Usability Engineering

Jakob Nielsen

Usability Engineering

This page intentionally left blank

Usability Engineering

JAKOB NIELSEN

SunSoft
2550 Garcia Avenue
Mountain View, California

AP PROFESSIONAL

Boston San Diego New York
London Sydney Tokyo Toronto

This book is printed on acid-free paper. ♾

AP PROFESSIONAL
955 Massachusetts Avenue, Cambridge, MA 02139

An Imprint of ACADEMIC PRESS, INC.
A Division of HARCOURT BRACE & COMPANY

United Kingdom Edition published by
ACADEMIC PRESS LIMITED
24–28 Oval Road, London NW1 7DX

ISBN 0-12-518406-9

Printed in the United States of America
94 95 96 97 BB 9 8 7 6 5 4 3 2 1

Table of Contents

Preface

Software developed in recent years has been devoting an average of 48% of the code to the user interface [Myers and Rosson 1992]. It would thus seem justified to allocate a reasonable proportion of the effort in software development projects to ensuring the usability of these user interfaces. This book tells you what to do if you decide to improve usability.

The main goal of the book is to provide concrete advice and methods that can be systematically employed to ensure a high degree of usability in the final user interface. To arrive at the perfect user interface, one also needs genius, a stroke of inspiration, and plain old luck. Even the most gifted designers, however, would be pressing their luck *too* far if they were to ignore systematic usability engineering methods.

Audience

The book has a very wide intended audience. First of all, it is naturally intended for the people who actually design and develop computer systems and user interfaces since these individuals have the ultimate power to improve usability. The book is crammed with practical advice for including usability considerations in the

software engineering process, and developers and project managers should read through the entire book. The book is also intended for people who design documentation, help systems, and training courses, since these are elements of the "total user interface" just as much as the screen designs. This book is not intended to teach technical writing as such, but it can help writers produce support materials that users will find easier to use.

Furthermore, large parts of the book should be helpful to the users themselves and to computer support managers who need to determine which computer systems and software to recommend to their users. Even though it is fairly rare for customer organizations to perform their own usability testing, there is no reason why a large organization should not use some of the techniques in Chapter 6, Usability Testing, to compare software packages and whole systems before deciding on what to buy. Smaller organizations and individual users can use the definitions in Chapter 2, What Is Usability?, and the usability principles in Chapter 5, Usability Heuristics, as a checklist to consider whether an interface seems usable before buying it. Multinational corporations and other international organizations should benefit from Chapter 9, International User Interfaces, when planning the requirements for their information systems. Finally, user organizations that contract out for software development can use Chapter 4, The Usability Engineering Lifecycle, and Chapter 8, Interface Standards, to help set requirements that will ensure the usability of the product they will eventually receive from their vendor.

The executive summary in Chapter 1 is intended to help those readers who may not have time to read the entire book. It is especially intended for managers who are considering whether their companies are devoting sufficient effort to usability and what concrete steps they can request to ensure improved usability of their systems. It should be read by all readers, however, as it is not just a summary; it also addresses several topics that are not covered in the rest of the book, such as the cost/benefit trade-offs of taking human factors seriously.

Most of the examples in the book come from user interfaces to computer systems. The methods can be used for the development of interfaces to any kind of interactive system, including most consumer electronics products, and they are even useful for the development of certain information-intensive types of noninteractive products such as computer printouts, time tables, and driving directions. For example, van Nes and van Itegem [1990] describe the use of a logging method (see also page 216 ff.) in a usability study of an advanced car radio with 37 functions. For half a year, four drivers had every use of practically all of these functions from their new car radio automatically recorded. The results showed that some of the novel features went unused and that others were used differently than the designers had intended. A follow-up user interview revealed that the users still had not understood some features after half a year of use. One user complained that the auto-search tuning mechanism skipped some radio stations, whereas in fact it operated at three successive sensitivity levels and would pick up the missing stations at the second or third scan.

Any object, product, system, or service that will be used by humans has the potential for usability problems and should be subjected to some form of usability engineering. Human–computer interaction serves as the main focus of this book because it is the author's special area of expertise and because the potential for usability problems seems to be especially severe in computers, due to their ability to implement complex features and intricate interactions. For other kinds of interfaces, slight modifications may have to be made, but the main principles in this book should still hold. For example, questionnaires and user testing have been applied to improve the usability of railroad cars [McCrobie 1989].

Teaching Usability Engineering

Several universities have developed both traditional courses and continuing education efforts in various aspects of human–computer interaction [Baecker 1989; Carey 1989; John *et al.* 1992; Mantei 1989; Mantei *et al.* 1991; Preece and Keller 1990, 1991; Strong

1989; van der Veer and White 1990]. The Association for Computing Machinery's Special Interest Group on Computer–Human Interaction (ACM SIGCHI) has even developed a recommended curriculum for the teaching of human–computer interaction [ACM SIGCHI 1992]. Typical topics covered in such courses include theoretical approaches to human–computer interaction, the implementation of user interfaces, and the actual design of user interfaces. The latter is often taught through exercises [Nielsen *et al.* 1992; Winograd 1990]. In a survey of skills needed by usability practitioners [Dayton *et al.* 1993], the four skills rated as having an importance of more than 9.0 on a 1–10 scale were oral presentation, dialogue design, task analysis, and usability evaluation. The presence of presentation skills at the top of the list indicates that no usability project is conducted in isolation: to be successful, it needs to impact a larger development team.

Usability engineering as such also seems to be taught more these days, either as part of a general HCI (human–computer interaction) course or as a course in its own right [Nielsen and Molich 1989; Perlman 1988, 1990]. This is especially true of courses taught by corporate training departments or offered as continuing education for software engineers.

My main advice for the teaching of usability engineering would be to base the course firmly in the laboratory. Even though there is a substantial amount of theory and principles that can be taught in the auditorium, the most important aspects of design and evaluation require a hands-on approach. Certainly, a required part of any usability engineering course should be to have the students conduct a user test with a small number of real users. Not only is this a good way to teach proper evaluation methodology, but more important, it is the only way to achieve the required revolutionary change in student attitudes. Most professional programmers and computer science students gain profound insights the first time they actually sit down with test users and observe them struggle with supposedly "easy" software. This is especially true if the software was designed by the programmers or students themselves!

Appendix A lists several practical exercises touching upon important aspects of usability engineering. The way these exercises are described is mostly intended for self-study readers, but they can easily be expanded into more elaborate assignments for class use.

Acknowledgments

Many colleagues graciously answered questions about specific issues, provided comments on the treatment of their special interests, or even read through the entire manuscript. For this help, I would like to thank Jeff Abbott (Tivoli Systems), David Ackley (Bellcore), Alfred V. Aho (Bellcore), Gregory H. Anderson (Anderson Financial Systems), Mary M. Anthony (Tivoli Systems), Sonia D. Bot (Bell-Northern Research, Canada), Andreas Buja (Bellcore), Mike Coble (TRIPOS Associates), Bill Curtis (Software Engineering Institute), Tom Dayton (Bellcore), Susan T. Dumais (Bellcore), Lawson J. Dumbeck (Western Washington University), Tom Emerson (Symantec Corporation), Peter W. Foltz (University of Colorado), Ellen Francik (Pacific Bell), George Furnas (Bellcore), Marc Fusco (Bellcore), Thom Gillespie (University of California, Berkeley), Michael Good (Digital Equipment Corporation), Peter Henriksen (Microsoft), Hiroshi Ishii (NTT Human Interface Laboratories, Japan), Robert E. Jackson (Space Telescope Science Institute), Janice James (American Airlines), Jeff Johnson (Hewlett-Packard Laboratories), Peter R. Jones (Symantec Corporation), Anker Helms Jørgensen (Copenhagen University, Denmark), Hannah Kain (Citibag), Alistair Kilgour (Heriot-Watt University, U.K.), Thomas K. Landauer (Bellcore), Jonathan Levy (Bellcore), Robert L. Mack (IBM T. J. Watson Research Center), Miles Macleod (Hatfield Polytechnic, U.K.), Deborah J. Mayhew (Deborah J. Mayhew & Associates), Rolf Molich (Baltica Insurance, Denmark), Michael Muller (U S WEST), Robert M. Mulligan (AT&T Bell Laboratories), Gerhard Nielsen (Denmark's Radio), Randy Pausch (University of Virginia), Gary Perlman (Ohio State University), Steven Poltrock (Boeing Computer Services), Dan Rosenberg (Borland International), Kjeld Schmidt (Risø National Laboratory, Denmark), David Schnepper (Borland International), Tom Semple

(Symantec Corporation), Brian Shackel (Loughborough University of Technology, U.K.), Ben Shneiderman (University of Maryland), Scott Stornetta (Bellcore), Kurt Sussman (Symantec Corporation), Desirée Sy (Information Design Solutions), Michael Tauber (University of Paderborn, Germany), Bruce Tognazzini (SunSoft), Hirotada Ueda (Hitachi Central Research Laboratory and FRIEND21 Research Center, Japan), Gerrit van der Veer (Free University of Amsterdam, The Netherlands), Floris L. van Nes (Institute for Perception Research/Philips Research Laboratories, The Netherlands), Robert Virzi (GTE Laboratories), Christopher A. White (GTech Corporation), Richard Wolf (Lotus Development Corporation), and Peter Wright (University of York, U.K.).

The resulting book is solely the responsibility of the author, and the people mentioned above should not be held responsible for the way I have interpreted their comments and advice. Most of this book was written while I was on the applied research staff of Bellcore but it should not be taken as necessarily representing any official views or policies of Bellcore.

This printing of the book has been updated with a number of literature references and comments on developments since the book was first published. Among other things, I added the new synergy review method, which I invented just a few weeks after sending in the final copy for the first printing. Mostly, the book is unchanged, though, since the basics of usability engineering remain fairly constant and do not vary from year to year.

Jakob Nielsen
Mountain View, California
April 1994

Chapter 1 *Executive Summary*

Have you ever seen one of the people who will be users of your current project?[1] Have you talked to such a user? Have you visited the users' work environment and observed what their tasks are, how they approach these tasks, and what pragmatic circumstances they have to cope with? Such simple user-centered activities form the basis of usability engineering. More advanced methods exist and are covered later in this book, but just a simple field trip to observe users in their own environment working on real-world tasks can often provide a wealth of usability insights.

In one example, three one-day visits to branch offices of a medium-sized insurance company produced a list of 130 usability problems [Nielsen 1990b]. The system design was sound, and most of the problems were simple enough to fix once they were known (but, of course, they would not have been known if it had not been for the field study). Many of the 130 items were serious problems only for novice users. However, even very experienced users were estimated to waste at least 10 minutes every day because of usability

1. Note that you have to talk to the individuals who will be using the system. Talking to the users' manager or vice president for data processing does not count since these people are likely to have a completely different understanding of the job than the actual users.

problems, costing the company large amounts of money in both labor costs and lost sales opportunities.

The staff was often interrupted by telephone calls or walk-in clients. Unfortunately, several subsystems were not designed for interruptions—users lost all of their work if a transaction was not carried to completion. At one small branch, an agent stated that she never used the damage-claims subsystem during periods where she was the only person in the office and had to answer all calls. In some cases, agents were observed using other agents' terminals (and "borrowing" their passwords) to deal with interruptions rather than quit one of the unforgiving subsystems in the middle of a transaction.

In another case, the system allowed only one line for error messages, so it had to give an obscure, truncated version of a long message. The full message was available by pressing the help key, PF1, an action the developers in the central data-processing office felt was very natural. But users in the branch office had not made the conceptual leap that told them the help key was doubling as an extended-error-message key. Instead, they wasted a lot of time trying to understand the truncated message. A better design would have used the one line on the screen for a brief indication of the error, followed by "`PF1 for more information`" or a similar instruction.

1.1 Cost Savings

There are several well-documented examples of cost savings from the use of usability engineering methods.[2] For example:

- When a certain rotary dial telephone was first tested, users were found to dial fairly slowly. A human factors expert spent one

2. There are more examples of cost savings that are less well documented. As noted by Chapanis [1991], most case studies fail to meet the rigorous methodological requirements that are necessary to be absolutely sure what cost savings can be attributed to user interface improvements since there are often several other changes made simultaneously (e.g., [Thompson *et al.* 1986]).

hour to come up with a simple graphical interface element which speeded up users' dialing behavior by about 0.15 seconds per digit, for a total annual saving of about $1,000,000 in reduced demands on the central switches [Karlin and Klemmer 1989].

- An Australian insurance company had annual savings of A$536,023 from redesigning its application forms to make customer errors less likely [Fisher and Sless 1990]. The cost of the usability project was less than A$100,000. The old forms were so difficult to fill in that they contained an average of 7.8 errors per form, making it necessary for company staff to spend more than one hour per form repairing the errors.
- A major computer company saved $41,700 the first day the system was in use by making sign-on attempts faster for a security application. This increased usability was achieved through iterative design at a cost of only $20,700 [Karat 1990].
- The 25 "human factors success stories" discussed by Harris [1984] include the improvement of the Boeing 757 flight deck interface to allow operation by two instead of three pilots, the 35% increase in alignment speed in a production line for integrated circuits, the reduction from 3,000 words to 150 words of instructions needed to operate a paging device, and even an improvement in a drunk-driver detection system that increased the arrest rate per police officer patrol-hour by 12%.

Unfortunately, the cost savings from increased usability are not always directly visible to the development organization since they may not show up until after the release of the product. As an extreme example, Fisher and Sless [1990] report that the Australian government can process a tax return for A$2.25 on the average. At the same time, the average Australian resident spends 11 hours filling in the form, and 62% of Australians have to use agents to help do the job. If the complexity of the tax forms were reduced, these "customers" might therefore realize huge savings in time and advisor fees, but the government might only save a few cents in processing costs. In the same way, making a spreadsheet easier to learn might only save the vendor a small amount in reduced hotline staffing levels, even though each customer might save several hours of unnecessary work.

Distributed benefits of a few hours per user are hard to measure and do not immediately add up to hard cash [Sassone 1987]. For example, redesigning the interface to an oscilloscope increased user productivity by 77% during the time they were using the scope [Bailey *et al.* 1988], but the productivity impact on the total workday of an engineer was much less dramatic and therefore had less impact. The customers *do* save with better interfaces, though, and these savings presumably translate into a better reputation for the product and therefore eventually increase sales. Unfortunately, the effect of having increased usability lead to increased sales has mostly been documented only anecdotally.[3] In several cases, the relative usability of competing products is well known in the industry, and computer salespersons often recommend certain software packages on the basis of their usability.

Because much of the financial payoff from usability methods shows up after the release of the product, some usability specialists [Grudin *et al.* 1987] have advocated shifting parts of the responsibility for usability engineering toward middle and upper management levels instead of the development managers. Even the development manager may see some immediate benefits from usability engineering, however, in the frequent case when early usability studies reveal that there is no need for certain contemplated features. If users' needs are not known, considerable development efforts may be wasted on such features in the mistaken belief that some users may want them. Users rarely complain that a system can do too *much* (they just don't use the superfluous features), so such over-design normally does not become sufficiently visible to make the potential development savings explicitly known. They are there nevertheless.

3. In one of the few documented cases, a usability study of the first version of a fourth-generation database system revealed 75 usability problems. Twenty of the most serious problems were fixed in the second release, which generated 80% higher product revenues than the first release [Wixon and Jones 1994]. This revenue increase was 66% higher than sales projections and so is probably due to the improvements in usability since field test customers were reported to point to the user interface as the most significant improvement in the product.