Successful User Interface Design from Efficient Computer Algorithms

Gary Marsden
Computing Science
University of Cape Town
Rondebosch 7701, South Africa
+27 21 650 2666
gaz@cs.uct.ac.za

Harold Thimbleby, Matt Jones & Paul Gillary
Computing Science
Middlesex University
London, N11 2NQ, UK
+44 208 362 6061
harold@mdx.ac.uk

ABSTRACT
Exploiting standard computer science algorithms, we designed a more efficient user interface for a mobile phone. In experiments, the new design was found to be not only more efficient but preferred by users.

Keywords
Design ideas, mobile computing, cellular telephone

INTRODUCTION
Where does an interface design project start? Where can design ideas be found, and how can the different ideas be compared?

The authors faced this problem when approached by a cellular telephone service provider and asked to design a new interface to the handsets (mobile phones) they sold to their clients. Every handset in the range was designed around hierarchical menus, the only differences being in the way functions were arranged with the menu structure. As customers experienced problems with every handset (the ordering of the menus was not the problem), it was necessary to design a completely new paradigm for accessing the functions.

Computer Science
As the authors of this paper are all computer scientists, we started by viewing the problem as one of data access each handset has a number of functions, arranged in a hierarchical network, which the user searches to find the desired function. One handset (with 74 functions) needed the user to make, on average, 8.2 key presses to access a function; to access all menus required *at least* 110 key presses! As a data structure, it was clear that this type of structure was sub-optimal. What data structure would be optimal?

After studying several alternative data structures and algorithms, we designed a technique based on hashing which reduced the average number of key presses for accessing functions to 3.1, and for accessing every function to 74 key presses. We then experimentally compared the designs.

FROM ALGORITHMS TO USER INTERFACES
By using data structures as a source of inspiration, the designer has a source of well-understood alternatives (books such as [4] contain thousands of algorithms). Relative theoretical performances of designs are then well understood.

When designing the new interface to the handset, we considered structuring the interface around a linear list, a binary tree, a hash table and an ordered list with a binary search algorithm. Because the strengths and weaknesses of each of these techniques is theoretically well understood, we confidently choose hash tables as an implementation which would require the user to make the least number of key presses.

Using data structures also has the interesting consequence that the corresponding parts of the user manual describe can be viewed as an algorithm description. In a conventional design for a handset, where the menu choices are based on designer intuition, the manual must describe the location and nesting of each function in detail (which is also a learning burden on the user).

However, it could be argued that data structures are not the best way of designing user interfaces. We are not claiming that it is. Instead, we are claiming that there does not seem to be any strong methodology for designing interaction, and whilst considering interfaces as data structures may not ultimately prove to be the best way to design, we feel that it is a step in the right direction it also draws on a large body of theory [4].

The design solution chosen exploited the handset s alphabetic characters printed on its numeric keys. We used a hash table to map between the alphanumerics and the names of the functions. To access Call Divert the user would start by typing the keys 2255 (the key 2 has the letters ABC, and key 5 has letters JKL). The handset effectively searches the function names for any that start with the letter A or B or C, with second A or B or C and with third J or K or L.

In contrast IBM s *Speech Filing System* [1], which also used alphanumerics, required users to enter unambiguous input (thus typing a single letter C would require three key presses). A solution that does permit ambiguous input is Tegic s *T9* software [2], used for natural language text entry on devices with a reduced keyboard (like handsets). Rather than match function names, the T9 software disambiguates the input for natural language words which, for handsets, would reduce the key presses

required for messaging. T9 technology, however, is not used to access interface functionality as in our design.

Method

The hashing algorithm was prototyped in Bongo and then implemented as a full Java 1.1 applet. The Java version is available on the Web [3].

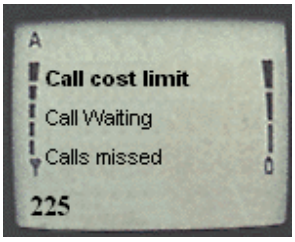

Figure 1. Screen shot of the mobile. The top menu item is in bold, showing that it will be invoked if the Function button is pressed, otherwise a call 225 can be made.

The new handset design is modeless: users press (standard) alphanumeric keys, whether to enter a phone number or to spell a function name; then press the Call button or press the Function button. Additionally, the user can use the scroll buttons to move to the function they require. Therefore, continuing from Figure 1, if the user had been looking for Call Waiting they need not spell the rest of the name, but simply press the down key once. When *searching* for a particular function, the scroll keys reduce the key pressing to access a function. The scroll keys allow functions to be searched in a linear fashion, hence achieving 74 key presses for browsing all functions.

EXPERIMENTAL EVALUATION

Whilst an optimal user (who knows all the function names and is not confused by the interaction paradigm) can access functions in fewer key presses, we need to evaluate how real users perform with the new interface.

A simulation of an existing handset and a simulation of the equivalent new design was evaluated with thirty users. Subjects were split in two groups, matched for gender, age and experience. Subjects were given twenty four randomly ordered scenarios to work through on each simulation. An example scenario was, Text messages are announced with a quiet beep: how would you change this to something more interesting? here the subject would try to access the appropriate function to allow them to change the message tone. All scenario descriptions used words different to those in menu descriptions to avoid biasing users towards the target words used by the new design.

As we were only interested in the structure of the interface, not the functionality of the handset, subjects were required only to find the function, not to change any settings. To remove order effects, one group used the standard design first, whilst the other group stated with the modified design first.

Each subject was videotaped and the number of keystrokes made in completing each task was recorded. After all tasks were completed, subjects participated in an interview and the video was used to analyze any scenario where they experienced difficulties. They were also asked which design they had enjoyed using and which features they found frustrating.

It should be noted that issues of screen size and font were not considered, nor did we compare simulations with real physical devices.

Results

Scenarios completed on the new design took, on average, 9.54 key presses to complete, in comparison to the standard design where 16.52 key presses were required. This is a strongly significant result ($p<0.001$) with users requiring approximately 7 *fewer* key presses, on average, to access the functions.

Observations

From the subjects feedback, and from analysing the video recordings we also made the following observations:

- Almost every subject preferred the modified design. This may be due to the fact that it was novel, but many subjects gave convincing reasons such as I was able to see all the items in the list nothing was hidden.
- The frustration of subjects when using the standard design was obvious. They would repeatedly become lost as they could not memorize, or visualize, the menu structure. Many subjects became caught in cycles within the menu structure (connections between menu siblings are in a continuous loop) from which they could not escape.

CONCLUSIONS

Ad hoc design provides no framework for specifying design criteria or objectively comparing design alternatives. By considering the user interface as a data structure, the designer can calculate the impact of design decisions using standard algorithm analysis.

We showed that our approach led to a more efficient user interface (in theory and in experiments), but one that was preferred by users. Further work should apply the many other untried algorithms to more user interface issues and evaluate the results. We anticipate designers would find many more better user interfaces.

ACKNOWLEDGEMENTS

The work was funded by EPSRC grant GR/M14548. Orange plc. provided handsets.

REFERENCES

1. Gould J.D. and Boies, S.J. Speech Filing — An Office System for Principals, *Readings in Human Computer Interaction* (Baecker, R. and Buxton, W. eds), 8—24, 1987.
2. Tegic Home Page http://www.tegic.com/ 1999.
3. Marsden, G. Java Handset Implementation. http://www.cs.uct.ac.za/~gaz/ 1999.
4. Cormen, T.H., Leiserson, C.E. and Rivest, R.L. *Introduction to Algorithms*, MIT Press, 1990.