RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS
No. 85 2009
Advanced Information Technologies for Management – AITM 2009

Waldemar Bojar

University of Technology & Life Sciences, Bydgoszcz, Poland
e-mail: waldemar.bojar@utp.edu.pl

TENDENCIES IN USER INTERFACE DEVELOPMENT (MODELLING) FOR DSS NEEDS

Abstract: In the paper some principles and trends in User Interface (UI) design were shown. Especially selected functional and exploitation characteristics of well designed UI in context of Decision Support Systems user needs were highlighted. Review of appropriate literature allows to conclude that evolutionary approach and interactive process of UI updating should be preferred but some standardized exploitation principles in application design have to be saved to avoid errors and too extended costs.

1. Introduction

Because of spreading IT as a tool of majority of workers and spending much labour time using computer it is necessary to design the most friendly and effective user interface ensuring not only business needs of human beings but also their psychic and emotional requirements. In recent years the development of highly interactive software systems with graphical user interfaces has become increasingly common. The acceptance of such a system depends to a large degree on the quality of its user interface.

The only such individual customized way for user interface design, which ensures maximum comfort of computer users, can face current challenges in this matter [Glushko, Tabas 2009].

Especially the problem arises due to two areas of effective Decision Support Systems: their functional requirements and exploitation constraints. On the one hand effective DSS can satisfy differentiated, sometimes mutually excluding themselves, business requirements. On the other hand some technical requirements ensuring safety of information or limited costs should be considered. Huge area networks and common availability of different UI designs make it possible that UI process designing mistakes made by certain UI designers can be quickly multiplied by others, which causes danger of wrong solutions replication and even sanctioning them as “standards”.

Particularly, the circumstances mentioned above ought to be implemented in communication interface project, which can be defined as possibilities of work of expert with DSS through differentiated alternatives of graphical, text and sound files.

A project of interface for complicated, hybrid DSS can be a unique solution or can become commercial tool of extended applications.

Functionality of DSS can be expressed through, among other, intuitiveness of UI [Takeuchi, Sugimoto 2009].

2. UI design principles

A fundamental trend in application development is user interface understood as a system for users. User needs are to be crucial premises for programmers to create applications. Programmes ought to satisfy them and simultaneously be simply in operation. In the opinion of Constantine [1999] user interface lets people, who understand essential problem features, work with application, avoiding reading an operation software guide or being trained. Better user interface is important because of some reasons. First, it encourages higher number of customers to use the application. Second, good user interface decreases training costs. Third, high quality user interface increases satisfaction of users applying software and gives a chance for growth in number of customers. These circumstances determine UI design principles described below.

The simplicity principle means that UI design should make common tasks simple, communication clear and simple in the user's own language, and provide good shortcuts that are significantly related to longer procedures. UI design **visibility principle** is expressed in keeping all needed options and materials for a given task visible without distracting the user with irrelevant or redundant information. Good designs do not overwhelm users with too many alternatives or confound them with needless information. **The feedback principle** implies UI design which is to keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unmistakable language well-known to users. **The tolerance principle** denotes that UI design ought to be flexible and open-minded, reducing the cost of mistakes and mistreatment by allowing undoing and redoing, while also preventing errors wherever possible by standing for varied inputs and sequences and by inferring all reasonable actions reasonable. **The reuse principle** indicates such UI design which should reuse internal and external components and behaviours, maintaining consistency with purpose rather than just arbitrary consistency, thus reducing the need for users to rethink and remember.

With regard to the principles mentioned above user interface should satisfy the following characteristics:

- efficiency, effectiveness and functionality,
- adaptability and a low level of difficulty,
- flexibility,
- users preferences,
- rule of simplicity and elegance,
- rule of correct screen,
- rule of appropriate graphic representation.

Among basic rules some key trends expressed in different models of User Interface can be observed. In my opinion, the model by Bob Baxley, described below, shows well contemporary tendencies in UI design [Ambler 2002].

3. Increasing role of communication rules against architecture functions to satisfy user needs

Increasing role of communication rules against architecture functions is expressed in Bob Baxley's Universal Model of a User Interface [Ambler 2002]. It begins on the established model of structure-behaviour-presentation but adds additional levels of granularity and specificity. Structure-behaviour-presentation can be seen across many models of user experience. Although the traditional delineation between structure, behaviour, and presentation served as an obvious starting point, those three elements alone did not provide sufficient granularity to describe the full set of issues and considerations involved in more complex forms of interactive media such as Web applications.

Baxley's model also does a great job of cementing the role of a generalist (or strong design lead) on complex product designs. Someone on the project team needs to carry out the interface design from conceptual model all the way to tone and voice (the text in the presentation layer) in a consistent and cohesive manner. When each tier is owned by a specialist and no one owns the top-level interface vision, the user experience lacks the focus needed to communicate a unified and clear message to users. Like other sophisticated, multi-dimensional forms of communication, interactive media require the designer to harmonize and balance a variety of differing and often opposing concerns. Even though a user encounters an interactive product as a single, unified experience, the designer has to construct and understand the experience one element at a time. This requires the designer to proceed with an understanding of discrete interface elements as well as appreciation of their influence on the whole.

Characteristics and trends mentioned above can be reached only when designer will respect more detail rules like those specified below.

4. User Interface implementation recommendations – know-how to design UI effectively

To satisfy compliance. User interface should work in a regular way, consistently. UI consistency lets users to build up precise psychic model of their functioning and this way facilitate work with software and decrease training and support costs.

Setting standards and meeting them. The only way of ensuring compatibility of the application is setting UI design standards and maintaining them. It has to be at least Agile Modeling's (AM) introducing and usage of those norms in practice.

Be prepared to hold the line. When one can develop the user interface for his system one will be able to find out that holders often have some unusual ideas as to how the user interface should be developed. Definitely the attention should be paid to these ideas but it is also necessary to make our holders aware of business UI standards and the need to conform to them.

Explaining the rules. UI users need to know how to work with the application built for them. When the application works consistently, it means we only have to explain the rules once. This is a lot easier than explaining in detail exactly how to use each feature in the application step-by-step.

Navigating between major user interface items is important. If it is difficult to get from one screen to another, then the users will quickly become discouraged and give up. When the flow between screens matches the flow of the work the user is trying to accomplish, then the application will make sense to our users. Because different users work in different ways, your system needs to be flexible enough to support their various approaches. User interface-flow diagrams should optionally be developed for further understanding of the flow of the user interface (see Figure 1).

User interface-flow diagrams are typically used for one of two purposes. First, they are used to model the interactions that users have with our software, as defined in a single use case. For example, a use case can refer to several screens and provide insight into how they are used. Based on this information, one can develop a user interface-flow diagram that reflects the behavioural view of the single use case. Second, as we see on Figure 1, they enable us to gain a high-level overview of the user interface for your application. This overview is effectively the combination of all the behavioural views derived from our use cases, the result being called the architectural view of our user interface [Constantine, Lockwood 1999]. The high-level overview approach is also referred to as the architectural approach, because it enables to understand the complete user interface idea.

Navigating within a screen is important. In Western societies, people read left to right and top to bottom. Globalization and common, global IT receivers force UI designers to invent screens that are also organized left to right and top to bottom when designing a user interface for people from such culture. One can want to organize navigation between widgets on your screen in a manner users will find familiar to them.

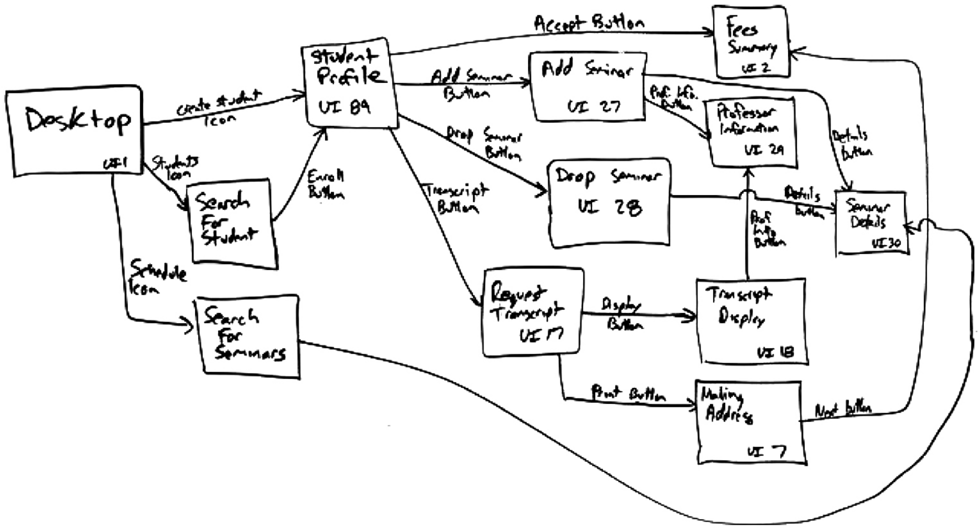

Figure 1. An UI flow diagram

Source: [Ambler 2008].

Word your messages and labels effectively. The text displayed on screens is a primary source of information for users. If the text is worded poorly, then our interface will be poorly recognized by your users. Using full words and sentences, as opposed to abbreviations and codes, makes our text easier to understand. UI messages should be worded positively, imply that the user is in control, and provide insight into how to use the application right. For example, which message do we find more engaging: “You have input the wrong information” or “An account number should be eight digits in length”. Furthermore, our messages should be worded consistently and displayed in a consistent place on the screen. Although the messages “The person’s first name must be input” and “An account number should be input” separately are worded well, together they are incoherent. In light of the first message, a better wording of the second message would be “The account number must be input” to make the two messages consistent.

Understanding the UI widgets. Widget is a combination of a graphic symbol and some programme code to perform a specific function, e.g. a scroll-bar or button. Windowing systems usually provide widget libraries containing commonly used widgets drawn in a certain style and with consistent behaviour. One should use the right widget for the right task, helping to increase the consistency in our application and probably making it easier to build the application in the first place. The only way we can learn how to use widgets properly is to read and understand the user-interface standards and guidelines our organization has adopted.

Looking at other applications with a grain of salt. Unless we know another application has been verified to follow the user interface-standards and guidelines of our organization, we should not assume the application is working correctly. Although looking at the work of others to get ideas is always a good idea, until we know how to distinguish between good user interface design and bad user interface design, we must be careful. Too many developers make the mistake of imitating the user interface of inadequately designed software.

Using colour appropriately. Colour should be used scarcely in our applications and, if we do use it, we must also use a secondary indicator. The problem is that some of our users may be colour blind and if we are using colour to highlight something on a screen, then we need to do something else to make it stand out if we want these people to notice it. We also should use colours in our application consistently, so we have a common look and feel throughout our application.

Following the contrast rule. If we are going to use colour in our application, we need to ensure that our screens are still readable. The best way to do this is to follow the contrast rule, using dark text on light backgrounds and light text on dark backgrounds. Reading blue text on a white background is easy, but reading blue text on a red background is difficult. The problem is insufficient contrast between blue and red to make it easy to read, whereas there is a lot of contrast between blue and white.

Aligning fields effectively. When a screen has more than one editing field, we want to organize the fields in a way that is both visually appealing and efficient. The best manner to do so is to left-justify edit fields: in other words, make the left-hand side of each edit field line up in a straight line, one over the other. The corresponding labels should be right-justified and placed immediately beside the field. This is a clean and efficient technique to organize the fields on a screen in a right way.

Expecting our users to make mistakes. One can ask how many times have we accidentally deleted some text in one of our files or in the file itself? Were we able to recover from these mistakes or were we forced to redo hours, or even days, of work? The reality is that in majority of cases the reason of errors is a human being. Hence, we should design our user interface to recover from mistakes made by our users.

Justifying data appropriately. For columns of data, common practice is to right-justify integers, decimal align floating-point numbers, and to left-justify strings. UI design should be intuitive. That means, if our users do not know how to use our software, they should be able to determine how to use it by making educated guesses. Even when the guesses are wrong, our system ought to supply reasonable results which our users can easily understand and from which they can easily learn.

Not to create busy user interfaces. Crowded screens are difficult to understand and, hence, are difficult to use. Experimental results show that the overall density of the screen should not exceed 40 percent, whereas local density within groupings should not exceed 62 percent.

Grouping things effectively. Items that are logically connected should be grouped together on the screen to communicate they are connected, whereas items that have not any relations to each other should be separated. It is wise to use white space between collections of items to group them and/or we can put boxes around them to accomplish the same thing.

Taking an evolutionary approach. Techniques such as Agile Model Driven Development (AMDD) are critical to our success in UI modelling process. Some other new techniques like mechanical tools versus digital ones are also recommended [Biddle, Noble 2003; Murphy 2009].

Knowing UI design principles, trends and know-how one can imagine that it is enough to produce very effective user interface. We can say that knowing the principles, tendencies and know-how we satisfy so called necessary condition of well-done design but still we are not equipped with sufficient designing methods. A very good method of effective UI design is UI prototyping which was described below.

5. User interface prototyping characteristics

Prototyping is an excellent means for generating ideas about how a user interface can be designed, and it helps to assess the quality of a solution at an early stage [Ambler 2002]. No project applied a traditional life-cycle approach, which is one of the reasons why most of them were successful. Prototypes are increasingly used as a vehicle for developing and demonstrating visions of innovative systems. As we see in the activity diagram depicted on Figure 2, there are four high-level steps in the UI prototyping process. The first step is to analyze the user interface wants of our users. User interface modelling moves from requirements definition into analysis at the point we decide to evolve all or part of our essential user interface prototype into a traditional UI prototype. This implies converting our hand-drawings, flip--chart paper, and sticky notes into something more substantial. One can carry out this process by making platform decisions, which in effect is an architectural decision. For example, do we intend to deploy our system so it runs in an Internet browser, as an application with a windows-based graphical user interface (GUI), as a cross--platform Java application, or as a mainframe-based set of "green screens?" Different platforms lead to different prototyping tools, for a browser-based application, we call for to use an HTML-development tool, whereas a Java-based application would require a Java development tool and a different approach to the user interface design. So how do we use sticky notes and flip-chart paper to create an essential user interface prototype? Let us start by defining several terms. A major user interface element represents a large-grained item, potentially a screen, HTML page, or report. A minor user interface element represents a small-grained item, widgets such as user input fields, menu items, lists, or static text fields such as labels. When a team is creating a fundamental user interface prototype, it iterates between described below tasks [Ambler 2002].

Exploring system usage. The team will explore system usage by several means. First, it is wise to work together on a whiteboard (Agile Modelling standard element) to discuss ideas, work on initial drawing together, and generally take advantage of the dynamic nature of whiteboards to come to understanding quickly the portion of the system which is discussed. For example, with the university system, we may gather around a whiteboard to make an initial drawing of what a university transcript would contain or what a seminar enrolment submission would contain. Second, as we have seen, essential use case modelling is an effective technique for understanding the behavioural requirements for our system.

Modelling major user interface elements. Major user interface elements, such as potential screens and reports, can be modelled using flip-chart paper. One can say "potential" because whether something is a screen or printed report is a design decision – a university transcript could be implemented as an HTML page of our users view in a browser, as a paper report that is printed and mailed to students, or as an application screen. Each piece of flip-chart paper is given a name, such as *Student Transcript* or *Seminar Enrolment Request*, and has the appropriate minor user interface elements added to it as needed. Pieces of flip-chart paper have several advantages: they can be taped onto a wall; they are good for working in groups because they make it easier for everyone to see and interact; they are large enough so we can put many smaller items such as sticky notes on them; we can draw on them; and they can be stored away between modelling sessions.

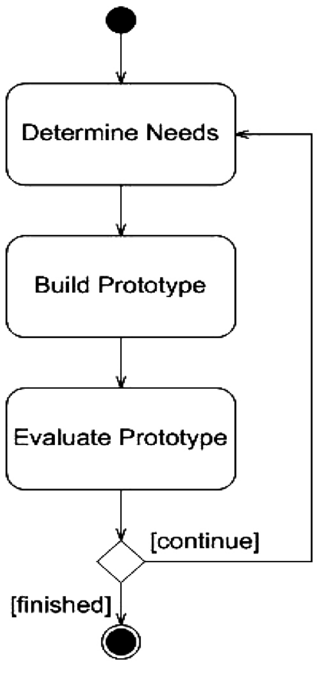

Figure 2. UI prototyping process

Source: [Ambler 2008].

Modelling minor user interface elements. Minor UI elements, such as input fields, lists, and containers (minor UI elements that aggregate other minor UI elements) are modelled using sticky notes. Constantine and Lockwood (2002a; 2002b) suggest using different colour notes for different types of components, for example, bright colours (yellow or red) for active user interface elements such as input fields versus subdued colours (white or tan) for passive interface elements such as containers. One can notice that each sticky note has to be a name that describes its purpose, but not how it is implemented. One can look at the sticky note and immediately shall know how it is used. Different sizes of sticky notes are also used, indicating the relative size of each UI element. The relative order of the UI elements is also indicated by the order of the sticky notes. This ordering may change during design but it should be close enough. Whenever we realize we may need a minor user interface element, we simply take a sticky note, label it appropriately, and place it in the general area on a major user interface element. Sometimes we identify a minor UI element that may not have a major UI element on which to place it. It is not important problem. This is an iterative process, so attempt to identify and consider an appropriate major UI elements is necessary. The very fact that sticky notes do not look like a real GUI widget is a constant visual reminder to our team that we are building an abstract model of the user interface and not the real thing. Each sticky note is, effectively, a placeholder that says we need something there, but we do not know yet the best way to implement it, so for now, we want to keep it open.

6. Experiences showing successful UI implementation

Described in previous sections main requirements for effective user interface design are obligatory even more in the context of Decision Support Systems that software applied by managers is designed mainly for economical profits. So, the money invested by companies in DSS solutions should be given back as quickly as possible. It will be possible only under condition that installed software and its exploitation will satisfy managers.

Company corporations are able to use sophisticated IT tools in institutional and formalized way ensuring permanent training and support for their workers. Second, big companies' agreements with software providers ensure them such conditions which give guarantee to develop their solutions in accordance with client demand. Other DSS solution usage conditions have SMEs. Their market power and financial possibilities are essentially poorer than those of bigger companies, so DSS including user interface should be different from those for extended business units.

Among SMEs special place is occupied by agricultural enterprises where both training possibilities and supporting solutions differ strongly from companies representing other sectors of economy. One can say that DSS and UI solutions working well at farms will be for sure working efficiently also in other firms.

A good experience in this area have creators of American expert systems for farmers support called Ma'ayan [*Agricultural Production...*, 1999]. Seeking appropriate UI solutions designers of this system pay attention to the length of time spent by a farmer working with computers and the most common software used by them up to now. On the basis of the questionnaires one can find that majority of potential ES users used MS Excel spreadsheets. Hence, the ES UI designers concluded that also their model should save the same form (widgets) like in MS Excel sheets. Secondary the UI ES designers applied gradual and modular process of application implementation paying attention to the complexity level and the level of user preparation and experience. So, they have prepared the simplest input part for the least advanced farmers and more advanced modules for more experienced managers. This way, respecting user IT operating habits and their level of data inserting advancement, ES Ma'ayan designers have brought to commercial success and spreading this system in the market.

7. Concluding remarks

The user interface of an application should satisfy both functional (business) and exploitation (technical) requirements. Especially for the users of Decision Support Systems the functionality that an application provides them is the most important, although the way in which it provides that functionality is just as significant. An application that is hard to use will not be used.

Using commonly spread user interface design standards, one should not underestimate the value of user interface design or of usability. Effective UI designers can find methods to work closely with their holders. Active Stakeholder Participation principle and Evolutionary Approach rule are desired in current tendencies of user interface development. Our holders do much of the business-related modelling using inclusive modelling techniques. Furthermore, DSS final users should be closely involved in user interface prototyping efforts as well. Then not only their business or exploitation preferences will be considered but even their psychological needs, too. Some experiences in UI implementation have verified positively an advisability of the premises formulated above.

References

Agricultural Production Forecast 1999-2005, Ministry of Agriculture and Rural Development (the U.S.), Planning Authority, 1999.

Ambler W.S. (2002), *User Interface Prototyping Tips and Techniques*, Copyright 2002 by Scott W. Ambler.

Ambler S.W. (2008), *Maturing Usability Quality in Software, Interaction, and Value*, Springer-Verlag, Berlin.

Biddle R., Noble J. (2003), *From Essential Use Cases to Objects*, Victoria University of Wellington (New Zealand), Ewan Tempero University of Auckland (New Zealand).

Constantine L.L. (1999), *Simplifying User Interfaces by Simplifying Use Cases*, Constantine & Lockwood, Ltd., University of Technology, Sydney.

Constantine L.L., Lockwood L.A.D. (1999), *Software for Use: A Practical Guide to the Essential Models and Methods of Usage-Centered Design*, Addison-Wesley, Reading, MA.

Constantine L.L., Lockwood L.A.D. (2002a), Instructive interaction, *User Experience*, Vol. 1, No. 3, pp. 14-19.

Constantine L.L., Lockwood L.A.D. (2002b), *Process Agility and Software Usability*, Constantine & Lockwood, Ltd., University of Technology, Sydney.

Glushko R.J., Tabas L. (2009), Designing service systems by bridging the “front stage” and “back stage”, *Information Systems and eBusiness Management*, Vol. 7, No. 4.

Murphy N. (2009), Mechanical vs. digital: a GUI isn’t always the answer – User interface design is not always an either/or decision, *Embedded Systems Design*, Vol. 22, No. 1.

Takeuchi Y., Sugimoto M. (2009), A user-adaptive city guide system with an unobtrusive navigation interface, *Personal and Ubiquitous Computing*, Vol. 13, No. 2.