

A Usability and Accessibility Oriented Development Process*

María Dolores Lozano, Francisco Montero, Pascual González

Computer Science Department
Laboratory of User Interaction and Software Engineering
Computer Science Research Institute of Albacete
University of Castilla-La Mancha
Albacete -SPAIN-
[mlozano, fmontero, pgonzalez]@info-ab.uclm.es

Abstract. Usability has become a critical quality factor of software systems. Although it was defined as one attribute of software quality in the first quality decompositions at the end of the 70's, it has not been really considered and it has been historically relegated in relation to other quality factors such as efficiency, reliability, safety and so on. For this reason, most software development methodologies do not include mechanisms to take into account this critical factor in the software process. Fortunately, nowadays the point of view is changing and not only the original concept of usability is getting more and more important but also new concepts are emerging such as universal usability, accessibility and so on. In this paper we aim to analyze this problem and give a step forward proposing a usability and accessibility-oriented methodological framework which takes into account these criteria from the very beginning of the software development process.

1. INTRODUCTION

Universal access has emerged as a new critical quality target due to the wide spread of the new information technologies that are changing the current society. The appearance of these new technologies might be considered as the origin of a new Era, called the Information Society. The most difficult goal is to achieve an Information Society “for all”, with an easy access for anyone, anytime and anywhere [Ste01a]. To attain this, it is very important that software developers take these matters into account when designing these new interactive systems.

As Stephanidis states in [Ste01b], in the context of human-computer interaction Universal Access introduces a new perspective that attempts to accommodate a very wide range of human abilities, skills, requirements and preferences in the design of computer-based products. Even more, “access” is not enough to ensure successful usage [Shn00]. In this sense a new concept, Universal Usability, has emerged as an important issue and a topic for computing research.

The concept of usability is not a new idea, as a matter of fact, from the end of the 70's, usability was already considered one more attribute of software quality, but historically its importance has been relegated in relation to other quality factors such as efficiency, reliability, safety and so on. For this reason, most software development methodologies do not include mechanisms to take into account this critical factor in the software process.

Fortunately, nowadays the point of view is changing and not only the original concept of usability is getting more and more important but also new concepts, such as accessibility which help to attain the goal of an Information Society for all, are emerging.

In this paper we aim to analyze this problem and give a step forward proposing a usability and accessibility-oriented methodological framework, which takes into account these criteria from the very beginning of the software development process.

* This work is supported in part by the Spanish PBC-03-003 grant.

The structure of the paper is the following. In section 2 we analyze the problematic of integrating usability and accessibility into the software development process, we analyze some approaches and discuss their deficiencies. In section 3 we present a usability and accessibility-oriented methodological framework which aims to give a step forward in this field and contribute some solutions to the problem of incorporating usability and accessibility criteria within the software process. We finish in section 4 with some concluding remarks.

2. USABILITY AND ACCESIBILITY IN SOFTWARE DEVELOPMENT PROCESS

Computer systems built with usability criteria have several benefits such as productivity improvement, cost and learning time reduction and a notable increment of final user autonomy.

Landauer [Lan95] stated that 80% of maintenance costs – what represents the 80% of software development total costs – are due to user-system interaction problems and not to technical problems. Besides, he indicates that 64% of these costs are related with usability problems.

This situation highlights the importance of usability and justifies the need to incorporate usability criteria before, during and after software systems development.

Some proposals have arisen trying to contribute new ideas in this field. Deborah Mayhew [May99] proposes a Usability Engineering Lifecycle in which she defines a lifecycle structured in three stages: Requirements Analysis, Design/Testing/Development and Installation. This development process follows the waterfall lifecycle. It cannot be considered as an iterative approach as it starts with Requirements analysis, then goes to the second stage of Design/Testing/Development and ends with the installation but it does not return to requirements analysis unless there is lack of functionality to be developed. This methodology is focused on user interface development. It is supposed to be complemented with the OOSE Methodology [Jac93] to cover the rest of the system development.

Constantine and Lockwood [Con99] propose a Usage-centered Design which can be considered as a method that describes the techniques to use more than a development process as defined in Software Engineering. This usage-centered design consists on a set of usability-oriented coordinated activities. They include Task modeling, Interface content modeling, etc. The most important feature they introduce is the concept of essential use cases. This is a variation of the original definition of use cases and it is the foundation of Usage-Centered Design.

Costabile [Cos01] faces the usability problem trying to integrate user-centered methods into the software process. She focuses on three basic concepts:

- Users and Task Analysis.
- Iterative system design and implementation by using prototypes of increasing complexities.
- Evaluation of prototypes with users.

The main idea of her proposal starts with the waterfall lifecycle. She proposes a modification of this lifecycle including two new usability oriented activities. The first one is related to users and task analysis and scenarios and user interface specifications; the second one is related to prototyping and tests. She defines the possibility of returning to a previous stage from any stage within the lifecycle. These backwards returns within the waterfall together with the prototyping activity and tests which occur twice in the lifecycle are considered the iterative nature of her approach.

With the aim to achieve universal access, Stephanidis [Ste01b] proposes the Unified User Interface Development Methodology. This approach seeks to convey a new perspective on the development of user interfaces, and provide a principled and systematic approach towards coping with diversity in the target user groups, tasks, and environments of use, through the use of automatic

user interface adaptation. It aims to identify and enumerate possible design alternatives suitable for different users and contexts of use; identify abstractions and fuse alternatives into abstract design patterns; and rationalize the design space.

Unfortunately, a standard definition of how usability has to be included in the software lifecycle does not exist. The integration of accessibility criteria within the software process is even more unusual. Usually, accessibility is validated at the end of the software development process using the existing validation tools such as TAW, Bobby, APROMPT, and so on which are based on the evaluation of the accessibility guidelines defined by the WAI of the W3C Consortium. It is difficult to find proposals trying to integrate accessibility criteria within the software development process. This paper tries to give a step forward in this sense.

3. IDEAS: A USABILITY-ORIENTED METHODOLOGICAL FRAMEWORK

As stated in the previous section, we can conclude that there are still important lacks in current software development approaches regarding the integration of usability and accessibility within the lifecycle. Some of these lacks can be summarized in lack of real iterativeness, lack of user interface modelling languages, lack of formalization and definition of the different artefacts to produce, and above all, lack of real user-centred approaches.

B. Shneiderman [Shn98] explains that, within the “usability engineering”, one of the bases to reach a high quality level is using iterative design methods. These methods allow the developers to use prototypes, getting feedback from user reactions.

J. Nielsen [Nie93] states that the best way to successfully apply usability criteria is putting the maximum effort before the design process of the interface layout starts. With this concern, the most important thing is to make a previous study that allows the designer to “know the users”. This knowledge is based not only on the study of users’ characteristics or individual abilities, but also it is necessary to know the tasks they need to carry out. In this sense, it is very important that users could perform evaluations (usability tests) upon the prototype. With this practice we can obtain improvements in next versions of the product.

For the reasons stated before, our proposal includes high-level models which allow us to “know the users”. For this matter, IDEAS includes use case, task and user models. The first ones allow us to describe user tasks accurately. The last one describes the individual characteristics of the different kind of users interacting with the system.

After knowing the users and the tasks they perform, it is time to set the goals or basic usability and accessibility criteria the system must reach. The definition of these criteria will allow us to perform the usability tests associated to such criteria further on. In this stage, a quality model gathering these usability criteria and metrics is defined.

In last phases of the development when designing the presentation model, the designer, considering the quality model previously defined, must take into account different heuristics, usability guidelines and accessibility criteria to improve the quality of the layouts. Starting from the presentation models, the user interface prototype is generated. Final users evaluate these prototypes with the usability and accessibility tests based on the quality model previously defined.

Finally, all the previously stated phases and models are incorporated into an iterative design process. This way of doing allows us to perform a feedback process by introducing the information collected in the tests performed by the users in the user interface development. This way, the usability of the designed interfaces improves notably. In figure 1, this iterative process integrating usability is depicted.

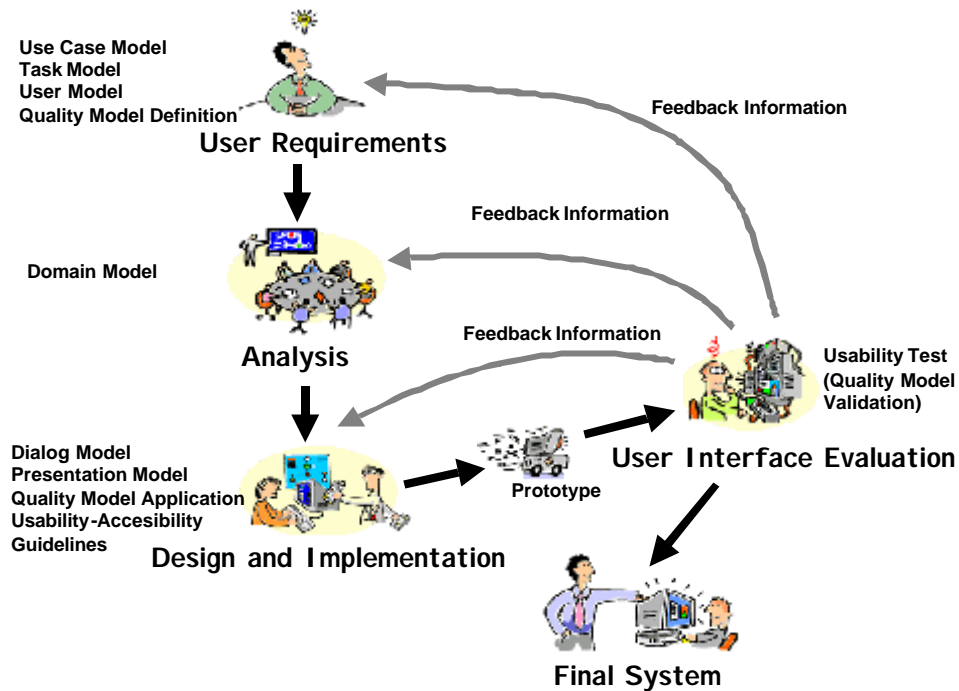


Figure 1: Iterative development process.

The aim of this software process is the integration of a user interface model including usability and accessibility criteria within an object oriented software production environment. The user interface specification process is tackled in parallel to the application development, and according to the common principles of Model-based User Interface Development Environments [Sch96]. This development process is depicted in more detail in figure 2.

It is important to highlight that this user interface development process is independent of the final platform where the Graphical User Interface (GUI) is going to run. It could be a web interface, a personal computer interface, a PDA or any other platform, as a decision of this kind is only made in the last phase of the methodology, i.e., the same design may be used for different implementations in different devices.

As shown in figure 2, the development process is split into two parts: On the one hand the development of the GUI is performed and on the other hand the application itself is developed.

These two independent and parallel developments take as starting point the same information: a set of use cases providing the requirements of the domain activity to be supported. Starting from this common information, on the one hand a domain model is created, that is, a conceptual object model that describes the objects identified within the system together with the relations among them.

But on the other hand, and in order to develop the GUI, we need additional information apart from the one gathered by the use cases. Use cases are not enough to describe user tasks from the point of view of the final user. Use cases describe the systems in terms of functional requirements; nevertheless, one of their main deficiencies is that they do not capture non-functional requirements, such as reliability, efficiency, maintainability and above all, *usability*, which undoubtedly are critical factors for final users to accept a computer application. Use cases do not capture user needs either, which are essential to develop user interfaces. For these reasons, task analysis techniques are included at this point. Starting from task analysis, the User Interface Model is built in the way described below.

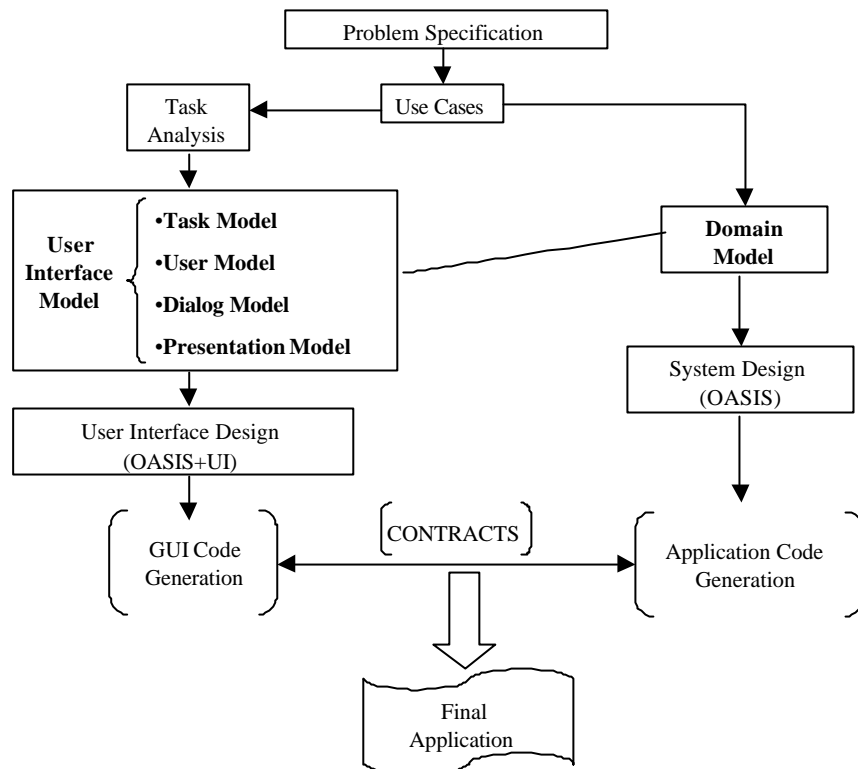


Figure 2: Development Process within IDEAS.

The approach we propose is to take a use-case-driven development process adding task analysis techniques to enable the specification of the user interface in a usable way. These techniques mainly help to define who the users of the system are, what tasks should they perform, what the main goal of those tasks is, and so on.

On the other hand (the right side in figure 2), the design of the system supporting the solution is developed. In IDEAS, the declarative models composing the user interface model are the following: Task, User, Domain, Dialogue and Presentation Models. In next section, these models will be defined following the different stages of the development process.

3.1. User Interface Development Process.

The user interface development process proposed in IDEAS is depicted in figure 3.

At **Requirements Level** three models are created: the Use Case Model, the Task Model and the User Model.

Firstly, the **use cases technique** is used. In this first stage and taking into account the use cases, the different kinds of users are identified. Subsequently, each use case is refined and the business rules, entities and actors that participate in it are specified.

The **Task Model** defines the ordered set of activities and actions the user has to perform to achieve a concrete purpose or goal. Artifacts are essential for task development. They are modeled as objects and represented in the domain model. There exist a strong relation between the task model and the domain model.

The **User Model** describes the characteristics of the different types of users. The purpose of this model is to support the creation of individual and personalized user interfaces at design time. Firstly, it defines, for each kind of user, the set of tasks he/she can perform. Secondly, for each kind of user in relation with a concrete task, a projection on the actions within the task that he/she can

perform is established. And finally, depending on the user's particular characteristics (child, adult, handicapped...), the information and the interaction established by the Dialog Model to show the information contained in the Domain Model is adapted to the user. All these characteristics are used to define the properties to be included in the quality model.

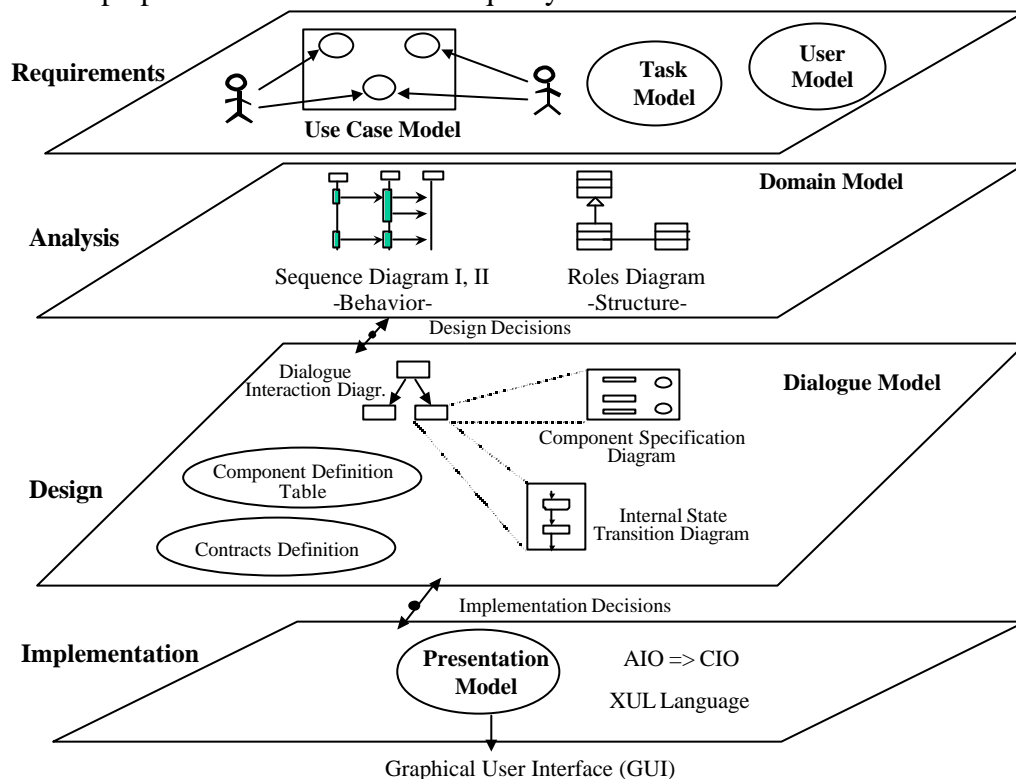


Figure 3: IDEAS: Interface Development Environment within OASIS.

At **analysis level** the **Domain Model** is performed. This model consists of two diagrams. The first one is the **Sequence Diagram** a slight variation based on the original UML sequence diagrams, which defines the system behavior. The second one is the **Roles Model**, a kind of class diagram which defines the structure of the classes that take part in the associated sequence diagram together with the relationships among these classes, specifying the role of each one of them.

At **design level**, the **Dialogue Model** is performed [Loz00]. All the models that have been generated up to now do not contain any graphical aspect of the final user interface. It is from now on that these issues start to be addressed and the way in which the user-system interaction will be performed is especially important. The dialogue model includes the generation of four different kinds of diagrams. Herein and due to space constraints, we highlight the Dialogue Structure Diagram and the Component Specification Diagram. The first diagram specifies the user interface behavior. It represents the windows and dialogues the user needs to complete all the tasks he/she requires from the system and the user selections to pass from one window to another. For the generation of this diagram, we have to take into account the sequence diagram obtained in the analysis phase. The second diagram entails establishing, for every one of the windows and dialogues obtained in the first diagram, the set of Abstract Interaction Objects (AIOs) needed to give the user the functionality he requires to perform the corresponding task. At this level the user interface is designed by means of Abstract Interaction Objects (AIOs), so that we do not have to decide yet the user interface concrete widgets. This decision is postponed to the next step considering the final implementation platform.

At **implementation level** the **Presentation Model** is performed. This model describes the concrete interaction objects (CIOs) composing the final GUI. These CIOs are defined taking into account implementation decisions, the final implementation platform and according to the style guide followed. In IDEAS, the final GUI generation is performed by using XUL*, an XML based language, in order to make it as much independent as possible from the final platform where the application is going to run. The use of different abstract levels (AIOs-CIOs) allows us to attain the “technology variety” challenge stated by Shneiderman [Shn00] as this way of doing allows us to perform different implementation under different platforms starting from the same design.

Next we show a fragment of a case study developed with the tool IDEAS-CASE [Loz02] that supports the process explained before. The example is based on a web application for a Conference Review System. Figure 4 shows the use case diagram for the Conference Review System, showing also the different development stages in which the tool is structured.

Following the development process proposed within IDEAS, at design level we finally get the Component Specification Diagram showed in figure 5. This diagram shows the Abstract Interaction Objects (AIO) the user needs to perform the task of logging into the system.

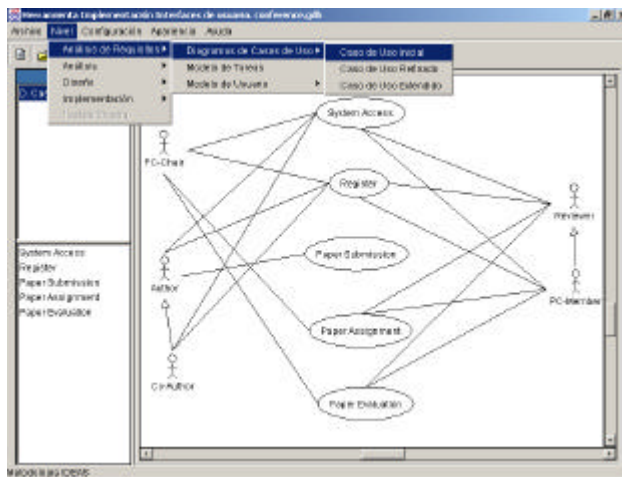


Figure 4. IDEAS Tool: Initial Use Case Diagram.

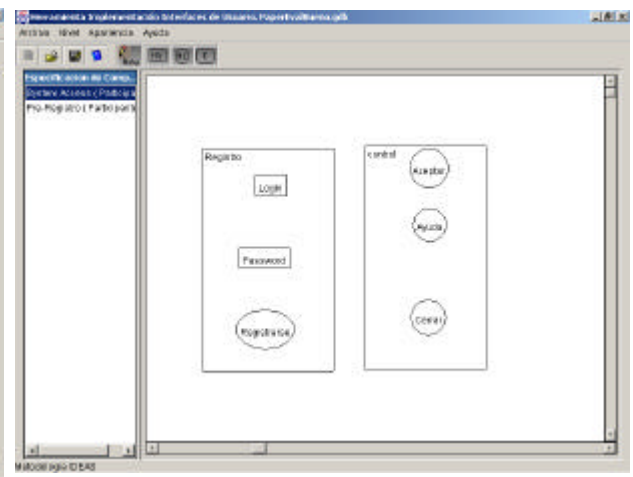


Figure 5. Component Specification Diagram (AIOs)

Starting from this diagram, the AIO are translated into CIO as depicted in figure 6, automatically generating the final GUI shown in figure 7.

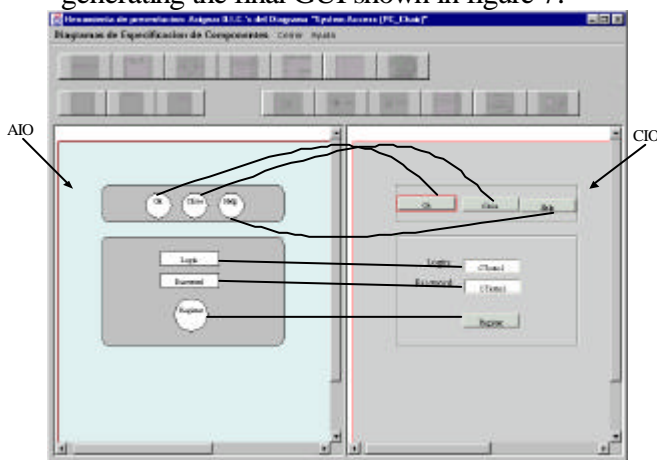


Figure 6. Conversion from AIO to CIO.

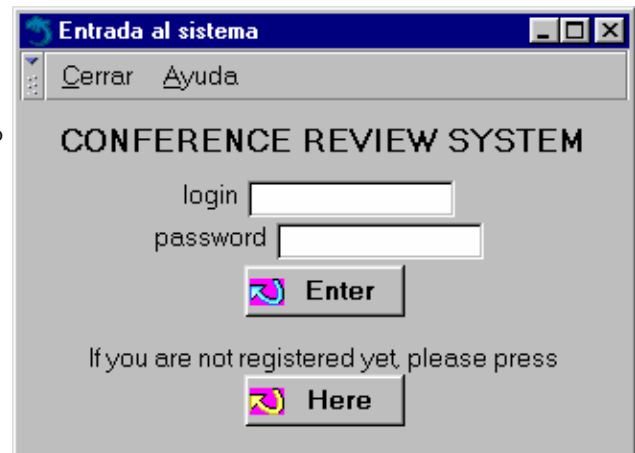


Figure 7. Final Graphical User Interface Generated.

* XML-Based User Interface Language

4. CONCLUSIONS

In this paper we have tackled the problem of improving the user interface quality to get closer to the concept of Universal Access and allow so that the highest number of citizens in the highest number of conditions can access the new Information Technologies.

To achieve this general goal it is essential to incorporate usability and accessibility criteria within the software development process. But, currently there are very few proposals including these criteria accurately. In this sense, we have tried to give a step forward and we propose a methodological framework incorporating a quality model in its process model as a mechanism to define and validate usability and accessibility criteria. This development process not only allows us to generate adaptable user interfaces depending on the particular characteristics of the different kinds of user considered in the proposed User Model, but also the definition of different abstraction levels allows us to generate different implementations from the same abstract specification.

REFERENCES

- [Con99] Constantine, L. and Lockwood, L. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley, New York, 1999.
- [Cos01] Costabile, M.F. Usability in the Software Life Cycle. *Handbook of Software Engineering and Knowledge Engineering*. World Scientific Publishing, pp. 179-192. Singapore, 2001.
- [Lan95] Landauer, T.K. "The Trouble with Computers: Usefulness, Usability and Productivity". MIT Press, 1995.
- [Loz00] M. Lozano, I. Ramos, P. González. User Interface Specification and Modeling in an Object Oriented Environment for Automatic Software Development. 34th International Conference on Technology of Object-Oriented Languages and Systems. TOOLS-USA 2000. Santa Barbara, CA. 30 July - 3 August, 2000.
- [Loz02] M. Lozano, J.P. Molina, F. Montero, P. González, I. Ramos. A Graphical User Interface Development Tool. *Proceedings of the 6th Annual Conference on Human Computer Interaction (HCI'02)* Ref.: ISBN: 1-902505-48-4, Londres, Inglaterra, 2002.
- [May99] Mayhew, D.J. *The Usability Engineering Lifecycle*. Morgan Kaufmann, San Francisco, CA. 1999
- [Nie93] J. Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993.
- [Sch96] Schlungbaum, E.. *Model-Based User Interface Software Tools - Current State of Declarative Models*. Technical Report 96-30, Graphics, Visualization and Usability Center, Georgia Institute of Technology, 1996.
- [Shn98] B. Shneiderman. *Designing the User Interface. Strategies for Effective Human-Computer Interaction*. Addison Wesley, 1998.
- [Ste01a] Stephanidis, C. User Interfaces for all: new perspectives into HCI. In *User Interfaces for all- concepts, methods and tools*. Lawrence Erlbaum Associates, Mahwah, NJ. pp. 3-17, 2001.
- [Ste01b] Stephanidis, C., Savidis, A. Universal Access in the Information Society: Methods, Tools and Interaction Technologies. *UAIS 1*: 40-55. 2001.