Flight Searching – A Comparison of Two User-Interface Design Strategies

Antti Pirhonen[1] and Niko Kotilainen[2]

[1] Department of Computer Science and Information Systems
[2] Deparment of Mathematical Information Technology
P.O. Box 35, FI-40014 University of Jyväskylä, Finland
{pianta,npkotila}@jyu.fi

Abstract. The most usable user-interface is not necessarily the most popular. For example, the extent to which an interaction is based on graphics can depend highly on convention rather than usability. This study compares contemporary flight search applications in order to investigate whether a more extensive use of graphics can enhance usability. Two user-interfaces are compared: one follows the ideal principles of graphical user-interfaces and direct manipulation, while the second interface requires text to be entered with a keyboard. The results of the comparison indicate that even an early prototype of the graphics based alternative performed better than the typical formula based search application for several measurements of usability.

Keywords: Flight search, direct manipulation, graphical user interface.

1 Introduction

Why did the graphical user-interface (GUI) become the standard in personal computing after an era of command line interfaces (CLI)? This is a contentious question which has no simple answer. When the Microsoft Windows GUI was introduced, its superiority over CLIs was not at all clear [2, 5]. Some 20 years later, it can be argued that the first sceptical comments were caused by the relatively slow and clumsy graphics of the computers of that time. However, the critique against GUIs was not based on this kind of simple technical argument – everyone knew that the graphics would inevitably become more fluent over time. Therefore the target of critical statements was not the first clumsy implementations of GUI but the underlying logic and principles, recently also trust [8].

Much of this critique can be understood in terms of "user profiles". Before the advent of the GUI for personal computers, a typical computer user was a technically oriented computing specialist. The CLIs and their cryptic commands had been developed to satisfy the needs of expert users, for whom the essence of computing was effective code and underlying logic. The only value of the user interface was its ability to reflect the underlying computation.

J.A. Jacko (Ed.): Human-Computer Interaction, Part III, HCII 2009, LNCS 5612, pp. 179–188, 2009.

The introduction of the GUI coincided with a major revolution in computing, which was not technical by nature. When the personal computer became mass-marketed, there was a clear need to develop the computer with ordinary people and their practical needs in mind. From this point on, a typical user was not considered interested in the effectiveness of the code or other technical issues, as long as the performance of a given task is effective. Meanwhile, the development of microprocessors has provided so much computational capacity that the optimisation of code and other traditional virtues of computing no longer hold the same value. A computer can even be programmed with the help of graphical tools, which may not produce the neat and compact code that a human programmer could produce, but sufficiently for a contemporary processor.

At the start of the GUI revolution, applications were bursting with elaborate graphics. It is easy to argue that these graphical fireworks were designed to impress prospective computer buyers, and that most of the new features did not have much to do with the actual tasks to be performed within the application.

A famous rational argument for GUIs was based on the notion of direct manipulation [7]. The idea was that with a GUI, the user has direct access to the entities to be worked with, without the need to remember complex syntaxes.

In this study, we first briefly analyse the current usage of user-interfaces (UIs), and the arguments for and against the various types. We then examine the differences in the approaches by comparing a formula-based and a graphics-based UI for a flight search application. The pros and cons of each strategy are analysed, based on a usability evaluation.

2 Differences in UIs in Terms of the Usage of Graphics and Input Devices

The use of graphics divides user interfaces to different genres. In addition to the use of graphics as part of a visual display unit (VDU), the use of a graphical input device also creates a distinction between the main categories of user interfaces. Interaction in a CLI relies on text input, and therefore the dominating input device is a keyboard. Input for a GUI largely relies on mouse or another two-dimensional pointing device.

After the heyday of an elaborate use of graphics in GUIs, the novelty effect of the new interaction style has faded out. The GUI has established itself as the dominant paradigm for interaction design but the use of graphics alone is no longer considered the hallmark of an effective program. At the same time, the borderline between GUIs and CLIs has become somewhat ambiguous. A CLI is often implemented within GUI. For example, many GUIs require text input in certain text fields, making the interaction with the application similar to a CLI. Therefore, it is not necessarily appropriate to make a distinction between the categories anymore. Rather, a CLI and a GUI in their original meaning represent two ends of a continuum. Most user interfaces fall somewhere between these extremes.

In addition to the use of output graphics and the input device, a third property is often connected with GUIs: the simulation of real world objects. These kinds of user interface elements are sometimes referred to as metaphors. Although the use of the

word metaphor in this context is contentious[1], it is clear that the imitation of real world entities is one of the most typical interaction design strategies for creating intuitive mappings within an application. However, the imitation of physical objects with digital technology can sometimes hinder the development of technology. For example, the physical design of the first generation digital cameras highly resembles that of film cameras. Only a few manufacturers had the courage to completely redesign the camera. Indeed, most digital cameras continue to carry the outdated physical restrictions of the cameras of the past with them. This could be due to conservative consumers who want that their digital cameras to look like cameras. Presumably, these design inefficiencies will be gradually reduced as new designs gain mainstream acceptance.

In computer applications, the same kind of evolution can be observed. For instance, the first graphics based self-service banking applications imitated the familiar paper forms. After a while, they were replaced with more efficient forms based on a clear step-by-step procedure instead of the completion of a one single form.

A related debate from the early 1990's was evoked by Bonnie Nardi and the notion of visual formalisms [4]. She strongly opposed 'metaphors' in design, i.e., the imitation of real world objects such as the famous 'desktop metaphor'. Nardi implies that not everything in a user-interface needs to have a direct counterpart in the physical world. The concept of visual formalism was intended to combine direct manipulation with a user interface designed in terms of the capabilities of the computer rather than the technology of the past.

While it is possible to discuss different UI-styles and their properties endlessly, it appears that the borderline between the concept of a GUI and a CLI is usually technically defined. A GUI is graphical because the output is graphics. For instance, rather than using letters as the atomic units of words, GUIs draw each letter in VDU with a large number of pixels. However, in terms of interaction design this kind of technical definition is not necessarily appropriate. Even in GUIs, CLI-like interaction can be implemented. Therefore more interesting than comparing a GUI and a CLI is to analyse how current practices of using GUIs fulfils the original ideas of direct manipulation. Do UIs provide direct access to underlying entities, or is interaction based on rules and syntaxes which need to be learned?

3 User-Interface for a Database Query

Databases have inherited much of their terminology from the technology of the past, like files and folders. This nomenclature has made it easy to adopt highly abstract computing concepts. What is new compared to paper files and folders is the ease and

[1] GUIs are sometimes said to be based on metaphors. However, precisely what is meant by metaphors in the context of user interfaces is questionable [6]. In metaphor theories, metaphors are the mental entities on which human conceptualisation processes are built. On the contrary, in the context of UIs the term metaphor denotes the imitation of real world objects. Since we find the latter use of the term metaphor inaccurate, we reject the use of the term in this study in favour of more accurate expressions. In other words, we don't argue that the difference between GUI and CLI is that GUIs would be based on metaphors.

speed of managing the database. For example, making a query to a large database is revolutionarily easy compared to paper files. Perhaps, the superiority of digital databases has made it easy to accept a rather 'techy' interaction with the databases. For example in most applications, the queries require text input similar to that required in a CLI. The user enters text, and confirms this text input by pressing a physical (e.g. 'Enter' in a keyboard) or a virtual push-button.

Text-based data entry has become so widely accepted in database queries that it has been implemented in all types of database applications and in a wide variety of contexts. However, in the following sections we describe an application design project, which made us consider whether formula based data query is necessarily the most usable strategy.

3.1 Searching for a Flight: The Creation of FlightMapper

Air traffic has recently continued in growth, partly because of the emergence of budget airlines that have made flying a possibility for many travellers who could not previously afford it.

In an effort to cut costs, many airlines have embraced online services. In particular, flight searches and booking are widely accomplished via the Internet. The quality of online services is a major competitive issue among airlines, and this should motivate investment into the development of these services [3]. Flight search facilities and their quality have been cited as the most important feature of online service from a users' point-of-view [1].

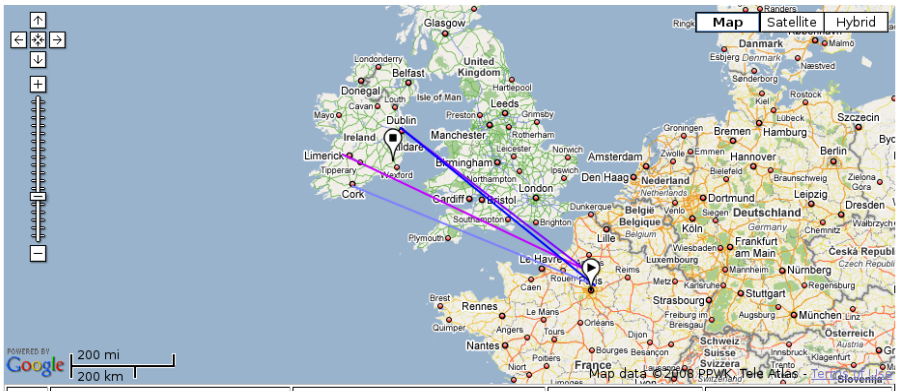
Online flight searching applications are now available for most flight operators. In addition, there are a number of services which have access to the databases of several different operators. This has made it possible to easily search and compare different flight operators.

All of the flight search applications we found were based on the same kind of user-interface design concept. The user chooses the departure airport, destination and date. This information is typed in the fields of a formula and then the search is launched by clicking a push-button. An intermediate screen shows that the search is being performed. After a while, the search results are presented.

The kinds of database query formulas described above are so familiar that we rarely question their appropriateness. However, in the context of a flight search, a much easier and illustrative process can be implemented with a fairly simple[2] web-application.

The creation of new kind of flight search application, FlightMapper, arose from very practical needs. It was found that the traditional formula based queries, which share resemblances with a CLI, might be satisfactory when the exact time, date, and departure location are fixed and known. However, consumers who have a flexible schedule but a small budget for travelling are often happy to choose the destination and the whole itinerary according to the most affordable option rather than be fixed to a specific date and route. In UI-design terms, these consumers present a different use

[2] The system consists of a web server replying to users' queries for flights and a JavaScript client running in the users' web browser. Queries are made using the AJAX technique. Google Maps API was used in implementing map functionalities.



Color	From	To	By	Total distance to airports
	Paris (CDG), France (23 km)	Dublin (DUB), Ireland (104 km)	Aer Lingus(Direct)	128 km
	Paris (CDG), France (23 km)	Cork (ORK), Ireland (147 km)	Aer Lingus(Direct)	170 km
	Beauvais (BVA), France (71 km)	Dublin (DUB), Ireland (104 km)	Ryanair(Direct)	176 km
	Beauvais (BVA), France (71 km)	Shannon (SNN), Ireland (156 km)	Ryanair(Direct)	228 km

☐ Show only direct flights

Fig. 1. Screenshot of search results with FlightMapper

scenario than the ordinary business traveller. In this scenario, the traveller is trying to rapidly figure out the available flight routes between somewhere here to somewhere there. Rather than having certain airport or city or even country in mind, the user in our scenario scans opportunities to different areas and is content to continue or start the journey by train, bus, or some other vehicle to reach an interesting travel destination.

The starting point for the creation of the user-interface of FlightMapper was an ordinary map. This map is the abstraction of geography that best meets the requirements of a nomadic traveller. It illustrates the physical directions and distances, major towns and borders of countries.

When providing information for the application about the preferred place of departure, the map is utilised by simply pointing and clicking on a place on the map with a mouse pointer. The same is done for the preferred destination. FlightMapper then starts searching flights roughly (within a given radius) from the selected departure point to roughly the selected arrival point. The application then draws the available routes on the map, and prints the names of the airports and operators for each alternative.

If no flights exist, no routes are drawn on the map. The user must then try another place of departure and/or destination. This change again utilises familiar GUI routines. The marked point of departure and the point of destination can be dragged-and-dropped elsewhere on the map. In this way, the user can effortlessly scan a large number of travel plans.

Figure 1 illustrates the search results when setting the departure point near Helsinki, Finland and destination point near Amsterdam, The Netherlands. The application returns several options. The pins indicating the departure and destination points can be dragged to new locations. Once the pins are dropped again, the application immediately returns new results.

The user-interface of FlightMapper relies on graphics and mouse operations like pointing, choosing, and drag-and-drop. It can therefore be seen as a representative of GUI and direct manipulation. We decided to contrast it with a typical, formula based flight search application. We hypothesised that for the needs of the above described use-scenario, FlightMapper would be faster and generally more appropriate. To verify this assumption, we organised a usability evaluation to compare the two design strategies.

3.2 Usability Evaluation Setting

In the usability evaluation, we wanted to compare the overall usability and performance time in particular, of FlightMapper and a typical flight search application. From the typical, formula based flight search applications we chose www.amadeus.net as the representative of this kind of applications. Amadeus was chosen because it is widely used and has all the typical properties and functionalities of flight search applications.

Twelve people from the University of Jyväskylä took part. Eight participants were male and four female. Five of them were staff and seven were students. The ages ranged from 21 to 48. The completion of the tasks took about 10-15 mins per participant, and the participants were rewarded with a movie ticket. The sessions were videotaped so that both the actions of the participant and the screen events could be traced. We used an external tft-display pointing towards the camera to enable this.

Each of the participants had a set of ten simple tasks to be performed with both FlightMapper and Amadeus. Six of the participants started with Amadeus, and six with FlightMapper. The tasks were to find out if there are flight routes available between given areas. The participants had a printed table of 10 pairs of cities. After each search, the participant was supposed to mark on the table ‘yes’ or ‘no’.

Before the first task with each application, each participant was given a very short demonstration about the use of the application. In practice, this meant that the researcher demonstrated how the application works. No hands-on trials were allowed before the first task, because we wished to get data about the learnability of the applications.

Concerning usability, our focus was on performance time. To measuring the time taken for each task, we did not create any separate log file but relied on the time code of the digital video. After the sessions, we used the video to record the starting and completion time of each task. However, in this kind of task, the definition of starting time might be ambiguous; is it when the gaze is in the next task printed in the paper, or perhaps when the first physical movement toward the departure city is done with the mouse? To maintain reliability, we defined the starting time of one task as the completion time of the previous task. In practice, this meant the point of time when the participant wrote the search result on the paper. In the case of the first task, the starting time was defined as the first movement of the mouse.

After the completion of all the search tasks, each participant was asked to complete a usability evaluation form. The form was a slightly modified (and translated) version of James Lewis’ post-study system usability questionnaire (PSSQU). The modifications included the elimination of questions concerning help-options, since there were none available for FlightMapper. This is because FlightMapper is only a

prototype and lacks many of the functionalities of a final application. The problem of comparing a prototype with a widely used application is discussed later. Another modification was due to the comparative setting: the participants were asked the same questions concerning both applications.

3.3 The Results of the Usability Evaluation

Performance Time

We performed statistical analysis on the recorded performance times. Figure 2 illustrates the summary of the task completion times.

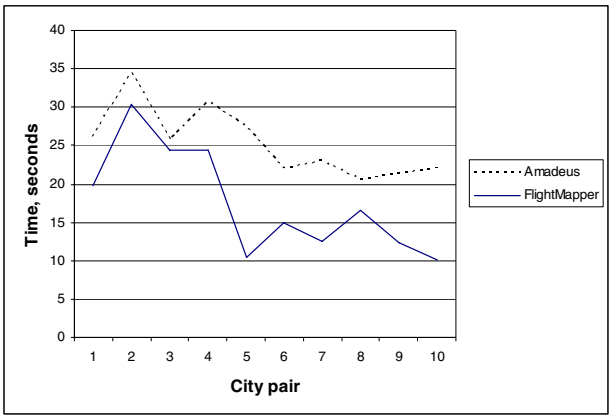


Fig. 2. Mean performance time, with the fastest and slowest removed

As described in the caption of Figure 2, we rejected the fastest and slowest performances. This is because according to our observations, both exceptionally fast and exceptionally slow times were due to experimental shortcomings. For instance, in some cases when the participant was searching for a flight, the route of the next task could be seen in the same view, resulting in zero performance time. This occurred when FlightMapper was looking for airports within a certain range, and by chance, two airports which were related to consecutive tasks, were in range of one single search.

Figure 2 illustrates the differences in the use of the applications. First of all, it confirms our hypothesis that FlightMapper is more efficient than a traditional form based application. As can be seen from the curves, performance times shorten towards the end of the set with both applications, but with FlightMapper the drop is much clearer. However, the more drastic drop of performance time with FlightMapper cannot be interpreted as a higher level of learnability. Presumably, the participants were familiar with with Amadeus, or at least some similar kind of application. Interacting with FlightMapper, on the contrary, was not that familiar, although it was based on familiar GUI routines. Therefore it is natural that the drop of performance times in the beginning of the session is clear. In other words: the use of familiar Amadeus did not require learning to the extent as FlightMapper did, and therefore the learning curves should not be compared.

The drop of performance times is not steady with either of the applications. This is easy to understand, having followed the sessions: it simply took different times to either find a place on the map (FlightMapper) or to type the name of the city (Amadeus). The other common feature with these curves is that the second task took more time than the first one, but this is probably due to difference in time measurement, as reported above: the starting point of the first task was defined differently from the starting points of the other tasks. Also, especially with FlightMapper, it took more time to complete a task when no flights were found; the user waited for a while before noticing that there were no search results.

In order to summarise the results of performance time measurement, we derived a t-test and a Wilcoxon test (because of the small number of participants) for the whole data. In the t-test for medians $t = -3.519$, $df = 11$, $p = .005$ (FlightMapper's times were shorter), and for mean performance times $t = -4.339$, $df = 11$, $p = .001$ (FlightMapper's times were shorter). In Wilcoxon test for medians $Z = -2.275$, $p = .023$ (FlightMapper's times were shorter), and for mean values $Z = -2.903$, $p = .004$ (FlightMapper's times were shorter).

Subjective Evaluation

The post-study questionnaire (PSSQU) provided with subjective observations about essential usability issues. The questionnaire measured overall usability, system usefulness, information quality, and user interface quality, on scale from 1 to 7. The number 1 indicated the highest value of usability. In the modified and translated version of the questionnaire there were 14 questions concerning each application. The mean values of each factor are presented in Table 1.

From the table it can be seen that there were no large differences in the experienced usability. Concerning all other factors than interface quality, FlightMapper was assessed slightly better. In usefulness, difference in favour of FlightMapper was clearest.

Table 1. Average usability scores in subjective evaluation

	Overall usability	System usefulness	Information quality	Interface quality
FlightMapper	2,38	2,13	2,61	2,86
Amadeus	2,59	2,49	2,70	2,81
Difference in favour of FlightMapper	0,21	0,36	0,09	-0,05

4 Conclusions and Discussion

We started this report by contrasting CLIs and GUIs. We then compared two user-interfaces, of which one had inherited essential properties from CLIs, and the other one had clearly GUI with its extensive use of graphics, use of the mouse as the dominating input device, and the application of the principles of direct manipulation.

What did we learn then? From a single case study, obviously, no universal conclusions can be drawn. However, we argue that this case study illustrates issues which are worth consideration in several contexts.

On the basis of the results of this study, which one is better, formula based (analogous with CLI) or graphics based (analogous with GUI) user-interface? Let's have a look at the quantitative results of this study – FlightMapper, the representative of GUI-style, was significantly faster, it was found more useful, slightly more usable and slightly better in terms of information quality. In terms of user-interface quality, the scores were practically equal.

Are these results clear enough basis for arguing that FlightMapper is a better tool for finding a flight, not to speak about the comparison of text based and graphics based user-interfaces? In order to interpret the results, we will need to remember the setting:

First, the form based application (Amadeus) is widely used, and probably has had a respectable evolution with numerous re-design iterations, where as FlightMapper is still an early prototype. Taking into account this David vs. Goliath setting, the results were relatively positive for FlightMapper.

Second, FlightMapper was based on the use scenario of a flexible traveller. The tasks in the evaluation can be argued to be a direct reflection of that scenario. Undoubtedly, with tasks that did not reflect FlightMapper use scenario, the results would have been very different.

Third, being a prototype, FlightMapper lacks many functionalities that are common in database queries. Therefore the role of this evaluation should be seen only as a test of which application best enables a fast scan of available routes. In the further development of FlightMapper, more functionality will emerge, and a broader evaluation will be carried out.

This evaluation also showed that there are database query tasks in which 'traditional' GUI is faster and generally more usable than a text input based formula. It is quite understandable, that when more and more public services are going online, there are things to which people simply get used while using them regularly. They become de-facto standard. However from the point-of-view of usability, the result is not necessarily ideal.

The comparison of the two UIs in this study does not follow the traditional division be the pro-users' CLI and the consumers' GUI. The graphics version was faster with all of the participants, whether technically oriented or not. This shows that different interaction designs should be considered by all user groups.

The purpose of this study was not to prove one user-interface design strategy as superior to another. Rather, we are illustrating that in terms of usability, current GUI design conventions do not always propose the best design practice but can still rely on the interaction paradigm of CLIs.

References

1. Benckendorff, P.: An exploratory analyisis of traveler preferences for airline website content. Information Technology & Tourism 8, 149–159 (2006)
2. Hazari, S.I., Reaves, R.R.: Student preferences toward microcomputer user interfaces. Computers & Education 22(3), 225–229 (1994)
3. Long, F., Poskitt, H.: Aerlingus.com – A Usability Case Study. In: Proceedings of the Irish Ergonomics Society Annual Conference, pp. 42–47 (2003)
4. Nardi, B.A., Zarmer, C.L.: Beyond models and metaphors: Visual formalisms in user interface design. Journal of Visual Languages and Computing 4, 5–33 (1993)
5. Petre, M., Green, T.R.G.: Is graphical notation really superior to text, or just different? Some claims by logic designers about graphics in notation. In: Proceedings of the Fifth Conference on Cognitive Ergonomics, Urbino, Italy, September 3-6 (1990)
6. Pirhonen, A.: To simulate or to stimulate? In search of the power of metaphor in design. In: Pirhonen, A., Isomäki, H., Roast, C., Saariluoma, P. (eds.) Future Interaction Design, pp. 105–123. Springer, London (2005)
7. Shneiderman, B.: Designing the user interface: Strategies for effective human-computer interaction. Addison Wesley Longman, Reading (1998)
8. Takayama, L., Kandogan, E.: Trust as an underlying factor of system administrator interface choice. In: Extended abstracts of CHI 2006, pp. 1391–1396. ACM Press, New York (2006)