SUN'S APPROACHES TO AN INTERACTIVE DESIGN

Ing. Josef Pavlíček Ph.D,
Sun Microsystems, Evropská 33e,
Praha 6 Dejvice 165 00.
Josef.Pavlicek@sun.com

ABSTRACT

To develop user friendly software is the goal for a lot of software companies. By the term "user friendly" we mean smart software, which has a high focus on usability and ergonomics. The software has to have a high level of Quality. Exists many approaches to achieve the desired quality. This article talks about the approaches the Interaction designers use in the Software eXperience Design Team of Sun Microsystems. The author evaluates known approaches and suggests a suitable way to build and develop the NetBeans Java development tool with a high focus on integration, usability and ergonomics.

KEYWORDS
User Interface, User Interaction, Design, Use Case, Scenario, Prototype, Quality

Introduction

I have been working for Sun Microsystems for three years. I am a member of the Software eXperinece Design Team. The goal of this team is to do the UI design as well as possible. If we want to be good in this subject, we have to study a lot of materials. In addition to we have to follow the International standards like ISO/IEC 25062, ISO/IEC 9000. The discussion in the scientific conferences is necessary to. And that is the reason why I wrote this paper.

User Interface and Interaction design

The UI (User Interface) is the gate into the computer world. If the gate is wrong and the user has a problems opening it, he/she cannot enter. The user feels uncomfortable and he/she can get angry. That is the first typical mistake in UI. Alan Cooper says [Alan Cooper] "the user cannot feel stupid". If we want to make good software, we have to keep this note in mind.

Interaction versus Interface design

Alan Cooper also says [Coo99a] "I prefer the term Interaction design over the term interface design because interface suggests that you have code over here, people over there and an interface in between that passes messages between them". This is a good approach. The UI is crucial for the first user experience with the software. The user evaluates the

software according to his/her first experience. If the software looks smart the user feels well and wants to try to use it.

But UI is not everything. UI can help the user to start to use the software. If the software is intuitive and the user doesn't have problems using it, everything is OK. If not, the user probably has "Usability Problems". The software has to be intuitive and user friendly. This behaviour we can call Usability.

If we join the User Interface and Usability together we can speak about User Interaction design.

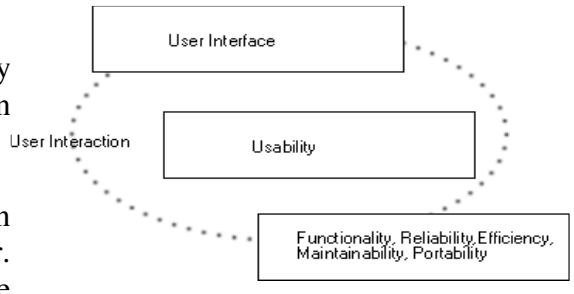

Illustration 1: User Interaction

Cooper says [Coo99a] "Interface design only tells how to dress up an existing behaviour. For example, in a data reporting tool, interface design would eliminate unnecessary border and other visual clutter from a table of figures, color code important points, provide rich visual feedback when the user clicks on data elements, and so on. That is better than nothing, but far from sufficient".

Evidently we have to consider the "behavioural design" too. This kind of design tells us how the elements of the software should act and communicate. Cooper says [Coo99a] "We can go deeper into what we call "conceptual design", which considers what is valuable for the users in the first place. Conceptual design might tell you that examining a table of figures is only an incidental task – the users' real goal is spotting trends, meaning that you don't want to create a reporting tool at all, but a trend–spotting tool. To deliver both power and pleasure to users, you need to think first Conceptually, then in terms of Behaviour, and last in terms of Interface".

Software User Interface, Usability and Interaction specification

If we want to realize new functionality for the NetBeans Java programming tool, the Interaction Designers have to develop Software User Interface, Usability and Interaction specification. We call it "UI Specification" and the reader can read a lot of specifications on Sun's web page http://ui.netbeans.org .

This UI specification has three main parts.

- UI specification Use Cases
- UI specification Scenarios
- new Software specification

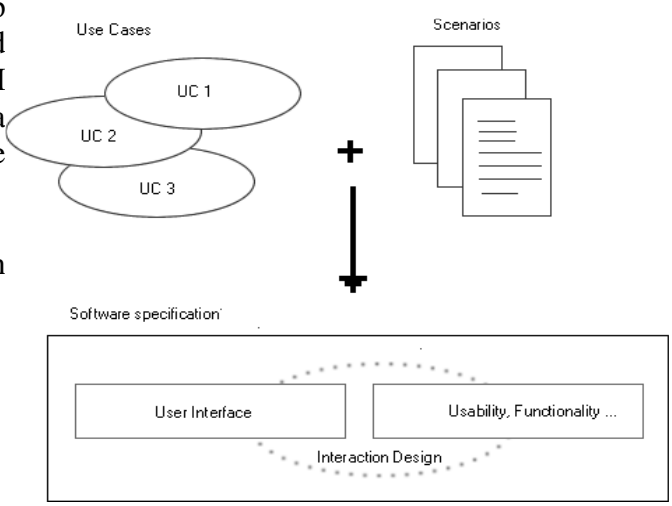

Illustration 2: Software UI specification

UI specification Use Cases

Use Cases are very informative for the Interaction design. We are using them because they show the users' main work flow. On the basis of the Use Cases we are designing the functional elements of the UI. The Use Cases tell us which functionality has to be accessible from the first window. Besides that they help us to determine the appropriate level of complexity for the software.

Each part of the software product has a different level of complexity. The welcome screen has a very low level. It's used for new users, who don't know the software and have to feel comfortable. The Database browser has a high level of software complexity. It's designed for advanced users.

Also, we can use the number of Use Cases to predict software complexity. For this prediction we are using artificial intelligence – the neural network PETRA.

UI specification Scenarios

Scenarios tell us how the system will answer the user's needs. Scenarios are usually an inverted view of Use Cases. Scenarios show us the first view of functionality of the software. They are also very useful for checking software functionality. Use Cases tell us what the user wants. Scenarios tell us how the software allows it.

Scenarios can also be used for software complexity prediction.

New Software specification

New software specification is the biggest part of UI specification. User Interface design depends primarily on this part. Use Cases and Scenarios tell us what the user wants and how the system allows it – we call this Usability. New software specification tells us how the software has to look. This part also describes in detail the software behaviour. It represents the main part of Interaction design.

Note:

We cannot say “The Interaction design depends on the Software specification part only”. Of course not. UI specification make sense only if is it complete (with Use Cases, Scenarios, Software Specification). This includes Usability, User Interface, User Interaction and Reliability, Functionality and other Software Quality attributes [Ras00a].

The feature software usability is dependent on this part. The number of usability problems depends on the quality of UI design in the UI specification. I have gained this knowledge during my human interface design work for SUN.

I can say (in general):

- The software complexity depends on the number of Use Cases.
- The software Usability depends on the quality of User Interface design placed in the New Software Design chapter in the UI specification.
- User centred Interaction design joins both.

Analysis of the current situation

Number of Use Cases

The method Use Case point by Gustav Karner [Kar93] uses the Use Cases as the basis of software functionality. Karner designed the Use Case point method according to this idea. This approach is not new. The Function Point method used a similar approach in the 1980ties. Function point is designed for structured programming. This approach is old now. Today we are using Object oriented programming. Generally – Use Case point is similar to Function point but for Object oriented programming.

Unfortunately Karners' approach doesn't make sense for software prediction from the point of view of the theory of measurement [Van99],[Lac01]. On the basis of this fact I designed an alternative method. In this method are prediction steps which are based only on subjective classification given to solve the problems with the artificial intelligence - neural networks.

The quality of User Interface design

We have a lot of possibilities how to do the User Interface design. We can use three [Pav05] elementary approaches. Each approach has some pluses and a few minuses.

- ASCI design,
- Graphical design,
- Functional prototype.

ASCI design

- Pluses
  - is easy for developing,
  - shows position of buttons and functional elements (in comparison with text description).
- Minuses
  - doesn't shows colours (it is useful because colours can give the user a feedback. For example, inactive icon is grey),
  - the developer has to think about the colour form and it takes him/her the time for programming,
  - we don't see the graphical problems (wrong colours, state of functional elements etc.),
  - doesn't show the Interaction design.

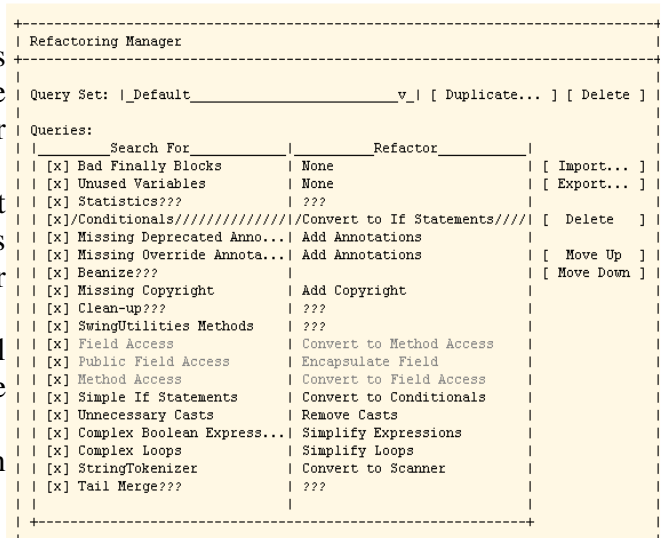

Illustration 3: ASCI design

Summary:

We can say “the saved time during the software UI specification, we have to loose during the implementation process with the developers”. This kind of the design is useful for an easy design, where is a low focus on the usability and interaction. For example the Dialogues like “ The field NAME is empty, do you want to close the dialogue?”.

Graphical design

- Pluses
  - can be finished in the short time,
  - shows position of buttons and functional elements,
  - shows colours and the UI,
  - shows the developer, how the software has to look.
- Minuses
  - Takes more time than ASCI design,
  - doesn't show the Interaction design.

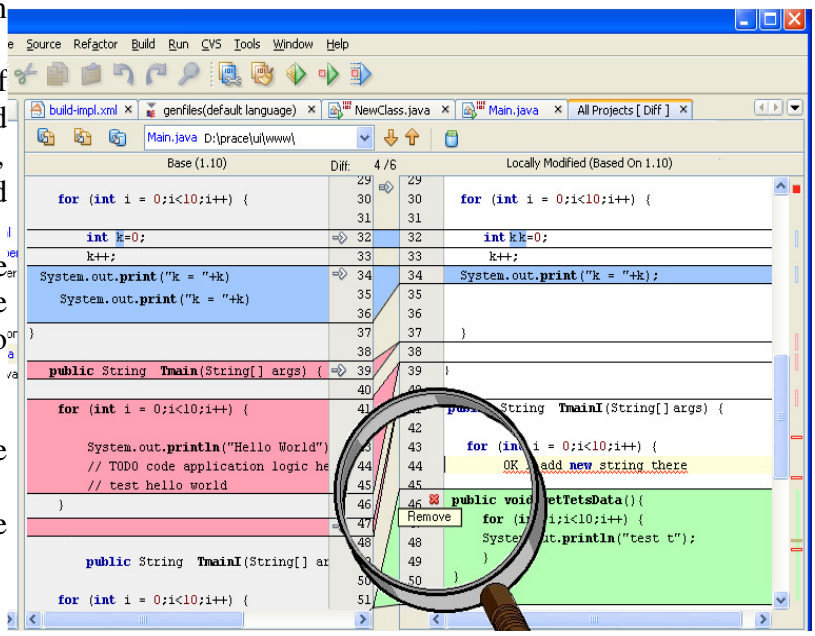

Illustration 4: Graphical design

Summary:

I can say: “this type of design is the best in general”. We can use it in the wide area of software tools. It shows all details in the high quality. We can imagine the interaction design much better than in the ASCI design. It is much cheaper than a functional prototype.

Functional prototype

We can use each programming language, or some special language for modelling), for working prototype.

- Pluses
  - shows position of buttons and functional elements,
  - shows colours and User Interface design,
  - shows the developer, how the software has to look,
  - shows Interaction design
- Minuses
  - is expensive,
  - takes a lot of time
  - is hard to change it, sometimes the designer is solving the problems during programming the prototype and is not solving the Interaction design problems.

Summary:

The working prototype is useful if we need to test interaction design primarily. In the case when we are not able to imagine it. This prototype is the best for testing. It's expensive and takes a lot of time.

Conclusions

It is useful to do the Software UI specification before we start to develop some software. We can save a lot of time, which we will need for tuning, if we will build the software without UI specification. In general to have some Software UI specification is useful for planning and managing the software develop process. On the basis of Use Cases we can approximate the software complexity. This is very important for manager decision type: realize the software, how much it will cost and how long it will take to build it.

For the purpose of the software complexity approximation I developed a special neural network system. This system has the name PETRA and I will talk and show it during my presentation.

References:

- [Coo99a] Cooper A., The intimates are running the asylum, pp.0-672-31649-8.
- [Ras00a] Raskin J., The Humane Interface,pp. 0-201-37937-6.
- [Pav05] Pavlíček J., Objekty 2005, pp.80-248-0595-2.
- [Van99] Vaníček J., Měření a hodnocení jakosti informačních systémů, pp 80-213-1206-8.
- [Lac01] Lacko B., Aplikace metody RIPRAN v softwarovém inženýrství, Sborník celostátní konference Tvorba Software Ostrava.
- [Kar93] Karner G., 1993, "Metrics for Objectory". Diploma thesis, University of Linköping, Sweden. No. LiTHIDA - Ex-9344:21