. Prototype description (don't have to be complete, but must be detailed).

2. Task description (frequent task).

3. Complete list of the actions needed to complete the task with the given prototype.

4. Indications about the users and their experience.

Cognitive Walkthrough

- Evaluators step through the sequence of actions (step 3) and, for each action, try to perceive if the users:
  - try to execute the action to perform the task
  - notice that the correct action is available (visibility)
  - identify the action they should do (see the button and know that is the right button)
  - understand the action feedback.

Cognitive Walkthrough

- Results:
  - Form with information 1,2 and 4, date, hour, evaluator.
  - Separate form for every action (from 3) to answer the former questions.

User testing

User tests

- We can not tell the quality of an interface before it is used or adopted.

- It is difficult to preview what the real users would do.

- Select participants:
  - Representative users in terms of domain knowledge
  - Doctors vs Medical Students

- Select tasks
  - Realistic
  - Not fragmented
  - Avoid long tasks

User tests

- Formative evaluation
  - Identifies usability problems to be corrected in the next iteration
  - Evaluates a prototype or implementation in a controlled environment (lab) on selected tasks
  - Qualitative observations (usability problems)
  - May not say enough about the use of an interface in a real environment with real tasks.
- Field study
  - Evaluates an implementation in the real context with real tasks
  - Mostly qualitative observations (users in the real environment)
- Controlled experiments
  - Test an hypothesis (ex: interface X is faster than interface Y)
  - Evaluate a prototype or implementation in a controlled environment (lab) on selected tasks
  - Quantitative observations (time, nº of errors).

User tests

- Pressure on users:
  - performance anxiety
  - feels like an intelligence test
  - afraid to fail
  - feel observed

User tests

- Treat the user with respect
  - Time
    - Don' t waste time, prepare it ”à priori”
  - Comfort
    - Make the user comfortable (physically and psychologically)
  - Consent
    - Inform the user, be transparent
  - Privacy
    - Preserve the user' s privacy, do not identify the user
  - Control
    - The user can stop at any time.

User tests

- Before the test
  - Time
    - Pilot-test all materials
  - Comfort
    - “We’ re testing the system; we’ re not testing you.”
    - “Any difficulties you encounter are the system’ s fault. We need your help to find these problems.”
  - Information and consent
    - Brief description of the purpose of the test
    - Inform about the observers and taping
    - Answer any questions before hand (as long as it doesn’ t influence the test).
  - Privacy
    - “Your test results are confidential.”
  - Control
    - “You can stop at any time”.

User tests

- During the test
  - Time
    - Eliminate unnecessary tasks
  - Comfort
    - Calm, relaxed, not distracting environment
    - Take breaks in long sessions (water, coffee, move)
    - Never act disappointed
    - Give tasks one at a time
    - First tasks should be easy to encourage users
  - Information and consent
    - Answer questions (where they won’t bias).
  - Privacy
    - User's boss shouldn't be watching
  - Control
    - User can give up a task and go on to the next
    - User can quit entirely.

User tests

- After the test
  - Comfort
    - Thank you
    - Tell users how helpful they were
  - Information and consent
    - Answer any questions that you couldn't answer before.
  - Privacy
    - Don’t publish user-identifying information
    - Don’t show video or audio without user’s permission

Formative evaluation

- Select some appropriate users
  - Should be representative of the target user group, based on the user analysis.
- Give each user some tasks
  - Should be representative, based on the task analysis
- Watch users do the tasks

Formative evaluation

- Roles
  - User
  - Facilitador
  - Observers

Formative evaluation

- User (use the interface to execute tasks)
  - Think aloud
    - windows to the user reasoning process
  - Problems
    - Feel weird or ashamed
    - Thinking aloud may alter behaviour
    - Disrupts concentration
    - May be quiet when performing demanding tasks (facilitator intervention)
  - Alternative: constructive interaction
    - Users working in pairs – natural conversation

Formative evaluation

- Facilitator (test leader)
  - Does the briefing
  - Give tasks
  - Development team spokesman
  - Coaches the user to think aloud by asking questions
    - "What are you thinking?"
    - "What are you trying to do?"
    - "Why did you try that?"
  - Controls the session

Formative evaluation

- Observer (watch and annotate)
  - Be quiet! (invisible, as far as possible)
    - Don’t help, don’t explain, don’t point out errors
  - Take notes
    - In order to recreate the users’ behaviour
    - Critical incidents: events that strongly affects task performance or satisfaction
    - Negative:
      - Errors
      - Repeated attempts
    - Positive: denote satisfaction.

Formative evaluation

- Recording observations
  - Pen and paper notes
    - Prepare forms
  - Audio recording
    - Think aloud
  - Video recording
    - Two cameras: one for the user and one for the screen
    - Facial expressions
    - Allows observation from another room
    - Generates too much data, but allows replay (eventually with the user).
  - Screen capture and logging
    - Cheap and unobtrusive
    - Huge quantity of data to analyse.

Formative evaluation

- How many users?
  - Every tested user finds a fraction L of usability problems (typical L = 31%)
  - Assuming that user tests are independent (since users do not talk to each other), n users find a fraction 1- $(1\text{-}L)^n$.
  - So 5 users find 85% of the problems.

Nielsen, J. and Landauer, T., A mathematical model of the finding of usability problems, Proceedings of INTERCHI 93, Amsterdam, The Netherlands, 1993, pp. 206-213.

Formative evaluation

- Which one is better?
  - Using 15 users to find 99% of problems with one design iteration
  - Using 5 users to find 85% problems with each of three design iterations

- For multiple user classes, get 3-5 users from each class

Formative evaluation

- How many users?
  - 5 is the magic number, but L can be much smaller (Spool and Schroeder L = 8%, so 5 users only find 35% of the problems)
  - L may vary from problem to problem
    - Different problems have different probabilities of being found, caused by:
      - Individual differences
      - Interface diversity
      - Task complexity
  - It is not possible to predict with confidence how many users may be needed.

Spool, J. and Schroeder, W., Testing web sites: five users is nowhere near enough, CHI' 01 Extended Abstracts, Seattle, Washington, 2001, pp. 285-286.

Controlled experiments

- Formulate an hypothesis to test (quantifiable)
  - Prediction of the outcome of the experiment
    - Interface X is faster than interface Y
- Manipulate the independent variables to produce different conditions for comparison
  - Different interfaces, tasks, number of menu items
- Measure dependent variables
  - Times, errors, user’s preferences
- Use statistical techniques to analyse how variations in the independent variables affect the dependent variables and to accept or reject the hypothesis.

Controlled experiments

- Users
  - Should be representative of the target users population.
    - (= age, education level, experience – whenever possible do it with real users).
  - Sample size
    - Depends on the users and resources availability
    - Minimum 10 (..., but more is better, depending on the statistical methods and the interface).

Controlled experiments

- Hypothesis
  - Experiment results preview
  - 1 variation in the independent variables cause changes in the dependent variables.
  - Experiment goal: demonstrate the hypothesis is true.

“The Macintosh menu bar, which is anchored to the top of the screen, is faster to access than the Windows menu bar, which is separated from the top of the screen by a window title bar.”

Controlled experiments

- Variables
  - Independent – manipulated to produce different conditions to allow comparison of results (ex: interaction techniques, icon design, menu configuration).

“Kind of interface: Mac menubar or Windows menubar.”

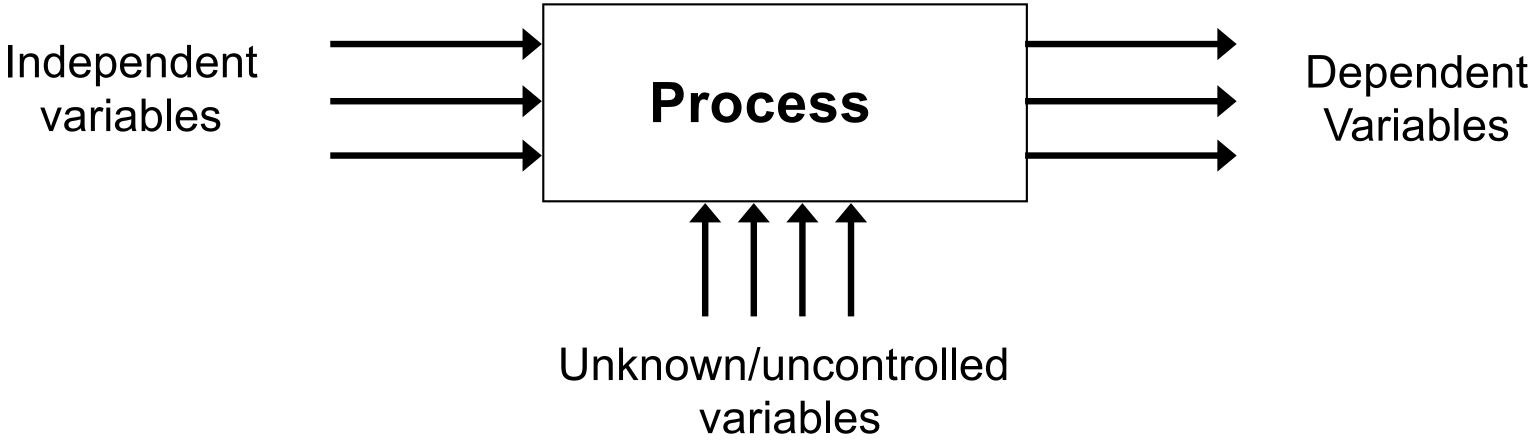
Controlled experiments

- Variables
  - Dependent – affected by the independent variables. Their resulting values are measured (ex: task execution time, number of errors).

“Task completion time.”

Controlled experiments

- Unknown variables may affect the process in unpredictable ways. Goal: eliminate the effect of these unknown variables.



Controlled experiments

- Internal validity – the effect produced in the output variables are caused by variations in the input variables (not by variations in the uncontrolled variables).
  - All variables, except the independent variables, should be constant: room, light, machine, mouse, keyboard, tasks.
  - Big sample
- External validity – the observed effect can be generalized for the real world outside the lab, when we can not control the unknown variables.
- Reliability - will consistent results be obtained by repeating the experiment? A unique test is not meaningful.

Controlled experiments

A

B

- Hypothesis: A has a different number of balls than B
- Independent variables: identity of the box
- Dependent variables: number of balls inside the box

Controlled experiments

- Reliability
  - Manually counting is reliable for a few number of balls
  - Repeated counting improves reliability, but it is slow...

- Internal validity
  - Weight the boxes instead of counting the balls
  - Ball A may have a different weight than ball B
    - Dependent variable (total weight) is a function not only of the number of balls
  - Box A may have a different weight than box B
    - Use the same box C to weight both sets of balls

- External validity
  - Does this result apply to all boxes in the world?

Controlled experiments

- Internal validity
  - Ordering effects
    - Users learn and get tired.
    - Don’t present tasks in the same order to all users – Randomize.
  - Selection effects
    - Don’t use pre-existing groups (unless group is an independent variable).
    - Randomly assign users to groups.
  - Experimenter bias
    - Experimenter may prefer an interface over the other
    - Give training and briefing in paper, not in person
    - Double-blind experiments
      - Essential if measurement of dependent variables requires judgement.

Controlled experiments

- External validity
  - Population
    - Draw a random sample from your real target population
  - Ecological
    - Make lab conditions as realistic as possible in important respects
  - Training
    - Training should mimic how real interface would be encountered and learned
  - Task
    - Base tasks on task analysis

Controlled experiments

- Reliability
  - Previous experience
    - Novices and experts: separate
  - User differences
    - Fastest users are **10 times** faster than slowest users
    - Intelligence, visual acuity, memory
  - Tasks design
    - Do tasks measure what you’ re trying to measure?
  - Measurement errors
    - Time on task includes coaching, distractions
  - Solutions
    - Eliminate uncontrolled variables
      - Select users with the same experience
      - Give all users the same training
      - Measure dependent variables precisely
    - Repetition
      - Many users, many trials
      - Standard deviation of the mean shrinks like the square root of N (i.e., quadrupling users makes the mean twice as accurate)

Controlled experiments

- **Blocking**
  - Eliminate uncontrolled variation, and therefore increase reliability
    - Divide samples into subsets which are more homogeneous than the whole set
    - Apply all conditions within each block
    - Measure difference within block
    - Randomize within the block to eliminate internal validity threats

Controlled experiments

- **Method**
  - **“Between-subjects” design**
    - Each group tests only one interface
    - Results are compared **between** different groups
      - Eliminate ordering effect
        - » User can’t learn from one interface to do better on the other
      - Do not take into account the user’s differences → needs more repetition
  - **“Within-subjects”**
    - Each user test all the interfaces (in random order)
    - Results are compared **within** each user
      - Eliminates variation due to user differences
        - » Cheaper – fewer users
      - Ordering effect – ≠ ordering to ≠ users
      - Fatigue effect
  - Which is better?
    - User differences cause much more variation than ordering effects
    - First option requires more users
    - First option may be more externally valid
    - Thread-off solution (when there is more than one independent variable).

Controlled experiments

- **Counterbalancing**
  - **Reduce ordering effects by systematically varying the order of conditions**
  - **Latin square design**
    - Randomly assign users to equal-size groups
    - A, B, C, ...are the experimental conditions
    - Each experimental condition occurs the same number of times at each position in the order

A	B	C
B	C	A
C	A	B

Balanced latin square

A	B	D	C
B	C	A	D
C	D	B	A
D	A	C	B

Statistical testing

- Compute statistical measures on the experiment data
  - Mean
  - Standard deviation
- Apply a statistical test
  - t test: are two means really different?
  - ANOVA (ANalysis Of VAriance): are three or more means different?
- Inspect p value
  - p value = probability that the observed difference happened by chance
  - If $p<0.05$, then we are 95% confident that there is a difference between Windows and Mac.

Measurements

- Self-report
  - Cheap
  - Biased by reactive effects (politeness, social desirability)
  - Surveys - satisfaction
- Observation
  - More expensive
  - More objective
  - As unobtrusive and discrete as possible

Questionnaires

- Pre-defined questions – less flexible and faster than interviews.
- Can be used in various phases of the design process.
- Simple and cheap to execute, provides subjective information.
- Types of questions:
  - General
  - Open-ended
  - Scalar
  - Multiple choice
  - Ranked

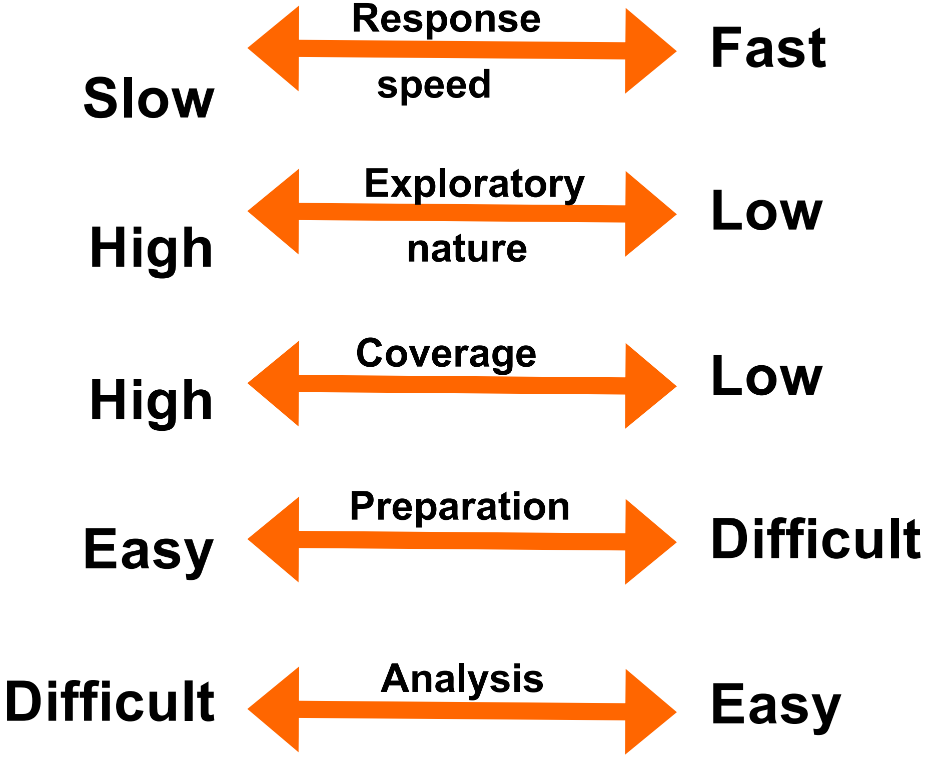
Questionnaires

- Language selection:
  - Simple: Use the user’ s vocabulary (units vs departments).
  - Be specific (not vague).
  - Short questions.
  - Do not influence the answers.
  - Technical precision.

Questionnaires

- Types of questions:
  - General
    - Help to define the user profile (age, gender, experience).
  - Open-ended
    - Collect subjective data
    - Easy to create, difficult to analyse
    - Suggestions to improve the interface
  - Scalar
    - Judge a statement according to a numeric scale (Agreement or disagreement 1-5; 1-7).
  - Multiple choice
  - Ranked
    - Sort a list of items
    - Show user’s preferences.

Questionnaires



Open-ended questions		**Closed questions**
Slow	**Response speed**	**Fast**
High	**Exploratory nature**	**Low**
High	**Coverage**	**Low**
Easy	**Preparation**	**Difficult**
Difficult	**Analysis**	**Easy**

Questionnaires

- Scalar:
  1. Evaluate the user's attitude and characteristics.
  2. Know the user’s judgement related to interface details.
- Ex:

“The system is easy to use”

Strongly Disagree **Strongly Agree**

1 2 3 4 5

Questionnaires

- Scales:
  1. Nominal
     - Classify objects
     - “What kind of applications you use more?”
       1. Word processor
       2. Spreadsheet
       3. DB
       4. Drawing application
     - Results: totals.

Questionnaires

- Scales:

  2. Ordinal
     - Ordered classification
     - "The support team is:"
       1. Excellent
       2. Good
       3. Satisfactory
       4. Not good
       5. Bad

  - Difference between 1 and 2 may be different from difference between 3 and 4.

Questionnaires

- Scales:

3. Interval
   - Ordered classification
   - Equal ranges between each element
   - Allow mathematical calculations.
   - Ex: temperature: Celsius and Fahrenheit.
   - “How useful is the support given by the support team?”

Inútil				**Muito útil**
1	2	3	4	5

   - Allows a more quantitative analysis.

Questionnaires

- Scales:

4. Ratio
   - Similar to interval, but includes 0.
   - Ex: ruler scale.
   - "How many hours you spend with your computer per day?"

0 2 4 6 8

   - Infrequent use

Questionnaires

- Legibility: alignment, space, font ...
- Available space for the anwser
- Ask user to sign the choice with a circle (avoid ambiguous answers).
- Consistency: appearance, help location, scales format.
- Question’s ordering: important questions first (from the user’s point of view).
- Group related questions.

Questionnaires

- Session (methods)
  - Group session
  - Individual session
  - Fill in and return centralized in the working place.
  - Distribution and collection by post
  - Electronic

Evaluation

- Environment
  - Laboratory studies
    - allow controlled experimentation and observation
    - looses naturalness of the user’ s environment
  - Field studies
    - do not allow control over user activity
  - Both studies should be made:
    - Lab studies dominating early stages
    - Field studies for new implementations

Evaluation

- Measurements
  - Quantitative
    - numeric
    - can be easily analysed using statistical techniques
  - Qualitative
    - non-numeric
    - difficult to analyse
    - provide importantt detail which can not be determined from numbers.
  - Numeric scales can be used to gather subjective data – Likert scales.

User testing

- Bruce Tognazzini

  “ I have spent much of my twenty-five year career in software design troubleshooting projects that are in trouble, helping them get back on course. The single thread that ran through every one of them was a lack of user testing.”

  “If your people don’t know how to user test, find someone who does. Whether you bring in an outside design or hire your own people, make sure they are telling you all about their plans to test, because if they don’t test, your customers will, and it will cost you a whole bunch of money.”

References

- Apple human interface guidelines

- Card, S., Morn, T. And Newell, A., The keystroke-level model for user performance with interactive systems, Communications of ACM, 23, 1980, pp. 396-410.

- Card, S. K., Moran, T. P. and Newell, A., The Psychology of Human Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

- Dix, Alan, Finlay, Janet, Abowd, Gregory, Beale, Russel. Human-Computer Interaction. Prentice Hall Europe, London, 1998.

- John, B. and Kieras, D., Using GOMS for user interface design and evaluation: which technique?. Journal ACM Transactions on Computer-Human Interaction (TOCHI), v.3 n. 4, 1996.

- Nielsen, J., Usability Engineering, Academic Press,1993.

References

- Nielsen, J., Ten Usability Heuristics - http://www.useit.com/papers/heuristic/heuristic_list.html

- Nielsen, J. and Landauer, T., A mathematical model of the finding of usability problems, Proceedings of INTERCHI 93, Amsterdam, The Netherlands, 1993, pp. 206-213.

- Norman, Donald Design Rules Based on Analyses of Human Error. *CACM*, v.26 n.4, 1983.

- Norman, Donald. *The Design of Everyday Things*. MIT Press, 1998.Spool, J. and Schroeder, W., Testing web sites: five users is nowhere near enough, CHI' 01 Extended Abstracts, Seatle, Washington, 2001, pp. 285-286.

- Tog Interaction Design Principles - http://www.asktog.com/basics/firstPrinciples.html