Towards A Better Model Based User Interface Development Environment: A Comprehensive Survey

Mohammed Gomaa, Akram Salah and Syed Rahman
Computer Science Department
North Dakota State University
258 IACC, Fargo, ND 58105
{mohammed.gomaa, akram.salah, syed.rahman}@ndsu.edu

Abstract

The introduction of user devices with built-in computer programs has introduced a number of challenges to the design of user interfaces. Automating the management and generation of interfaces greatly improves their quality and maintainability and significantly reduces the cost of development. Model-based user interface development environments (MBUIDEs) are tools that help designers with building interfaces through automating the generation of interfaces using high-level declarative models.

In this paper, surveyed different interface generation techniques and built a framework to compare and analyze their suitability to handle the changes imposed by universal usability. The paper points out limitation with current techniques and proposes the use of a multi-model conceptual layer that will be used as a management system to control the specification, creation, and manipulation of the interfaces. We claim this framework will be able to overcome many of the limitations of today's techniques in facing the above mentioned challenges.

1. Introduction

User interface is a very important part in software development. An average of 48% of the code of applications is devoted to user interface, and about 50% of the implementation time is devoted to implementing the user interface portion [46].

Automating user interface improves the quality of developed interfaces and makes the creation of interfaces more economical and maintainable. Several approaches have been proposed to automate the creation of user interfaces. User Interface Management Systems (UIMS) were first proposed with an analogy to Database Managements Systems (DBMS).

Model-based user interface development environments (MBUIDEs) were introduced later to overcome problems faced with UIMSs. They are tools that support the design and development of user interfaces through the use of abstract interface models. There are two generations [44] of MBUIDEs that appeared as improvements to the previous UIMSs. The first generation was basically aiming at providing a strategy to generate a user interface from the high-level models. The tools of this generation emphasized the automatic generation of an interface instead of a user interface design process. The second generation of MBUIDEs stressed the involvement of users in the development process of interfaces and started MBUIDEs that are user-centered. In second generation, the interface model was described in better ways. Tools of this generation supported the incremental interface design.

The last decade has seen the introduction of a large number of appliances and devices that have built-in computer programs and are used to access several information sources. This has introduced a number of challenges to the design and creation of user interfaces. The wide range of users, applications and devices requires interface designers to provide user interfaces that will be flexible enough to satisfy the wider range of requirements without being redesigned for every use. The term universal interfaces is used to describe interfaces that run on several platforms and are multilingual. A number of other terms were coined to describe this like ubiquitous computing, smart interfaces, universal usability, interface plasticity and universal access. Universal usability describes products that should be usable by the widest range of people possible.

In this paper, we survey current and past interface generation techniques to find their suitability to face the challenges introduced by universal usability. We analyzed the current approaches and discussed their shortcomings in facing the universal usability problem. The work in this paper is part of a research project that aims to introduce a framework capable of addressing the challenges introduced by universal usability.

This paper is structured as follows. Section 2 gives a brief description of the surveyed techniques. Section 3 provides a framework to compare abstract models in different MBUIDEs and gives an analysis for the surveyed techniques. Conclusions and future work are provided in Section 4

2. Overview of environments

This section provides a brief discussion about the techniques surveyed. Table 1 presents the tools surveyed, the organizations in which they were developed, and suggested references for that technique. Follows is a brief discussion about those techniques and tools.

Tool	Year	Place	References
ADEPT	1995	Queen Mary and Westfield College	[1,3,4]
HUMANOID	1993	University of Southern California	[6, 7]
MASTERMIND	1995	University of Southern California, and Georgia Institute of Technology	[8, 9]
TADEUS	1995	University of Rostock	[10, 11]
MECANO	1995	Stanford University	[12, 13]
GENIUS	1993		[14]
TRIDENT	1993	Namur University	[15, 16, 17, 18, 20]
UIDE	1991	Georgia Institute of Technology	[21, 22, 23, 25, 26]
AME	1996	Fachbereich Informatik, Germany	[27]
FUSE	1996	University of Manchester, Napier University, University of Glasgow	[28, 29]
MOBI-D	1997	Stanford University	[30, 31, 32, 33, 34]
JANUS	1995	Ruhr University	[44]
ITS	1989	IBM T.J. Watson Research Center	[35]
Teallach	1999	University of Manchester, Napier University, University of Glasgow.	[38, 39, 40, 41, 42]
DRIVE	1995		[45]
Markopoulos approach	2000	Eindhoven University of Technology	[43]
UIML	1999	Garvin Innovation Center, Virginia Tech	[47, 48]
XUL	N/A	Mozilla	[58]
XIML	1999	RedWhale Software	[50,51]
Xforms	2003	W3C	[52]
Aurora	2000	IBM Almaden Research Center, NehaNet Corporation	[53]
Dygimes	2004	Limburgs Universitair Centrum, Belgium	[54]
TERESA	2003	Consiglio Nazionale delle Rierche, Italy	[55]
Pebbles	2002	Carnegie Mellon University, MAYA Design Inc	[49, 56]
AIAP	N/A	INCITS/V2	[57]

"Table 1: Surveyed techniques"

2.1. User Interface Management Systems (UIMS)

UIMS seemed a very promising approach in the early 80s. UIMSs were to abstract the details of input and output device. According to Myers [46], UIMSs failed due to what he called the moving-target problem; the standardization of the user interface elements in the late 80's on the desktop paradigm made the need for abstractions from the input devices unnecessary. We will give a brief description of ITS, one of the UIMSs.

2.1.1. ITS

ITS (WIECHA, 1989) [35] is a UIMS that offers a frame-based language for specification of interface in its logical structure. It also allows the specification of style rules, which describe the mapping between logical user interface (dialogue content) and style. ITS has no graphical specification technique. It employs different notation for describing dialogue content and presentation design rules. ITS defines action Layer, Dialog layer, style rule layer, and style program layer. Action layer implements back-end application functions. It is similar to application layer in other tools. Dialog layer defines the content of the user interface, independent of its style. Content specifies the objects included in each frame of the interface, the flow of control among frames, and what actions are associated with each object. Style rule layer defines the presentation and behavior of a family of interaction techniques. Style program layer implements an extensible toolkit of objects that are composed by the rule layer into complete interaction techniques.

2.2. First generation model based tools

2.2.1. HUMANOID

HUMANOID [6, 7] aimed to help maintaining a balance between having the designer handle a tremendous number of design details (as in interface builders), or limiting his control over design decisions (as in automatic interface tools). HUMANOID uses pre-defined presentation templates to solve layout generation problems. The models are not explicitly defined in HUMANOID. Rather, they are defined as five dimensions, namely Application Model, Presentation, Manipulation, Sequencing, and Action side effects.

HUMANOID uses a declarative language to express application semantics, presentation, input gestures and results, constraints on the ordering of commands and inputs, and side-effects of user actions. Presentation and gestures are defined using templates, whereas the dialogue constraints are derived from the application semantics. HUMANOID also provides a run-time system to control the designed interface. HUMANOID provides specialized editors to construct presentation templates and to specify all their attributes. For specifying layout, HUMANOID has a library of templates of commonly use layout methods such as rows, columns, tables and graphs. HUMANOID has a behavior model, based on Myers' Interactor, model that is used to specify behavior in the presentation. HUMANOID uses manipulation, sequencing, and action side effects models for what is called dialogue models in other tools.

2.2.2. UIDE

UIDE (Sukaviriya, 1992) [21, 22, 23, 24, 25, 26] was one of the early MBUIDEs in which the designer had to specify application actions, interface actions, and interactions techniques. Parameters, pre and post conditions were then assigned to each action. Pre and post conditions are used to control the interface. An extension to UIDE introduced at 1995 added some more features. The research on UIDE and HUMANOID was joint in the MASTERMIND project.

2.3. Second generation model-based tools

2.3.1. ADEPT

ADEPT (Johnson, 1995) [1, 3, 4] stands for "Advanced design environment for prototyping with tasks". It is a task-based [2], user-centered (user task-based) design environment that emphasizes the involvement of users. In design, task model is transformed into the specifications of logical interface components. Based on the design rules in a user model, a concrete interface is derived from the logical interface. The dialogue structure is included in the Abstract Presentation Model. Objects are modeled in Concrete Interface Model by the set of events it can recognize, the sequencing of events, and the behavior of the objects. ADEPT has tools namely Task, User, and Interface model editors for capturing and editing the task, user and interface models respectively.

2.3.2. MASTERMIND

MASTERMIND (Szekely, 1996) [8, 9] stands for Models Allowing Shared Tools and Explicit Representations Making Interfaces Natural to Develop. It is a continuation of the work on HUMANOID and UIDE and aims at inheriting the strengths of both environments. HUMANOID's strength lied in the presentation model, modeling tools and performance, where as UIDE's strength lied in the dialogue model, the design critics, and the help generation tools.

Models in MASTERMIND are shared via the model server. Whenever a model element in the model server is modified (by request of any tool), all tools that depend on the modified element are informed so that they can update their state. MASTERMIND's presentation model is similar to that of HUMANOID and ITS. MASTERMIND's main contribution is that its presentation model is designed to support graphical specification of presentations similar to that of interface builders.

2.3.3. TADEUS

TADEUS (Elwert, Schlungbaum 1995) [10, 11] stands for Task based development of user interface software. In TADEUS, the development process is divided into the phases: Requirements Analysis, Dialogue Design, and Realization. In the Requirements Analysis phase, hierarchy of goals, a class hierarchy, and the user models are specified. In Dialogue Design phase, static and dynamic aspects of the interface are described in terms of views and dialogue graphs (an extension of dialogue nets of GENIUS). In Realization phase, the logical user interface is transformed into interface description for a UIMS by applying software ergonomic guidelines specified through decision tables.

2.3.4. MECANO

MECANO [12, 13] was one of the early tools that were improved afterwards to MOBI-D to include more models to define the interface. A model editor is used to visualize and edit the domain model. The domain model is then used to generate high and low level dialogue specifications. High-level dialog defines all interface windows, assigns interface objects to windows, and specifies the navigation schema among windows in the interface while low-level dialog defines specific dialog elements (widgets) to each interface object created at the high level and specifies how the standard behavior of the dialog element is modified for the given domain.

2.3.5. GENIUS

GENIUS (Janssen, Weisbecker, & Ziegler 1993) [14] stands for Generator for user interfaces using software ergonomic rules. It generates interfaces for database oriented applications. The problem domain model is represented by an ERA (entity-relationship-attribute) diagram. Based on this ERA diagram, static aspects of the logical user interface are described in terms of views (abstract representations of windows). For each view in the logical user interface, the static UI layout is generated by applying software ergonomic guidelines which are described as decision tables.

2.3.6. TRIDENT

TRIDENT (Vanderdonckt, 93) [15, 16, 17, 18, 19, 20] stands for Tools for an interactive development environment. It is a tool for developing interfaces for business-oriented applications. Development in TRIDENT starts with task analysis which results in a hierarchical decomposition of the application into tasks and sub-tasks, goals, actions, objects of tasks, relationship between tasks, prerequisite of tasks, user stereotypes, and a description of the workplace. An extended version of entity-relationship-attribute (ERA) model is then built to describe the object structure and relations to be manipulated by the user. An activity chaining graph (ACG), which is a graph describing the information flow between the application domain functions which are necessary to perform the task goal is then built. Interaction styles (natural language, command language, query language, questions/answers, function keys, menu selection, form filling, multi-windowing, direct manipulation, iconic interaction...) are defined next. Once the appropriate interaction styles have been selected and the ACG has been completed, we are ready to start the definition of the presentation.

2.3.7. AME

AME (Martin, 1996) [27] stands for Application Modeling Environment. AME is a prototypical MBUIDE that is targeted at business applications. AME introduced the idea of concurrent lifecycles where the development process is divided into activities for the user interface and activities for the domain functionality of the system. Therefore, AME integrates object-oriented and knowledge-based tools and is able to model, prototype and generate business applications with graphical user interfaces.

2.3.8. FUSE

FUSE (Lonczewski, 96) [28, 29] stands for Formal User Interface Specification Environment. It is a tool for formal specification and automatic generation of UIs. It also emphasizes the automatic generation of user help modules.

2.3.9. MOBI-D

MOBI-D [30, 31, 32, 33, 34] stands for Model-Based Interface Designer. It is the successor of MECANO. It aims at solving the mapping problem between different abstraction levels for objects in models of interfaces e.g. purely abstract units in such as a user tasks and very concrete units, such as scrollbars and pushbuttons. Developers of MOBI-D believe that the mapping problem defies automation because of the number of variables that can impact each possible mapping. Instead of automation, they propose that model-based systems provide tools that allow developers to interactively set the mappings. The mapping issue is addressed according to three aspects: mapping of domain objects with interactors according to some priorities; style attributes controlling some graphical and textual attributes; and strategy preferences indicating the preferred number of windows, the preferred way to implement sequential constraints, and the preferred interaction and navigation modalities.

2.3.10. JANUS

JANUS [44] emphasizes the use of object-oriented domain model to generate the interface. The output is source code for C++.

2.3.11. TEALLACH

The goal of the Teallach [38, 39, 40, 41, 42] project was to provide facilities for the systematic development of interfaces to object databases in a manner which is independent of both a specific underlying database and operating system. It also allows the creation of interfaces to non database applications in a platform-independent manner. There is a contradiction in Teallach's literature about the models used in the environment. User model is referenced and used in some of the literature and not in others. Teallach's interfaces are generated as compiled Java applications.

2.3.12. DRIVE

DRIVE (Mitchell, 1995) [45] stands for Database Representation Independent Visual Environment. It was explicitly aimed at producing interfaces to databases rather than applications in general.

2.3.13. Markopoulos approach.

Markopoulos' approach [43] considers using UML for modeling due to its popularity within the software developers. It has a scenario model that is used for envisioning systems at early design stages.

2.4. Tools for universal computing

Several researchers are currently working to face the challenges imposed by the universal usability standards. A number of tools are currently under development as a result of the research made. A brief description of those tools is given below.

User Interface Markup Language (UIML) [47, 48] is an appliance-independent XML meta-language for describing interfaces. It takes the approach of building applications using a generic vocabulary that could then be rendered for multiple platforms. A special renderer for each target device is needed. It could be argued that having multiple renderers is a disadvantage. UIML does not take into account any of the advances of Model Based User Interface Development Environment reached in the past decade because it does not allow the abstraction of interaction functionality.

XML User Interface Language (XUL) [58] is a markup language used for describing user interfaces. It has its focus on window-based UIs by XML. Its disadvantage is the limitation to such graphical user interfaces, which makes it unsuitable to interfaces of small mobile devices.

Extensible Interface Markup Language (XIML) [50, 51] is an XML based language that is intended to be a universal user interface specification language. Research is still underway with XIML but it seems to be promising.

The W3C consortium has recently delivered a new standard, Xforms [52], that presents a description of the architecture, concepts, processing model, and terminology underlying the next generation of web forms based on the separation between the purpose and the presentation of a form.

Aurora [53] is a model based tool that uses a transaction model to adapt web pages to support universal usability. Aurora has a limited scope of web pages.

Dygimes [54] exploits Distributed User Interfaces that allow users to share information on their devices with other people using different devices by allowing mobile users to use the surrounding devices. It introduces an interaction model that describes the communication between the generated interface and the application logic for which the interface is rendered.

TERESA [55] is also an XML based language that uses abstract models to generate interfaces for multiple platforms and devices. Currently they only concentrate on web applications.

Developers of the Pebbles project [49, 56] propose that every user would carry a personal universal controller device that allows him to interact with all the existing appliances in the environment. Personal universal controllers will be different depending on the users' needs. It could be a regular handheld machine, or one with Braille surface, or one with speech output and recognition.

The International Committee for Information Technology Standards (INCITS) is currently developing the Alternative Interface Access Protocol (AIAP [57]), which is a standard to help disabled people use everyday appliances. It contains a description language for appliances that can be used by interface generators.

3. Models

One goal of MBUIDEs was to face the limitations of UIMSs. MBUIDEs are suites of software tools that support designing and developing user interfaces by creating interface models. Puerta [59, 60] defines an interface model as a computational representation of all the relevant aspects of a user interface. Szekely [61] defines an interface model as high-level declarative specification of some single coherent aspect of a user interface, such as its appearance, layout characteristics and dynamic behavior. By focusing attention on a single aspect of a user interface, a model can be expressed in a highly specialized notation. Different MBUIDEs define different set of models to describe the interface.

In order to analyze the different techniques, we created a framework where we defined task, application, user, presentation, and dialogue models. One problem we faced was the conflicts in the literature in the definitions and scopes of different models used by different environments where authors were referring to the same models with different meaning and so on. In order to overcome this problem, we have used our defined models as a basis for comparison. Table 2 below presents the different models. Between square brackets, the name as used by the creator of the tool will be mentioned to the reader.

• *Task model.* A task is described by a goal, actions needed to achieve the goal, and a plan of how to select the actions. The task model describes the tasks that the user can perform with a system including sub-tasks, their goals, and the procedures used to achieve the goals.

• *Application model.* Application model specifies the information about an application that is independent of how the objects are displayed, and how the operations are invoked. It is usually a hierarchy of classes

• *User model.* User model describes prospective users of the systems in terms of their abilities, knowledge and styles of information processing.

• *Presentation model.* Presentation model describes the visual aspects of the interface. It is divided into Abstract presentation model and concrete presentation model. Abstract presentation model provides an abstract view of an interface that is independent from the underlying concrete model. Concrete Presentation model is the concrete instance of an interface which can be presented to a user; there may be many concrete instances of an abstract presentation model.

• *Dialogue model.* Dialogue model defines the procedural characteristics of the human-computer dialogue in an interface model.

Environment	Task Model	Domain Model	User Model	Abstract Presentation Model	Concrete Presentation Model	Dialogue Model
ADEPT	⊠	⊠ [Problem domain]	⊠	⊠ [abstract interface model]	⊠ [concrete interface model]	
HUMANOID		⊠ [application model]		⊠ [presentation model]		⊠
MASTERMIND		⊠ [application model]		⊠	⊠	
TADEUS	⊠	⊠ [problem domain]	⊠	⊠	⊠	⊠
MECANO	⊠	⊠	⊠	⊠ [Interface model]	⊠ [Presentation Model]	⊠
GENIUS		⊠ [Data model]		⊠		⊠

TRIDENT	⊠	⊠ [data model]		⊠		
UIDE		⊠ [application model]		⊠ UI model (after 1995)		
AME		⊠				
FUSE	⊠	⊠ [problem domain model]	⊠	⊠		
MOBI-D	⊠	⊠ [application model]	⊠	⊠ [presentation model]		⊠
JANUS		⊠ [problem domain model]			⊠ [user interface model]	
ITS						⊠
Teallach	⊠	⊠	⊠	⊠	⊠	
DRIVE		⊠ [application model]	⊠	⊠ [interface model]		
Markopoulos approach	⊠	⊠		⊠	⊠	

"Table 2: Different Models in MBUIDEs"

3.1. Analysis of models

• Domain model and application model were used interchangeably in literature. We believe that these should be considered as two different models. We will adopt the current definition for application model but add a domain model that describes the characteristics of the domain for which the application is developed e.g. accounting, engineering …
• Application model is only referenced in HUMANOID, MASTERMIND, UIDE, MOBI-D, and DRIVE. DRIVE uses this model to describe non database applications. The definition of application model in UIDE, HUMANOID and MASTERMIND is similar to the definition of task model in other tools.
• Most MBUIDEs have task, domain models, and presentation models. Some of them divide the presentation to concrete and logical and some do not.
• All MBUIDEs use either dialogue or presentation models or both.
• User, implementation, and workplace models are referenced in some of the literature but never actually applied towards the implementation of the interface.

• In recent years, research in MBUIDEs to support universal usability has been adopted by several companies rather than just academic interest.
• Some MBUIDEs define tools that are not used in any other tools. MECANO has a *design model* which provides a collection of design mappings that establish design relationships among interface objects. MOBI-D is an extension to MECANO that adds task, domain, user, dialogue and presentation models. TRIDENT has an *architectural model* that is a hierarchical object-oriented architecture relying on the use of three kinds of objects: application objects, dialog objects, and interaction objects. Markopoulos' model has a *scenario model* that is used for envisioning systems at early design stages.
• Different notations are used in the different MBUIDEs. Some MBUIDEs use existing modeling technique and other tools developed their own notations. Some use textual notation while others use graphical notations. Also, some MBUIDEs use the same notation for describing all the models e.g. MOBI-D's MIMIC. ITS uses style rules. Other tools use other notations or extensions to other notations like CORBA IDL, CTT, UML, UAN, HIT, ACG, and Petri-nets. Most MBUIDEs tend to use more than one notation. We believe this better fits the different types of information included in different models.

3.2. Limitations of current modeling techniques

Automated user-interface generation environments have been criticized for their failure to deliver rich and powerful interactive applications [62]. The main limitations of today's model-based tools are:
• The modeling languages of existing model-based tools suffer from a lack of flexibility. They are not expressive enough to give developers adequate ways to control all the features of the interface needed for real applications.
• Most model-based tools are experimental, and thus not tuned for performance.
• Most model-based tools are hard to use, especially when compared with interface builders. Most model-based tools require models to be specified in a specialized modeling language. Thus modeling becomes a form of programming, which is not a skill many interface developers have or wish to learn.
• User models are supported in some UIMs (ADEPT, MECANO, and TADEUS). They are a very challenging aspect of the UI not well addressed in MBUIDEs. In those MBUIDEs that have a user model, it appears that the user model can be replaced by design guidelines.
• No standard framework and notation are used in the second generation of MBUIDE
• Editing generated interfaces after the generation process is usually not possibly. The generated interfaces are rigid and un-editable because they are compiled into the application. There is no management system to control the specification, creation, and manipulation of the interfaces.
• Mapping between models of different abstract levels represents a common problem in MBUIDEs.

4. Conclusions & Future work

We performed a comprehensive survey for interface generation tools. We analyzed the surveyed tools to see the extent of their suitability to support universal usability issues and identified existing limitations.

Most researchers agree that a multi-model tool is required to provide an abstract way to describe different characteristics of the users and application to which the interfaces are to be developed.

We believe that we need a multi-model conceptual level that uses abstract models to allow the specification, manipulation, management and configuration of user interfaces. This layer will separate the interface from underlying applications, users, platforms and devices. We believe that this architecture will have enough flexibility to solve many of the challenges introduced by universal interfaces. The work in this paper is part of a research project that aims to introduce a framework capable of addressing the challenges introduced by universal usability. This framework will have its architectural basis built upon the conclusions learned from this survey

References

[1] P. Markopoulos, J. Pycock, S. Wilson, and P. Johnson, "Adept - A task based design environment", In Proceedings of the 25th Hawaii International Conference on System Sciences, IEEE Computer Society Press, 1992, pp. 587-596.
[2] S. Wilson, and P. Johnson, "Empowering users in a task-based approach to design, Designing Interactive Systems: Processes, Practices, Methods, & Techniques", In Proceedings of the conference on Designing interactive systems: processes, practices, methods, & techniques, ACM Press, New York, 1995, pp. 25-31.
[3] P. Johnson, S. Wilson, P. Markopoulos, J. Pycock, "ADEPT-Advanced Design Environment for Prototyping with Task Models" , In Proc. of InterCHI'93, Amsterdam, 24-29 April 1993, ACM Press, New York, pp. 56.
[4] S. Wilson, P. Johnson, C. Kelly, J. Cunningham, and P. Markopoulos, “Beyond Hacking: a Model Based Approach to User Interface Design.” HCI'93, Loughborough, U.K. Cambridge University Press, 1993, pp. 217-31.
[5] S. Wilson and P. Johnson, "Bridging the Generation Gap: FromWork Tasks to User Interface Designs", In Computer-Aided Design of User Interfaces, Namur University Press, Namur, Belgium, 1996, pp. 77-94.
[6] P. Luo, P. Szekely, and R. Neches, "Management of interface design in HUMANOID", In Proceedings of InterCHI'93, ACM Press, New York , April 1993, pp.107-114.
[7] P. Szekely, P. Luo, and R. Neches, "Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design", In Proceedings of SIGCHI'92, ACM Press, New York, May 1992, pp. 507-515.
[8] P. Szekely, P. Sukaviriya, P. Castells, J. Muthukumarasamy, and E. Salcher, "Declarative Interface Models for User Interface Construction Tools: the MASTERMIND Approach", In

Engineering for Human-Computer Interaction, Chapman & Hall, London, UK, 1996, pp. 120-150.
[9] T. Browne, D. Dávila, S. Rugaber, and K. Stirewalt, Formal Methods in Human-Computer Interaction, Springer, Verlag, 1997.
[10] E. Schlungbaum, and T. Elwert, "Dialogue Graphs - A Formal and Visual Specification Technique for Dialogue Modeling", In proc. of the BCS-FACS Workshop on Formal Aspects of the Human Computer Interface, Springer, London, 1996.
[11] T. Elwert, "Continuous and Explicit Dialogue Modelling", In Proceedings Conference on Human Factors in Computing Systems, ACM Press, New York, 1996, pp. 265 - 266.
[12] A. Puerta, H. Eriksson, J.H. Gennari, and M.A. Musen, "Model-Based Automated Generation of User Interfaces", In Proceedings of the National Conference on Artificial Intelligence, 1994, pp. 471477.
[13] A.R. Puerta, "The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development", In Proc. of the 2nd International Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Presses Universitaires de Namur, 5-7 June 1996), 1996, pp. 19-36.
[14] C. Janssen, A. Weisbecker, and J. Ziegler, "Generating user interfaces from data models and dialogue net specifications", In Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press, New York, 1993, pp. 418 - 423.
[15] F. Bodart, A.M. Hennebert, J.M. Leheureux, and J. Vanderdonckt, "Computer-Aided Window Identification in TRIDENT", In Proc. of the 5 IFIP TC13 Conf. on Human Computer Interaction Interact'95 ,Chapman & Hall, London, 1995, pp. 331-336.
[16] F. Bodart, A.M. Hennebert, J.M. Leheureux, I. Provot, J. Vanderdonckt, and G. Zucchinetti, Key Activities for a Development Methodology of Interactive Applications , Critical Issues in User Interface System Engineering, Springer Verlag, Berlin, 1995.
[17] F. Bodart, A.M. Hennebert, J.M. Leheureux, I. Provot, and J. Vanderdonckt, "A Model Based Approach to Presentation: A Continuum from Task Analysis to Prototype." In proc. of 1st Eurographics Workshop on Design Specification and Verification of Interactive System (DSVIS '94), Eurographics, Carrara, Italy, 1994, pp. 25-39.
[18] F. Bodart, J. Vanderdonckt, "On the Problem of Selecting Interaction Objects", In Proc. of People and Computers IX (HCI`94), Cambridge University Press, Cambridge, 1994, pp. 163-178.
[20] F. Bodart, A.M. Hennebert, J.M. Leheureux, I. Sacre, and J. Vanderdonckt, Architecture elements for highly-interactive business-oriented applications. Lecture Notes in Computer Science (volume 753 of LNCS), Springer, Verlag, 1993.
[21] J. Foley, W. Kim, S. Kovacevic, and K. Murray, "Defining Interfaces at a High Level of Abstraction ", IEEE Software (Vol. 6, No. 1), IEEE, 1989, pp. 25-32.
[22] J. Foley, A van Dam, S. Feiner and J. Hughes, Computer Graphics: Principles and Practice, Addison-Wesley, 1990.
[23] J. Foley: User Interface Software Tools. Research report GIT-GVU-91-29
[25] J. Foley, C. Gibbs, W. Kim, S. Kovacevic, "A Knowledge-Based User Interface Management System", In Proceedings of the 1988 Conference on Human Factors in Computer Systems (CHI'88), ACM, New York, 1988, pp. 67-72.
[26] P. Sukavariya, J.D. Foley, and T. Griffith, "A Second Generation User Interface Design Environment: The Model and The Runtime Architecture", In Proceedings of the Conference on Human Factors in Computing Systems INTERCHI '93, ACM Press, Amsterdam, April 1993, pp. 375-382.

[27] C. Martin, Software Life Cycle Automation for Interactive Applications: The AME Design Environment, Computer-Aided Design of User Interfaces, Namur University Press, Namur, Belgium, 1996.
[28] F. Lonczewski, and S. Schreiber, "The Fuse System: an Integrated User Interface Design Environment", In Proceedings of 2nd International Workshop on Computer-Aided Design of User Interfaces CADUI `96, Presses Universitaires de Namur, Namur, 1996, pp. 37-56.
[29] F. Lonczewski, "Providing User Support for Interactive Applications with FUSE", In Proceedings of the 2nd international conference on Intelligent user interfaces, ACM Press, Orlando, 1997, pp. 253 - 256.
[30] J. Eisenstein and A. Puerta, "Adaptation in Automated User-Interface Design", In Proc. of ACM International Conference on Intelligent User Interfaces IUI'2000, ACM Press, New Orleans, 2000, pp. 74 - 81.
[31] A.R. Puerta, and D. Maulsby, "MOBI-D: A Model-Based Development Environment for User Centered Design", in proceedings of CHI'97, ACM Press, Atlanta, March 1997, pp. 4-5.
[32] A. Puerta, and J. Eisenstein, "Towards a General Computational Framework for Model-Based Interface Development", In Proceedings of IUI 1998, ACM Press, San Francisco, 1999, pp. 171-178.
[33] A.R. Puerta, and D. Maulsby, "Management of interface design knowledge with MOBI-D", in proc. international conference on intelligent user interfaces, ACM Press, Orlando, 1997, pp. 249-252.
[34] J. Eisenstein, and A. Puerta, "TIMM: Exploring Task-Interface Links in MOBI-D", CHI98 Workshop on From Task to Dialogue: Task-Based User Interface Design, ACM Press, Los Angeles, April 1998.
[35] C. Wiecha, W. Bennett, S. Boies, J. Gould, and S. Greene, "ITS: A Tool for Rapidly Developing Interactive Applications", ACM Transactions on Information Systems 8, ACM Press, New York, 1990, pp. 204-236.
[36] F. Paterno, Task Models for Interactive Software Systems in Handbook of Software Engineering & Knowledge Engineering, World Scientific Publishing Co., 2001.
[37] P. Gray, R. Cooper, J. Kennedy, J. McKirdy, P. Barclay, and T. Griffiths, "A Lightweight Presentation Model for Database User Interfaces", In Proc. of 4 th ERCIM Int. Workshop on User Interfaces for All, Stockholm, 19-21 October 1998.
[38] T. Griffiths, P.J. Barclay, J. McKirdy, N.W. Paton, P.D. Gray, J. Kennedy, R. Cooper, C.A. Goble, A. West, and M. Smyth, "Teallach: A Model-Based User Interface Development Environment for Object Databases", in Proc. User Interfaces to Data Intensive Systems (UIDIS99),IEEE Computer Society Publishers, Edinburgh, 5-6 September, 1999, pp. 86-96.
[39] P.J. Barclay, T. Griffiths, J. McKirdy, N.W. Paton, R. Cooper, and J. Kennedy, "The Teallach Tool: Using Models for Flexible User Interface Design", in pro. 3rd International Conference on Computer-Aided Design of User Interfaces (CADUI'99), Louvain-la-Neuve (Belgium), 21-23 October 1999.
[40] P. da Silva, T. Griffiths, N. Paton, "Generating User Interface Code in a Model Based User Interface Development Environment", in Proceedings of Advanced Visual Interfaces, 2000, pp. 155-160.
[41] P. Barclay, J. Kennedy, "Teallach's Presentation Model", in proc. AVI, Palermo, 23- 26 May 2000, pp. 151-154.

[42] T. Griffiths, J. McKirdy, N. Paton, J. Kennedy, R. Cooper, B. Barclay, C. Goble, P. Gray, M. Smyth, A. West, and A. Dinn, "An Open Model-Based Interface Development System: The Teallach Approach", in proc. DSV-IS, Eurographics, June 1998, pp. 32-49.
[43] P. Markopoulos, and P. Marijnissen, "UML as a representation for Interaction Designs", in Proceedings of OZCHI 2000, Academic Press ,Sydney, 2000, pp.240-249.
[44] H. Balzert, "From OOA to GUI - The JANUS-System", in Proceedings of INTERACT'95, Chapman & Hall, London, June 1995, pp. 319-324.
[45] K. Mitchell, J. Kennedy, and P. Barclay, "Using a Conceptual Data Language to Describe a Database and it Interface", in Proceedings of British National Conference on Databases 13, Manchester, 1995, pp. 101-119.
[46] B. Myers et al. Past, Present, and Future of User Interface Software Tools ACM Transactions on Computer-Human Interaction (TOCHI) 7(2000), no. 1, 3-28.
[47] Abrams, M., Phanouriou, C., Batongbacal, A. Williams, S., and Shuster, J., UIML: An appliance-independent XML User Interface language, in Proc. Of WWW'8 (Toronto, May 1999)
[48] Mir Farooq Ali, Manuel A. Perez Quinones, Eric Shell, Marc Abrams. Building multi-platform user interfaces with UIML.
[49] Jeffrey Nichols, Brad Myers, Kevin Litwack, Michal Higgins, Joseph Hughes, Thomas Harris. Describing Appliance User Interfaces Abstractly with XML.
[50] Angel Puerta and Jacob Eisenstein. XIML: A Common Representation for Interaction Data, in Proceedings of the 2002 International Conference on Intelligent User Interfaces, January 13-16, 2002, San Francisco, California, USA. ACM, 2002.
[51] Angel Puerta and Jacob Eisenstein. XIML: A Universal Language for User Interfaces
[52] Xforms 1.0, October 2003, Available at: http://www.w3.org/TR/xforms/.
[53] Anita Huang, Neel Sundaresan. Aurora: A conceptual Model for Web-content Adaptation to support the Universal Usability of Web-based services, In Proceedings on the 2000 conference on Universal Usability, Arlington, Virginia, ACM Press, 2000, pp. 124-131
[54] Chris Vandervelpen. Karin Coninx. Towards Model-Based Design Support for Distributed User Interfaces. NordiChi' 04. October 2004, Tampere, Finland, ACM.
[55] Giulio Mori, Fabio Paterno, Carmen Santoro. Tool Support for Designing Nomadic Applications. IUI'03. January 12-15, 2003, Miami, USA. ACM
[56] Jeffrey Nichols, Brad Myers, Thomas Harris, Michal Higgins, Joseph Hughes. Requirements for Automatically Generating Multi-Modal Interfaces for Complex Appliances, Proceedings of the 4th IEEE International Conference on Multimodal Interfaces, October 14 - 16, 2002, Pittsburgh, Pennsylvania , pp. 377, IEEE.
[57] AIAP, www.incits.org/tc_home/v2.htm.
[58] XUL, www.xulplanet.com
[59] AR. PUERTA, A Model-Based Interface Development Environment, IEEE, 97.
[60] AR Puerta, E Cheng, T Ou, J Min. MOBILE: User-Centered Interface Building, Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, Pittsburgh, Pennsylvania, 1999 pp. 426-433, ACM.
[61] P Szekely, P Sukaviriya, P Castells, J. Declarative interface models for user interface construction tools: the MASTERMIND approach. Engineering for Human-Computer Interaction, 1996
[62] REK Stirewalt, S Rugaber. The model-composition problem in user-interface generation. Automated Software Engineering, 2000.