

# TIMM: Exploring Task-Interface Links in MOBI-D

Jacob Eisenstein and Angel Puerta

Stanford University

251 Campus Drive – MSOB x215

Stanford, CA 94305-5479 USA

+1 650 723 5294

puerta@smi.stanford.edu

<http://www.smi.stanford.edu/projects/mecano>

## ABSTRACT

One of the central elements of model-based interface design is the mapping of abstract task models into concrete interface designs. It is also one of the least understood parts of model-based technology. Most previous solutions to this problem focused on determining automatically such mappings or on hardwiring the mappings into software. We propose an alternative solution in which we build interactive tools that allow developers to view and manipulate such mappings. To enable these capabilities, we exploit the features of the interface modeling language in MOBI-D (Model-Based Interface Designer), which is able to represent declaratively mappings between abstract and concrete units of an interface model. We present the MOBI-D environment emphasizing its interactive tools for task-interface mapping and illustrate the use of the tools via examples. We claim that MOBI-D affords its users a high level of flexibility in model-based interface design and a comprehensive set of views into the design knowledge and structure of an interface.

## INTRODUCTION

The central appeal of model-based interface development [4] is its professed ability to support the design of an interface from an abstract representation to a concrete design. Typically, this takes the form of a mapping between a model of a user task and a model of the presentation and of the dialog of an interface. Unfortunately, our knowledge about how to define such mappings is very limited. We lack theoretical foundations to define them, and consequently to apply them.

Notwithstanding this absence of a well-understood framework, many model-based systems have attempted to deliver specific solutions for the task-interface mapping problem. The usual approach is to embed into the code of the system one such solution. This is the case in systems such as Mecano [3], UIDE [1] and ADEPT [6] among several others. The result typically is an inflexible interface design process whose inner workings are beyond the reach of the users of the system. In addition,

because of the limited scope of the implemented mapping solution, the associated model-based system can effectively work only within a restricted subset of the design space for interfaces.

We claim that mappings between abstract and concrete elements of an interface model are *loose* or *fuzzy* at best. The mappings are highly dependent on specific characteristics of a given user-task model. Thus, it is more appropriate to try to build interactive tools with which interface developers can create the mappings and operate on them. Naturally, to build such tools we must first be able to represent those mappings declaratively as part of an interface model.

In this paper, we present an initial interactive approach to establish mappings between user-task models and concrete interface designs. The approach is built into the MOBI-D (Model-based Interface Designer) [4] model-based development environment. It leverages off the ability of the MOBI-D interface modeling language to represent explicitly the desired mappings. We have built into MOBI-D capabilities for various types of interface-model mappings between the various components of a MOBI-D interface model: user-task, domain, presentation, dialog, and user. We consider the mappings of task to domain, domain to presentation, and task to dialog the crucial ones in any interface design. It is our goal that by allowing interface developers to map interface model elements interactively, we can establish with experience solid patterns of usage of these mappings. Detecting such patterns can then lead to more automated tools for model-based interface development.

The rest of the paper is organized as follows. We first introduce the MOBI-D interface modeling language and the architecture of MOBI-D. We then describe the MOBI-D development cycle and illustrate it via an example. Throughout the example, we will emphasize the MOBI-D tools that developers use to map interface elements to each other in order to bridge the abstract-to-concrete gap in the design of an interface via models. We close by summarizing the paper and pointing some of the potential benefits of our approach.

## THE MIMIC INTERFACE MODELING LANGUAGE

The foundation for any model-based interface development environment is its knowledge representation for interface models. In MOBI-D, we have developed a declarative interface modeling language called MIMIC, which is fully described elsewhere [3]. For the purpose of this paper we will concentrate on the aspects of MIMIC that enable support for interactive task-to-interface mappings.

MIMIC is a meta-language that structures and organizes interface models. It divides an interface model into *model components*. The current components in MIMIC are user-task, domain, presentation, dialog, user, and design models. Of these components, the design model is the unit that represents all of the mappings among the elements in an interface. In the context of MIMIC, an *interface* is made up of all the elements defined in the user-task, domain, presentation, dialog, and user models. Correspondingly, an *interface design* is the set of mappings among those elements represented by the design model. The MOBI-D tools that afford interactive manipulation of the mappings do so by providing various views into the design component of MIMIC.

## THE MOBI-D INTERFACE DEVELOPMENT ENVIRONMENT

The Model-Based Interface Designer environment [4] supports end users and interface developers in designing and implementing user interfaces under a user-centered development cycle. The environment presents an open architecture and overcomes many of shortcomings of previous model-based systems.

MOBI-D integrates a number of design-time and runtime tools, including:

- A user-task elicitation tool (U-TEL) to obtain user-task models directly from domain experts.
- A set of interactive model editors. Each category of interface elements (user task, domain, presentation, dialog, and user type) is handled via a model editor with specific functionality pertinent to that category. In addition, a *design model* editor allows visualization and editing of the mappings among interface elements (i.e., the interface design as defined by the MOBI-D modeling language)
- A task-interface model mapping tool (TIMM) that acts as an interface design assistant. It allows developers to make global and specific design choices for presentation and dialog.
- A task-based interface builder. Similar to the familiar palette-and-canvas builders but where operations are dictated by a user-task model.

## FROM ABSTRACT TO CONCRETE IN MOBI-D

MOBI-D supports a user-task centered development cycle. A domain expert provides an outline of a user task in textual form. Using the MOBI-D interactive model editors, an interface developer refines this outline into user task and domain models. The elements of these models are then integrated by specifying the corresponding relations among them (i.e., which domain objects are needed in each user subtask).

In the next phase, the developer uses the interface-design assistant tool to specify which styles, guidelines, and dialog strategies will be applied. Then, the developer, in possible cooperation with the domain expert or with end users, completes the presentation and dialog using the task-based interface builder. This interface builder is customized according to the user-task and domain models and the selections made for styles and strategies.

In general, the first two phases in the development cycle define the abstract elements of the interface while the following phases map those abstract elements to concrete interface components. In the example that follows we will traverse the complete development cycle and highlight the mapping operations that take place. The target application domain in the example is that of military logistics.

### Eliciting the User Task

Domain experts can use the U-TEL tool [5] in the MOBI-D environment to specify directly the outline of the user task for a target interface. In U-TEL, the end user first types in a free-text description of the task and then categorizes key terms as objects, actions, or actors (users). Using the categories and the text, the end user creates an outline of the task that gives a sense of task decomposition and ordering. The outlines and categories are employed in the next phase to build user-task and domain models. Our evaluation of this tool [5] has shown that this method of elicitation is effective for both experienced and non-experienced computer users.

### Creating User-Task and Domain Models

The outline provided by the end user must be transformed into an interface model, and specifically into the user-task and domain components of such model. MOBI-D automatically builds a skeleton of those components from the outline and category information given by the end user. The interface developer uses the interactive model editors to refine that skeleton into fully specified model components. Typical operations to be performed here are to assign data types to each domain object, to specify subtask ordering (e.g., sequence, unordered, optional sequence), and to input ranges of allowed values and default values where appropriate. This phase transforms a textual description into a model in the MOBI-D interface modeling language.

### **Task-Domain Mappings: Integrating the User-Task and Domain Models**

The elements defined in the user-task and domain models developed in the previous phase must be incorporated into a design in the MOBI-D sense of the word. That is, user tasks and domain objects must be mapped to each other in order to specify which domain objects play a role in each of the subtasks in the user task model. The interface developer uses a MOBI-D model editor to accomplish this integration via drag and drop operations. The editor presents the full hierarchy of the interface model, and allows browsing of the mappings among elements. The interaction with this editor is very similar to that of a file browser. In addition, MOBI-D users also map user types to tasks in order to define which user does what part of the task model. After this phase, MOBI-D users will have created the design foundation that drives the rest of the development cycle.

### **Task-Interface Mappings: Assisting the Design of the Interface**

Under the direction of the interface developer, MOBI-D will now prepare the interface model for the presentation and dialog design phases. With the support of an interface-design assistant tool, called TIMM (Task-Interface Model Mapper) the developer creates a number of mappings between the task/domain models and the concrete interface design. There are three types of operations here as depicted in Figure 1:

- *Interactor assignments.* MOBI-D suggests what interactor(s) (or widget) is preferred (high priority) to display each of the domain objects. The assignment depends on the data type of the object and on a knowledge base of interface design guidelines. The developer can browse and change the assignments. Global operations are allowed. For example, the developer can set that all Boolean objects be displayed with checkboxes. This tool affords a level of control to the developer that is missing in previous model-based systems that automatically generate layouts using data types. In the automated approach, the developers of the model-based system preset the mappings between domain objects and widgets. In MOBI-D, these mappings are set interactively, can always be visualized, and can be changed as per the needs of specific designs.
- *Styles.* The developer can browse and select a number of standard styles for the interface. Styles include features such as font groups, preferred location of OK/Cancel button pairs and so on. It works in a similar fashion to that of styles in a word processing program. A style is also a type of task-interface mapping in that it assigns entire sets of

concrete interface characteristics to the overall user-task model

- *Strategies.* The developer can view and modify the strategies that will be used to set up the dialog and navigation characteristics of the interface. A strategy is a type of task-dialog mapping. It answers the question: How are the characteristics of a user-task model reflected in the dialog of the resulting interface. Strategies that can be set by the user in MOBI-D include among others:
  1. The number of windows desired (e.g., one window per major subtask in the user task model).
  2. The enforcement of sequential task requirements (e.g., follow strictly the sequences specified in the user-task model).
  3. The enforcement of value ranges for domain objects accessible through the interface.

Note, for example, that the number-of-windows strategy is a simple tree-to-tree mapping. Assume a user-task model (represented as a tree in MOBI-D) with multiple levels of decomposition. Setting the number of windows slide bar completely to the left will mean that the resulting interface design will be mapped to a single window. This would mean that the top task and all of its subtasks in the user-task model will be accomplished by the end user via this window. Setting the slide bar to next allowed discrete value to the right will break the resulting interface design into as many windows as there are immediate children of the top task in the user-task model. Moving the slide bar further to the right will be going down level by level on the user-task tree and consequently increasing the number of windows in the resulting interface.

As a group, these features give the developer control over how the interface will be designed. The tool functions as a collection of *automation knobs* that can be set by the developer to full, none, or somewhere in between. Our initial evaluation indicates that developers “turn the knobs” up as they gain familiarity with the environment (and possibly develop confidence in it).

### **Task-Interface Mappings: Presentation and Dialog Design**

This phase is completed by the interface developer (possibly cooperating with end users) using an interface-builder like tool as shown in Figure 2. This tool, however, differs significantly from conventional interface builders in the sense that it is guided by the user-task model and by the selections made in the styles and strategies phase. The developer moves down the user-task model and for each leaf in the user task model tree (top window) selects a widget. MOBI-D orders the possible widgets for each task according to their priority (as set in

the previous phase). The developer is free to use a different widget at this point, however. Essentially, for each widget selected and placed on the canvas, the developer has established a mapping between an abstract user-task element and a concrete interface one.

Because the presentation and dialog design are at all times guided by the interface model, there is a clear connection between each widget and its relevance in the overall user task. Moreover, the developer is assured to have provided interaction functionality for the complete user task model. Furthermore, end users find it easy to understand the selection of a particular widget since a connection is clear between the widget selection and the task outline that they provided at the beginning of the design process.

**SUMMARY**

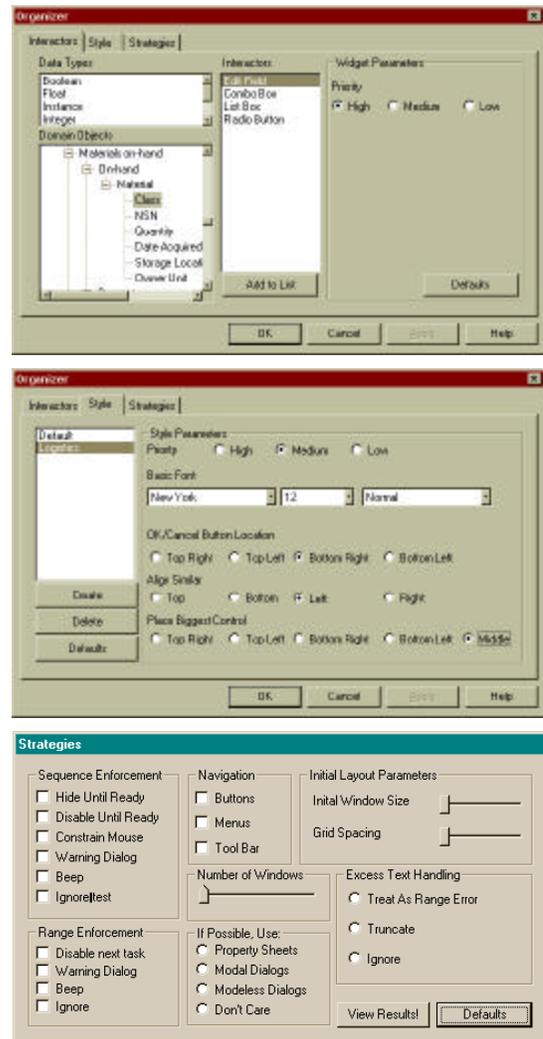
We have presented some initial prototypes to allow interface developers to map interactively task models to concrete interface designs. We claim that developing a thorough understanding of such a mapping process is the key to the success of model-based systems. We expect that by gathering experience with these tools, we will start detecting patterns of use of these mappings. The patterns can be the foundation for a theory of task-interface mapping.

**ACKNOWLEDGEMENTS**

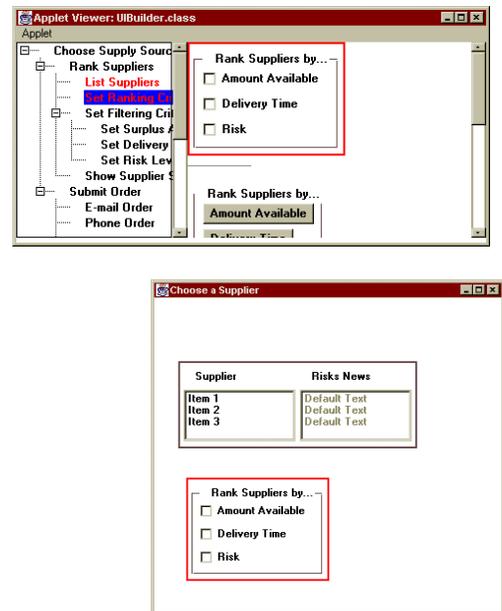
The work on MOBI-D is supported by DARPA under contract N66001-96-C-8525.

**REFERENCES**

1. Foley, J., et al., *UIDE-An Intelligent User Interface Design Environment*, in *Intelligent User Interfaces*, J. Sullivan and S. Tyler, Editors. 1991, Addison-Wesley. p. 339-384.
2. Puerta, A. and Eriksson, H. *Model-Based Automated Generation of User Interfaces*, in Proc. of AAI'94. 1994: AAAI Press.
3. Puerta, A. R. *The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development*, in Proc. of CADUI96: Computer-Aided Design of User Interfaces. 1996. Namur, Belgium.
4. Puerta, A. R. A Model-Based Interface Development Environment. *IEEE Software*, (14) 4, July/August 1997, pp. 40-47.
5. Tam, R. C.-M., Maulsby, D., and Puerta, A. *U-TEL: A Tool for Eliciting User Task Models from Domain Experts*, in Proc. of IUI98: 1998 International Conference on Intelligent User Interfaces. 1998. San Francisco, CA: ACM Press.
6. Wilson, S. and Johnson, P. *Beyond Hacking: A Model-Based Approach To User Interface Design*, in Proc. of HCI'93. 1993.



**Figure 1.** The Task-Interface Model Mapper.



**Figure 2.** The task-based interface builder.