

## Chapter 20

# IDEALXML: AN INTERACTION DESIGN TOOL

## *A Task-based Approach to User Interface Design*

Francisco Montero and Víctor López-Jaquero

*Laboratory on User Interaction & Software Engineering (LoUISE)*

*Instituto de Investigación en Informática (I3A)*

*University of Castilla-La Mancha - 02071 Albacete (Spain)*

*E-mail: {fmontero, victor}@info-ab.uclm.es*

*Tel: +34 967/599200 – Fax: +34 967/599224 – Web: <http://www.i3a.uclm.es>*

**Abstract** Task modeling has become one of the cornerstones of model-based user interface design. In this paper, a task-based approach to user interfaces design is introduced. This approach is supported by a tool, namely IDEALXML, that allows for the animation of the specified user interfaces to generate a *hi-fi* prototype of the future user interface while still in the first development stages

**Keywords:** Model-based design, Specification animation, Task modeling, User interfaces design tools, User interface extensible markup language

## 1. INTRODUCTION

Nowadays, software engineers use rapid prototyping to discover requirements by analyzing the prototypes built early in the development process and gathering feedback. In this paper, we address fast *hi-fi* prototyping within a model-based user interface (UI) environment [17]. This approach is supported by a powerful visual tool, namely IDEALXML [12,13]. UI design following the proposed approach is driven by task and domain models using a seamless mapping technique. The task and the domain models are mapped together thanks to a set of mappings which express how data from the domain model is manipulated in the task and how methods from the domain model are executed in tasks. This paper is organized in three sections. First, an overview of model-based UI generation is presented. Next, UI description languages are introduced, focusing on UsiXML [10]. Finally, our approach to fast *hi-fi* prototyping is described.

## 2. USER INTERFACE GENERATION

We can find proposals in the literature that provide frameworks that enable UI development. At the beginning, most of those proposals would generate the UI out of a domain model. However, currently most approaches drive their development out of a task model. Some of these proposals will be introduced in the next sections, describing the pros and cons of both.

### 2.1 Domain-Based Generation of User Interfaces

Domain model encapsulates the important entities of a particular application domain together with their attributes, methods and relationships. Elements in the domain model possess attributes that are often relevant to UI presentation elements selection. Examples for these attributes are the data type, the range, the minimum and maximum value, etc. Meaningful examples of this strategy in UI generation out of different types of domain models are Janus [2], Vampire [7], OlivaNova [11], Teallach [9] for desktop application, in web-based environments WebRatio [4] and VisualWade [8] and in hypermedia applications OHDM [21]. These domain-based UI generation approaches produce complex UI, because users can see many elements at the same time. Moreover, as long as the user-task is not contemplated, the dialog within the UI is rather limited and constrained, producing UI quite static.

### 2.2 Task-Based Generation of User Interfaces

Most model-based development approaches define a dialog model by using a task model. ConcurTaskTree (CTT) [16] is a well-accepted notation in the UI research development community used for the specification of task models. Information from the task model is exploited in order to automatically or interactively derive the navigational structure of the application. TERESA [14] exploits structural information as well as temporal relationships in order to generate an *activation set*, which is later used to automatically generate the dialog model and the widgets of presentation model. Task-based design as opposed to domain-based one incorporates information regarding the tasks the user will carry out through the UI as well as the temporal relationships between those tasks. This kind of information allows addressing usability aspects such as UI overload, presentation grouping, etc.

### 2.3 User Interfaces Prototyping

Some of the main drawbacks of model-based user interface development have been the unpredictability of the final results and the lack of techniques for the evaluation of the final UI given a set of declarative models [15]. To

overcome these drawbacks, and some other ones, different techniques have been introduced. One of those techniques introduced is user interface prototyping. Prototyping consists in the creation of a preliminary version of the future UI (prototype) so that the user and the experts can find possible problems in the design of the UI, both from the functional and from the usability points of view. Prototyping techniques fall into two main categories:






1. *Lo-fi* (low-fidelity) techniques: this family of techniques is mostly used in requirements analysis stage to validate the requirements with the user in user-centered approaches.
2. *Hi-fi* (high-fidelity) techniques: they are aimed at the creation of preliminary version of the UI with an acceptable degree of quality. This kind of techniques produces a UI prototype which is closer to the final future one.

Although paper is still the most widely tool used in prototyping, some other tools have been proposed to try to make prototyping faster, easier to change or more accurate. In this sense, sketching tools like SketchiXML [6] or CanonSketch [3] try to replicate the facilities in paper prototyping into a computer. A different point of view is pushed in UI Pilot [19]. *Hi-fi* prototypes could be considered to be better than *lo-fi* prototypes, since they are closer to the final user interface the user will interact with. Nevertheless, a set of disadvantages have been identified [20].

### 3. MODEL-DRIVEN DEVELOPMENT IN USER INTERFACES DESIGN

In our proposal, we use models precisely because they actually speed up development and help us to get to a better solution more quickly. Good models clarify design issues and highlight tradeoffs, so design issues can be resolved rapidly. Models also help us to deliver better and more robust systems. In this sense, abstract prototyping was devised because it was found that the sooner developers started drawing realistic pictures or positioning real widgets, the longer it took them to converge on a good design. Abstract models are always much simpler than the real thing. Nowadays, a series of models are used within MB-UID approaches to describe UI. These models need to be stored in a repository so that they can be manipulated by the different tools used during UI generation stages. In most cases these models are stored using an XML-based format. In [22], a review of the most prominent XML-based UI description languages can be found. UIML [1], XIML [18], DiaMODL [11] or UsiXML [10,23] are meaningful examples of these kinds of languages. UsiXML provides an abstract UI model that represents a canonical expression of the renderings and manipulation of the domain concepts and functions in a way that is as independent as possible from modalities and computing platform specifics.

Table 1. Abstract interaction objects and facets in UsiXML and icons used in IDEALXML

Abstract object	Icon	Facet	Icon
Container		Input	
Component		Output	
		Control	
		Navigation	

We are using the abstract UI specification proposed in UsiXML because it provides a reduced set of elements that allow the description of an abstract UI in a platform and modality independent manner. In Table 1 the set of icons used within our tool to represent the different elements of the abstract UI are shown.

#### 4. FAST GENERATION OF HI-FI USER INTER-FACE PROTOTYPES














One of the advantages of using a formal modeling language to specify the task model, such as CTT, is the ability to simulate the system before it is built. Simulation can help to ensure that the system that is built will match users’ conceptual model as well as to help to evaluate the usability of a system at a very early stage. Several task models simulators have been built for CTT. For example, in CTTE the designers can specify a task model, which can be simulated. In IDEALXML, designers can specify a task model and simulate the UI derived from the designed task model in an abstract manner by using CTT, UsiXML and a set of heuristics to transform the task model specification into an abstract UI. Currently, these heuristics are hard coded in IDEALXML application code, but there is an ongoing work to support the use of transformation rules that the designer can modify following approach similar to the one proposed in [10].

##### 4.1 Abstract User Interfaces Prototyping

The previously mentioned hard coded transformation rules are gathered in this section. *Strightforward rules* govern transformations (Table 2):

- Each cluster of interrelated task cases becomes an interaction space in the navigation map, so an abstract task is a container.
- A container also can be an interaction task or an application task if any of them are leaf in a hierarchical task decomposition.
- A component rises when we found an interaction or application task in a hierarchical task decomposition.
- A component can have several facets (input, output, control and navigation). These facets allow the user to interact with a system.

Table 2. From task model to abstract presentation model

Task model	is	Abstract presentation model
Abstract task 	is	a container 
Interactive task 	is	<i>input</i> 
		<i>output</i> 
		<i>control</i> 
		<i>navigation</i> 
		<i>not a leaf: container</i> 
		<i>not a leaf: container</i> 
Application task 	is	<i>a leaf: component</i> 
		<i>output</i> 
		<i>navigation</i> 

## 4.2 Abstract User Interfaces Prototypes Animation

IdealXML supports the animation of the abstract user interface resulting from the designed task model. This animation is grounded in the identification of the *enabled task set* (ETS) [16]. Having identified the ETC for a task model, the next step is to identify the effects of performing each task in each ETS. The result of this analysis is a *state transition network* (STN), where each ETS is a state and transitions occur when tasks are performed. In our proposal, the task model specification is split into states. Each state is a set of interrelated tasks, including temporal relationships between those tasks, usually connected to an *essential use case* [5].

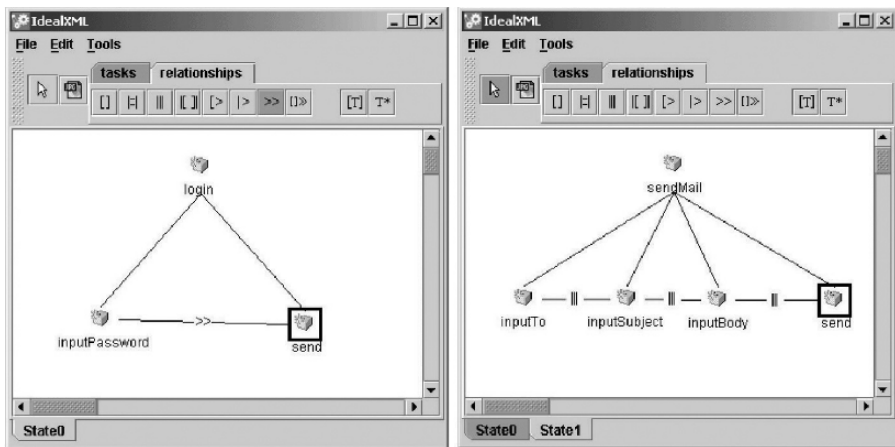


Figure 1. Task model specification in IDEALXML for e-mail sending task

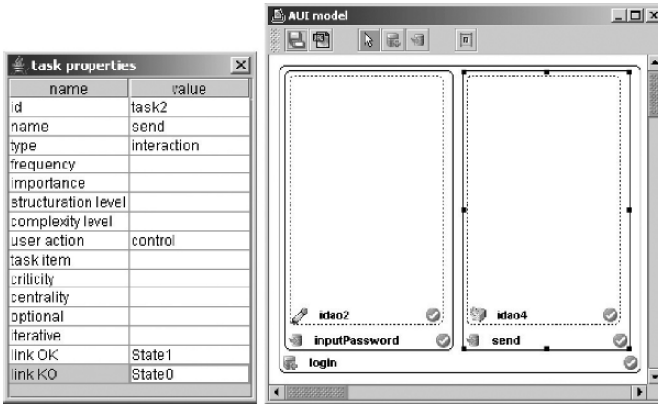


Figure 2. Abstract UI specification out of task model

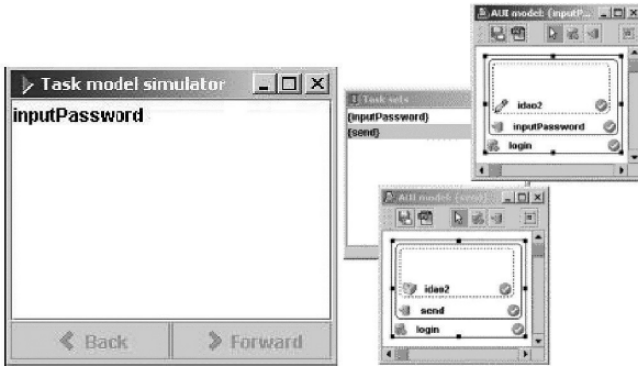


Figure 3. Simulation, ETS and abstract UI specification are available in IDEALXML

In Fig. 1, the task model for sending an e-mail message can be found. Two states have been identified in this case. The first one is related to user identification in the mail server and the second one is related to sending the e-mail message. By splitting the task model into states the task model complexity is drastically reduced and the legibility is really boosted. States are connected by establishing links between them. Two different kinds of links are proposed *linkOK* and *linkKO*. *LinkOK* specifies which state the system should go to when the goal of the current state is successfully achieved. In a similar manner, *linkKO* is state the system should go to when the goal of the current state fails. For example, in Fig. 2, *linkOK* points to the state where the user can send the e-mail (it means that the user password provided was successfully validated) and *linkKO* points to current state (identification state, because the verification of the user password provided failed).

As in CTTE, the designer can simulate task model specification in a textual manner (Fig. 3a). In IDEALXML, the designer is allowed also to animate the specification in a visual manner interacting with the abstract UI. Moreover, at any time designers can select any set of tasks in the task model and get the abstract UI specification for the selected task in a graphical manner.

## 5. CONCLUSION

A good user interface design is essential to ensure the acceptance of a new software. It is a complex subject, but we can overcome this complexity by raising the level of abstraction in the design by using models. Abstract prototyping is a way to avoid the seduction of attractive prototypes that disguise weak designs. By making better use of modern visual development tools, abstract prototyping can speed up and simplify the design of highly usable systems and help us to produce improved and more innovative software products. In our fast abstract prototyping proposal we address most of the *hi-fi* prototypes shortcomings identified in [20], providing an environment that allows the creation of the prototypes quickly in an abstract level enough to avoid focusing more on *look & feel* than in functional or usability issues and providing prototypes that can be easily modified.

## REFERENCES

- [1] Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S.M., and Shuster, J.E., *UIML: An Appliance-Independent XML UI Language*, Computer Networks, Vol. 31, 1999.
- [2] Balzert, H., Hofmann, F., Kruschinski, V., and Niemann, C., *The JANUS Application Development Environment - Generating More than the User Interface*, in J. Vanderdonckt (ed.), Proc. of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'1996 (Namur, 5–7 June 1996), Presses Universitaires de Namur, Namur, 1996, pp. 183–208.
- [3] Campos, P., and Nunes, N., *Canonsketch: a User-Centered Tool for Canonical Abstract Prototyping*, in R. Bastide, P. Palanque, J. Roth (eds.), Proc. of 9th IFIP Working Conf. on Engineering for Human-Computer Interaction EHCI-DSVIS'2004 (Hamburg, 11–13 July 2004), Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 146–163.
- [4] Ceri, S., Fraternali, P., and Bongio, A., Web Modeling Language (WebML): a Modeling Language for Designing Web Sites, in Proc. 9th Int. Conf. on World-Wide Web (Amsterdam, May 2000), 2000, accessible at <http://www9.org/w9cdrom/177/177.html>.
- [5] Constantine, L.L., and Lockwood, L.A.D., *Software for Use*, Addison-Wesley, Reading, 1999.
- [6] Coyette, A., and Vanderdonckt, J., *A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces*, in Proc. of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'2005 (Rome, 12–16 September 2005), Lecture Notes in Computer Science, Vol. 3585, Springer-Verlag, Berlin, 2005, pp. 550–564.
- [7] Eisentein, J., and Rich, C., *Agents and GUIs From Task Models*, in Proc. of 7th ACM Conf. on Intelligent User Interfaces IUI'2002 (San Francisco, 13–16 January 2002). ACM Press. New York, 2002, pp. 47–54.
- [8] Gómez, J., *Model-Driven Web Development with VisualWADE*, in N. Koch, P. Fraternali, M. Wirsing (eds.), Proc. of 4th Int. Conf. on Web Engineering ICWE'04 (Munich, 28–30 July 2004), Lecture Notes in Computer Science, Vol. 3140, Springer-Verlag, Berlin, 2004, pp. 611–612.
- [9] Griffiths, T., Barclay, P., McKirdy, J., Paton, N., Gray, P., Kennedy, J., Cooper, R., Goble, C., West, A., and Smyth, M., *Teallach: A Model-Based User Interface Development*

- Environment for Object Databases*, in N.W. Paton, T. Griffiths (eds.), Proc. of 1st Int. Workshop on User Interfaces to Data Intensive Systems UIDIS'99 (Edinburgh, 5–6 September 1999), IEEE Computer Society Press, Los Alamitos, 1999, pp. 86–96.
- [10] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Lopez, V., *UsiXML: a Language Supporting Multi-Path Development of User Interfaces*, in Proc. of 9th IFIP Working Conf. on Engineering for Human-Computer Interaction EHCI-DSVIS'2004 (Hamburg, July 11–13, 2004), LNCS, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 200–220.
  - [11] Molina, P., *User Interface Generation with OlivaNova Model Execution System*, in J. Vanderdonckt, N.J. Nunes, Ch. Rich (eds.), Proc. of ACM Int. Conf. on Intelligent User Interfaces IUI'2004 (Funchal, 13–16 January 2004), ACM Press, New York, 2004, pp. 358–359.
  - [12] Montero, F., López Jaquero, V., Lozano, M., and González, P., *A User Interfaces Development and Abstraction Mechanism*, in Proc. of V Congreso Interacción Persona-Ordenador Interacción 2004 (Lérida, 3–7 May 2004).
  - [13] Montero, F., López-Jaquero, V., Vanderdonckt, J., Gonzalez, P., Lozano, M.D., and Limbourg, Q., *Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML*, in S.W. Gilroy, M.D. Harrison (eds.), Proc. of 12th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2005 (Newcastle upon Tyne, 13–15 July 2005), Lecture Notes in Computer Science, Vol. 3941, Springer-Verlag, Berlin, 2005, pp. 161–172.
  - [14] Mori, G., Paternò, F., and Santoro, C., *Tool Support for Designing Nomadic Applications*, in Proc. of the ACM Int. Conf. on Intelligent User Interfaces IUI'2003 (Miami, 12–15 January 2003), ACM Press, New York, 2003, pp. 141–148.
  - [15] Myers, B., Hudson, S.E., and Pausch, R., *Past, Present, and Future of User Interface Software Tools*, ACM Transactions on Computer-Human Interaction, Vol. 7, No. 1, March 2000, pp. 3–28.
  - [16] Paternò, F., *Model-Based Design and Evaluation of Interactive Applications*, Springer-Verlag, Berlin, 1999.
  - [17] Puerta, A.R., *A Model-Based Interface Development Environment*, IEEE Software, Vol. 14, No. 4, July/August 1997, pp. 41–47.
  - [18] Puerta, A.R., and Eisenstein, J., *XML: A Common Representation for Interaction Data*, in Proc. of ACM Int. Conf. on Intelligent User Interfaces IUI'2002 (San Francisco, 13–16 January 2002), ACM Press, New York, 2002, pp. 216–217.
  - [19] Puerta, A.R., Micheletti, M., and Mak, A., *The UI Pilot: A Model-Based Tool to Guide Early Interface Design*, in Proc. of ACM Int. Conf. on Intelligent User Interfaces IUI'2005 (San Diego, 10–13 January 2005), ACM Press, New York, 2005, pp. 215–222.
  - [20] Rettig, M., *Prototyping for Tiny Fingers*, Communications of the ACM, Vol. 37, No. 4, 1994, pp. 21–27.
  - [21] Schwabe, D., and Rossi, G., *The Object-Oriented Hypermedia Design Model*, Communications of the ACM, Vol. 38, No. 8, 1995, pp. 45–46.
  - [22] Souchon, N., and Vanderdonckt, J., *A Review of XML-Compliant User Interface Description Languages*, in J. Jorge, N.J. Nunes, J. Cunha (eds.), Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003 (Madeira, 4–6 June 2003), Lecture Notes in Computer Science, Vol. 2844, Springer-Verlag, Berlin, 2003, pp. 377–391.
  - [23] Vanderdonckt, J., *A MDA-Compliant Environment for Developing User Interfaces of Information Systems*, in O. Pastor, J. Falcão e Cunha (eds.), Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13–17 June 2005), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16–31.