Usability Engineering

JAKOB NIELSEN

SunSoft
2550 Garcia Avenue
Mountain View, California

Morgan Kaufmann
An Imprint of Elsevier

Amsterdam Boston Heidelberg London New York Oxford
Paris San Diego San Francisco Singapore Sydney Tokyo

ACADEMIC PRESS
An Imprint of Elsevier
525 B Street, Suite 1900, San Diego, CA 92101-4495 USA
http://www.academicpress.com

Academic Press
Harcourt Place, 32 Jamestown Road, London, NW1 7BY UK

Morgan Kaufmann
340 Pine Street, Sixth Floor, San Francisco, CA 94104-3205
http://www.mkp.com

Library of Congress Catalog Number: 93000488
International Standard Book Number: 0-12-518406-9

Printed in the United States of America
04 05 06 07 08 / 12

Metaphors also present potential problems with respect to internationalization, since not all metaphors are meaningful to all cultures. For example, a Danish interface designer might choose to use the pause signal as a metaphor for delayed system response, drawing upon the common knowledge that radio stations play a special endless tune of the same 13 notes repeated over and over when one show finishes before the scheduled starting time of the next. However, the concept of a pause signal would be quite foreign to users in many other countries, such as the United States, where radio stations fill every available moment with commercials and would never put on a special signal just to fill up time.

5.3 *Minimize User Memory Load*

Computers are very good at remembering things very precisely, so they should take over the burden of memory from the user as much as possible. In general, people have a much easier time at recognizing something that is shown to them than they have at having to recall the same information from memory without help. This phenomenon is well known to anybody who has learned a foreign language: Your passive vocabulary is always much larger than your active vocabulary. And of course, computers really speak a foreign language as far as the users are concerned.

The computer should therefore display dialogue elements to the users and allow them to choose from items generated by the computer or to edit them. Menus are a typical technology to achieve this goal. It is also much easier for the user to modify information displayed by the computer than to have to generate all of the desired result from scratch. For example, when users want to rename an information object, it is very likely that they will want the new name to be similar to the old one, so the text edit field in which the user is supposed to enter the new name should be prepopulated with the old name, allowing users to make modifications instead of typing everything.

Interfaces based on recognition rely to a great extent on the visibility of the objects of interest to the user. Unfortunately, displaying too many objects and attributes will result in a relative loss of salience for the ones of interest to the user, so care should be taken to match object visibility as much as possible with the user's needs [Gilmore 1991]. As usual we find that "less is more."

Whenever users are asked to provide input, the system should describe the required format and, if possible, provide an example of legal and sensible input, such as a default value. For example, a system asking the user to enter a date could do it as follows:

- `Enter date (DD-Mmm-YY, e.g., 2-Aug-93):`

An even better dialogue design would provide the example in the input field itself as a default value (possibly using today's date or some other reasonable date), thus allowing the user to edit the date rather than having to enter all of it.

There is no need for the user to have to remember or guess at the range of legal input and the unit of measurement that will be used to interpret it. Instead, the system can supply that information as part of the dialogue, such as, for example:

- `Left margin:___10 points [0-128]`

A famous example indicating the need to display measurement units to help the user's memory was the positioning of a space-based mirror by the space shuttle *Discovery* [Neumann 1991]. The mirror was supposed to be aimed at a mountain top in order to reflect a laser beam, and the user had ordered the computer to point the mirror toward a point with an elevation of "10,023 above sea level." The user apparently entered the elevation as if it were measured in feet, whereas, in fact, the system used miles as its measurement unit, causing the mirror to be aimed *away* from the Earth, toward a point 10,000 miles out in space.[5]

To minimize the users' memory load, the system should be based on a small number of pervasive rules that apply throughout the

5. With respect to measurement units, other usability principles often lead to a need allow users to select between several alternative units, such as inches, feet, miles, centimeter, meter, and kilometer, depending on their needs.

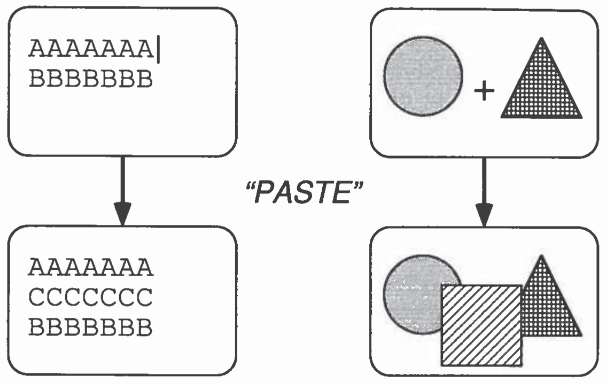

Figure 13 *A generic command: "Paste" can be used to insert a line of* C's *(text) as well as a striped square (graphics) at the insertion point.*

user interface. If a very large number of rules is needed to determine the behavior of the system, then the user will have to learn and remember all those rules, making them a burden. On the other hand, if the system is not governed by any rules at all, then the user will have to learn every single dialogue element on its own, and it is impossible to predict the behavior of a dialogue element without already knowing (and remembering) how it works.

The use of generic commands [Rosenberg and Moran 1984] is one way to let a few rules govern a complex system. As shown in Figure 13, generic commands make similar things happen in different circumstances, making it sufficient for the user to learn a few commands in order to work with many different types of data. One of the main advantages of generic commands is that they support transfer of learning from one application to the next, since users do not need to relearn those commands they already know [Ziegler *et al.* 1986].

Generic commands need not perform exactly the same function in all circumstances, as long as the user can think of the command as a single unified concept, such as "insert the object from the clipboard" in the case of a `paste` command. As shown by Figure 13, this generic command may actually insert the clipboard *and* move some old objects out of the way, when it operates on text, but perform the insert operation without moving anything when it operates on graphics. The designer of a generic command will need to determine what "naturally" feels like the same command to users, even if some details will differ due to the requirements of the different parts of the system.

5.4 Consistency

Consistency is one of the most basic usability principles. If users know that the same command or the same action will always have the same effect, they will feel more confident in using the system, and they will be encouraged to try out exploratory learning strategies because they will already have part of the knowledge needed to operate new parts of the system [Lewis *et al.* 1989].

The same information should be presented in the same location on all screens and dialog boxes and it should be formatted in the same way to facilitate recognition. For example, my heating bill contains a comparison between my current heating use and my use in the same month in the previous year, listed as a table with the current year in the left column and the previous year in the right. To facilitate my interpretation of these numbers, a footnote on the bill furthermore contains information about the average temperature in each of the two years. Unfortunately, the footnote mentions the previous year before (that is, to the left of) the current year, thus inverting the relation compared to that used in the table. Consistency considerations would have implied a design of this printout with the same spatial relation between the two periods for both kinds of information. An order where the previous year was mentioned before the current year might be preferred as being consistent with the way timelines work, but unfortunately one can also argue that the reverse order achieves a better match with the user's task of assessing current heat usage. As is often the case in user interface design, one would have to decide which of these two considerations was most important; once this decision had been made, one should follow it consistently and not mix the two layout rules.

Many aspects of consistency become easier to achieve to the extent that one is following a user interface standard in the design, since the standard will then have specified many details of the dialogue, such as, for example, how to indicate a pop-up menu or which typeface to use in a list of font sizes. See Chapter 8 for a discussion of user interface standards and ways to increase compliance and thereby consistency. Unfortunately, standards compliance is not sufficient to ensure consistency, since the standards leave a fair amount of leeway for the designers. See the discussion of user interface coordination in Section 4.6 (page 90) for ways to promote consistency during interface design.

Consistency is not just a question of screen design, but includes considerations of the task and functionality structure of the system [Kellogg 1987, 1989]. For example, Eberts and MacMillan [1987] found that subjects were more confused when they switched between using a command-line mainframe and a command-line personal computer than when they switched between the command-line personal computer and a graphical personal computer. From a screen design perspective, the two command-line interfaces were very similar, but the underlying operating systems were in fact very different. And the two personal computer interfaces were built on top of systems with the same basic philosophy and features.

A study of a popular spreadsheet program found 10 consistency problems causing common errors for novice users [Doyle 1990]. Seven of these problems were due to inconsistencies between the spreadsheet and the users' task expectations, three were due to inconsistencies between the spreadsheet and other user interfaces, and only two problems were due to inconsistencies within the spreadsheet itself. The spreadsheet's menu navigation method was classified as being inconsistent in all three ways and was therefore counted in all three categories. Of course, other systems may have

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.