11

HOW MANY NOVICES DOES IT TAKE TO MATCH THREE EXPERT DESIGNERS? LESSONS FROM AN EXERCISE IN PARALLEL DESIGN

Saila Ovaska and Kari-Jouko Räihä

University of Tampere, Department of Computer Science
P.O. Box 607, FIN-33101 Tampere, Finland

ov@cs.uta.fi, kjr@cs.uta.fi

KEYWORDS: Parallel design, Expert and novice designers, Usability engineering

ABSTRACT: In parallel design, several interface designers work independently on a design problem, and produce a set of ideas that can be studied further. In a panel at INTERCHI'93 three expert designers presented their first designs of a system for selecting and playing songs from a remote CD database. We have worked on the same exercise with novice designers. We show that also inexperienced interface designers can produce designs with many features worth investigating further. By increasing the number of novice designers also the number of new design ideas increases, and (in this exercise) finally exceeds the number of ideas produced by expert designers.

INTRODUCTION

In parallel design [Nielsen, 1993; Nielsen, 1994] several designers work simultaneously and independently of each other on the first drafts of a user interface. Parallel design is a method intended to quickly produce ideas for a new interface, thus exploring the design space. The resulting designs give the design team an understanding of what is important, and they also make it possible to pick one of the designs for further development. The main goal is not to spot usability errors in designs but to find design ideas worth of working on in the future.

A panel on parallel design was organised by Jakob Nielsen and Heather Desurvire [Nielsen and Desurvire, 1993] at the INTERCHI'93 conference. Three expert designers were given a design problem to work on. The designers were asked to produce a mock-up design before the conference. The members of the panel heuristically evaluated the designs.

The panel discussion was entertaining, but the designs produced by the experts have not been analyzed in detail. An interesting question is how many designers are needed in parallel design to get close to saturating the space of good design ideas.

In other studies [Nielsen, 1992] it has been observed that although experts, of course, are better qualified to carry out usability engineering tasks, novices can also do reasonably well, provided that sufficiently many novices are used. Another question that we wanted to investigate is whether this also holds for parallel design.

We have used the INTERCHI'93 design exercise [Nielsen and Desurvire, 1993] as a student project assignment in our HCI class. Here we analyze the designs produced by both the experts and our students. The results reiterate the value of quantity in usability engineering:

- A single designer, expert or no expert, is likely to overlook many promising design possibilities.
- Sufficiently many novice designers can come up with more good design ideas than a handful of experts.

In the following sections we first briefly describe the design problem. Next, the setting for the experiment, i.e. the arrangements of our course are described. Then the various problematic points of the design task are discussed to explain some items in the list of

K. Nordby et al. (eds.), *Human-Computer Interaction*

good or interesting features that we have collected from the designs produced. The list is used to evaluate the solutions of both experts and novices, and the data is used to predict the number of good design ideas found by these groups for this particular problem. We conclude by discussing why the results have to be interpreted with caution and by presenting some questions for further study.

THE REMOTE CD PLAYER ASSIGNMENT

The task as described in the INTERCHI'93 proceedings is to design a remote control panel for a hypothetical CD-player with a jukebox-like functionality and over 10 000 discs. The system operates like a central database, from which the songs are played via a broad band network to a set of speakers at the user's workstation. It is possible to select songs by different search criteria, and play them. A user can select

- one song with various artists
- one artist with various songs
- multiple selections for a certain time interval (e.g. 45 minutes of music)
- random songs in a selected genre (by singer, musician, or style of music)

While listening to the music, the user should also be able to use the usual CD control buttons like Pause, Fast Forward, Skip and Rewind.

The functionality description as given to the expert designers has (on purpose) been left vague by Nielsen and Desurvire; they claim that in practice no detailed functionality descriptions are available in the first place.

OUR HCI CLASS

The HCI course taught at our department is patterned after one of the model courses presented in the HCI Curriculum of ACM [Hewett *et al*, 1992, Appendix E]. The course is mainly intended for second year undergraduate students. The prerequisite for the course is an introductory course in programming. In 1993, the course was a required course for all computer science majors. Some of the students were more competent as programmers than the average second year student; for all of them, though, the course was the first one in HCI.

During the course, the students work in groups of one to three persons. In 1993, there were altogether 35 groups (67 students). The students used the whole semester for their designs, contrary to the few days that the experts reported in the panel. They carried out an assignment in four phases, each phase taking three weeks and involving feedback from the course assistant. The phases were the following.

(1) Task analysis: Analysis of a task somehow analogous to the CD player problem. The method used was (iterative) design and testing of a questionnaire.
(2) Designing of a paper mock-up for the CD player problem and usability testing of the mock-up.
(3) Implementation of a user interface prototype.
(4) Usability testing of the prototype. Test users were observed as they were asked to think aloud while carrying out a given set of tasks. Use of video equipment was optional.

We used the same task description (translated into Finnish) in our student projects that the experts had. The designers were given free choice of implementation platform; the most popular choices were Toolbook, HyperCard and Visual Basic. The solutions produced by the students evolved through several iterations. This paper is based on the last versions of their designs. The students did not see the list of design features that was prepared for this study.

ANALYSING THE TASK

The CD player interface presents several design challenges. First of all, the user may choose music from 10 000 discs; she may know what she wants, but how should the system show what is available for those who have not made up their minds? A database containing information on 10 000 discs would include data of at least 100 000 songs, and the songs may be searched with several criteria. Just listing the songs in alphabetical order is not sufficient. Design choices range from information filters [Loeb, 1992] to more conventional query systems.

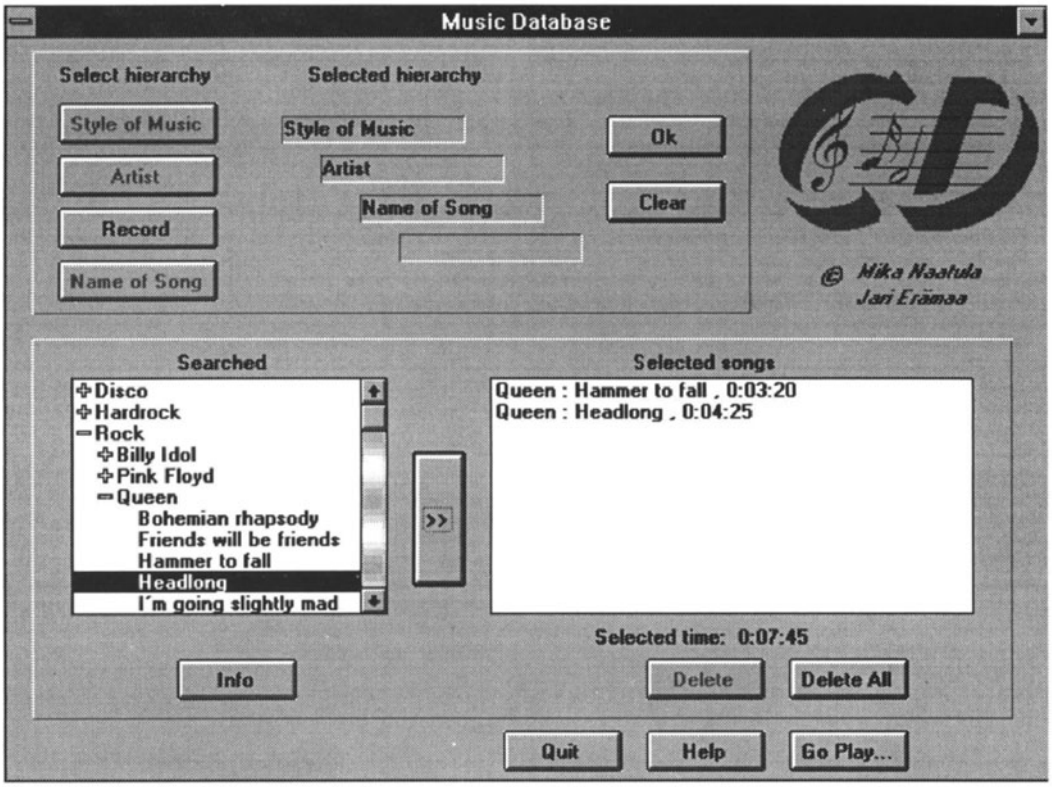
A major design decision is how to separate the selection and play phases. Should the system maintain a separate list of what has been currently selected and is now playing, and what happens to the play list when the user makes another selection – or when a song has been played? Typically, the user may make selections while music is playing; but in this case at least two lists are needed: the list from where selections can be made, and the play list. Moving data (and control) between the lists is a delicate matter that needs to be studied carefully, and it is partially connected with separating the lists into their own windows.

Another major decision is interaction style, since the CD player is planned to be used in a home computer environment. In the task description a graphic interaction style with mouse was assumed. This can be achieved using menus, buttons, or even direct manipulation, or a mix of these interaction styles.

As the task description allowed various interpretations of the task, also an extensive functionality can be planned. For example, saving play lists and loading them into use, managing time and cost, and sup-

porting a wider variety of search combinations will make the user more content with the system. Some of these options (like finding a song with just a phrase of its lyrics) may be hard to implement, but nevertheless, we have included them as desirable design features in our analysis.

In practice a necessary attribute for searching the songs would be the compact disc name. It was not mentioned in the task description, perhaps because including it makes the interface even more complicated. Some releases even consist of a set of CDs.



The screen shot above shows one of the better solutions produced during the course. It illustrates one further issue that the designer has to decide: how should the search space be visualized? One possibility is to view it as a set of songs, each with a set of attributes. Then solutions such as those used in the FilmFinder [Ahlberg and Shneiderman, 1994] could be used. Another possibility is to view the search space as an hierarchical structure: an artist has released a sequence of albums, each containing a sequence of songs. This forms what we call a "static" hierarchy. The user, however, might be interested in another kind of hierarchy: she could start with an artist, then select a style, and expect to see a list of all songs by the artist in the given style. A "dynamic" hierarchy adjusts to the search path of the user, and this is exactly what the shown solution facilitates. A drawback in the solution above is the need to sort the database according to the required hierarchy. The list of selected songs is always in the same format, though also whole albums and even the entire production of an artist may be moved into it with a single selection.

ANALYSIS OF THE DESIGN SOLUTIONS

To be able to compare the various solutions analytically, we made a list (presented in the Appendix) of all the good or interesting features that came up in the solutions. We have tried to include all features that are at least worth further consideration. Notably bad design features have been omitted from this analysis. This is in line with the goal of parallel design: the purpose is to come up with food for thought in developing the next version. An interface that may have serious overall problems but still contains some unique bright ideas can be very useful for further phases of the design process.

In this respect, our approach is fundamentally different from, e.g., that of Bailey [1993], who found that human factors experts produced notably better designs than programmers. A comparison of the overall quality of expert designs and novice designs would have produced a similar result also in our case. We do not compare the quality of the designs; instead, we are interested in how the group of novices can collectively help in parallel design.

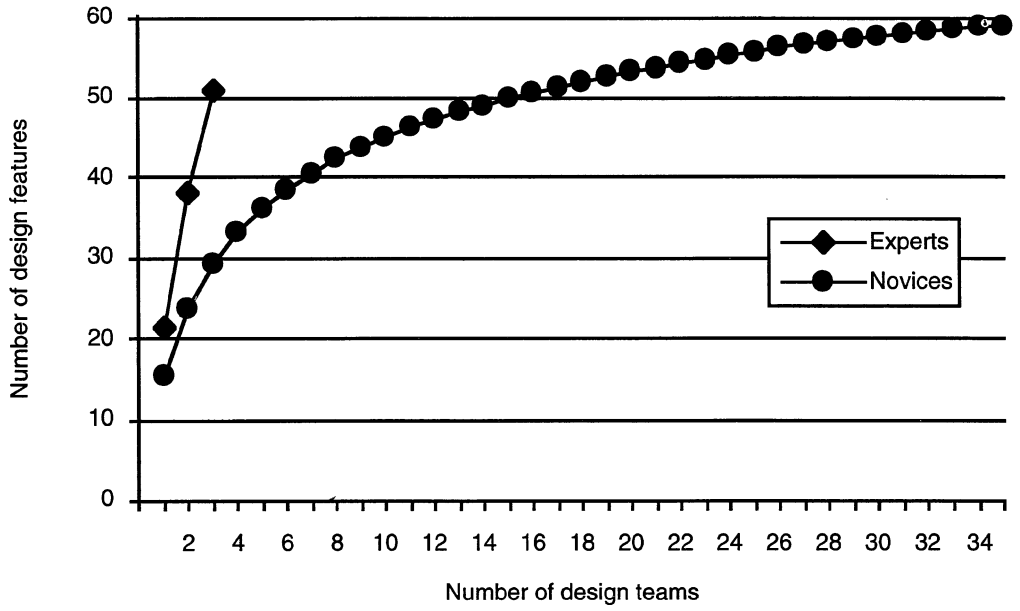
Altogether, our list of interesting design features contains 67 design ideas – a considerable number in view of the relatively limited functionality. Obviously, some design choices are mutually exclusive: a search list, for instance, cannot be simultaneously presented as a flat list and as a hierarchy. Therefore it is impossible for any single design to incorporate all the features in the list.

How did the designs fare? The variance was high in both groups. For the expert designs, the scores (number of interesting features appearing in the designs) were 16, 20, and 26. For the student designs, the scores varied between 9 and 24. It is interesting that quite a few student designs scored 20 or more, which is quite good when compared to the expert designs. On the other hand, 9 is a really low score – in some cases the list contains all possible design choices, so any design will automatically score a few points.

The first conclusion to be drawn from this is the one familiar from other contexts in usability engineering: never trust a single opinion. The really interesting questions are:

- how many parallel designs should be carried out to produce a large percentage of the useful design ideas; and
- how much does the experience of the designers affect this number.

In the following diagram we present the number of good design features that can be expected to appear in at least one design, when the number of parallel designs grows.



First of all, we observe that the designs produced by the experts were radically different. The slope of the curve is steep, and taken together, the designs of the three experts exhibit 51 of the 67 features. On the other hand, the novices were thus able to come up with 16 good ideas that did not appear in any of the expert designs. Inversely, the experts suggested 8 ideas that did not appear in any student designs.

It is remarkable that the expected value for just two novices is greater than that for a single expert! From there on the novice curve grows slowly, as could be expected. The expected value of 51 features for three experts is matched when the number of novices exceeds 16.

DISCUSSION

There are many reasons why the results obtained in this study should be considered critically, and why more similar experiments are needed to produce results that can be generalized.

The two groups approached the problem differently. The experts had other commitments and could spend only from a couple of days to perhaps a few weeks on the assignment. The students, on the other hand, worked with the assignment for almost three months and carried out at least two usability evaluations. However, we feel that this somewhat compensates for the fact that the students were really complete novices to interface design. The majority had never designed an interface before. In reality, one would expect even novices working in a parallel design project to have at least a basic training in interface design.

The fact that the students carried out usability evaluations also matches the goals of parallel design: it should represent one (and only one [Nielsen and Bergman, 1993]) step in the iterative design process. It is a commonly accepted principle that each iterative design phase should include usability evaluation.

It can also be argued that the expert designers were asked to produce one complete design. They were not asked to list possible features of the interface. They can, in principle, have considered some features on our list and abandoned them for some reason. The same is true, of course, for the novice designers. In our evaluation of the various designs, we have not relied solely on the proposed interfaces. We have also counted features that have been discussed by the designers but not included in the interface. We estimate that in this respect, neither group has benefited more than the other. On the other hand, the fact that the students had to implement a prototype could bias our analysis in favor of the experts. The students may have silently omitted some design ideas as too hard to implement.

The list of interesting design features used in the analysis is unavoidably somewhat subjective. Moreover, the features are of varying importance, ranging from details of visualization to fundamental design decisions. It is noteworthy, though, that both groups – experts and students – could come up with minor *and* major ideas not discovered by the other group.

The way that the analysis is presented may give the impression that we look at interface design as a competition in featurism. This, of course, is not our intention. A good interface is more than just a collection of nice features. For the next phase of the iterative design process, there were not many student solutions that could have competed with the best expert solutions from the status of the interface that is the basis for the next development phase. But again it is interesting that a couple of the best student solutions were on a par with the expert solutions.

These reservations notwithstanding, some observations can be made safely.

- In parallel design, as in most usability engineering methods, it is necessary to use at least a couple of subjects – in this case, designers.
- To some extent, substituting quality with quantity is possible: a sufficiently large number of novices can produce most of the design ideas generated by a few experts. In this experiment, the number of novices required was reasonable and in line with similar results obtained in studying usability inspection methods.

FURTHER WORK

This exercise was repeated in 1994 with a new slate of students [Ovaska and Räihä, 1995], with an interesting result: the number of expected design features grew more slowly than in the study described here. It could be expected that since we were ourselves more familiar with the problem, the students would receive different tutoring than the previous group. Moreover, the course is not a required course any more (very few of our courses are), so in 1994 there were only 22 groups, not 35 as in 1993.

There are several interesting directions to which we could take our study from here. For instance, how can we know what is a good design feature, and in particular, how well can the designers recognize those features? Another issue worth further study is the effect that the chosen prototyping tool might have on the existence of good design ideas in the solutions.

ACKNOWLEDGEMENTS

We owe a lot to the students that made this paper possible by working hard with the design and implementation of their prototypes. We all learned from this experience. We would also like to thank the expert designers of the INTERCHI panel: Randy Kerr, Dan Rosenberg, and Gitta Salomon, for kindly sharing with us their panel presentation material, and the panel moderator, Jakob Nielsen, for providing us with the complete assignment given to the designers (a slightly extended version of the one printed in the proceedings).

REFERENCES

[Ahlberg and Shneiderman, 1994] Christopher Ahlberg and Ben Shneiderman, The Alphaslider: A Compact and Rapid Selector. Proceedings of CHI'94, 365–371.

[Bailey, 1993] Gregg (Skip) Bailey, Iterative Methodology and Designer Training in Human-Computer Interface Design. Proceedings of INTERCHI'93, 198–205.

[Hewett *et al*, 1992] Tom Hewett *et al*, ACM SIGCHI Curricula for Human-Computer Interaction. ACM Press, 1992.

[Loeb, 1992] Shoshana Loeb, Architecting Personalized Delivery of Multimedia Information. *Comm. ACM* 35:12 (December 1992), 39–48.

[Nielsen, 1992] Jakob Nielsen, Finding Usability Problems Through Heuristic Evaluation. Proceedings of CHI'92, 373–380.

[Nielsen and Bergman, 1993] Jakob Nielsen and Marco G. P. Bergman, Independent Iterative Design: A Method That Didn't Work. Manuscript, 1993.

[Nielsen and Desurvire, 1993] Jakob Nielsen and Heather Desurvire, Comparative Design Review: An Exercise in Parallel Design (Panel). Proceedings of INTERCHI'93, 414–417.

[Nielsen, 1993] Jakob Nielsen, *Usability Engineering*. Academic Press, 1993.

[Nielsen, 1994] Jakob Nielsen, Diversified Parallel Design: Contrasting Design Approaches (Panel). Proceedings of CHI'94 (Conference Companion), 179–180.

[Ovaska and Räihä, 1995] Saila Ovaska and Kari-Jouko Räihä, Parallel Design in the Classroom. Short paper to be presented at CHI'95.

APPENDIX. THE LIST OF DESIGN FEATURES

Few windows (simple navigation)
Search criteria
- search by lyrics / era / substring of title / album / composer / lyricist / origin

Entering search conditions
- from a menu or list / from an adjusting list (type-ahead) / by typing
- names can appear in either order

Search method
- search list appearing after entering of one attribute / all attributes / arbitrary combination of attributes
- possibility to enter multiple choices for one attribute

Manipulation of search list
- manual selection / subset selection for given duration / selection of given number of songs / selection of most popular songs / selection of all songs in list
- possibility to sort the list
- possibility to gather selections together
- possibility to expand and collapse hierarchies
- fast sampling of songs

Presentation of search list
- selected songs highlighted / selected songs marked with checkmark / separate list of selected songs (for transfer into playlist)
- as strings on a line / as a list with fields in columns / as a static hierarchical structure / as a dynamic hierarchical structure
- duration of each song visible
- duration of selection visible

Handling of search and play lists
- separate search and play list
- several lists open simultaneously
- searching possible while music is playing

Transfer from search list to play list
- by pressing a button / drag and drop / by clicking or doubleclicking

Manipulation of play list
- possibility to shuffle the play list / to remove songs from playlist / to move songs to new positions in the playlist / to name and store the playlist / to clear the playlist
- played songs removed automatically

Presentation of play list
- duration of each song visible
- current song shown separately
- costs visible

Control of play
- start by using a button / by doubleclicking / by a menu command
- extensive / simple (easy to use)
- automatic repetition of play
- random play

Visualization
- display of CD cover
- play list as personal CD
- play mode as CD player

Error messages
- user generated errors not possible / as standard error messages / as synthesized speech / as non-speech audio

Miscellaneous
- possibility to name and store the state of the system
- expandability to other media
- additional information about songs available
- lyrics visible