

Integrating Usability within the User Interface Development Process of Web Applications*

MARÍA LOZANO, PASCUAL GONZÁLEZ, FRANCISCO MONTERO, JOSE PASCUAL MOLINA

Computer Science Department
University of Castilla-La Mancha
Albacete -SPAIN-

ISIDRO RAMOS

Dept. of Information Systems and Computation
Valencia University of Technology
Valencia -SPAIN-

The main goal of information systems development in general, and web oriented applications in particular, is obtaining a system that allows final users to reach their goals and perform the tasks needed to achieve the goals in an efficient and effective way. To achieve this aim it is necessary, not only taking into account system requirements, but also incorporate new techniques that help to capture user needs considering usability criteria. In this way, it is possible to develop intuitive and easy-to-use user interfaces that help the final users to derive the maximum benefit from web applications. With this aim, a new user interface development environment within the object-oriented software development process is proposed in this paper.

Categories and Subject Descriptors: H5.2 [**Information Interfaces and Presentation**]: User Interfaces - *User-centered design; Interaction styles; Theory and methods.*

General Terms: Human Factors

Additional Key Words and Phrases: Human-computer interaction, object oriented modeling

1. INTRODUCTION

Most of the research developed by industry and academy has led to several object-oriented methods highly adequate for the development of systems. But most of these methods do not include mechanisms (models) for identifying and specifying user needs and requirements as well as testing and validating requirements with end-users before, during and after development. These considerations are especially important in web applications and interactive systems in general, where the user interaction is very high and the user interface is a significant part of the system.

As a result of this weakness, interactive systems developed using such methods can meet all technical requirements, be very robust, and yet be unusable by the end-user. This problem explains a large part of the frequently observed phenomenon whereby large numbers of change requests to modify the services of a web application are made after its deployment. As a solution to this problem we propose a method for integrating a user interface model, taking into account user needs and requirements, in the software life cycle. Besides, this approach to the development process ensures the quality of delivered applications from the point of view of the end user.

There exists several reasons that justify the complexity of design and implementation of user interfaces [1], but perhaps the main reason is the difficulty of understanding the

* This research is supported by the GEOZOCO project from the Spanish government commission of Science and Technology with reference CICYT-TIC 2000-1106-C02-02.

tasks the user must perform with the system and the characteristics of the different kind of users managing it [2]. In this sense, it seems necessary to define new models that allow collecting user requirements and specifying the different characteristics of the interface associated to a concrete application. These models should allow defining the specification of the interface independently [3], that is, the models used to specify the user interface should be different from those used to specify the application.

In this paper we present IDEAS¹ an object oriented and model-based approach for developing user interfaces within an automatic software production environment. This environment is supported by an object-oriented model called OASIS² [4], [5], which covers the software life cycle of analysis, design and implementation within the principles of the object oriented paradigm. So, for better understand our proposal, we first present the state-of-the-art of model-based user interface development environments, and then we finally present IDEAS with an example of use.

2. MODEL-BASED USER INTERFACE DEVELOPMENT ENVIRONMENTS. STATE OF THE ART

In this section we briefly describe the most understanding features of current model-based user interface development environments. In a state of the art report, B. Myers presents a classification of user interface software tools [6]. This classification is based on the way the user interfaces developers can specify the layout and dynamic behavior of a user interface. This classification is the following:

- Language-based Tools: These tools require the developer to program in a special purpose language.
- Interactive Graphical Specification Tools: These tools allow an interactive design of the user interface.
- Model-based Generation Tools: These tools use a high-level specification or model to generate the user interface automatically.

The last classification seems to be the most adequate to tackle the user interface development. The first two support the specification of either the dynamic behavior or the layout of the user interface in an easy way, but not both parts at the same time. And the same as most current tools, they only support the last phase of the user interface life-cycle development, and the abstractions they provide do not have a direct connection with the task analysis results.

On the other hand, model-based user interface development approach and its corresponding tools is an emerging technology that remedies these shortcomings. Interfaces are developed by using specialized design-time tools to build and refine models. Developers specify “what” features the interface should have, rather than write programs that specify “how” to make the computer exhibit the desired behavior.

The main shortcoming of the model-based approach is that building models is difficult. A rich model describing all the features of an interface is complex and non-trivial to specify.

The most representative feature of the model-based approach is that all aspects of the user interface design are represented using declarative models. This paradigm offers several benefits:

¹ Interface Development Environment within oASis.

² Open and Active Specification of Information Systems

- User-centered development life cycle.
- Centralized user interface specification.
- Design tools for an automated and interactive development.
- Reuse of the user interface designs.

For a user interface tool to be considered as a Model-based user interface development environment, it must accomplish the following criteria:

1. It must include a high-level, abstract and explicitly represented (declarative) model about the system under development.
2. It must clearly establish a technologically well-supported relation between 1) and the final running user interface.

Currently there exist several model-based user interface software tools, as for instance, UIDE [7], HUMANOID [8], MOBI-D [9], MASTERMIND [10], [11], TRIDENT [12], FUSE [13], JANUS [14] and IDEAS [15]our proposed model explained in following sections.

3. IDEAS: INTERFACE DEVELOPMENT ENVIRONMENT WITHIN OASIS

The aim of this research is the integration of a user interface Model 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, as exposed in previous section. This development process is depicted in figure 1.

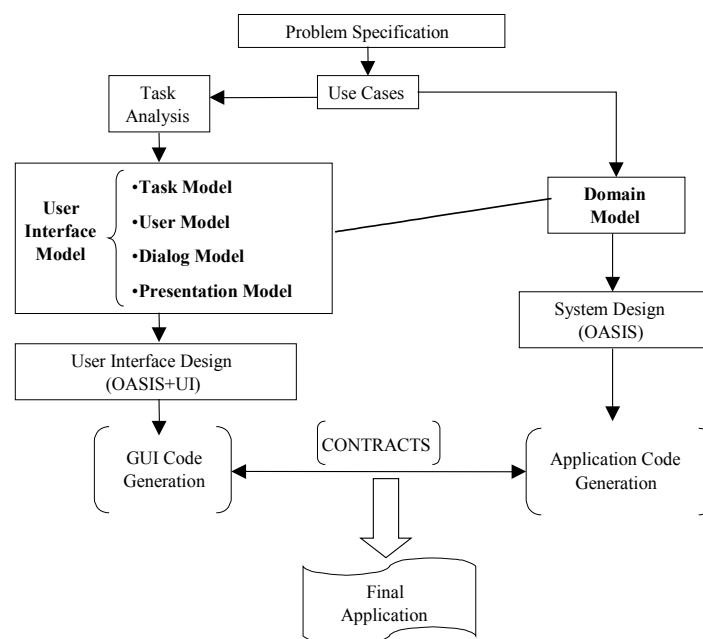


Fig. 1. IDEAS: Interface Development Environment within OASIS.

IDEAS aims to be an automatic user interface development system integrated within the framework of the object oriented model OASIS [5] to support the automatic production of high-quality user interfaces. This is possible due to the fact that this environment is based on declarative models, with the advantages of this approach as stated in the previous section.

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 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 1, 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. The relationship between use cases and the tasks identified with the task analysis technique is one to many, this is, one use case may correspond to one or many tasks. So, the system functionality described by a use case is equivalent to one or many user tasks.

All in all, 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. 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 1), the design of the system supporting the solution is developed. In many ways these two designs (the user interface and the application) specify two separate but interdependent systems.

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 Development Process

The user interface development process proposed in IDEAS is depicted in figure 2.

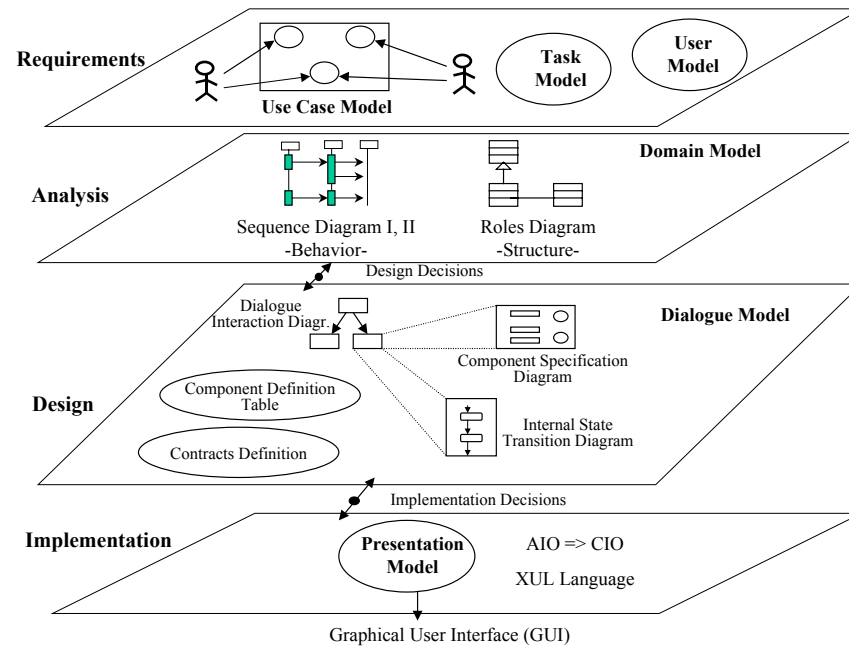


Fig. 2. User Interface Development Process.

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.

According to [16], a task description should include a goal, a non-empty set of actions necessary to achieve the goal, a plan for the selection of actions and a model of the artifact involved in the task development. This description of what a task is and must contain can be used to construct a task definition language. We propose a template (fig. 6), based on the one proposed by [17], to describe in natural language all these issues. The question now is what tasks should be modeled in the task model. The answer is all those tasks the final users have to perform with the system.

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. 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.

At analysis level the Domain Model is performed. This model consists of two diagrams. The first one is the Sequence Diagram, which defines the system behavior. The second one is the Roles Model, 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 [18]. 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, this is, it represents the windows and dialogues that 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 control and visualization tools needed to give the user the functionality he requires performing the corresponding task. This diagram defines the state of the user interface.

At implementation level the Presentation Model is performed. This model describes the concrete interaction objects (CIOs) composing the final GUI, its design characteristics and visual dependencies among them. These CIOs are defined taking into account implementation decisions, the final implementation platform and according to the style guide followed. The final GUI has a static and a dynamic part. The first one is the standard widgets presentation and the second one shows the data dependent on the application that change at run time. In IDEAS, the final GUI generation is performed by using XUL³, 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 and so get the maximum compatibility with new web technologies.

This development process considers the three-level software architecture: boundary, control and entity as proposed in the Unified Software Development Process [19], but the scope of our proposal is to specify the boundary classes corresponding with the final graphical user interface. Users interact with these boundary classes (user interface classes) to perform their tasks. The boundary classes will be associated to the corresponding control class/es to achieve the task goal, and the entity classes correspond to long-lived and persistent information. The specification of boundary classes is the goal of user interface design whereas the specification of control and entity classes is the goal of domain analysis model. Thus, we center on boundary classes to represent the application user interface.

4. USABILITY IN USER INTERFACE DEVELOPMENT

The main aim of software construction is to help users to achieve their goals in an easy way. That is, make the system to be easy to learn, easy to manage and useful, i.e. it should contain all the functionality the users need to accomplish their tasks. All in all, the basic goal is to get a computer application with a high grade of “usability” [2].

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 [20] 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

³ XML-Based User Interface Language

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. This means that computer application development is a multidisciplinary activity that must incorporate human factors and ergonomic techniques with the aim of improving the work conditions of final users. This implies the participation of different kind of experts, not only computer engineers but also experts on disciplines such as psychology, ergonomics and human factors and so on.

4.1. The ISO Definition of Usability

The ISO 9241 standard [21] defines usability as the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use".

- effectiveness: The "accuracy and completeness with which users achieve specified goals".
- efficiency: The "resources expended in relation to the accuracy and completeness with which users achieve goals".
- satisfaction: The "freedom of discomfort, and positive attitude to the use of the product".

The components and the relationships between them are illustrated in figure 3.

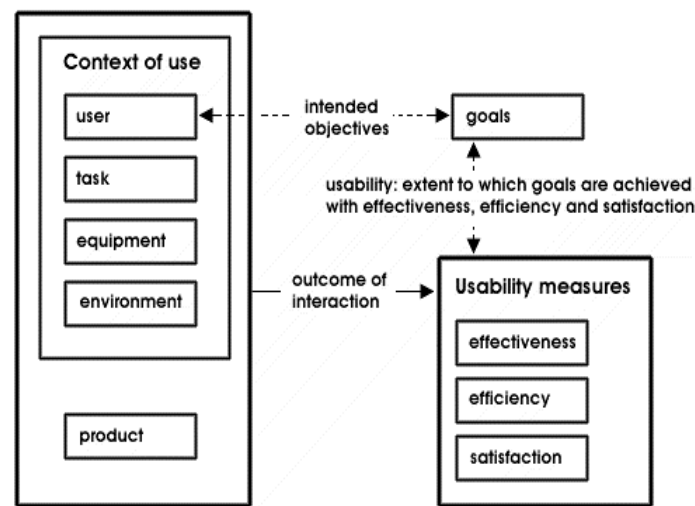


Fig. 3. Usability framework according to ISO/DIS 92411.2.

The standard states that when specifying or measuring usability, the following information is needed:

1. A description of the intended goals.
2. A description of the components of the context of use including users, tasks, equipment and environments. This may be a description of an existing context, or a specification of intended contexts. The relevant aspects of the context and the level of detail required will depend on the scope of the issues being addressed. The description of the context needs to be sufficiently detailed so that those aspects of the context which may have a significant influence on usability could be reproduced.

3. Target or actual values of effectiveness, efficiency, and satisfaction for the intended contexts.

4.2. Usability Context of Use.

The context of use defined by the standard includes the following factors [22]:

Description of users: Characteristics of the users need to be described. These can include knowledge, skill, experience, education, training, physical attributes, and motor and sensory capabilities. It may be necessary to define the characteristics of different types of user, for example users having different levels of experience or performing different roles.

Description of tasks: Tasks are the activities undertaken to achieve a goal. Characteristics of tasks which may influence usability should be described, e.g. the frequency and the duration of the task.

Detailed descriptions of the activities and processes may be required if the description of the context is to be used as a basis for the design or valuation of details of interaction with the product. This may include description of the allocation of activities and steps between the human and technological resources. Tasks should not be described solely in terms of the functions or features provided by a product or system. Any description of the activities and steps involved in performing the task should be related to the goals which are to be achieved.

Description of equipment: The description of the hardware, software and materials may be in terms of a set of products, one or more of which may be the focus of usability specification or evaluation, or it may be in terms of a set of attributes or performance characteristics of the hardware, software and other materials.

Description of environments: Relevant characteristics of the physical and social environment need to be described. Aspects which may need to be described include attributes of the wider technical environment (e.g. the local area network), the physical environment (e.g. workplace, furniture), the ambient environment (e.g. temperature, humidity) and the social and cultural environment (e.g. work practices, organizational structure and attitudes).

Usability measures: Usability measures include effectiveness, efficiency and satisfaction. These are measured in user trials of the product.

Goals of trial identify what the projects are trying to find out in their trials. The goal of the user trial may be to help in defining user requirements, to validate that the technology works in real conditions, to measure user attitudes, etc.

All in all, usability means that the people who use the product can do so quickly and easily to accomplish their own tasks. [Dum93].

B. Shneiderman [23] explains that, within the “usability engineering”, one of the basis to reach a high quality level is using iterative design methods. These methods permit developers to use prototypes, getting feedback from user reactions.

J. Nielsen [24] states that the best way to successfully applying 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.

4.3. Usability in IDEAS.

For the reasons stated before, our proposal includes high-level models which permit “know the users”. For this matter, IDEAS includes use case, task and user models. The first ones permit 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 criteria the system must reach. The definition of these criteria will allow performing the usability test associated to such criteria further on.

In last phases of the development, when designing the presentation model, the designer must take into account different heuristics and style guides to improve the usability of the layouts. Starting from the presentation models, the user interface prototype is generated. Final users evaluate these prototypes with the usability tests prepared previously.

Finally, all the previously stated phases and models are incorporated into an iterative design process. This way of doing permits 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 4, this iterative process integrating usability is depicted.

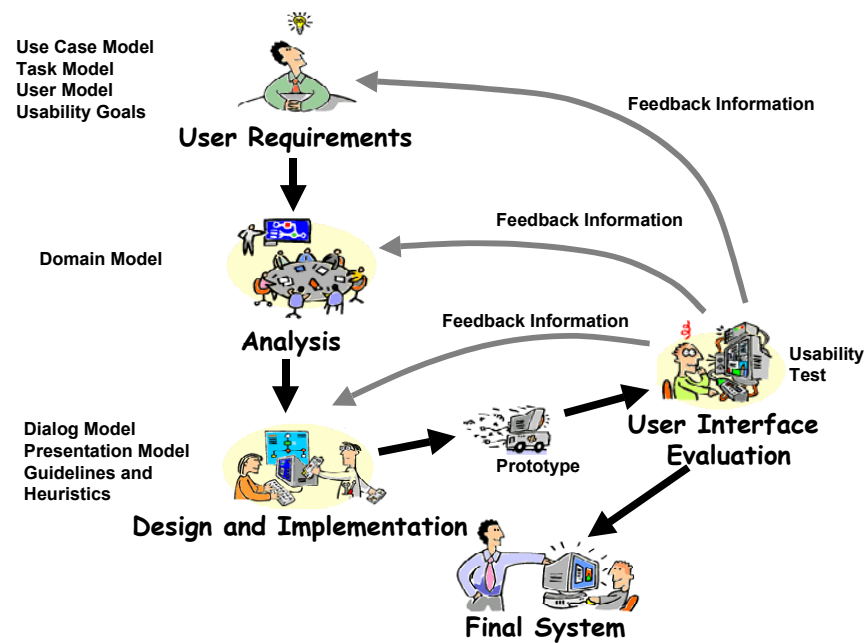


Fig. 4. Iterative development process.

5. EXAMPLE OF USE

To illustrate the proposed methodology, in this section we apply it to the simplified case of a Rental Car. The initial use case model is shown in figure 5.

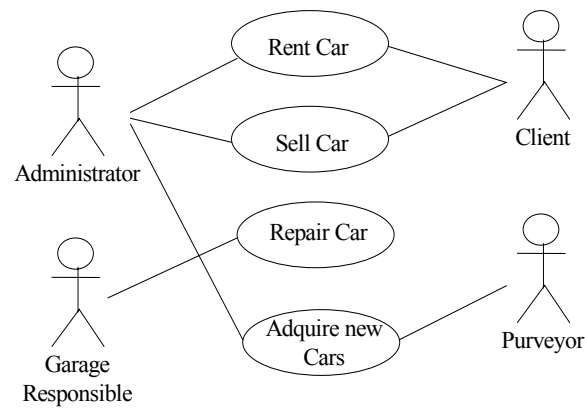


Fig. 5. Use Case Diagram.

The next step is to identify user tasks from use cases. In this case, the “Rent Car” use case corresponds to the task “Rent a Car”. In this case the correspondence is one to one. The task model specification follows the template shown in figure 6.

Task Model: Template (Example: Rental Car)

Task: Rent a car.
GENERAL FEATURES:
GOAL: The administrator rents a car to the client.
PRECONDITION: - The car is available.
 - The car is in good conditions.
 - The client has not more cars than permitted
SUCCESS CONDITIONS: - The administrator checks the car’s availability and state.
 - The administrator brings the client’s state up to date.
 - The client receives the solicited car.
FAILURE CONDITIONS: - The car is not available.
 - The client already has rented the maximum number of cars permitted.
PRIMARY ACTORS: Client, Car.
SECONDARY ACTORS: Administrator.
TRIGGER ACTION: The client request to rent a car.
NORMAL SCENARIO:
 .The client request to rent a car.
 .The administrator checks the state and the availability of the requested car.
 .The administrator checks that the client has not rented the top number of cars permitted.
 .The administrator gives the client the car.
 .The client receives the requested car.
VARIANTS:
 .The car is not available.
 .The car is not in good state.
 .The client already has the top number of cars permitted to rent.
RELATED INFORMATION:
Priority: High
Duration: 15 mins.
Frequency: 10/day.

Fig. 6. Task Model Template.

The Domain Model uses sequence diagrams [25] to model the system behavior and roles model to describe the system structure. This is depicted in figure 7:

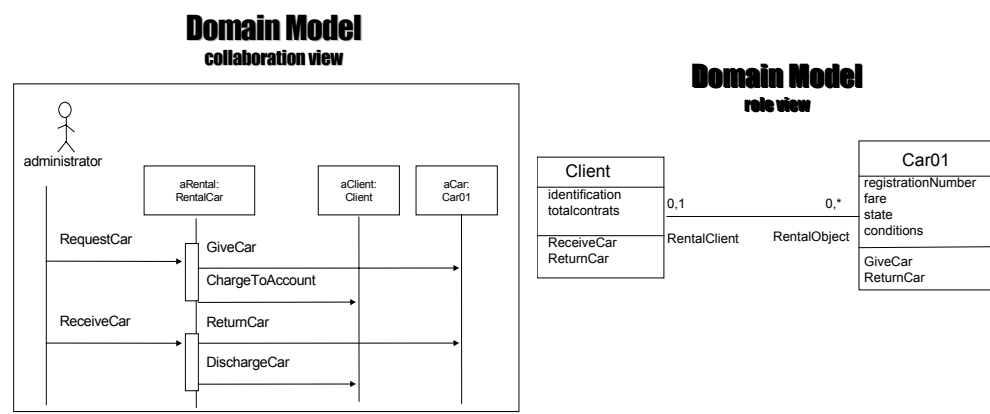


Fig. 7. Domain Model.

The Dialog Model includes the generation of three different kinds of diagrams: the Dialog Structure Diagram, the Dialog Structure Diagram, the Component Specification Diagram and the Component Behavior Definition Table. For the Rent Car case study these diagrams are shown in figure 8.

The Dialog Model₁
Dialog Interaction Diagram:
Windows and Dialogs the user needs to perform the tasks.

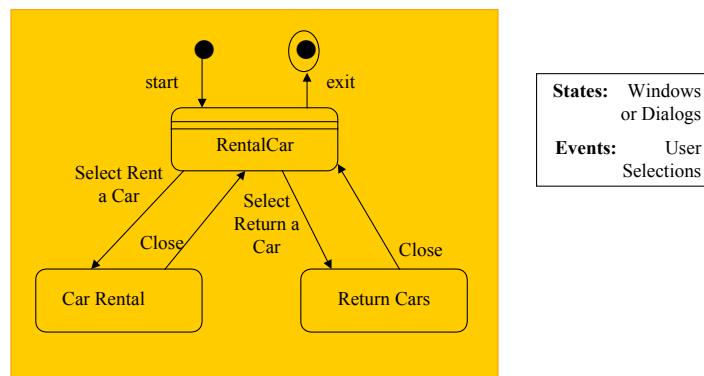
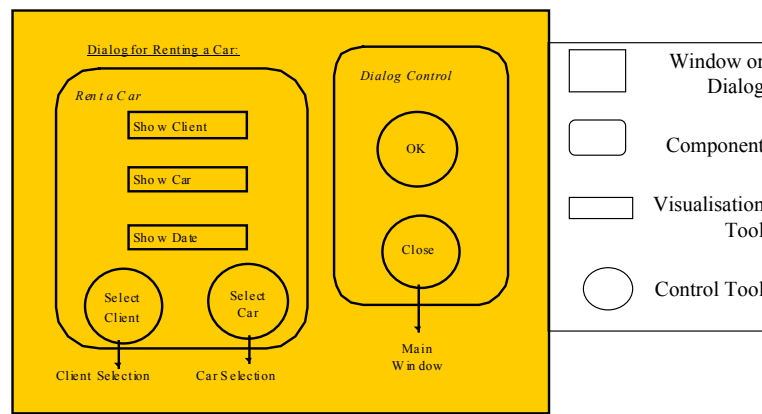


Fig. 8. Dialog Model.

The Dialog Model₂

Component Specification Diagram:

Models the contents of every one of the windows and dialogs represented in the previous diagram.



The Dialog Model₃

Component Functional Model:

Table in which every one of the components and tools represented in the previous diagram are described in detail.

Component	Control Tool	Visualisation Tool
Rent a Car	<i>Select_Client</i> : Allows the selection of a Client or the introduction of a new one. <i>Select_Car</i> : Allows the selection of a car from the database.	<i>Show_Client</i> : Shows the client selected through the control tool. <i>Show_Car</i> : Shows the car selected through the control tool. <i>Show_date</i> : Shows the system date corresponding to the rental date.
Dialog Control	<i>Ok</i> : Allows the user to introduce a new rental in the system. <i>Close</i> : Close the dialog and returns the user to the main window.	—

Fig. 8. Dialog Model (Cont.)

Finally, the Presentation Model is built from the dialog model taking into account style guides. This is, the abstract interaction objects modeled in the dialog model are now translated into the concrete interaction objects conforming the GUI.

6. CONCLUSIONS

In this paper we propose a user interface development environment within the object-oriented software development process. This development process is performed applying use cases and task analysis techniques, from which a user interface model is built. This model consists of five declarative models that describe different aspects involved in user interface construction.

It has been also described the way to incorporate usability criteria in order to achieve high usable user interfaces. Finally, we have stated an iterative design process that permits the incorporation of the consecutive improvements the users introduce in the design process.

And finally, remark the more and more importance that is getting the user interface, above all in interactive systems, as a means of interacting with a system and taking the most profit of it. This is the main motivation of this research.

7. REFERENCES

- [1] B.A. Myers: Why are Human-Computer Interfaces Difficult to Design and Implement. Tech. Report CMU-CS-93-183, CMU, Jul 1993.
- [2] J. Gould, C. Lewis: Designing for Usability – Key Principles and What Designers Think. Commun. ACM, Vol 28, 1985, pp.300-311.
- [3] W. Swartout, R. Blazaer: The Inevitable Intertwining of Specification and Implementation. Commun ACM, Vol 25, No. 7, Jul 1982, pp.438-440.
- [4] O. Pastor, I. Ramos. OASIS 2.1.1.: A Class-Definition Language to Model Information Systems Using an Object-Oriented Approach. Octubre 95 (3 ed.)
- [5] P. Letelier, I. Ramos, P. Sanchez, O. Pastor. OASIS versión 3.0: A Formal Approach for Object Oriented Conceptual Modelling. SPUPV-98.4011. Edited by the Technical University of Valencia, Spain, 1998.
- [6] B. A. Myers: User Interface Software Tools. ACM Transactions on Computer-Human Interaction 2 (1995), 1, 64-103
- [7] J. Foley, P. Sukaviriya: History, Results and Bibliography of the User Interface Design Environment (UIDE), an Early Model-based System for User Interface Design and Implementation. In F. Paterno (ed.): Interactive Systems: Design, Specification and Verification. Berlin: Springer, 1995, 3-14.
- [8] P. Szekely et al: Beyond Interface Builders: Model-Based Interface Tools. In: Bridges between Worlds. Proceedings InterCHI'93 (Amsterdam, April 1993). New York: ACM Press, 1993, 383-390.
- [9] A. Puerta: The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development. In: Computer-Aided Design of User Interfaces. Namur: Namur University Press, 1996, 19-36.
- [10] R. Neches et al.: Knowledgeable Development Environments Using Shared Design Models. In: Proceedings of the 1993 International Workshop on Intelligent User Interfaces (Orlando, Enero 1993). New York: ACM Press, 1993, 63-70.

- [11] P. Szekely, et al.: Declarative Interface Models for User Interface Construction Tools: The MASTERMIND Approach. In: Engineering for Human-Computer Interaction. Proceedings of the IFIP Conference on Engineering for Human-Computer Interaction (Yellowstone Park, Agosto 1995). London: Chapman & Hall, 1996, 120-150
- [12] F. Bodart et al.: Towards a Systematic Building of Software Architectures: the TRIDENT Methodological Guide. In: Design, Specification and Verification of Interactive Systems. Wien: Springer, 1995, 262-278.
- [13] F. Lonczewski: The FUSE System: An Integrated User Interface Design Environment. In: Computer-Aided Design of User Interfaces. Namur: Namur University Press, 1996, 37-56.
- [14] H. Balzert et al.: The JANUS application Development Environment Generating More than the User Interface. In: Computer-Aided Design of User Interfaces. Namur: Namur University Press, 1996, 183-205.
- [15] M. Lozano: A Methodological Approach for the Specification and Development of Object Oriented User Interfaces. Ph.D. Thesis. Department of Information Systems and Computation. Valencia University of Technology. 2001.
- [16] G. Storrs: The Notion of Task in Human-Computer Interaction. In: People and Computers X. Proceeding British HCI'95 (Huddersfield UK, Agosto 1995). Cambridge: Cambridge University Press, 1995, 357-365.
- [17] A. Cockburn: Structuring Use Cases with Goals. Journal of Object-Oriented Programming, 10(5) y 10(7), 1997.
<http://members.aol.com/acockburn/papers/usecases.htm>
- [18] 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.
- [19] I. Jacobson, G. Booch, J. Rumbaugh. The Unified Software Development Process. Addison Wesley. 1998.
- [20] Landauer, T.K. "The Trouble with Computers: Usefulness, Usability and Productivity. MIT Press, 1995.
- [21] ISO/DIS 9241: Ergonomic Requirements for Office Work with Visual Display Terminals (VDT's). Geneva: International Standards Organization. 1998.
- [22] Concejero, P. Clarke, A., Ramos, R. "ACTS Usability Evaluation Guideline" Report of USINACTS Usability in Acts. June, 1999.
- [23] B. Shneiderman. Designing the User Interface. Strategies for Effective Human-Computer Interaction. Addison Wesley, 1998.
- [24] J. Nielsen. Usability Engineering. Morgan Kaufmann, 1993.
- [25] G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language. Addison Wesley. 2000.