Management of Interface Design Knowledge with MOBI-D

Angel R. Puerta and David Maulsby
Knowledge Systems Laboratory
Stanford University
MSOB x215
Stanford, CA 94305-5479 USA
+ 1 415 723 5294
{puerta,maulsby}@camis.stanford.edu - http://camis.stanford.edu/projects/mecano

ABSTRACT
Effective guidelines for interface construction require developers to apply a user-centered approach in their designs. Yet, developers lack integrated tools that would allow them to work with high-level concepts, such as user tasks, and to relate them to lower level elements, such as widgets, in their interface designs.

The Model-Based Interface Designer (MOBI-D) is a suite of tools for the management, visualization, editing, and interactive refinement of interface-design knowledge at multiple levels of abstraction. MOBI-D represents knowledge via declarative interface models that assign specific knowledge roles to each model component. Developers work in an integrated environment with abstract concepts such as user tasks, domain objects, presentation styles, dialogs, and user types while being able to relate those concepts to concrete interface elements such as push buttons. MOBI-D is the first development environment to integrate the disparate elements of interface design into structured conceptual units—interface models—and to define an interface design as an explicit declarative element of such units.

Keywords
Model-Based Interface Development, Interface Models, User Interface Development Tools

INTRODUCTION
Designing an interface is a complex task that requires developers to organize and manage multiple types of concepts at various levels of abstraction. On one hand, designers must identify and understand the target user's tasks in order to produce an effective user-centered interface. On another, these designers must deal with minute details of layout and presentation that many times have an important impact on the usability of the interface. Furthermore, designs become even more complicated when, among others, issues of multiplicity of target user types, effective sequencing of dialogs, or platform dependencies come into play.

There is, however, a lack of tools that help designers put all the pieces of an interface design together. Currently available tools focus on specific portions of an interface design. For example, interface builders are excellent to create layouts and manipulate widgets but do not help in managing user tasks. Other tools help with high-level designs, or mock-ups, but have no way of relating those designs to actual working interface elements.

A potential solution to create the kinds of tools that interface developers need is to view interface design as essentially a knowledge engineering activity. From that point of view, the problem becomes one of defining appropriate knowledge representations for interfaces and their designs, and one of building the tools to support the definition and editing of such representations.

In this paper, we take such a knowledge engineering approach and present MOBI-D, an interface-development environment that allows developers to define, organize, visualize, and edit interface design knowledge. Furthermore, we introduce the knowledge representation associated with MOBI-D: Declarative interface models. We show that interface models can be used to define how an interface design is created by relating high-level elements, such as user tasks, to lower level ones, such as widgets.

RELATED WORK.
The research area concerned with the explicit representation of interface-design knowledge is known as model-based interface development. Most efforts until now in this field have focused on the definition of a single part of an interface model (e.g., a user-task model), and on driving automatically the generation of interfaces from such specifications.

Mecano [2], which is the precursor of MOBI-D, allowed developers to generate form-based interfaces from explicit

models of an application's domain. ADEPT [1] drives interface generation from models of user's tasks. HUMANOID [4] emphasized presentation models as a basis for producing interfaces. Other approaches utilized Entity-Relationship data models or Petri Nets as dialog models to produce interfaces.

These approaches are limited in two aspects. First, by defining only part of the design knowledge of an interface, they offer developers only partial views into the design of an interface. As a result, these technologies tend to be successful in narrowly defined situations. Second, because of the emphasis in automated interface generation, the developers using these tools have a very restricted control over generated interface designs. More importantly, however, none of these research efforts attempts to define the key components of an interface model (user tasks, domain objects, users, presentation styles, and dialogs) as forming a declarative unit. Consequently, there is no way for developers to define and visualize relations among, say, a user task, a domain object, and a dialog element.

THE MECANO PROJECT

To address the limitations stated in the previous section, we have created The Mecano Project. This project is a research effort aiming to enable comprehensive and integrated support for the design and implementation of interfaces via declarative interface models [2]. The are three key elements provided by The Mecano Project:

- A modeling language for interface models (MIMIC)
- A set of generic interface models for user interface development (MIMs)
- A development environment that supports the conceptual framework of MIMIC and its associated MIMs (MOBI-D)

THE MIMIC MODELING FRAMEWORK

MIMIC is a language to express interface models [2]. It takes a componential approach that defines interface models as sets of components with specific knowledge roles. There are two key concepts encapsulated by the MIMIC language:

- An *Interface* is a collection of interface objects ordered under specific interface model components. The following classes of model components are defined: user-tasks, domain, presentation, dialog, and user type. There may be more than one instance of each class of component in any interface. The ordering, or schema, of interface objects dictated by one instance of a component may be different from that of another instance of the same class of component.
- An *Interface Design* is an expression of the relationships among the objects of the various components of an interface. A design is defined by a design model component in MIMIC.

The implications for interface software tools are that developers can: (1) construct interfaces by working with abstract elements such as user tasks, domain objects, and dialog representations; (2) visualize how the various objects form a design (e.g., what presentation styles are assigned to specific domain objects and which user tasks involve access to those domain objects); and (3) manage the use of, and experiment with, multiple versions of interface model components (e.g., two presentation styles for different screen sizes, or two user task models with different ordering of tasks).

MIMIC should be distinguished from previous modeling languages that model only interfaces [3]. The primary purpose of MIMIC is to define the components of interface models, to designate the roles of those components within an interface design, and to express the relationships among the objects belonging to various components. MIMIC can thus be seen to be at a metalevel above current interface modeling languages.

THE MOBI-D ENVIRONMENT

The tools in MOBI-D provide the functionality to manage, visualize, edit, and refine each of the components of an interface model as defined by MIMIC. In a typical, although not fixed, MOBI-D design process developers follow these steps using the environment's interactive tools: (1) define a user-task model; (2) define an application domain model and relate the objects in the domain to the defined user tasks; (3) interactively accept, modify, or discard MOBI-D recommendations for presentation styles and dialog interaction techniques; (4) optionally define user type models and adapt interface elements to each user type.

There is, however, no need to specify completely every element of an interface. MOBI-D supports the use of generic interface models, called Mecano Interface Models (MIMs) that already provide many of the elements that typical interfaces need. Figure 1 shows a MOBI-D editor view for user-task model components. The view is split into three panes. On the top right pane, the developer can view the generic objects belonging to the MIM. The top left pane has the hierarchical view of the particular user task model for the target interface. The bottom pane lists the properties of the object selected in either of the top panes. To use a MIM object, developers simply select the object and drag it to the hierarchical view. Developers can also define objects on the fly on the hierarchical view and add them, if appropriate, to the MIM. The properties of any object can be edited via the properties pane. Other

model components, such as domain and dialog models, are edited and refined in MOBI-D with tools similar to that in Figure 1.

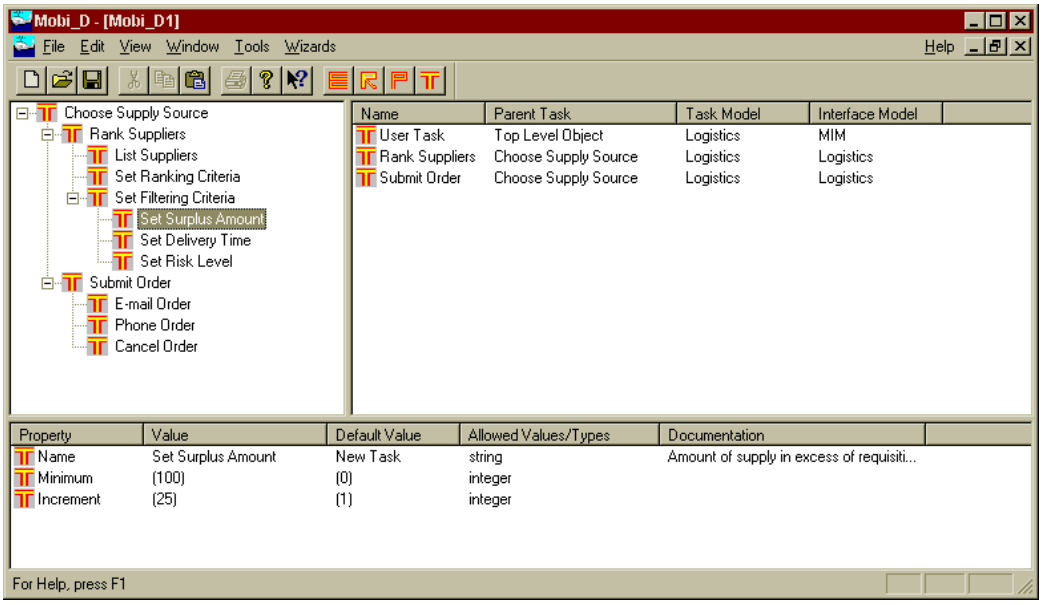

Figure 1. User-task model editor in MOBI-D showing the task model for the interface in Figure 3.

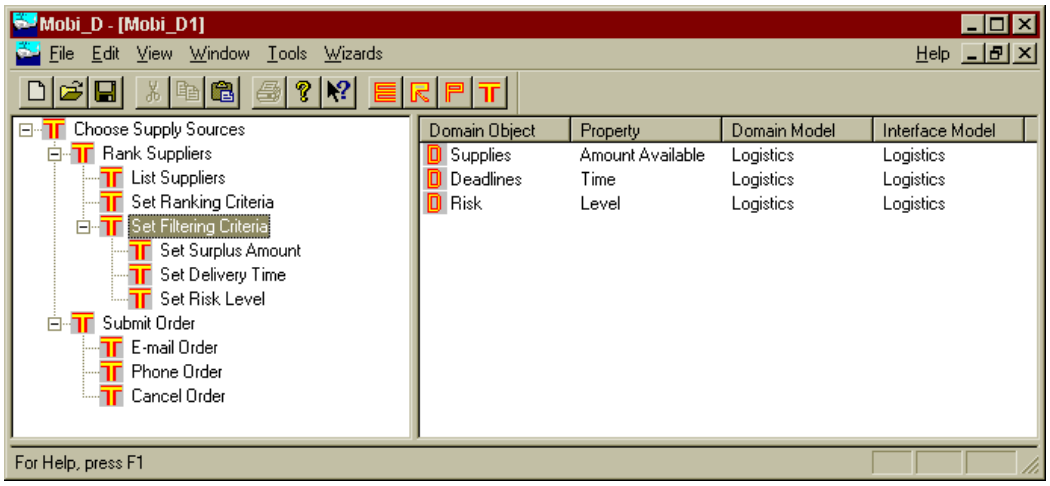

Figure 2. The right panel shows the design view for the selected user task with the relations to all the domain interface model elements affected by that task.

Interactive Design Views

As defined earlier, an interface design is created when the developer relates the objects in one model component with those in other components. In MOBI-D this is achieved via design views. Figure 2 shows a design view for the interface in Figure 3. A design view shows, for example, what domain elements take part in a given user task, or, what widgets provide user access to a domain object. In the example of Figure 2, a design view allows a developer to link the task "Set Filtering Criteria" with the domain objects "Supplies", "Deadlines" and "Risk".

Design views allow developers to quickly understand the impact on the resulting interface of changes to abstract elements such as user tasks. They also provide an integrated framework for visualizing all the aspects of an interface design.

Developer-Driven Interface Generation

Unlike previous model-based systems, we have discarded the idea of automatically generating the interface from partial models. Instead, the goal of MOBI-D is to use interface models to assist developers in making design decisions. For example, there are several choices for

widgets to specify deadlines in our sample interface design. MOBI-D may recommend one choice based on data type characteristics and a knowledge base of interface guidelines [3], but will allow the developer to modify interactively the

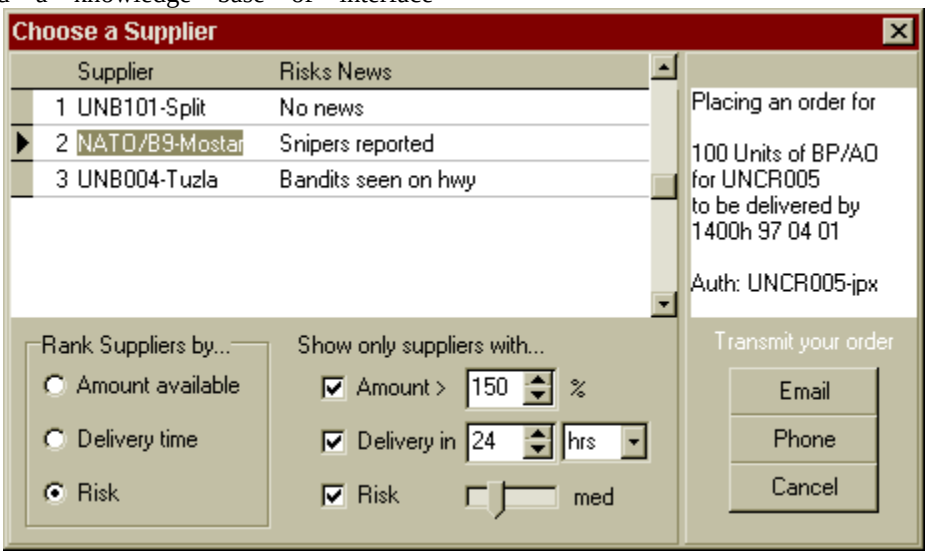

Figure 3. A prototype window of an interface for the requisition of supplies. A requisitions officer chooses the best candidates to supply specified items. The officer can rank the suppliers according to three factors, filter the list of candidates, and submit the order by e-mail or phone. This window is part of a larger, more complex application in the logistics domain, called an Interactive Logistics Map, whose interface is being designed with MOBI-D Tools.

choice, or even to change the way MOBI-D makes recommendations for that type of object. In a similar manner, the environment tools will guide a developer in deriving dialog models from user task models, and in assigning presentation styles based on established guidelines. The result is that the interface in Figure 3 can be created by formulating a user-task model and a domain model, and by following MOBI-D recommendations for presentation styles and interaction techniques.

CONCLUSIONS

The MOBI-D environment allows developers to take advantage of declarative interface models for interface development. The environment is the first of its kind to integrate the multiple elements of an interface design into a conceptual unit—an interface model.

By working in MOBI-D, developers can complete user-centered designs in a structured integrated environment. MOBI-D facilitates interface construction by offering generic interface models as well as developer-driven interface generation tools.

The ability of developers to create interfaces via interface models opens the possibility for many types of decision support tools that take an interface model as input. Examples of such tools are guideline-enforcement tools, usability tools, critiquing tools, and automated help systems.

Our experience with model-based interface development indicates great potential for the technology, but also shows that such potential cannot be realized without effective tools for interface model definition such as MOBI-D.

ACKNOWLEDGMENTS

This work is supported by the Defense Advanced Research Projects Agency (DARPA) under contract N66001-96-C-8525.

REFERENCES

1. Johnson, P., Johnson, H., *Knowledge Analysis of Tasks: Task Analysis and Specification for Human-computer Systems*, in « Engineering the Human-Computer Interface », A. Downton (Ed.), McGraw Hill, London, 1991, pp. - .

2. Puerta, A.R., *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 (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur, 1996, pp. 19-36. ISBN 2-87037-232-9

3. Puerta, A.R., Eriksson, H., Gennari, J.H., Musen, M.A., *Beyond Data Models for Automated User Interface Generation*, in Proc. of British Conference on Human-Computer Interaction, HCI'94 (Glasgow, 23-26 August 1994), Cambridge University Press, Cambridge, 1994, pp. 353-366.

4. Szekely, P., Luo, P., Neches, R, *Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design*, in [CHI92], pp. 507-514.