DESIGNING UNIVERSAL USER INTERFACES THROUGH AN ONTOLOGY-BASED METHOD

Elizabeth Furtado
Elizabet@feq.unifor.br

Jean Vanderdonckt
Vanderdonckt@qant.ucl.ac.be

Abstract

In recent years, universal access has attracted considerable attention by the research communities, mainly from the perspective of system development. It comes from the fact that the design of user interfaces for all must cover design issues in multiple contexts of use where multiple types of users may carry out multiple tasks, possibly on multiple domains of interest. Existing design methods do not necessarily support designing such user interfaces. A new design method is presented for this purpose which allows a developer derives multiple user interfaces from models represented by an ontology of concepts, relationships, and attributes of the domain of discourse.

Keywords: *automated generation of user interfaces, model-based approach, modeling, ontology, physical level, universal design*

Resumo

Recentemente, o tema acesso universal tem atraído a atenção da comunidade, principalmente do ponto de vista dos métodos de desenvolvimento de software. Tal interesse se deve ao fato de que o projeto de interfaces usuário-computador *universais* deve considerar questões de projeto para diversos contextos de uso, onde diversos usuários interagem realizando várias atividades, possivelmente de diversos domínios. Os métodos de desenvolvimento de software existentes não necessariamente suportam o projeto destes tipos de interfaces. Um novo método de desenvolvimento é apresentado neste trabalho com este propósito, o qual permite o projetista derivar múltiplas interfaces a partir de modelos representados por uma ontologia de conceitos, relações e atributos do domínio em questão.

Palavras-chave: *geração automática de interfaces, enfoque baseado em modelos, ontologia, nível físico e projeto universal.*

1 Introduction

User Interfaces (UIs) for all (SAVIDIS, AKOUMIANAKIS & STEPHANIDIS, 2001) of interactive applications are developed for the widest population of users in the most different contexts of use by taking into account differences such as preferences, language, culture, habits, conventions, and system experience. One way to assure the universal access is to develop applications with adaptive user interfaces. These interfaces are able to *adapt* themselves to the user's characteristics, varying, for instance, in what concerns the kind of assistance that must be offered the user and how it must be shown to cope with individual differences induced by universal access. Along with the accessibility factor, the usability factor is another way to assure the quality of the applications. Universal design of usable UIs poses some difficulties due to the consideration of these multiple parameters depending on the supported differences. In addition, methods for developing UIs do not mesh well with this high variety of parameters as these parameters are not necessarily properly identified and manipulated in a structured way nor truly considered in the design process.

The goal of this paper is to present a structured method addressing parameters required for universal design. The method is supported by a suite of tools all based on an ontology of the domain of discourse and models that capture instantiations of concepts identified in this ontology for producing multiple UIs for one design situation. These different UIs exhibit different presentation styles, dialogue genres, and UI structures.

The remainder of this paper is structured as follows: section 2 provides a state of the art of methods for developing UIs with a focus on universal design. The method is progressively described in section 3. Section 4 summarizes the main points of the paper.

2 Related work

The Author's Interactive Dialogue Environment (AIDE) (GIMNICH, KUNKEL & REICHERT, 1991) is an integrated set of interactive tools enabling developers to implement UIs by directly manipulating and defining its objects, rather than by the traditional method of writing source code. AIDE provides developers with a more structured way to develop UI than with traditional, yet radically different, "rush-to-code" approaches where unclear steps possibly result in a poorly usable UI.

User-Centered Development Environment (UCDE) (BUTLER, 1995) is an object-oriented UI development method investigating how software development can function as an extension of business process improvements. Business-oriented components (BOCs) are software objects that model business rules, processes, and data from the end-user's perspective. They clearly map this information onto UI objects that are compatible by construction with the information. The advantage of UCDE is a smooth process starting from high-level abstractions to final UIs.

Another methodological framework for UI development is provided in [(FURTADO, 1997), (HIX, 1989) e (HARTSON & HIX, 1989)], which enables to integrate usability issues into the software development process from the beginning. The focal point of this approach is a psychologically based formal task description, which serves as the central reference for assessing the usability of the user interface under development. This contribution emphasizes the need of a task model as starting point for ensuring UI usability, whereas UCDE emphasizes the need of a domain model. The MUSE method (LIM & LONG, 1994) uses structured notations to specify other elements of the context of use such as organizational hierarchies, conceptual tasks, and domain semantics. Moreover, graphical structured notations are proved to communicate UI design to users more easily.

In the above contributions, we see the importance of having a structured way to capture, store, and manipulate multiple elements of the context of use, such as task, domain, and user. Although the above methods partially consider this information, they do not consider designing multiple UIs where task [(CARD, MORAN & NEWEL, 1983) e (GAINES, 1994)], domain, and user parameters are varying, possibly simultaneously. Only the unified process (SAVIDIS, AKOUMIANAKIS & STEPHANIDIS, 2001) suggests deriving multiple refinements of a task model to cope with individual differences induced by universal design. However, this contribution is focussing more on task modeling operations than on steps and information required to progressively take multiple users in multiple contexts of use into account. The following method attempts to fill this gap by dividing the main problem into the three subsequent levels.

3 The Method

3.1 The Basic Concepts

The propose method is based on an ontology of concepts, relationships, and attributes that need to be manipulated in a particular universal design [(GUARINO, 1995) e (TOP & AKKERMANS, 1994)].

The ontology notion comes from the Artificial Intelligence context where it is identified as the set of formal terms with one represents knowledge, since the representation completely determines what "exists" for the system. We hereby define a *context of use* as the global environments in which a population of users, probably with many different profiles, skills, and preferences, are carrying out a series of interactive tasks on one or multiple semantic domains. In universal design, it is expected to benefit from the advantage of considering any type of the above information to produce multiple UIs depending on the varying conditions. These pieces of information of a context of use can be captured in different models (PUERTA, 1997). A *model* is hereby defined as a set of postulates, data and inferences presented as a declarative description of a UI facet. Many facets do exist as well as related models: task, domain, user, interaction device, computing platform, application,

presentation, dialogue, help, guidance, tutorial, organizational environment. A model is typically built as a hierarchical decomposition of abstract concepts into several refined sub-levels. A model should also encompass relations between these concepts with roles, as well as for models, and between models.

Each level can be considered as a level of abstraction from the physical level as represented in Figure 1. The physical level is the instance level where instances of the case study are analysed, the logical level is the model level where theses instances are mapped onto relevant abstractions, and the conceptual level is the metamodel level where abstractions manipulated in the previous levels can be aggregated to identify the concepts, relationships, and attributes used in a particular method.

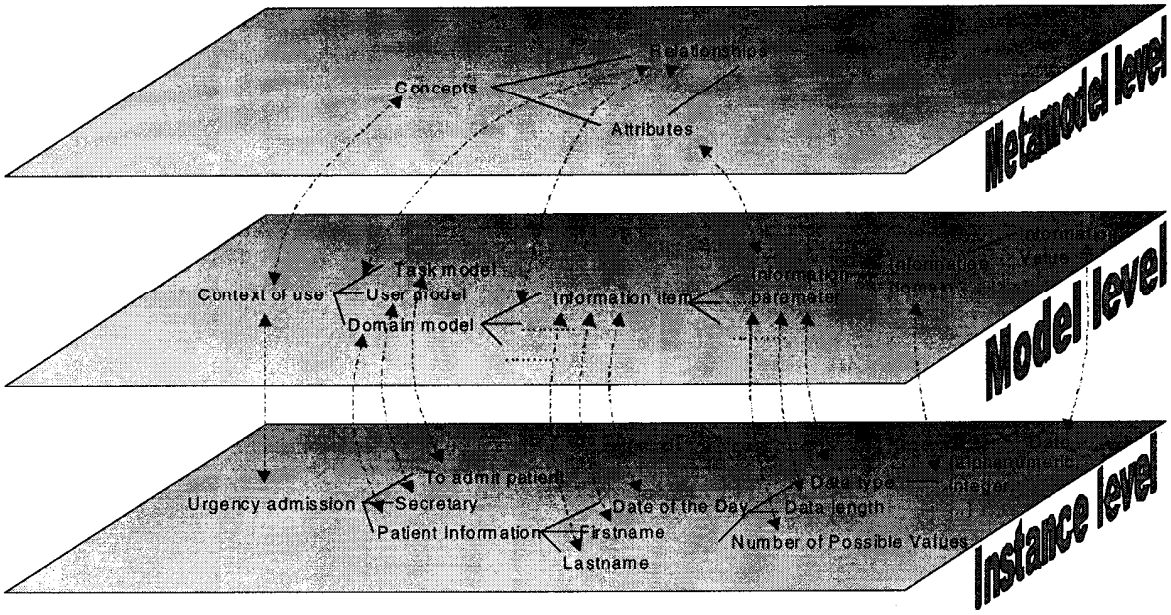

Figure 1- The concept levels of the proposed method.

3.2 The method in three levels of abstraction

The method structures the UI design in three levels of abstraction that can be viewed in Figure 2.

1. In the *conceptual level* is represented human factors and the expert domain by means of ontology of concepts, relationships, and attributes to produce multiple UIs.
2. The *logical level* allows designers to capture requirements of a specific UI design case by instantiating concepts, relationships, and attributes with a graphical editor. Each set of instantiations is stored in various models, thus resulting in a set of models for each considered design case (*n* designs in fig. 2).
3. The *physical level* helps developers in deriving multiple UIs from each set of models thanks to a model-based UI generator: in figure 2, *m* possible UIs are obtained for UI design #1, *p* for UI design #2,..., *r* for UI design #*n*. The generation is then exported to a traditional development environment for any manual edition (here, MS Visual Basic).

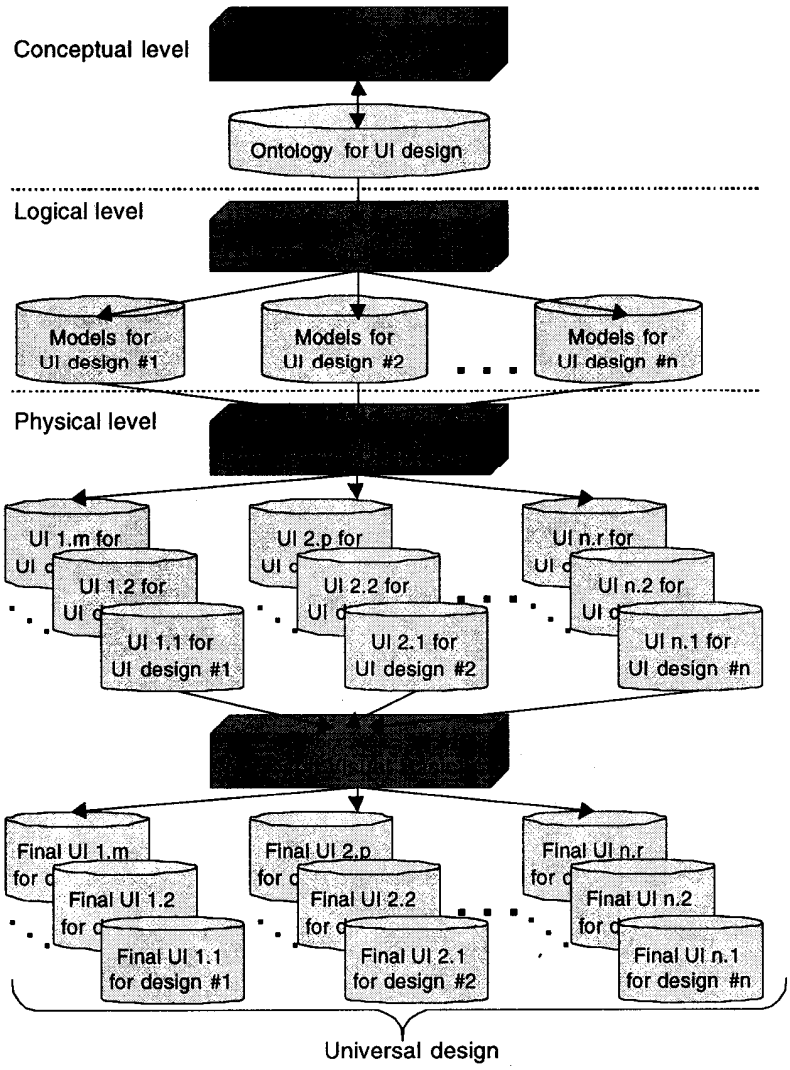

Figure 2- The different levels of the proposed method for universal design of user interfaces.

3.3 The Phases of the Method

The phases of the method are demonstrated throughout the paper by the usage of a specific series of supporting tools on the same case study: patient admission at a hospital.

3.3.1 The Definition of the Models for Designing of Universal Applications

The phase of the definition of the models used for represent any universal application is associate to conceptual level of abstraction of the proposed method. Here, human factors or domain of discourse experts identify the common concepts, relationships, and attributes that need to be represented in models to design universal applications.

For the simplicity of this paper, the models considered are the following:

- A *domain model* defines the data objects that a user can view, access, and manipulate through a UI. These data objects belong to the domain of discourse. A domain model can be represented as a decomposition of information items, any item may be iteratively refined into sub-items. Each such item can be described by one or many parameters (such as data type, length). Each parameter possesses its own domain of possible values.
- A *task model* is a hierarchical decomposition of a task into sub-tasks to end-up with actions which are no longer decomposed [(TOP & AKKERMANS, 1994), (LIM & LONG, 1994)]. The model can then be augmented with tempo-

ral relationships stating when, how and why these sub-tasks and actions are carried out. Similarly to the domain model, a task model may hold a series of parameters with domains of possible values. For instance, task importance (low/ medium/high), task structure (low/medium/high decomposition), task critical aspects (little/some/many), and required experience (low/moderate, high) are often considered.

- A *user model* consists of a hierarchical decomposition of the user population into stereotypes. Each stereotype gathers people sharing the same value for a given set of parameters. Each stereotype can be further decomposed into sub-stereotypes. For instance, the population diversity may be reflected by many user parameters such as language, culture, preference (manual input vs. selection), task experience (elementary/medium/ complex), system experience (elementary/ medium/complex), motivation (low/medium/ high), and experience of a complex interaction media (elementary/ medium/complex).

Figure 3 graphically depicts how the ontology editor can be used at the modeling stage to input, define, and structure concepts, relationships, and attributes of models used to describe a context of use. Here, the three models are represented and they all share a description by information parameters. Each parameter has a domain, each domain has a set of values, possibly enumerated.

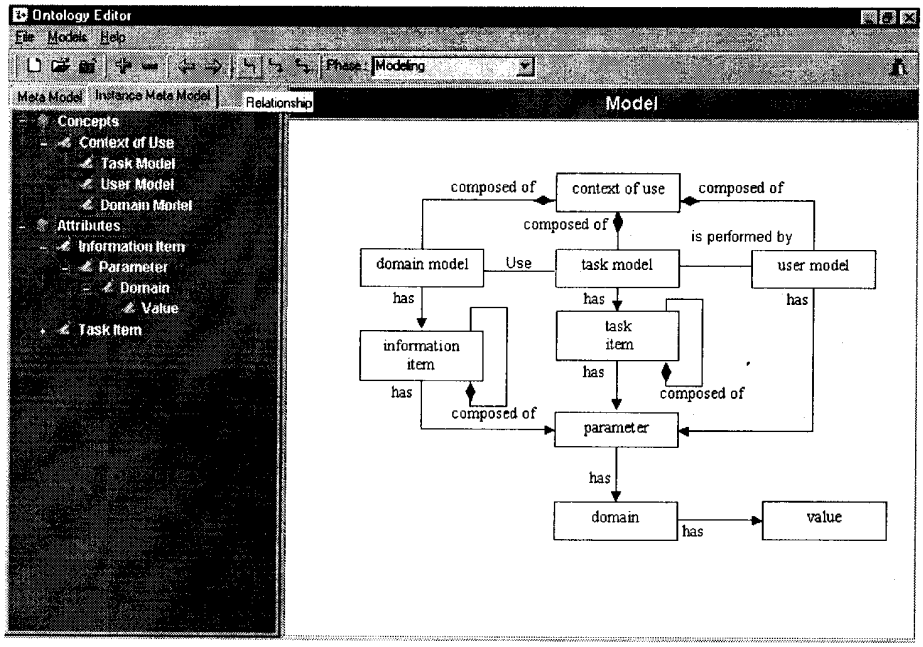

Figure 3- The ontology editor at the modeling stage.

The **big win** of this level is that the ontology can be defined once and use as many times as wished. When universal design requires the consideration of more information in models or more models, this ontology can be updated accordingly and so the method that supports universal design of UIs.

3.3.2 The Instantiation of the Models for Specifying of UIs

The UI specification is characterized by a set of design options, such as the selection of the interaction style, the selection of the dialogue attributes, the selection of the interaction objects, and so on. The problem of choosing UI design options for a design situation arises when, in the context of universal UI, there is no "average" user to ensure that interactive applications give high quality of interaction to all user stereotype. The employment of any particular user modeling approach, since there is no predefined/fixed set of parameters, shows the generality of this method for universal UI generation process. In this phase, which is associate to logical level of abstraction of the proposed method, each model defined before is now instantiated to. The instantiation occurs when the parameter values are defined for the user model, the task information and the domain information.

The ontology editor is used to instantiate the context of use, the relationships and attributes of models for the *Medical Attendance* domain. Figure 4 depicts the *Urgency Admission* context of use and the parameters of models of task, user and

domain. There are two tasks instantiated: *to admit patient* and *to show patient data*. The first one is activated by a *secretary* and uses *patient information* during its execution. The values of the user model parameter of the *secretary* include: *elementary* task experience level, input preference for *typing in data* rather than selecting it, information density preference in the screen with the enumerated values *low* and *high*.

The information items of a *patient* include: *date day*, *first name*, *last name*, *birthdate*, *address*, *phone number*, *gender* and *civil status*. The parameter values of an information item of a domain model depend on the UI design process. The parameter values of the *first name* information item required by the UI design process used here (*50* characters), domain type (this is a *known* domain with some *unknown* values the user can supply), interaction way (this data is to be used as *input*), mandatory status (this data is *required*), number of possible values (this is not applied to), number of principal values (this is not applied to), precision (the number of digits is *important*) and continuity whether all values of data are spread in a continuous range of values (this is not applied to).

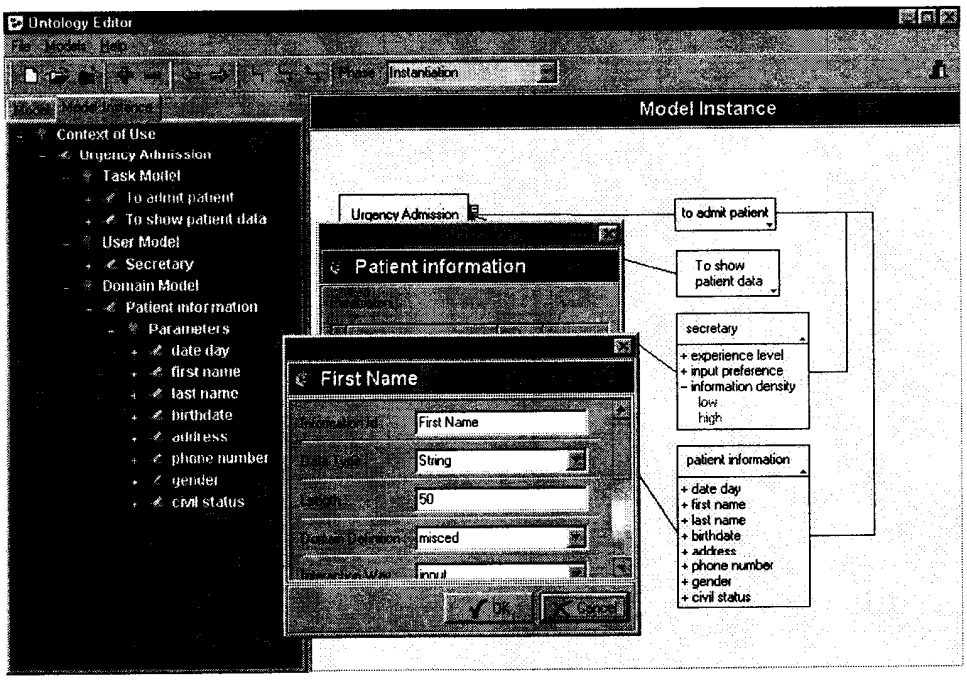

Figure 4- The ontology editor at the instantiation stage.

The models instantiated at the logical level are all based on the same ontology. The **big win** is that when the ontology changes, all associated models change accordingly since the ontology is used as a reference input for the graphical editor. The graphical nature of the editor improves the legibility and the communicability of information, while information which cannot be represented graphically is maintained in text properties. The models serve for both requirements documentation and UI production in the next level.

3.3.3 The Generation of UIs from the Models

The main goal of this phase relies in its ability to exploit instantiations captured in individual models to produce multiple UIs. It is highly desirable to have a tool to help developers in deriving multiple UIs from each set of models. Here, we are using SEGUIA (fig. 4), a model-based interface development that is capable of automatically generating MS Visual Basic code for a running UI from any specification file. The UIs generating process occurs in two stages: the first one, by generating the abstract UIs, which are independent of a particular graphical window manager, UIMS or toolkit and the second one, by generating the concrete UIs, which are dependent of them. The detailed generation process from the models is described in [(VANDERDONCKT & BERQUIN, 1999) e (FURTADO et al, 2001)].

Seguia (VANDERDONCKT, 1999) tool implements the abstract UIs generating process by selecting appropriated abstract interaction objects according to the model parameters and by placing these objects into screen with a visual and logical

arrangement. The activity of interaction object selection is possible by the application of selection rules over the model parameters. Selection rules are graphically represented in a decision tree (VANDERDONCKT & BODART, 1993). The abstract interaction objects are after transformed into concrete interaction objects, which are now environment dependent. The concrete interaction objects are distributed in the screen space following three techniques: physical object localization, appropriate and aesthetic sizing and ergonomical arrangement.

Figure 5 shows an example of a final UI generated to the *admit patient* task. The screen is composed of many concrete interaction objects related to *patient information*, such as first name, last name and so on. The concrete interaction object placement is based on the applicability of following placement rules: i) edit boxes and groups are arranged in a single vertical column and are left justified in the root window to which they belong; ii) identification labels of objects are placed in a single vertical column, are left justified between themselves and; iii) action are vertically justified.

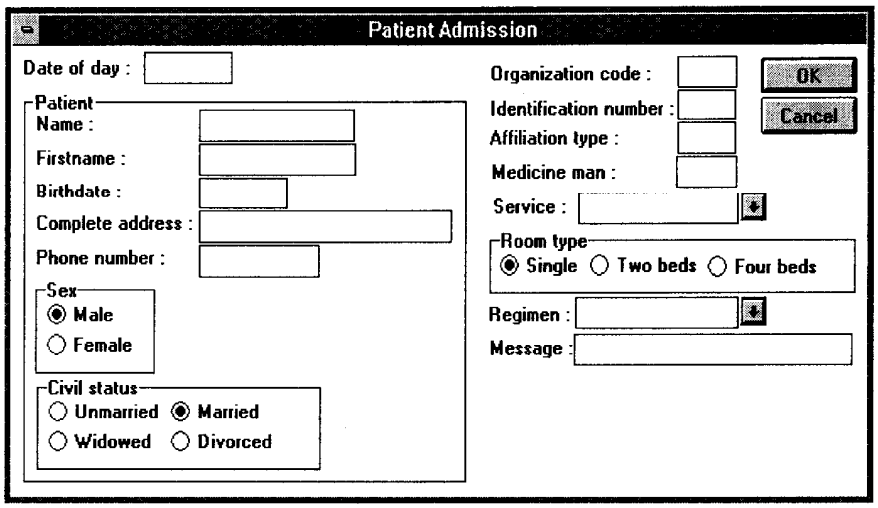

Figure 5- Example of the final UI.

This level allows sharing or reusing previously defined models for several UI designs, which is particularly useful when working in the same domain where similar information can be found. It also encourages users to work at a higher level abstraction than merely the code level and to explore multiple UI alternatives for the same UI design case. This flexibility may even produce UIs with unforeseen, unexpected or under-explored features. The **big win** is that when the set of models change, all UIs that can be created from this set can change accordingly. The *design space* is often referred to as the set of all possible UIs that can be created from an initial set of models for one UI design.

4 Conclusion

The main contributions and benefits of the method presented in this paper are: UI design method can be explicitly structured into three separate levels (i.e., conceptual, logical, and physical). The three levels make it possible to apply the "separation of concern" principle: (i) a definition of useful concepts first by someone who is aware of UI techniques such as user-centered design, task analysis, and human factors; (ii) a model definition where, for each UI design, multiple sets of models can be defined on the same basis with no redefinition of previously defined concepts; and (iii) multiple UI creation: for each set of UI models, several UIs can be created by playing with parameters supported by the UI generator and manual editing is allowed when needed, thus achieving the goal stated in the introduction.

5 Acknowledgements

We wish to thank CNPq and UNIFOR, who have sponsored this project, and extend a special thanks to our team. Our deepest thanks go to: Vasco Furtado, Leandro da Silva Taddeo, Daniel William T. Rodrigues, Wilker Bezerra Silva and Quentin Limbourg.

References

BUTLER, K. A. Designing deeper: towards a user-centered development environment design in context. In: ACM SYMPOSIUM ON DESIGNING INTERACTIVE SYSTEMS: PROCESSES, PRACTICES, METHODS & TECHNIQUES. 1995. New York. *Anais*... New York: ACM Press, 1995, p. 131-142.

CARD, S.; MORAN, T.P.; NEWEL, A. *The psychology of human-computer interaction.* Hillsdale: Lawrence Erlbaum Associates, 1983, 85 p.

FURTADO, E. *Mise en oeuvre d'une méthode de conception d'interfaces adaptatives pour des systémes de supervision à partir des spécifications conceptuelles.* 1997. Thèse de doctorat. AIV – Marseille III. Marseille.

FURTADO, E. et al. An ontology-based method for universal design of user interfaces. In: WORKSHOP ON MULTIPLE USER INTERFACES OVER THE INTERNET. 2001, Lille. *Anais*... Lille: Engineering and Applications Trends, 2001, p. 68-82.

GAINES, B. A situated classification solution of a resource allocation task represented in a visual language, special issue on models of problem solving,. *International Journal of Human-Computer Studies*, Netherlands, v. 40, n. 2, p. 243-271, Oct. 1994.

GIMNICH, R.; KUNKEL, K.; REICHERT, L. A usability engineering approach to the development of graphical user interfaces. INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION, 4., 1991, Stuttgart. *Anais*... Stuttgart: HCI International, 1991, p. 673-677.

GUARINO, N. Formal ontology, conceptual analysis and knowledge representation: the role of formal ontology in the information technology. *International Journal of Human-Computer Studies*, New York, v. 43, n. 5, p. 625-640, Oct. 1995.

HARTSON, H.R.; HIX, D. Human-computer interface development: concepts and systems for its management. *ACM Computing Surveys*, New York, v. 21, n. 1, p. 241-247, Oct. 1989.

HIX, D. Developing and evaluating an interactive system for producing human-computer interfaces. *Behaviour and Information Technology*, New York, v. 8, n. 4, p. 285-299, Aug. 1989.

LIM, K. Y.; LONG, J. Structured notations to support human factors specification of interactive systems notations and tools for design. *In: BCS CONFERENCE ON PEOPLE AND COMPUTERS, 9.*, 1994, Cambridge. *Anais*... Cambridge: Cambridge University Press, 1994, p. 313-326.

PUERTA, A. R. A Model-based interface development environment. *IEEE Software*, Los Alamitos, v. 14, n. 4, Jul./Aug. 1997. Disponível em: <http://www.arpuerta.com/pubs/ieee97.htm>. Acesso em 10 de maio de 2001.

SAVIDIS, A.; AKOUMIANAKIS, D.; STEPHANIDIS, C. *The unified user interface design method.* Mahwah: Lawrence Erlbaum Associates, 2001, 440 p.

TOP, J.; AKKERMANS, H. Tasks and ontologies in engineering modelling. *International Journal of Human-Computer Studies*, New York, v. 41, n. 4, p. 585-617, Aug. 1994.

VANDERDONCKT, J. Advicigiving systems for selecting interaction objects. In: INTERNATIONAL WORKSHOP ON USER INTERFACES TO DATA INTENSIVE SYSTEMS, 1., 1999, Edinburg. *Anais*... Edinburg: IEEE Computer Society Press, 1999, p. 152-157.

VANDERDONCKT, J., BERQUIN, P. Towards a very large model-based approach for user interface development, INTERNATIONAL WORKSHOP ON USER INTERFACES TO DATA INTENSIVE SYSTEMS, 1., 1999, Edinburg,. *Anais*... Edinburg: IEEE Computer Society Press, 1999, p. 76-85.

VANDERDONCKT, J. BODART, F. Encapsulating knowledge for intelligent automatic interaction objects selection. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 1993, Amsterdam. *Anais*... Amsterdam: ACM Press, 1993, p.424-429.

Elizabeth Furtado

B.Sc. (Informatics) at Universidade Federal do Ceará (1986). Master in Computer Science at Universidade Federal do Ceará (1993). Doctor in Computer Science - Engineering of Software at Universidade d'Aix Marseille (1997). Professor at Universidade de Fortaleza. Chairperson of NATI/EAD at Universidade de Fortaleza.

Jean Vanderdonckt

B.Sc. (Mathematics) at University of Notre-Dame de la Paix (1987). Master in Computer Science at University of Notre-Dame de la Paix (1989). Ph.D. in Comput Science at University of Notre-Dame de la Paix (1997). Post-Doctorate in Human-Computer Interaction at Stanford University (2000). Associate Professor, Information Systems and Quantitative methods Unit (QANT), School of Management (IAG), Catholic University of Louvain (UCL).