ISSN 0105-8517

Three Approaches to Teaching Software Ergonomics

Jakob Nielsen

DAIMI PB - 184
December 1984

Three Approaches to Teaching Software Ergonomics

Jakob Nielsen
Aarhus University
Aarhus, Denmark

Abstract
Instruction strategies and experiences are reported from

1. a university course for computer science graduate students
2. a continuing education course for programming school teachers
3. a software ergonomics awareness course for hardware ergonomists

It is concluded that software ergonomics courses should include specific examples of user-friendly systems and of human factors experiments as well as generally applicable theories and concepts. These theories and concepts should be used as a framework for the examples.

Keywords
Software ergonomics, human factors, training, continuing education, awareness course, windowing versus scrolling, color displays, design methodology.

Contents

1 Preface

1.1 Teaching an Undefined Subject

The problem with teaching software ergonomics is that the subject itself is poorly defined. Is it computer science or is it psychology? This is the least of our problems, since computer scientists in any case ought to have a solid foundation in the principles of user-friendly interfaces. So some software ergonomics should be a required part of any undergraduate computer science curriculum.

The worst problem is the lack of any really good basic theory that we can teach our students. Therefore is it necessary to rely on more narrow theories and on more or less fuzzy conceptual ideas. One important thing to do in a course at the present time is to include lots of examples of both research experiments and of working modern user interface designs.

Furthermore I have not yet found a textbook of the same quality as those used in other computer science subjects. The reason may be that given by Don Norman at the CHI'83 panel session on education: He claimed that no good textbook exists and added that he had not even written one himself (though he had started out to do so several times) because the subject area was not yet sufficiently established.

The solution to the textbook problem may be to use either one of the existing textbooks or one of the existing popular guides for designers of personal computer software systems. And then supplement the chosen book with a number of selected papers of your own choise.

One advantage of the primitive state of the art in software ergonomics is that the much touted close connection between teaching and research (that is supposed to result from having researchers as educators) is heavily utilized. Perhaps this even makes classes more interesting to the students if they can bear not getting exact answers to all the questions.

The present report tells how I have actually taught three quite different types of courses in software ergonomics. Perhaps some of the experiences reported will help other educators in the design of their own courses.

1.2 How Will I Teach My Next Course?

Certainly not the way I taught any of the courses reported in the following chapters!

For a university-level course I will try to mix together many of the elements from my earlier courses. The course will include both theoretical concepts such as interaction models and design ideas such as WYSIWYG. Then the students will analyse some programs available on local computer systems. These programs ought to be commercially widely used systems such as spreadsheets since computer science students should get an idea of how computers are used in the real world.

Finally the course will include some selected reserach results chosen both for the actual result of the experiment and for the reserach methodology used. And the course will then end by having the students carry out some small experiment themselves, probably focusing on qualitative methods such as thinking aloud.

2 An Experimental Course in Software Ergonomics

2.1 Abstract

An introductory graduate course was given in Experimental Software Ergonomics using a learning-by-doing style. Students did team projects on windowing versus scrolling using continuous data movement and on the utility of color in textual information.

2.2 Introduction

This article describes a course in software ergonomics given at the Computer Science Department, Aarhus University, Denmark in the spring semester 1984. The title of the article has a double meaning: 1) The course stressed experimental methodology and was entitled *"Experimental Software Ergonomics"*. 2) It was the first course of its kind given in Denmark and was as such an experiment in itself.

The course was an elective at the introductory graduate level and it was followed by 7 students of which 6 were computer science majors and one was a psychology major.

Since the field of user-friendly programming is as yet not blessed with a comprehensive theory, I from the start chose to run the class in a "learning-by-doing" style instead of relying on the more traditional lecture format. I would also say that - probably because of the lack of theory - no really good textbook exists in software ergonomics, which meant that the course did not rely on any one textbook.

Instead of a textbook I used some lecture notes of my own (in Danish) and copies of some relevant papers, most notably [Ramsey and Grimes 1983] to give an overview of the area, [Shneiderman 1983] to give a more specific example of modern thinking in the field, and [Moher and Schneider 1982] to give an introduction to the experimental process itself. I also distributed summaries of several additional papers.

Since I had chosen to base the course on student experiments, I started the course by handing out copies of [Shneiderman 1982], describing a similar course. The purpose was to give the students a feel for what to expect and also some basis for criticizing my teaching if it did not fit their expectations.

2.3 Course Schedule

The course ran over 15 weeks and had the following time schedule:

Week 1 Introduction to the course. What is software ergonomics anyway? The [Shneiderman 1982] course.

Week 2 Guidelines for better user-friendliness. Lists of good advise from the literature.

Week 3 Discussion of the possible student projects (see below).

Week 4 Lecture on models of the interaction.

Week 5 The students are by now divided into project groups and each group tells about the problem it will be investigating, including their hypothesis.

Week 6 Lecture on selected research results from the scientific software ergonomics literature.

Week 7 The project groups present their proposed experiments.

Week 8 Preparation of experiments.

Week 9 Actual experiments.

Week 10 Actual experiments.

Week 11 No class (Easter vacation).

Week 12 Discussion of problems with statistical treatment of the data.

Week 13 Statistical data processing.

Week 14 The project groups present their results.

Week 15 The final reports are handed in.

This was the intended schedule, but it turned out that both the preparations for the experiments (involving some programming work on new and unfamiliar computer equipment) and the data analysis took longer time than expected. So the students had to work for some weeks after the end of normal classes to write up their final reports and the date when reports were due was moved back one month.

2.4 Discussions of Student Work

The students handed out preliminary versions of the relevant chapters of their report at the class meetings in week 5 and week 7. This saved them some time when writing the final report but the primary purpose was to ensure that their proposed experimental hypothesis and methodology would make sense. It also supported one of the most important pedagogical aspects of the course, viz. that the students were able to learn from more than their own one experiment. The students were encouraged to comment on proposals from other students. Although a large share of the comments were made by myself, all the students were nevertheless present and participated in the discussions of all the projects. The students had the opportunity to change their preliminary writings, and all grading was done solely on the basis of the final reports.

The discussion of the experimental results was also carried out in class where all students were invited to comment on possible interpretations of each other's data.

Since only 7 students followed the course, it turned out that there was basis only for the formation of two project groups; one of 4 students and one of 3. This is the minimum acceptable but it still made possible the goal of having the students discussing some other project beside their own. On the other hand this course format would be impossible with more than, say 4 groups.

The students were allowed to choose their projects from a list of possible projects suggested by me. They were also allowed to suggest problems not on the list, but none did so. The problems investigated were *Windowing versus Scrolling* by the 4 student group and *the Utility of Color in Textual Information* by the 3 student group.

2.5 Windowing versus Scrolling

When you have more information than can be shown on the display at any one time, there are two different models of how to change the information displayed. Either you imagine moving the *window* to another part of the data or you imagine moving the *data* itself. These two models, windowing and scrolling, had earlier been investigated by among others [Bury et al. 1982] who found that windowing was superior to scrolling on an ordinary alphanumerical terminal. In our project a graphical workstation was used which enabled a continuous movement of the information on the display instead of the jumpy movement investigated by Bury et al. Also the experimental setup in this project used overlapping windows instead of the full screen. The hypothesis was that these changes would tend to make scrolling preferable because of what might be called "display inertia" (i.e. one would tend to think of the data moving because of the continuous movement and to think of the rest of the screen with the other windows as being stationary).

The test subjects used were 40 high school students from 3 classes that were given a tour of the Computer Science Department and were asked to participate in the experiment. Half of the subjects were randomly assigned to the windowing condition and half to the scrolling condition. They were given an explanation of the system using either a windowing model or a scrolling model and they were then asked each to solve 64 problems. Each problem required the subjects to use keys labelled with arrows to display a specific part of the information that was not initially visible.

The results shoved no significant difference in the average total time used to solve the problems, even though windowing was somewhat faster (519 seconds for windowing and 524 seconds for scrolling, $p=0.35$). The total number of pixels moved was also not significantly different in the two conditions, but here scrolling was slightly better (14899 pixels for windowing and 14146 pixels for scrolling, $p=0.06$). The only significant difference was in the reaction time from the display of the problem to the time when the subject hits the first key. The sum of reaction times for the 64 problems was lower and thus better for windowing (137 seconds for windowing and 164 seconds for scrolling, $p=0.02$).

In summary: This experiment showed no marked difference between windowing and scrolling and thus did not support the hypothesis that scrolling would be better in the condition tested. On the other hand [Bury et al. 1982] found large differences in favor of windowing in all measures taken in their experiment. So the change in equipment has probably meant some shift in favor of scrolling even though the shift did not go all the way to make scrolling best.

2.6 Utility of Color in Textual Information

The other group tested the utility of using color in textual information. Color is normally assumed to be an advantage in graphics, but here it was tested if the use of color as an additional information channel in a textually based application would be an advantage. The application tested was form filling in a simplified data base context. A form was shown on the display with all the information in the database about some person. And three different formats were used to display this form. One was simply black and white. The other highlighted the information that was entered by the user in the current session. And the third showed the name of the data fields in black text and the contents in red text if it had been entered by the user in the current session and in blue text otherwise.

As test subjects 24 first year computer science students were used. Each used the system to update 9 forms, 3 in each of the 3 formats. They were randomly assigned to one of four conditions determining the order of the formats.

Several time measurements were taken, including total time and reaction time from the presentation of a new problem to the first keystroke. None showed any notable difference between the three different formats! Maybe this is partly due to the fact that the instructions given to the test subjects did not sufficiently stress the importance of making the updates 100% correct. Therefore they may not have checked their entries after having keyed them in, and they would thus not gain the expected advantage of the highlighting or color-coding of the changes made.

On the other hand interview data showed a significant subjective preference for the color-coded system. 20 of 24 subjects said that it had been helpful being able to distinguish the text written by themselves. They said so, even though the objective measurements were not able to find any performance improvements when they were using formats with this feature. And when asked which of the three formats they would prefer, 14 wanted the color system, 5 wanted the highlighting system and 5 wanted the simple black and white system.

So in summary, this experiment showed that people like color better than black and white, even in a purely text based application. But perhaps due to experimental deficiencies it was not possible to show any objective performance improvements.

2.7 Conclusions

When evaluating the course, the students said that they would have liked to learn more about existing software ergonomics results and concepts before starting their own experiments. On the other hand they were all very excited about the benefits from doing an actual experiment.

So what might be done another time is to choose some experimental projects that have less of a research flavor and that perhaps do not rely so much on quantitative and statistical methods requiring lots of experimental subjects. Instead one could experiment with evaluations of existing software products or with simple prototypes of the students' own design, and then make some more qualitative experiments. The time saved on the experiments could then be used for more traditional lectures and perhaps for novel learning methods such as first reading about user friendliness guidelines and concepts and then dissecting some popular system to see how (and whether) the ideas have been implemented.

Acknowledgements

First of all, my students should be acknowledged for enthusiastically taking part in this experimental course. The color experiment was carried out on an IBM PC donated by IBM Denmark. Support was also received from the Danish Natural Science Research Council (grant 11-4296).

2.8 References

- Bury, Kevin F.; Boyle, James M.; Evey, R. James; and Neal, Alan S. "Windowing versus scrolling on a visual display terminal". *Human Factors* 24, 4 (August 1982), pp. 385-394.
- Moher, Thomas; and Schneider, G. Michael: "Methodology and experimental research in software engineering". *Int. J. Man-Machine Studies* **16**, 1 (Jan. 1982), pp. 65-87.
- Ramsey, H. Rudy; and Grimes, Jack D. "Human factors in interactive computer dialog". **In**: Williams, M.E. (ed.): *Annual Review of Information Science and Technology* **18** (1983), pp. 29-59.
- Shneiderman, Ben. "Teaching software psychology experiments through team projects". *Proc. Human Factors in Computer Systems* (Gaithersburg Md, March 15-17, 1982), pp. 299-301.
- Shneiderman, Ben. "Direct manipulation: A step beyond programming languages". *IEEE Computer* **16**, 8 (August 1983), pp. 57-69.

3 A Continuing Education Course in Software Ergonomics for Programming School Teachers

3.1 Abstract

A one-week continuing education course was given on software ergonomics for teachers in the Danish one to two year programmer training schools.

3.2 Introduction

In addition to university-based computer science education, Denmark also has a system of Computing Schools offering one or two year programmer training studies, mainly for high school graduates. The curriculum consists of such traditional subjects as business economics and programming in Cobol and assembler, but even so there has been some interest in the topic of user friendliness. Therefore a continuing education course in software ergonomics was given for some Computing School teachers.

3.3 Course Schedule

The course was a one-week course running over a period of five days, having the following contents:

1. Monday.
   (a) Participants write down their suggestions of the key design principles for user friendly software *(see Section 3.4)*.
   (b) Discussion of participants' design suggestions and of an actual design of a banking information system using a telephone as I/O medium.
   (c) Slides of examples of modern interfaces, e.g. 'Movie Manuals' from MIT Architecture Machine Group.
   (d) A virtual protocol model for computer-human interaction [Nielsen 1984a].
2. Tuesday.
   (a) Discussion of the scoring of participants' suggestions of key design principles *(see Section 3.4)*.

   (b) Software ergonomics as a science. How other disciplines may help us (e.g. Psychology and Advertising).
   (c) The paradigmes of interaction [Nielsen 1984b] and the five computer generations.
   (d) Users' learning difficulties [Nielsen 1984b].
   (e) Practical exercise: Design and implement a user-friendly program. Two problems were given:
      i. A calendar and alarm clock program.
      ii. A museum guide for tourists.

      The participants were divided into groups and each group did only one of these exercises. Some groups actually implemented a running computer program, while other participants made a mock-up using overhead transparencies simulating screen displays.

3. Wednesday.
   (a) Practical exercise: Thinking aloud evaluation of the systems designed Tuesday. The participants themselves were used as test subjects (for the program they had not themselves had as an exercise Tuesday).
   (b) Measurements of user friendliness.
   (c) Methodological difficulties in measurement experiments.
   (d) The spectrum of models in software ergonomics (cognitive models, intended and actual conceptual models, metaphors, user description models, etc.) [Nielsen 1984c].
   (e) Manuals and training.
   (f) Guidelines [Smith and Aucella 1983] and 'good advice' for programmers (e.g. [Hansen 1971]).
4. Thursday.
   (a) Principles of isomorfism (WYSIWYG, Direct Manipulation, Transparency).
   (b) Example: Different ways of doing text editing and document formatting.
   (c) Practical example: Apple Lisa and Macintosh.
   (d) Examples of current research results.
5. Friday.

Design Principle	Danish Teachers (N = 15)	American Programmers (N = 447)
Early Focus on Users	60	62
Interactive Design	53	9
Empirical Measurements	13	40
Iterative Design	20	20

Table 1:
Percent of respondents mentioning each particular design principle.

No. Principles	0	1	2	3	4
Danish Teachers (N = 15)	13	40	33	13	0
American Programmers (N = 447)	26	35	24	14	2

Table 2:
Percent of respondents mentioning a given number of the four key design principles.

(a) The literature (both textbooks and scientific literature) [Nielsen 1984e].

(b) How do we teach software ergonomics? [Nielsen 1984d].

(c) Evaluation of the course.

3.4 Key Design Principles

The participants were asked to write down their suggestions for the four or five most important principles to follow in a design process to ensure the user friendliness of a new software system. This is almost the same exercise as that put to 447 programmers and system designers from American industry by Gould and Lewis [Gould and Lewis 1983], and the results from this survey are compared to their results in Table 1 and Table 2.

The answers from each survey participant were scored for whether they could be said to imply the use of the following principles:

1. *Early Focus on Users.* Understanding potential users through direct contact with such users before starting the design.
2. *Interactive Design.* A panel of expected users should work closely with the design team during the early stages of the design.
3. *Empirical Measurements.* Intended users should actually use simulations and prototypes to carry out real work, and their performance and reactions should be measured.
4. *Iterative Design.* When problems are found in user testing (the previous principle), they must be fixed. There must be a cycle of design, test and measure, and redesign.

The scoring was 'liberal' in both studies, i.e. the participants got the benefit of doubt whenever there was a possiblity that one of their answers could be construed to imply one of the listed principles.

Table 2 shows that both groups (American programmers and Danish teachers) had about the same total score for number of principles mentioned.

Table 1 shows however, that there is an interesting difference in *which* principles were mentioned. The principles of Early Focus on Users and Iterative Design were mentioned the same number of times. But Interactive Design was mentioned significantly ($p \leq 0.01$) *more* times by the Danish teachers than by the American programmers, while the principle of Empirical Measurements was mentioned *fewer* times by the Danish teachers than by the American programmers ($p \leq 0.05$).

Some patriotic Danes would claim that the higher score on the Interactive Design principle was due to the Scandinavian tradition of placing a high value on the active involvement of users and on ergonomics. Though this point probably has some influence on the difference between the two groups, the largest effect is probably due to the fact that the Danish group consisted of teachers who were asked what principles they *thought* one should use. The American group consisted of industry practicioners who were asked what principles they *actually* used. The Danish Computing Schools have for a considerable number of years thaught a systems development method *('SYSKON')* that advocates a very interactive design process with heavy user involvement. But unfortunately this does not imply that the principle is always followed in practice.

The lower Danish score on the Empirical Measurements principle is on the other hand probably due to a real difference between the situation in the

US and Denmark. There is somewhat of a Danish tradition of preferring 'soft' talk and discussions to 'hard' measurements in areas such as user friendliness. Some Danes may feel that such measurements have too much of a 'Scientific Management' flavor. Furthemore, almost all Danish companies are very small on an international scale and have therefore until recently probably felt that they could not afford a real measurement effort.

3.5 Conclusions

The present course was a mixture of

1. Theoretical principles and concepts (such as 'Direct Manipulation').
2. Empirical research results and methods (e.g. the selection of a set of icons for the Xerox Star [Bewley et al. 1983]]).
3. Design guidelines and advice for designers.
4. Presentation and discussion of existing systems.
5. An exercise consisting of a design problem followed by a thinking-aloud evaluation.

The participants were very oriented towards current industri practice and wanted more emphasis on No. 3, guidelines and advice for designers. They also wanted more practical exercises, but were less interested in presentations of measurement and research methodologies which they felt would not be of use in practice. Also, they did not feel that the most modern interaction principles would be used in commercial environments in the immediate future.

I felt that presenting such modern principles would hopefully hasten their introduction in Danish practice. Also I find basic principles, concepts and theories of more general use than very specific design guidelines. However a course such as the one described here can not be taught without mentioning an appropriate selection of such guidelines.

3.6 References

- Bewley, William L., Teresa L. Roberts, David Schroit, and William L. Verplank: "Human factors testing in the design of Xerox's 'Star' office workstation", *Proc. CHI'83 Human Factors in Computing Systems* (Boston, December 12-15, 1983), New York: ACM, pp. 72-77.

- Hansen, Wilfred J.: "User engineering principles for interactive systems", *Proc. AFIPS Fall Joint Computer Conference*, vol **39** (1971), pp. 523-532.
- Gould, John D., and Clayton Lewis: "Designing for usability - Key principles and what designers think", *Proc. CHI'83 Human Factors in Computing Systems* (Boston, December 12-15, 1983), New York: ACM, pp. 50-53.
- Nielsen, Jakob: *"A Virtual Protocol Model for Computer-Human Interaction"*, Technical Report **DAIMI PB-178**, Computer Science Department, Aarhus University, 1984a.
- Nielsen, Jakob: *"Learning Difficulties of Programmers Faced With a New Programming Environment"*, Technical Report **DAIMI PB-181**, Computer Science Department, Aarhus University 1984b.
- Nielsen, Jakob: *"The Spectrum of Models in Software Ergonomics"*, Technical Report **DAIMI PB-183**, Computer Science Department, Aarhus University, 1984c.
- Nielsen, Jakob: *"Three Approaches to Teaching Software Ergonomics"*, Technical Report **DAIMI PB-184**, Computer Science Department, Aarhus University, 1984d.
- Nielsen, Jakob: *"An Annotated Bibliography of Selected Literature on Software Ergonomics (User-Friendly Programming)"*, Technical Report, Computer Science Department, Aarhus University, 1984e.
- Smith, Sidney L., and A. F. Aucella: *"Design Guidelines for the User Interface to Computer-Based Information Systems"*, Technical Report **ESD-TR-83-122**, U.S.A.F. Electronic Systems Division, Hanscom Air Force Base, MA, 1983 (NTIS No. AD A115 853).

4 A Software Ergonomics Awareness Course for Hardware Ergonomists

4.1 Abstract

A one-day course in software ergonomics was given to hardware ergonomists without previous computer knowledge. The course relied heavily on research examples and on illustrations of modern interfaces.

4.2 Introduction

Hardware ergonomists have a strong tradition for designing peoples' physical workplace. How do we avoid glare in the terminal screen? This question and many others like it are part of their usual consulting work. They are not used to dealing with the more psychological and computer technological aspects of the software inside these machines. This is the speciality of software ergonomists such as the author of this paper.

On the other hand many of these traditional hardware ergonomists have daily contact with actual users in small companies that could never afford to hire one of the extremely few Danish software ergonomical specialists as consultants just because they are buying a single personal computer. And even larger companies often have a tradition of relying on hardware ergonomists for advice on user friendliness of new equipment.

Therefore an awareness course was given for interested hardware ergonomists to give them an understanding of software ergonomics. The course even had to be repeated due to demand.

4.3 Course Schedule

The course was a one-day course only. As the participants had almost no previous computer science knowledge, it was necessary to cover rather elementary concepts in a broad way to make sure that participants were not contrained by their own previous experience with very specific computer systems.

The course covered the following issues:

1. Overview of supplementary literature.
2. The principle of the 'hidden tiger': When you learn something new you cannot see the 'tiger' initially. But once you know where the animal is,

is is easy to find the second time (actually, it is not even possible for you to *try* not to see it. Therefore it is e.g. also easy to understand the text in a manual, if you already know how the system works.

3. Different types of ergonomics: Hardware, software and organizational.
4. What is software ergonomics? Its research methodology. Report from the latest international conferences (CHI etc.).
5. The four design principles of [Gould and Lewis 1983] - in theory and actual practice.
6. Computer history from the first to the fifth generation. Changes in interface technology, operating systems, application areas, and typical user groups.
7. Slide photographs of different types of computer equipment and different types of graphical and other modern interfaces.
8. The virtual protocol model for computer-human interaction [Nielsen 1984a].
9. Different types of users, each having different requirements: Novices, casual users, and experts; programmers and non-programmers.
10. Selected concepts from software ergonomics. E.g. WYSIWYG and Direct Manipulation.
11. Selected reserach results and experimental methods.
12. An in-depth example: Consistency in menus [Teitelbaum and Granda 1983].
13. Discussion.

4.4 Research Feedback

As part of this course some slides were shown of the experimental setup of a software ergonomics experiment done at Aarhus University [Nielsen 1984b]. The hardware ergonomists in the audience immediately complained that the experimental subject was sitting in wrong working position due to the use of a non-adjustable chair and table. As one of them said: What is the use of designing a superb programming tool if the person using the system will end up shortly having a backache?

This is of course true and it is unfortunate that the university does not have a laboratory with modern ergonomical furniture. My only defence is, that the experimental system in question was actually not a production quality system designed for non-stop use over an extended period of time.

4.5 Conclusions

It is of course impossible to 100% satisfactory course in only one day to an audience lacking computer science sophistication. However this course did succeed in giving the participants an overview of the subject of software ergonomics. An important part of the course was concrete examples of modern interfaces. Without these examples much of the more theoretical material covered would have seemed to be without any realistic basis, as many participants only had experienced a few limited computer systems previously.

Acknowledgement

The analogy with the 'hidden tiger' is due to Clayton Lewis who calls this tthe 'hidden animal problem'.

4.6 References

- Gould, John D., and Clayton Lewis: "Designing for usability - Key principles and what designers think", *Proc. CHI'83 Human Factors in Computing Systems* (Boston, December 12-15, 1983), New York: ACM, pp. 50-53.
- Nielsen, Jakob: *"A Virtual Protocol Model for Computer-Human Interaction"*, Technical Report **DAIMI PB-178**, Computer Science Department, Aarhus University, 1984a.
- Nielsen, Jakob: *"Learning Difficulties of Programmers Faced With a New Programming Environment"*, Technical Report **DAIMI PB-181**, Computer Science Department, Aarhůs University 1984b.
- Teitelbaum, Richard C., and Richard E. Granda. "The effects of positional constancy on searching menus for information", *Proc. CHI'83 Human Factors in Computing Systems* (Boston, December 12-15, 1983), New York: ACM, pp. 150-153.