Incorporating Usability into an Object Oriented Development Process

Xavier Ferré
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo
28660 - Boadilla del Monte
Spain
xavier@fi.upm.es

Abstract

Usability has become a key issue for the perceived quality of a software product. In order to attain a high level of usability in the software product we need to apply the set of techniques belonging to Usability Engineering. In most software development organizations where these techniques are applied, they are not integrated with the Software Engineering development process. Object oriented methods offer a bridge between Software Engineering and Usability Engineering, by means of use cases. Giving use cases a supplementary user-centered focus we can make our way through object-oriented software development combined with usability techniques. From the numerous object-oriented approaches we have chosen Larman's method because it gives priority to the interaction design over the design to the internal part of the system to develop. A generally applicable joint Usability Engineering – Software Engineering development process has been defined by modifying Larman's method to integrate usability techniques where appropriate. The proposed process gives advice on the usability techniques to be used in every phase of such joint development process.

1 Introduction

More and more software development firms are beginning to take usability seriously [1]. They have identified the importance of usability, and they are applying usability techniques to improve the usability level of their products. Some firms have their own usability group, for example there is a usability group in IBM formed by 407 people, and Microsoft has one formed by 117 people [2]. Alternatively, some other companies hire a usability consulting firm (a list of more than 70 usability consultants can be found in [3]).

Despite the growing number of software firms including usability in their agenda, it is not solved yet how to integrate usability techniques into Software Engineering development processes [4][5]. Such an integration faces serious problems due to the multidisciplinary essence of Usability Engineering. While usability theory and practice foundations come from the disciplines of psychology, sociology, industrial design, graphic design, and so forth; software engineers have a very different approach, a typical engineering approach. Both fields speak a different language and they approach software systems from a different point of view.

Software Engineering has traditionally constructed software systems with development focused on internals, on processing logic and data organization [6]. Consequently, software quality has been identified with issues like efficiency, reliability, reusability or modularity. These are aspects of the system that the user is scarcely aware of. In contrast, the interaction with the user has been sometimes left as a secondary issue [5]. Despite the stated aim of building a software system that satisfies the user, after establishing a set of specifications the user is forgotten until the first release of the software product. Usability is sometimes wrongly identified to be a graphical user interface issue, which can be addressed after the main part of the functionality has already been developed [7]. When developers perceive usability in this way, they tend to think that after having constructed the "important" part of the system (the internal part), usability specialists can add a nice user interface in order to make the product usable. This attitude leads to systems where usability problems are very costly to fix when identified.

Usability practitioners, on the other hand, have focused on the user and the way he or she interacts with the system. They employ a set of techniques to better canalize the creative activity of interaction design, and to evaluate its products with real users. Focusing on the creative nature of interaction design, they haven't paid attention to issues central to Software Engineering such as how to make their

way of developing systems repeatable and structured, or how to estimate and plan their procedures.
Due to the increasing perception of usability as strategic for software development businesses, an increasing number of software development organizations are pursuing the aim of integrating usability practices into their Software Engineering processes. One of the leading journals for software practitioners, IEEE Software, has dedicated its January/February 2001 issue to the role of Usability Engineering in software development. Some proposals for integration ([8][9]) present ad-hoc solutions which have been created for particular software development organizations. They lack a generic approach to be applied to organizations with different characteristics.
In this paper we propose a joint process where Usability Engineering is embedded into the development. The approach we propose is generally applicable in the development of interactive software systems.
Usability Engineering groups a set of techniques aimed at getting a high usability level in the software product. These techniques can be classified as follows: early analysis techniques to gather user needs, domain knowledge and system concept inception; design techniques to guide design creativity in order to get an appropriate interaction design; and usability evaluation techniques, from which usability testing is the central technique.
Some usability techniques have a better applicability from a Software Engineering point of view, and we have chosen them as best candidates for integration in a Software Engineering development process.
Among software development techniques, use-case driven approaches are the closest to a usability perspective. They are considered in the next section. Usability-related activities are accommodated into a use-case driven method and the resulting process is described in section 3. The particular usability techniques chosen to be applied in each phase or activity are detailed in section 4. Finally, conclusions and future directions are presented in section 5.

2 Use Cases and Usability

Traditionally, software development has taken a variety of forms, with minor or big differences, but with a common focus: trying to build a system beginning from the inner part of it, the internal structure. A different approach stands out in object-oriented software development: use-case driven development.
Use cases are a user-centered technique in its conception, so they can fit well with the Usability Engineering approach. Therefore, use cases seem to be the best starting point for the integration of Software Engineering and Usability Engineering.
Nevertheless, for a real user-centric focus it is crucial that use cases are not converted into technical specifications in the sense of a production line. When use cases are taken away from the user sphere they loose their main benefit from a usability perspective. That is not to say that technical specifications will not be based in the use cases defined, but they are not the result of some direct use case transformation. Technical specifications reflect the development team decisions, which are taken according to the user needs stated in the use cases.
Today's possibly most popular object-oriented method, the Unified Process [10], is labeled as use-case driven. However, analyzing the role of use cases in the process, it can be observed that system architecture plays a much important role than the use-case model. Use cases in the Unified Process play a role in cycle planning, but they are treated as precursors of system classes and collaborations as soon as a specific development cycle begins. This approach is too close to low-level design, and it can prevent the development team from adopting a proper user-centered focus.
The usage of use cases in the early phases of software development is a great advance in usability terms compared with previous approaches, but it is still not enough. To make use cases more useful (from a usability point of view) we consider that they should include the concept of task used in Usability Engineering, specifically in Task Analysis techniques [11].
Larman proposes an iterative, incremental and use-case driven method for object-oriented software development [12]. It is focused on practical issues that a developer faces when developing real systems, instead of taking an academic approach. It is specially interesting for our purpose the division he makes between analysis and design. The analysis phase, while including traditional analysis activities like the creation of a Conceptual Model, it also addresses the design of the interaction of the system with the user, by viewing the system as a black box which communicates with the user, and specifying such communication by means of system sequence diagrams and system operation contracts. The details of the internals are left for the design phase. This approach is well suited for Usability Engineering, as it allows for an interaction design previous to the low-level design that will give form to the inner part of the system.
Unfortunately, Larman´s method has some flaws from a usability perspective that make it alien to a Usability Engineering process scheme. On one hand, the Conceptual Model (supposedly an analysis construct) is exploited as a rudimentary database, causing the developer to focus more on database modeling than on gathering user-domain knowledge. On the other hand, an event-driven interaction mode is implicit in its approach, causing the method not to be flexible enough to embrace a broad variety of software systems.

We consider that the initial approach in the structure of Larman’s method is valid from a usability point of view. This structure, when supplemented with the appropriate Usability Engineering activities, can yield a useful software development process from both a Software Engineering and a Usability Engineering point of view.

3 Joint Development Process

We have taken as basis for our joint development process the structure of Larman’s method, adapting some of the activities to a usability-focused interpretation. We have also adapted his terminology to make explicit the user-centered philosophy. We call *External Design* to the analysis phase in Larman’s method, and *Internal Design* to his design phase. External Design deals with the design of the communication between the outside world and the system, while Internal Design is concerned with the design of the internal structures to give service to that previously designed interaction.

The structure of the process is shown in figure 1, where activities at the same horizontal level can be performed in parallel, and activities which are placed higher inside a phase are performed before than activities placed lower. The process is formed by two preliminary phases and five phases inside the iterative cycles. The preliminary phases are as follows:

- **Early Analysis**: Before taking any decision about the future system, it is defined what the system is supposed to be in general terms (*System Scope Definition*), and what the intended users will be and their characteristics (*User Analysis*). Later on, a more detailed definition of what the system is going to offer to the user is specified (*Task Analysis - Use Case Definition*). To ensure that the set of tasks created suit the user needs, a *Usability Evaluation* is performed. A traditional set of requirements is then specified with the addition of particular usability requirements (*Requirements Specification*).
- **Cycle Planning**: Collaboratively with users and the customer use cases are assigned to development cycles (*Use Case Assignment to Cycles*). A previous activity of *Use Case Risk Analysis* may be optionally performed to serve as guide for decisions.

Once use cases are assigned to cycles, the first cycle begins. Each cycle is divided into the following phases:

- **External Design**: The tasks identified in the Task Analysis are defined more precisely (*Detailed Task Analysis*) in parallel with the definition of interaction objects and their behavior (*Interaction Conceptual Design*). The resulting interaction scheme is tested with users (*Usability Evaluation*).
- **Internal Design**: The classes to support the interaction designed in the previous phase are specified (*OO Class Diagrams Design*), along with their behavior (*OO Interaction Diagrams Design*). The graphical user interface that gives shape to the interaction design is built with the contribution of experts in graphic design (*Interaction Visual Design*).
- **Construction**: The structures designed in the previous phase are taken to a specific programming language, and converted into a working system.
- **Testing**: The system built is subject to tests to ensure that complies with the requirements (*Testing*). In particular, it is tested with users for compliance with usability requirements (*Usability Evaluation*).
- **Plan Refinement**: The results of the usability evaluation will be taken as basis for a possible re-planning of the next cycles, either to address the usability problems, or to add new tasks identified through user participation. After this phase a new cycle begins.

After the system is deployed, maintenance begins and it can be considered as additional cycles in the development. Maintenance cycles are driven by customer or user requests and some of the activities showed in figure 1 may be lighter than in normal development cycles.

Table 1 classifies the activities of the joint development process in the ones belonging to Larman's method and the ones originated from Usability Engineering. Note that there are some activities that combine both sources, such as *System Scope Definition* or *Requirements Specification*.

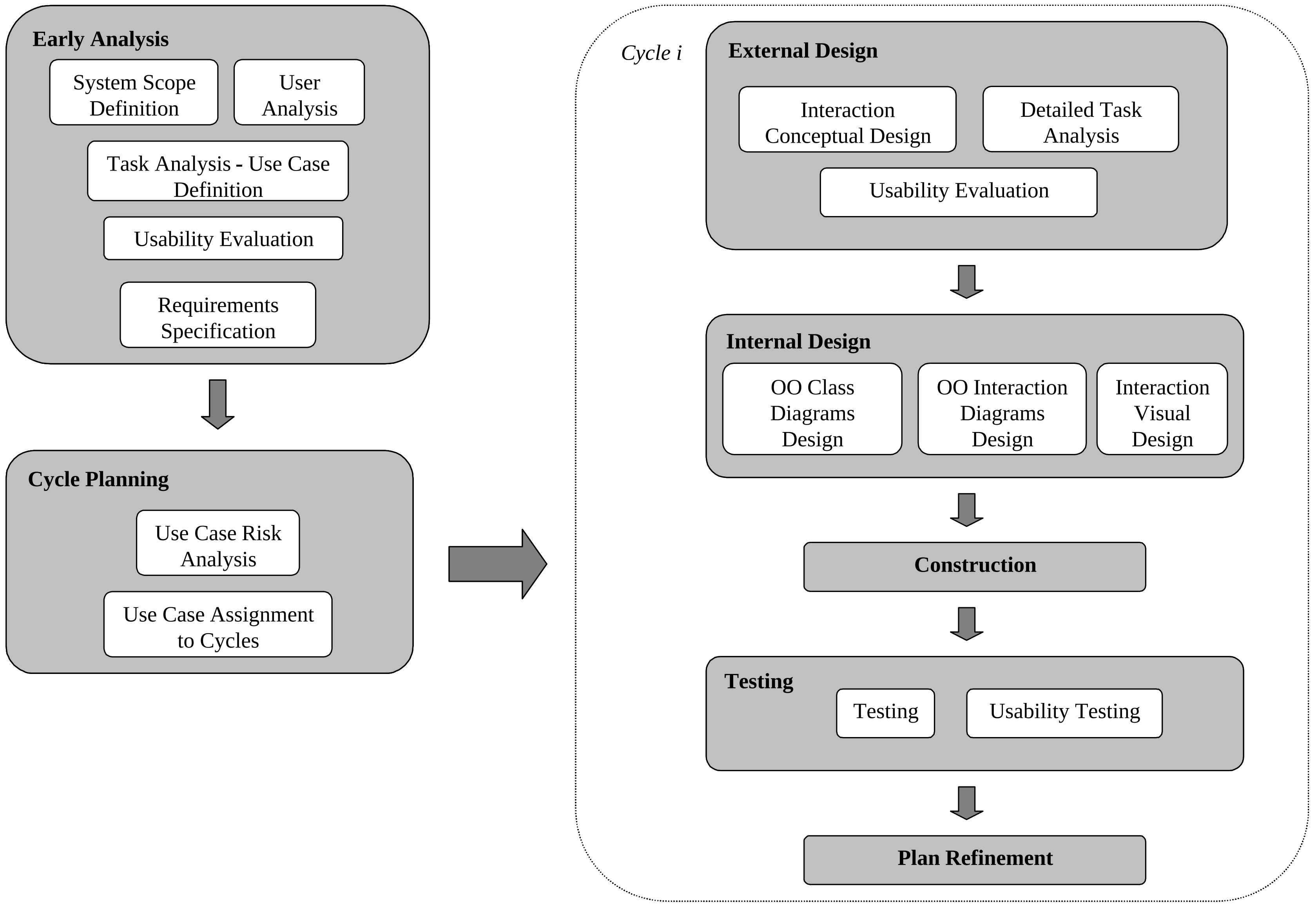

Figure 1 Joint Development Process

Joint Development Process	**Larman's Method**	**Usability Engineering**
Early Analysis	System Scope Definition	System Scope Definition
		User Analysis
	Use Case Definition / Task Analysis	Use Case Definition / Task Analysis
		Usability Evaluation
	Requirements Specification	Requirements Specification
Cycle Planning	Use Case Risk Analysis	Use Case Risk Analysis
	Use Case Assignment to Cycles	
External Design		Interaction Conceptual Design
	Detailed Task Analysis	Detailed Task Analysis
		Usability Evaluation
Internal Design	OO Class Diagrams Design	
	OO Interaction Diagrams Design	
		Interaction Visual Design
Construction	Construction	Construction
Testing	Testing	Usability Testing
Plan Ref inement	Plan Refinement	Plan Refinement

Table 1 Activities of the joint development process classified according to their belonging to either Larman's method or Usability Engineering

4 Usability Techniques in the Joint Development Process

Usability literature offers numerous techniques to be used for different project characteristics and for different usability purposes. We have chosen the ones more valuable for a broad variety of systems, considering specially their applicability from a software engineer point of view. Most of them can be applied with moderate usability training.

In the following sections we present the usability techniques we recommend. They are structured according to the phase of the joint development process where they can be applied. Table 2 summarizes this information.

4.1 Early Analysis

The activity of *System Scope Definition* will be highly dependent on the kind of system to be built. For traditional systems it can be just a short description of what the system is intended to do, but for innovative systems the set of techniques known as Holistic Design [13] can yield an adequate definition of what the system will do and how it will be like. This kind of definition can be called "Product Vision" [8]. For systems built from scratch, but not necessarily so innovative, Needs Analysis [5] would suffice.

User Analysis can be performed for in-site developments or tailored systems by means of site visits, being a variant of them the Contextual Design approach [14]. For commercial products with a broad range of users Market Analysis is more appropriate [15]. To help making user-centered the whole development process, the technique of "personas" can be employed [16].

Task Analysis should be performed considering the information gathered by the previous activities, and using as notation the Use Case Model of Larman's method. Use cases should be given the focus of tasks [11] as mentioned above.

Usability Evaluation can be performed on the results of Task Analysis, representing their functionality by means of paper prototypes (sometimes called paper mock-ups [7]), in addition to the textual descriptions of use cases in expanded format[12].

Requirements Specification will include a set of operationally-defined Usability Specifications [5], related to the tasks identified in *Task Analysis*.

4.2 Cycle Planning

In the activity of *Use Case Risk Analysis* the use cases are studied to decide the ones with a higher risk for the success of the project. The riskiest will be addressed first. As a result of *Usability Evaluation* in the previous phase, the set of use cases that support the core needs of the user are identified. Not addressing these use cases should be considered risky.

The decision of which use cases should be tackled and in which order (*Use Case Assignment to Cycles*) should be done collaboratively between developers and users. For that purpose some techniques to involve the user can be applied, like Participative Design[13].

4.3 External Design

Interaction Conceptual Design is a genuine creative activity, for which we cannot find a process whose application guarantees usable designs. Nevertheless, usability experts have gathered valuable Design Guidelines (like the ones in [6], [7] and [17]), which are the basic guidance for newcomers to the field. Most interaction notations can be considered too formal to software developers, but Constantine [6] presents some useful models to be applied in interaction representation.

Detailed Task Analysis should specify in detail each use case, all the interaction between the user and the system. Techniques described above for *Task Analysis* can apply. The differentiation between *Task Analysis* and *Detailed Task Analysis* is taken from the high level and expanded formats for use cases in Larman's method.

Usability Evaluation can be performed either by Heuristic Evaluation[7], by Collaborative Usability Inspection[6], or by informal usability testing with users, but with a focus on Formative Evaluation[5].

4.4 Internal Design

This phase is mostly devoted to traditional Software Engineering activities, being *Visual Interaction Design* the only Usability Engineering activity. Design tips for this activity can be found in [17] and [18].

4.5 Construction

No specific usability techniques can be applied in this phase.

4.6 Testing

Usability Testing is performed at a laboratory with real users [19] in order to assess the fulfillment of the usability specifications defined in the *Requirements Specification*.

For large projects, customers can plan the execution of Acceptance Tests [17], which can include usability testing of specific usability criteria.

4.7 Plan Refinement

Usability evaluation results should be considered along with traditional quality assurance results to decide which use cases should be refined and which ones are considered to be complete. Specially important for this phase is the identification through usability evaluation of new functionality which was not previously defined in Early Analysis.

4.8 Maintenance Cycles

After system deployment, maintenance can be performed with a usability perspective as well.

Usability evaluation can be enriched with User-Performance Data Logging [7][17], User Satisfaction Questionnaires [17], and other kinds of user feedback such as beta testing or trouble reporting[17].

When users are organized in Focus Groups [7][17], they can provide feedback information more valuable than individual interviews, as Focus Groups are more representative of the user population.

Table 2 Usability Techniques to be applied in each Phase of the Joint Development Process

Joint Development Process	**Usability Techniques**	**Bibliographical Source**
Early Analysis	Holistic Design	[13]
	Product Vision	[8]
	Needs Analysis	[5]
	Contextual Design	[14]
	Market Analyses	[15]
	"personas"	[16]
	Task Analysis	[11]
	Paper Prototypes	[7]
	Usability Specifications	[5]
Cycle Planning	Participative Design	[13]
External Design	Design Guidelines	[6] [7] [17]
	Heuristic Evaluation	[7]
	Collaborative Usability Inspection	[6]
	Formative Evaluation	[5]
Internal Design	Design tips for Visual Interaction Design	[17] [18]
Testing	Usability Testing	[19]
	Acceptance Tests	[17]

5 Conclusions and Future Directions

The need for integration of usability techniques into the development process has been discussed. Usability Engineering does not offer a well defined process. Additionally, current Software Engineering processes don't address usability issues, therefore producing unusable software products. Some proposals are beginning to emerge [8][9], but they are not generally applicable studies, instead they address specific cases in particular development organizations. A generic process where usability activities are integrated has been proposed, including the specific usability techniques to be applied in each phase according to the characteristics of the project.

Our experience has shown us that developers with a Software Engineering background regard usability activities as a nuisance that increases project time. We have proposed in the joint development process a parallelization of Usability Engineering activities with Software Engineering ones where possible. We have also made lighter some activities in Larman's method to avoid having a bulky joint development process.

Software engineers require a high degree of flexibility when adapting to the joint process, as the introduction of usability concepts can be hard to fit together with a traditional Software Engineering background. Organizational and cultural change needs to be managed carefully.

For a better applicability of the joint process the relationship between the usability-related activities and the object-oriented ones needs to be defined in more detail. New ways to parallelize the different activities must be pursued as well, since development time is a critical issue for software developers. The application of the process to a variety of projects is needed to define a detailed guide of how to tailor the joint development process to specific projects and software development organizations.

References

[1] L. Trenner, J. Bawa. *The Politics of Usability*. Springer, 1998.

[2] K. Halskov Madsen. *The Diversity of Usability Practice.*. Communicacions of the ACM, vol. 42 no. 5. 1999.

[3] *HCI-SITES: CONSULTANTS: Consultants with a focus on HCI.* http://www.acm.org/sigchi/hci-sites/CONSULTANTS.html

[4] X. Ferré, N. Juristo, H. Windl, L. Constantine. *Usability Basics for Software Developers*. IEEE Software, vol.18, no.1, January/February 2001. pp. 22-29.

[5] D. Hix, H.R. Hartson. *Developing User Interfaces: Ensuring Usability Through Product and Process*. John Wiley and Sons, 1993.

[6] L. L. Constantine, L. A. D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley, 1999.

[7] J. Nielsen. *Usability Engineering*. AP Professional, 1993.

[8] J. Anderson, F.Fleek, K. Garrity, F. Drake. *Integrating Usability Techniques into Software Development*. IEEE Software, vol.18, no.1. January/February 2001. pp. 46-53.

[9] K. Radle, S. Young. *Partnering Usability with Development: How Three Organizations Succeeded*. IEEE Software, vol.18, no.1, January/February 2001. pp. 38-45.

[10] I. Jacobson, G. Booch, J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999.

[11] J.T. Hackos, J.C. Redish.*User and Task Analysis for Interface Design.* John Wiley & Sons, 1998.

[12] C. Larman. *Applying UML and Patterns*. Prentice Hall, New Jersey, 1998.

[13] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey. *Human-Computer Interaction*. Addison Wesley, 1994.

[14] H. Beyer, K. Holtzblatt. *Contextual Design: A Customer-Centered Approach to Systems Design.* Morgan Kaufmann Publishers, 1997.

[15] A. M. Wychansky, C. N. Abernethy, M. E. Kotsonis; D. C. Antonelli; P. P. Mitchell. *Selling Ease of use: Human Factors Partnerships with Marketing.* Proc. Human Factors Society 32nd Annual Meeting, 1988, pp. 598-602.

[16] A. Cooper, P. Saffo. *The inmates are running the asylum.* Sams, 1999.

[17] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, Reading, MA, 1998.

[18] K. Mullet, D. Sano. *Designing Visual Interfaces: Communication Oriented Techniques.* Prentice Hall, 1994.

[19] J. S. Dumas, J. C. Redish. *A Practical Guide to Usability Testing*. Intellect, 1999.