Chapter 8

Usability Inspection Methods

Inspection of interface design using *heuristic* methods (based on analysis and judgement rather than experiment).

1. *Heuristic Evaluation*: A small team of evaluators inspects an interface using a small checklist of general principles and produces an aggregate list of potential problems.
2. *Guideline Checking*: An evaluator checks an interface against a detailed list of specific guidelines and produces a list of deviations from the guidelines.
3. *Cognitive Walkthrough*: A small team walks through a typical task in the mind set of a novice user and produces a success or failure story at each step along the correct path. [analyses learnability]
4. *Guideline Scoring*: An evaluator scores an interface against a detailed list of specific guidelines and produces a total score representing the degree to which an interface follows the guidelines.
5. *Action Analysis*: An evaluator produces an estimate of the time an expert user will take to complete a given task, by breaking the task down into ever smaller steps and then summing up the atomic action times. [analyses efficiency]

Would You Use Untested Software?

Would you knowingly use untested software?

- How many of you have written programs that are *used* by other people?
- How many of you have watched or observed users using your software?
- How many of you actually evaluated or tested your interface before it was used?
- In practice, most software developers do not actually conduct *any* kind of usability evaluation [due to perceived expense, lack of time, lack of expertise, lack of inclination, or lack of tradition].

References

+ Jakob Nielsen and Robert L. Mack (Eds.); *Usability Inspection Methods*; John Wiley, 1994. ISBN 0471018775 (com, uk) [Nielsen and Mack 1994]
+ Chauncey Wilson; *User Interface Inspection Methods*; Morgan Kaufmann, 2013. ISBN 012410391X (com, uk) [C. Wilson 2013]

Online Resources

+ Jeff Sauro; *What's the difference between a Heuristic Evaluation and a Cognitive Walkthrough?*; 02 Aug 2011. https://measuringu.com/he-cw/

8.1 Heuristic Evaluation

Heuristic evaluation is a formative usability inspection method.

First described by Nielsen and Molich [1990]:

- Small team of evaluators (usually usability specialists) systematically checks interface design against small set of recognised usability principles (the "heuristics").
- Often called an expert review.

References

+ Jakob Nielsen and Rolf Molich; *Heuristic Evaluation of User Interfaces*; Proc. CHI'90, May 1990. [Nielsen and Molich 1990]
  - Jakob Nielsen; *Finding Usability Problems Through Heuristic Evaluation*; Proc. CHI'92, May 1992. [Nielsen 1992]
  - Jakob Nielsen; *Enhancing the Exploratory Power of Usability Heuristics*; Proc. CHI'94, Apr 1994. [Nielsen 1994a]

Online Resources

+ Jakob Nielsen; *How to Conduct a Heuristic Evaluation*; https://nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/ [Nielsen 1994b]
+ Jakob Nielsen; *10 Usability Heuristics for User Interface Design*; https://nngroup.com/articles/ten-usability-heuristics/ [Nielsen 2020]
  - Jakob Nielsen; *Heuristic Evaluation*; https://nngroup.com/topic/heuristic-evaluation/ [Nielsen 2022]
+ Luke Chambers; *How to Run an Heuristic Evaluation*; https://uxmastery.com/how-to-run-an-heuristic-evaluation/ [Chambers 2016]
+ Enzinger et al; *Heuristic Evaluation of Enigmail*; https://projects.isds.tugraz.at/enigusab/he/he.html [Enzinger et al. 2017]

Usability Heuristics

The first nine heuristics were defined in Nielsen and Molich [1990]. Some of the heuristics were renamed and the tenth heuristic was added in Nielsen [1994a]. The category names in square brackets are the new names of the corresponding heuristics, in those cases where they are different.

1. **Feedback** [Visibility of System Status]

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

   For example: busy cursor [1–10s], progress indicator [>10s].

2. **Speak the Users' Language** [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 logical order.

   Match users' mental model. Beware of misleading metaphors.

3. **Clearly Marked Exits** [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.

4. **Consistency** [Consistency and Standards]

   Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. **Error Prevention**

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

   For example: select file from menu rather than typing in name, confirmation before dangerous actions, beware of modes, avoid similar command names, warning if Caps Lock is activated when entering a password, etc.

6. **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.

   Users' short-term memory is limited. Provide examples, default values, easily retrievable instructions.

7. **Accelerators** [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.

   For example: abbreviations, command keys, type-ahead, edit and reissue previous commands, menu of most recently used files, macros.

8. **Minimalist Design** [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. "Less is more"

9. **Good Error Messages** [Help Users Recognise, Diagnose, and Recover from Errors]

   Error messages should be expressed in *plain language* (no codes or jargon), *precisely* indicate the problem, and *constructively* suggest a solution.

   Phrase error messages *defensively*, never blame the user. Multilevel messages. Link to help system.

10. **Help and Documentation**

    Help and documentation should be easy to search, focused on the user's tasks, list concrete steps to be

carried out (recipes), and make liberal use of examples.

Limits on Response Times

- **0.1 sec.**: is the limit so that the system appears to react instantaneously.
  Important for direct manipulation, virtual world navigation.
- **1 sec.**: is the limit so that the user's flow of thought stays uninterrupted.
  Display a busy cursor if things will take longer than 1 sec.
- **10 secs.**: is the limit for keeping the user's attention on the task at hand.
  Display a progress indicator if things will take longer than 10 secs.

From Section 5.5 of [Nielsen 1993b] and [Nielsen 1993a].

Performing a Heuristic Evaluation

Step-by-step recipe for performing a heuristic evaluation, in five phases:

1. Preparation
2. Individual Evaluations
3. Aggregation
4. Severity Ratings
5. Debriefing and Report

1. Preparation

- The design may be verbal description, paper mock-up, working prototype, or running system. [when evaluating paper mock-ups, pay special attention to missing dialogue elements!]
- Provide evaluators with checklist of usability heuristics.
- Optionally provide evaluators with some domain-specific training.

2. Individual Evaluations

- Each evaluator works *alone* ($\approx$ 1–2 hours).
- Interface is often examined in two passes: first pass focuses on general flow, second on particular elements in detail.
- Notes taken either by evaluator or evaluation manager.
- Make a list of (potential) problems and a list of positive findings. [Not all problems fall under a heuristic. It is OK to include problems which are not covered by any heuristic.]
- Use screen video capture software such as OBS Studio, Camtasia, or AZ Screen Recorder to record each evaluation session (including the evaluator's voice if possible), since some problems may not be reproducible later on.

- Re-create a short video clip to illustrate each finding, concisely describing the finding in the audio track.

  Where a finding cannot be recreated, extract a video clip from your evaluation recording. If the existing audio commentary does not describe the finding adequately, record a new audio commentary to describe the issue.

  In some cases, a static image (still frame) may suffice to illustrate a finding. [But not in the practicals for this course!]

3. Aggregation

- Independent findings are now aggregated into one large list (by the evaluation manager) using a spreadsheet:
  - Start by copying the longest individual list of problems over into the spreadsheet. Then look through the other individual lists and compare each problem to see if it is already in the aggregated list.
  - If two or more evaluators find the same (or very similar) problem, combine them into one.
  - List many small related problems (such as 27 individual typos) as one problem (rather than 27).
- Proceed analogously to create an aggregated list of positive findings.

4. Severity Ratings

- The aggregated list of potential problems (with associated video clips) is distributed to each evaluator.
- Each evaluator now assigns severity ratings *individually* to each problem in the large list, unseen by the other evaluators. [For example, on a simple 0 (not a problem) to 4 (catastrophic problem) scale.]
- The individual severity ratings are averaged to obtain the final mean severity rating for each problem.
- The aggregated list is sorted in decreasing order of average severity.
- Proceed analogously for the positive findings and positivity ratings.

5. Debriefing and Report

- If possible, organise a group debriefing session to recommend possible solutions.
- Write up all of your results into a report.

How Many Problems are Found?

Nielsen and Molich [1990] ran four heuristic evaluations using "usability novices" as evaluators.

They compared the average number of problems found by each evaluator with the total number of known problems in each system, as shown in Table 8.1.

Aggregated Evaluations

- Individual evaluators found relatively few problems.

Evaluation Name	*Number of Evaluators*	*Total Known Problems*	*Average No. Problems Found per Evaluator*
Teledata	37	52	51%
Mantel	77	30	38%
Savings	34	48	26%
Transport	34	34	20%

Table 8.1: The average number of problems found by individual novice evaluators in each of four heuristic evaluations. [Data extracted from Nielsen and Molich [1990, Table 2].]

Aggregate:	*1*	*2*	*3*	*5*	*10*
Teledata	51%	71%	81%	90%	97%
Mantel	38%	52%	60%	70%	83%
Savings	26%	41%	50%	63%	78%
Transport	20%	33%	42%	55%	71%

Table 8.2: The average proportion of usability problems found by various sized aggregates of novice evaluators in each of the four heuristic evaluations. [Data extracted from Nielsen and Molich [1990, Table 4].]

- Aggregating the evaluations (merging the problem lists) of several individuals produced much better results.
- See Table 8.2 and Figure 8.1.
- Group debriefing session to suggest possible redesigns.

Experience of Evaluators

The experience of evaluators influences how many problems they find [Nielsen 1992].

Study of one interface, the Banking System, a touch tone "voice response" telephone banking system, by 3 groups of evaluators:

- 31 "novice" evaluators: computer science students with no formal knowledge of UI or usability (no usability expertise).
- 19 "regular" specialists: people with UI and usability experience, but no expertise in voice-response systems (usability expertise).
- 14 "double" specialists: people with expertise both in usability and in telephone-operated interfaces (usability and domain expertise).

Task: transfer $1000 from savings account to check account.

Sample Banking System Dialogue

Figure 8.2 shows the sample dialogue given to evaluators in the study.

- This dialogue actually took place!
- The problem was that the user did not have authority to make phone transfers.

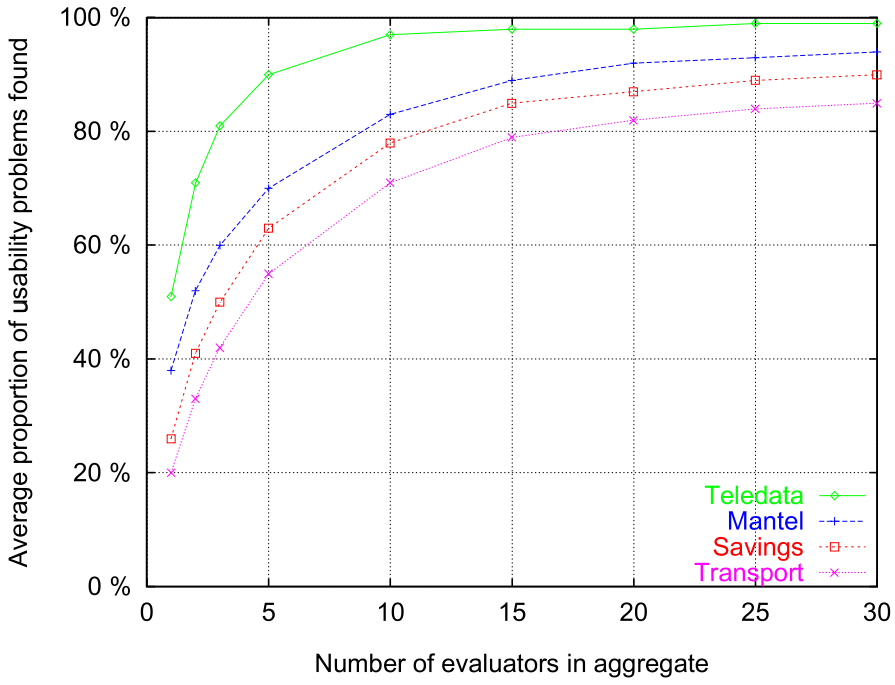

Figure 8.1: Average proportion of usability problems found by aggregates of size 1 to 30. [Results from Nielsen and Molich [1990, Figure 4].]

- The results are summarised in Table 8.3.
- Aggregating the results of individual evaluators gives the plots shown in Figure 8.3.

Conclusions from Studies

From these two studies, we can conclude that:

- Single evaluator finds only small subset of potential problems.
- Several evaluators should work *independently*, then aggregate results.
- Evaluators pay more attention to major problems without neglecting minor ones.
- In the second study [Nielsen 1992], for the Banking System interface, finding around 80% of usability problems required 2–3 double specialists, 3–5 regular usability specialists, and around 15 novice usability evaluators.
- This result does *not* generalise to all interfaces in all situations.

Heuristic Evaluation in Practice

When heuristic evaluation is used in standard industry practice today:

- It is often called an *expert review*.
- It is typically performed by 3–4 usability specialists.

[First there is a short dialogue in which the user is identified by entering an identification number and access code – this is not part of the evaluation exercise]

1. S: *Enter one for account information, three for transfers between your own accounts, . . .*
2. U: 3# {the user interrupts the system}
3. S: *Enter account to transfer from.*
4. U: 1234567890# {savings account number}
5. S: *Enter account to transfer to.*
6. U: # {an abbreviation for the checking account}
7. S: *Enter amount in cents.*
8. U: 100000#
9. S: *From account number twelve thirtyfour fiftysix seventyeight ninety to account number primary account, a transfer of one thousand dollars is to be made. Press one to confirm, zero to cancel.*
10. U: 1#
11. S: *You do not have access to this function.*

Figure 8.2: The sample Banking System dialogue used for the study. [Extracted from Nielsen [1992, Figure 1].]

No.	*Problem*	*Novice*	*Regular*	*Double*
Major Usability Problems				
1.	Error message appears too late.	68%	84%	100%
2.	Do not require dollar amount to be entered in cents.	68%	74%	79%
3.	The error message is not precise.	55%	63%	64%
4.	The error message is not constructive.	6%	11%	21%
5.	Replace term “primary account” with “checking account”.	10%	47%	43%
6.	Let users choose account from a menu.	16%	32%	43%
7.	Only require a # where it is necessary.	3%	32%	71%
8.	Give feedback as name of chosen account.	6%	26%	64%
Average for Major Problems		**29%**	**46%**	**61%**
Minor Usability Problems				
9.	Read menu item description before action number.	3%	11%	71%
10.	Avoid gap in menu numbers between 1 and 3.	42%	42%	79%
11.	Provide earlier feedback.	42%	63%	71%
12.	Replace use of 1 and 0 for accept and reject with # and *.	6%	21%	43%
13.	Remove the field label “number” when no number is given.	10%	32%	36%
14.	Change prompt “account” to “account number”.	6%	37%	36%
15.	Read numbers one digit at a time.	6%	47%	79%
16.	Use “press” consistently and avoid “enter”.	0%	32%	57%
Average for Minor Problems		**15%**	**36%**	**59%**
Average for All Problems		**22%**	**41%**	**60%**

Table 8.3: Proportion of novice, specialist, and double specialist usability evaluators finding problems in the Banking System. [Results from Nielsen [1992, Table 1].]

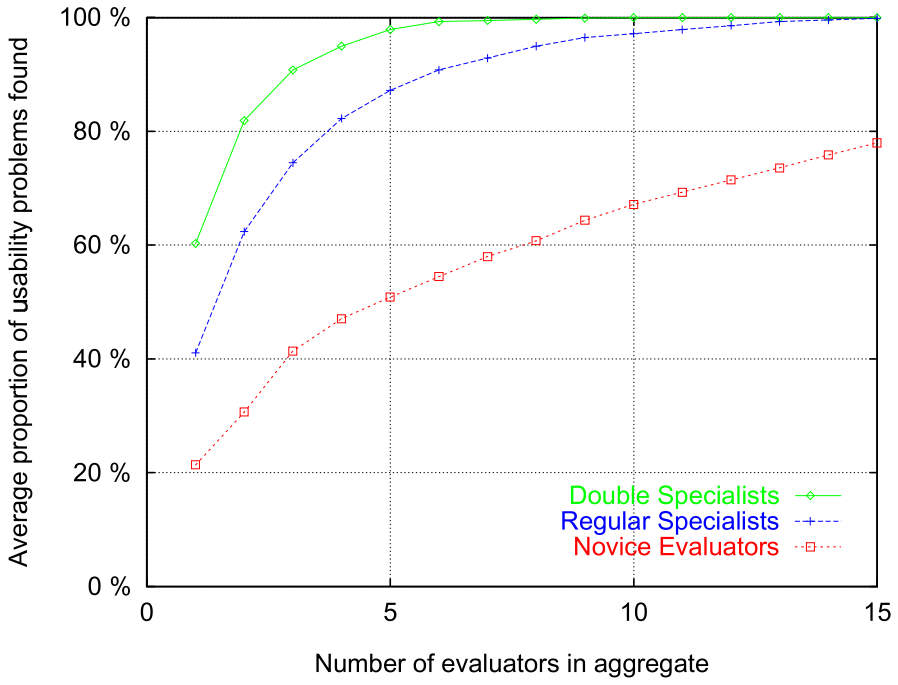

Figure 8.3: Average proportion of usability problems found by aggregates of novice evaluators, regular specialists, and double specialists. [Results from Nielsen [1992, Figure 2].]

Pros and Cons of Heuristic Evaluation

++ cheap

\+ intuitive

\+ usable early in development process

\+ finds many problems

\+ finds both major and minor problems

\- may miss domain-specific problems

8.2 Severity Ratings

Severity ratings can help prioritise the fixing of usability problems.

- After evaluation sessions, a complete aggregate list of usability problems is given/sent to each evaluator.
- Working independently, evaluators assign severity rating [on scale of 0–4] to each problem (≈ 30 mins.).
- Severity rating of single evaluator is unreliable, mean of 3–5 evaluators is satisfactory.

See the discussion of severity ratings in [Nielsen and Mack 1994], pages 47–55.

Five-Point Severity Scale

Score	*Severity*	*Fix Priority*
4	catastrophic problem	imperative
3	major problem	high
2	minor problem	low
1	cosmetic problem only	
0	not a problem at all	

Order of Criticality

To explicitly take problem frequency into account, assign *criticality* ratings.

Criticality = Severity Ranking + Frequency Ranking

Severity		*Frequency*	
4	catastrophic	4	>90%
3	major	3	51–89%
2	minor	2	11–50%
1	cosmetic	1	1–10%
0	none	0	<1%

8.3 Guideline Checking

Guideline checking is a formative usability inspection method.

Guidelines . . . specific advice about usability characteristics of an interface.

- An evaluator checks an interface against a detailed list of specific guidelines and produces a list of deviations from the guidelines.
- Whereas heuristic evaluation employs 10 broad principles, guideline checking often involves dozens (or hundreds) of more specific individual items on a checklist.

Example Sets of Guidelines

- Sidney Smith and Jane Mosier; *Design Guidelines for Designing User Interface Software*; The MITRE Corp., 1986. [944 guidelines] [S. L. Smith and Mosier 1986].

  Table 8.4 shows one of the 944 guidelines from Smith and Mosier.
- C. Marlin Brown; *Human-Computer Interface Design Guidelines*; Ablex, NJ, 1988. [302 guidelines] [C. M. Brown 1998].
- Deborah Mayhew; *Principles and Guidelines in Software User Interface Design*; Pearson, 2008. [288 guidelines] [D. Mayhew 2008].
- David Travis, Userfocus; Web Usability Guidelines; [247 guidelines in 9 categories] [Travis 2016]. `https://userfocus.co.uk/resources/guidelines.html`
- teamsuccess; Usability Checklist for web sites; [56 guidelines in 8 categories] [teamsuccess 2020]. `https://teamsuccess.io/UX`

Online Guideline Checking Application

- Testpad `ontestpad.com` provides an environment for creating online checklists (called scripts), which you can then check through on a tablet (say) for each interface. [Video: `https://vimeo.com/49168687`]
- The data can then be downloaded as a CSV file for offline analysis.

Pros and Cons of Guideline Checking

\+ cheap

\+ intuitive

\+ usable early in development process

\- time-consuming

\- can be intimidating – often hundreds or thousands of specific guidelines.

1.3 DATA ENTRY: Text

1.3/1 Adequate Display Capacity

Ensure that display capacity, i.e., number of lines and line length, is adequate to support efficient performance of text entry/editing tasks.

Example: For text editing where the page format of subsequent printed output is critical, the user's terminal should be able to display full pages of text in final output form, which might require a display capacity of 50-60 lines or more.

Example: For general text editing where a user might need to make large changes in text, i.e., sometimes moving paragraphs and sections, a display capacity of at least 20 lines should be provided.

Example: Where text editing will be limited to local changes, i.e., correcting typos and minor rewording, as few as seven lines of text might be displayed.

Comment: A single line of displayed text should not be used for text editing. During text editing, a user will need to see some displayed context in order to locate and change various text entries. Displaying only a small portion of text will make a user spend more time moving forward and back in a displayed document to see other parts, will increase load on the user's memory, and will cause users to make more errors.

Reference:
Elkerton Williges Pittman Roach 1982
Neal Darnell 1984

See also: 1.3/27

Table 8.4: One of the 944 guidelines by Smith and Mosier.

8.4 Guideline Scoring

Guideline scoring is a summative usability inspection method.

- The interface is scored according to its conformance against a weighted list of specific guidelines.
- A total score is produced, representing the degree to which an interface follows the guidelines.

An example checklist for the evaluation of a web site is shown in Figure 8.4.

Pros and Cons of Guideline Scoring

\+ cheap

\+ intuitive

\- must select and weight guidelines

\- guidelines or weightings often domain-dependent

Web Technologies - Checklist Homepage Design / Usability

Homepage (URL)		Date		Nr.	
Tester		Size	kB	Score	%

Nr	Topic	Recommended Design	Strength	Points	You
1	Download time	50 kB (<10 sec for your customer)	✶✶✶	3	
2	Window title	Start with Company Name	✶✶✶	3	
3	Title tag line	What about, Slogan	✶✶✶	3	
4	Readable URL	Hackable URL, URL is a UI part	✶✶	2	
5	Error page	Catch errors/dead links, to search	✶✶	2	
6	Meta tags	For search engines (trafficattack.de)	✶✶✶	3	
7	Alt Information	Images, accessibility, Lynx	✶✶	2	
8	Page width	770 pixel (620-1024)	✶✶	2	
9	• Liquid vs. frozen layout	Liquid is better	✶✶	2	
10	• Page length	<2 pages (1000-1600 px)	✶✶	2	
11	Frames	No, Don't use (search, bookmarks)	✶✶✶	3	
12	Logo placement	Upper left	✶✶✶	3	
13	• Logo size	Around 80x68 Pixel	✶✶	2	
14	**Search**	Yes, in a box, always	✶✶✶	3	
15	• Search placement	Upper part, right or left corner	✶✶	2	
16	• Search box color	White	✶✶✶	3	
17	• Search button	Call it "Search" or "Go"	✶✶	2	
18	• Width of search box	>=25 characters (30 best)	✶✶	2	
19	• Type of search	Simple search (Link to advanced)	✶✶	2	
20	**Navigation**	4 types: left, tabs, top, categories	✶✶	2	
21	• Footer navigation links	Max. 7 links, single line	✶	1	
22	• Sitemap link	Name "Site Map", Content	✶✶	2	
23	Routing page	No (www.logitech.com)	✶✶	2	
24	Splash page	No	✶✶✶	3	
25	Sign-In	"Account" or "Sign In"	✶	1	
26	**About** the company	Always include it	✶✶✶	3	
27	• About link	Call it "About <company>"	✶✶	2	
28	• Contact information	Call it "Contact us"	✶✶	2	
29	• Privacy policy	If you collect data	✶✶✶	3	
30	Home Button	Is there a home button visible	✶✶✶	3	
31	Job opening	Call it "Jobs" if you have it	✶✶	2	
32	**Help**	If it is a complex site (eBay, etc.)	✶	1	
33	• Help placement	Upper right	✶✶	2	
34	Auto-playing music	No	✶✶✶	3	
35	Animation	No	✶✶	2	
36	Graphics/illustration	5-15%	✶	1	
37	Advertising	<= 3 ads	✶✶	2	
38	**Body** text color	Black	✶✶	2	
39	• Body text size	12 points	✶	1	
40	• Body text size frozen	No (www.wired.com)	✶✶✶	3	
41	• Body text typeface	Sans-serif	✶	1	
42	• Background color	White	✶✶	2	
43	• Link color (unvisited)	Blue	✶✶	2	
44	• Link color (visited)	Purple	✶	1	
45	• Link color different	Yes (not light gray)	✶✶✶	3	
46	• Link underlining	Yes (except in navigation bar)	✶✶	2	
	Score of URL:			100	

✶ Default Recommendation
✶✶ Strong Recommendation
✶✶✶ Essential Recommendation

from Nielsen, Homepage Usability
adapted by Alexander Nischelwitzer V1.7
last update 10/2005 by NIS www.nischelwitzer.com

Figure 8.4: The Web Usability Checklist. The checklist covers 46 individual guidelines, each with a weighting of between one and three points, for 100 points total. [Adapted from [Nielsen and Tahir 2001] by Alexander Nischelwitzer, used with permission.]

8.5 Cognitive Walkthrough

Cognitive walkthrough is a formative usability inspection method.

Task-oriented walkthrough of interface, imagining novice users' thoughts and actions. Focuses explicitly on learnability.

- Design may be mock-up or working prototype.
- Analogous to structured walkthrough in software engineering.
- Based on *cognitive model* (CE+) of human exploratory learning.

  First described in [Lewis et al. 1990], discussion and practical guide in [Wharton et al. 1994].

Other References

- Clayton Lewis and John Rieman; *Task-Centered User Interface Design: A Practical Introduction*; 1993. http://hcibib.org/tcuid/
- John and Packer; *Learning and Using the Cognitive WalkthroughMethod: A Case Study Approach*; Proc. CHI'95, May 1995. http://www.acm.org/sigchi/chi95/proceedings/papers/bej1bdy.htm
- Rick Spencer; *The Streamlined Cognitive Walkthrough Method*; Proc. CHI 2000, [Spencer 2000]

Exploratory Learning

Rather than read manual or attend course, users often prefer to learn new system by "trial and error" → *exploratory learning* [Carroll and Rosson 1987]:

1. Start with rough idea of task to be accomplished.
2. Explore interface and select most appropriate action.
3. Monitor interface reactions.
4. Determine what action to take next.

The CE+ Model of Exploratory Learning

Based on psychological studies, the CE+ model describes exploratory learning behaviour in terms of 3 components:

- **Problem-Solving Component**

  User chooses among alternative actions based on similarity between the expected consequences of an action and the current goal.

  After executing selected action, user evaluates system response and decides whether progress is being made toward the goal. A mismatch results in an undo → "hill-climbing".

- **Learning Component**

  When above evaluation process leads to positive decision, the action taken is stored in long-term memory as a rule.

- **Execution Component**
  User first attempts to fire applicable rule matching current context. If none found, problem-solving component is invoked.

Cognitive Walkthrough Preparation

i) *Users*: Identify user population.

ii) *Task(s)*: Define suite of *representative* tasks.

iii) *Interface*: Describe or implement interface or prototype.

iv) *Correct Action Sequence*: Specify correct action sequence(s) for each task.

Cognitive Walkthrough Steps

For each action in solution path, construct *credible* "success" or "failure" story about why user would or would not select correct action.

Critique the story to make sure it is believable, according to four criteria:

a) *Will the user be trying to achieve the right effect?*
What is users' goal – will they *want* to select this action?

b) *Will the user know that the correct action is available?*
Is control (button, menu, switch, triple-click, etc.) for action *apparent* (visible)?

c) *Will the user know that the correct action will achieve the desired effect?*
Once users find control, will they *recognise* that it is the correct control to produce the desired effect?

d) *If the correct action is taken, will the user see that things are going ok?*
After correct action, will users realise progress has been made towards the goal (feedback)?

Note that CW always tracks the *correct* action sequence. Once the user deviates from the correct path their further progress is no longer considered.

Group Walkthrough

- Performed by mixed team of analysts (designers, engineers, usability specialist).
- Capture critical information on three group displays (flip charts, overheads):
  1. User knowledge (prior to and after action).
  2. Credible success or failure story.
  3. Side issues and design changes.
- Perhaps also videotape entire walkthrough.

Detailed Cognitive Walkthrough Example

Forwarding calls on a campus telephone system, from the perspective of a first time user; from [Wharton et al. 1994], pages 118–123.

i) *Users*: New faculty, staff, guests, and visitors. For this evaluation assume that the user is a new university professor.

ii) *Task*: Cancel current forwarding and forward calls instead to a colleague with the extension 1234.

iii) *Interface*: Standard-size, touch-tone phone on desk. Overlay template includes the following information:

FWD *2
CNCL #2
SEND ALL *3

iv) *Correct Action Sequence*: The seven correct actions for accomplishing this task are:

1. Pick up the receiver.
   Phone: *dial tone*
2. Press #2. {command to cancel forwarding}
   Phone: *bip bip bip*
3. Hang up the receiver.
4. Pick up the receiver.
   Phone: *dial tone*
5. Press *2. {command to forward calls}
   Phone: *dial tone*
6. Press 1234.
   Phone: *bip bip bip*
7. Hang up the receiver.

Example Walkthrough Steps

1. Pick up the receiver.
   Phone: *dial tone*

   Success story:

   Seems ok based on prior experience with phones.

2. Press #2.
   Phone: *bip bip bip*

   Failure story:

   a) *Will the user be trying to achieve the right effect?*

      How does the user even know that forwarding is in effect?

   b) *Will the user know that the correct action is available?*

      Probably yes, if forwarding is active, one must be able to cancel it. CNCL is visible on the template.

   c) *Will the user know that the correct action will achieve the desired effect?*

      Might not recognise CNCL as the control to cancel forwarding. Might think that just pressing '2' is sufficient, instead of '#2'. Might try to press the buttons simultaneously, rather than sequentially.

   d) *If the correct action is taken, will the user see that things are going ok?*

How do first-time users know they have succeeded? After some experience, they will recognise the *bips* as confirmation, but will they at first?

3. Hang up the receiver.
   Failure story:
   a) *Will the user be trying to achieve the right effect?*
      Big trouble. How do you know you have to hang up before reestablishing forwarding?

4. etc.

Pros and Cons of Cognitive Walkthrough

++ finds task-oriented problems

\+ helps define users' goals and assumptions

\+ usable early in development process

\- some training required

\- needs task definition methodology

\- applies only to ease of learning problems

\-- time-consuming

8.6 Action Analysis

Action analysis is a predictive usability inspection method:

- Quantitative analysis of actions to predict the time a skilled (expert) user requires to complete a task.
- The analysis is done on one or more interface designs, before the designs have been implemented.
- A task is broken down hierarchically into ever smaller subtasks.
- Time estimates are based on known (published) timings for typical interface actions.
- Focuses on the performance of skilled users (efficiency).

Flavours of Action Analysis

Two flavours (levels of detail) of action analysis:

a) Formal or "Keystroke-Level"

b) Informal or "Back-of-the-Envelope"

There are good examples in Lewis and Rieman [1993] and Raskin [2000].

8.6.1 Keystroke-Level Analysis

Described by [Card et al. 1983].

- Developed from GOMS (Goals, Operators, Methods, Selection) modeling.
- Extremely detailed, may often predict task duration to within 20%, but very tedious to carry out.
- Used to estimate performance of *high-use* systems (e.g. telephone operator workstations).

Procedure for Keystroke-Level Analysis

- Break down tasks hierarchically into subtasks until reach fraction of second level of detail.
- Use average values for action times (determined through extensive published research) to predict expected performance for particular task. See Table 8.5.
- The analysts do not measure action times themselves, but refer to published tables.

8.6.2 Back-of-the-Envelope Action Analysis

From [Lewis and Rieman 1993]. "Back-of-the-Envelope" uses the analogy of sketching out a rough analysis on the back side of an envelope while somewhere away from your desk ("Milchmädchenrechnung" in German).

- List actions required to complete a task (as before), but in much less detail – at level of explaining to a user:

  "Select Save from the File menu"
  "Edit the file name"
  "Confirm by pressing OK"

	Action	*Time*
Physical Movements	One keystroke	0.28
	Point with mouse	1.50
	Move hand to mouse or function key	0.30
Visual Perception	Respond to brief light	0.10
	Recognise 6-letter word	0.34
	Move eyes to new location on screen	0.23
Mental Actions	Retrieve one item from long-term memory	1.20
	Learn one step of a procedure	25.00
	Execute a mental step	0.075
	Choose among methods	1.20

Table 8.5: Average Times for typical keystroke-level actions, in seconds. From [J. Olson and G. M. Olson 1990], and cited by [Lewis and Rieman 1993].

- At this level of analysis, assume every action takes (say) 3 seconds (videotape a few users doing random tasks if you do not believe it takes this long!).
- Allows quick estimation of expected performance of interface for particular task.

Pros and Cons of Action Analysis

\+ predicts efficiency of interface before building it

\- some training required

\- - time-consuming