U-TEL: A Tool for Eliciting User Task Models from Domain Experts

R. Chung-Man Tam
Stanford University
251 Campus Drive - MSOB x215
Stanford, CA 94305-5479 USA
+1 415 723 5294
rtam@cs.stanford.edu

David Maulsby
Aurelium Inc.
430, 820 – 89 Av. SW
Calgary T2V 4N9 Canada
+1 403 252 2139
maulsby@aurelium.com
http://www.aurelium.com

Angel R. Puerta
Stanford University
251 Campus Drive - MSOB x215
Stanford, CA 94305-5479 USA
+1 415 723 5294
puerta@smi.stanford.edu
http://smi.stanford.edu/people/puerta

ABSTRACT
Eliciting user-task models is a thorny problem in model-based user interface design, and communicating domain-specific knowledge from an expert to a knowledge engineer is a continuing problem in knowledge acquisition.

We devised a task elicitation method that capitalizes on a domain expert's ability to describe a task in plain English, and on a knowledge engineer's skills to formalize it. The method bridges the gap between the two by helping the expert refine the description and by giving the engineer clues to its structure.

We implemented and evaluated an interactive tool called the User-Task Elicitation Tool (U-TEL) to elicit user-task models from domain experts based on our methodology. Via direct manipulation, U-TEL provides capabilities for word processing, keyword classification, and outline refinement. By using U-TEL, domain experts can refine a textual specification of a user task into a basic user-task model suitable for use in model-based interface development environments.

Our evaluation shows that U-TEL can be used effectively by domain experts with or without a background in programming or interface modeling, and that the tool can be a key element in promoting user-centered interface design in model-based systems.

Keywords
User-centered design, model-based user interface design, task models, knowledge elicitation

INTRODUCTION
In this paper, we present and evaluate a tool for eliciting informal user-task models from domain experts, in order to guide the design of a user interface. These models are formalized by a knowledge engineer and entered into the MOBI-D model-based user interface development environment [2]. The tool offers a possible solution to the problem of designing user-centered domain-specific user interfaces.

Model-based interface development [2] relies upon explicitly representing user-task and general design knowledge to guide the design of a user interface. Although there is increased evidence that user-task models are effective in driving a user-centered design approach [2,5], interface designers are having trouble identifying and understanding the target domain's tasks. These difficulties arise from the fact that interface designers are not experts in the target domain. Therefore, they cannot build interfaces that suit fully the needs of the end users in that domain.

On the other hand, domain experts are typically not interface designers. Therefore, they cannot develop a user interface using tools currently available. Interface design is a complex process involving organization and management of concepts at multiple levels of abstraction, some of which mean nothing to a domain expert.

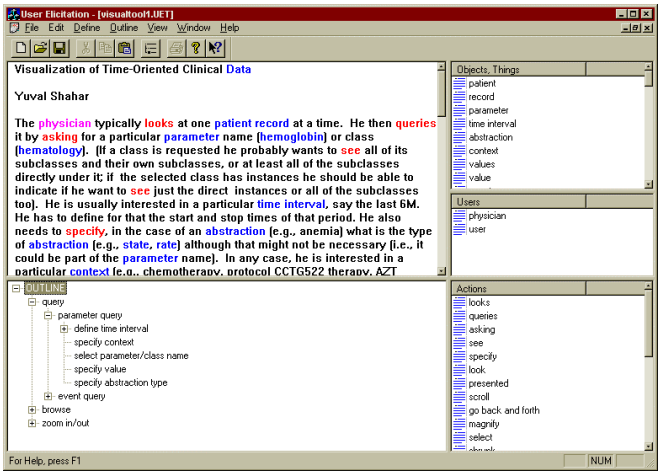

Figure 1. User Task Elicitation Tool. An example in the domain of visualization of temporal medical data is shown.

To address this problem, we built a prototype software tool that enables a domain expert to outline informal user-task models that an engineer can formalize using a model editor such as MOBI-D. The informal and formal models may then be used in conjunction to guide the user interface designer. Our goal is to make it possible for these three experts to communicate with one another effectively. Although there are a number of software tools built for task or workflow analysis, they are rarely aimed at domain experts and none can be used in an integrated fashion with a model-based interface development environment.

Our approach to eliciting task knowledge from domain experts to interface designers supplements or even replaces observational studies and *ad hoc* interviews with a structured yet flexible elicitation centered on sample scenarios. The methodology allows for domain experts to complete the elicitation process by themselves or by being aided by a facilitator, or interviewer. The elicitation begins with the domain expert recounting one or more examples of performing a task, and transcribing these examples into a text editor. The expert then refines these descriptions by performing three operations using U-TEL, illustrated in Figure 1. First, the expert identifies and classifies task terminology, in particular terms for objects, actions and actors involved in the task. The expert accomplishes this by selecting text and choosing its classification from a popup menu. The facilitator may prompt the expert by pointing at some phrase. Classified terms are highlighted and copied automatically to the appropriate on-screen list. Second, the scenario text is copied to an outline editor, where the expert segments it and organizes it to bring out task structure, putting each action on a separate line, and indenting and inserting headings to delimit subtasks. The facilitator may suggest groupings or headings. Third, the expert annotates subtask headings, using a popup menu to indicate whether actions need to be done in order, may be interleaved, or are conditional upon some event.

The output of this elicitation comprises: a) the original task scenarios; b) three lists of classified terms (objects, actions, actors); and c) an outline of the task that expresses its hierarchical decomposition and sequencing. Classified terms are highlighted in both the scenarios and outline. The outline and lists of terms are *not* intended to be a formal task model. Instead, they constitute a guide to the knowledge engineer, whose job is to eliminate ambiguities in the terminology, fill in missing details of actions, and further rationalize the task structure, in consultation with the domain expert. The knowledge engineer edits the task and data models using MOBI-D tools, which provide a formal language that helps the engineer think about task models in operational terms, that is, in terms of computer inputs, outputs, and knowledge representations.

In this paper, we first discuss the relationship between knowledge acquisition and model-based user interface design. We then describe our explorations of our elicitation method at two levels. At the first level, we employ Wizard-of-Oz simulations of an intelligent agent that suggests user-task model elements in order to guide the elicitation. At the second level, we partially implement the intelligent agent of our Wizard-of-Oz studies via the tool shown in Figure 1. We present a report of its evaluation and of its use by domain experts to model a rather complex task for which they desire computer support. Finally, we assess the utility of our method and of U-TEL, suggesting avenues for further research and development.

ACQUIRING DESIGN KNOWLEDGE

There has been considerable progress in the field of model-based interface development in recent years [2]. A number of systems have been implemented and have been successful at showing the potential of the model-based paradigm. Some of the well-known systems include ADEPT [5], HUMANOID [4], and Mecano [3]. However, none of the available model-based systems has implemented an effective process for eliciting user task models from domain experts, a key requirement for user-interface design. Some systems, including HUMANOID and Mecano, do not even have representations for user tasks in their interface models. Those that do so, like ADEPT, do not include tool support for user-task elicitation from domain experts.

Conversely, the field of knowledge acquisition has studied for a long time issues of identification, classification, and structure of knowledge terms. However, no tool support exists for the elicitation of user-task models for the explicit purpose of driving interface design. U-TEL will combine the progress made in modeling interface design knowledge with that of knowledge elicitation techniques in order to bridge a noticeable gap in the development cycle of model-based systems.

WIZARD OF OZ SIMULATIONS

We first tested our approach to refining informal descriptions in a Wizard-of-Oz experiment [1]. A researcher simulated an elicitation tool that follows the protocol outlined in the Introduction, but that provides "intelligent" assistance by suggesting transformations to the domain expert's description. These suggestions serve as examples to the user, and incidentally automate the elicitation. The experiment was conducted via email with four subjects, all of whom were asked to describe the same task—doing the laundry. We collected their email exchanges with the Wizard and their comments on the process.

Here is a typical description of the task:

First you sort the laundry into whites, colors and permanent press. You wash each load separately. Add the detergent to the washing machine and pile the clothes in loosely. If the load is whites, set the temperature to hot. If

bright colors, set the temperature to cold; otherwise warm. For permanent press or delicate items, set the speed to slow, otherwise regular. Set the number of wash/rinse cycles you want (more for grubby whites!). Push in the dial to start the machine. Then wait. When the machine beeps, it's done, and you can remove the clothes. Hang permanent press and delicates to dry, but put the rest in the dryer, setting the temperature as recommended on clothing labels.

The Wizard of Oz parsed the subject's initial description for non-function words (in particular, nouns and verbs) and suggested a domain terminology. For instance:

re. objects: laundry, whites, colors, permanent press, load, detergent, machine, pile, clothes, temperature, hot, etc.

re. actions: sort, wash, load separately, add, pile, loosely, set, wash, rinse, push, start, wait, beeps, remove, hang, etc.

re. actors: you

The subject could add or remove items. In both this and our user study of the elicitation tool, we found that subjects were "unclear" about what constitutes an action, sometimes selecting a verb, sometimes an entire sentence. But we do not wish to require such "clarity" of our domain experts: that is the knowledge engineer's job.

In the next round, the Wizard parsed the description for sentence breaks and verb clauses, to identify the steps of the task:

1) First you sort the laundry into whites, colors, etc.
2) You wash each load separately.
3) Add the detergent to the washing machine
4) and pile the clothes in loosely.
etc.

The subject was asked to critique this breakdown and rearrange the lines as appropriate. In general, subjects found this process easy and the researcher who reviewed their work considered the breakdown to be accurate. Subjects who were programmers spent more time on this than did non-programmers.

In the fourth round, the Wizard rearranged the line-by-line breakdown into a task-subtask hierarchy with conditionals. Since our Wizard was not supposed to have domain knowledge, it performed only a trivial task decomposition, by making a heading for the task as a whole:

Task: Do the Laundry
Subtasks:
1) First you sort the laundry into whites, colors, etc.

The subject was free to insert new headings and to group actions into subtasks. As an example, the Wizard showed them a decomposition of the "Grocery shopping" task. Non-programmers had difficulty with this step because they did not know where to begin. Programmers produced reasonable decompositions.

In the final round, the Wizard annotated the subject's task outline with sequencing information. By default, actions were assumed to be unordered. Keywords like "first", "then" and "after" alerted the Wizard to sequences:

Subtask: Washing
Do the following in order:
1) First you sort the laundry into whites, colors, etc.
2) If the load is whites, then
4a) set the temperature to hot
4b) Otherwise ??
3) If bright colors, then
5a) set the temperature to cold
5b) Otherwise warm.
etc.

All subjects had difficulty with this step. Non-programmers did not know where to begin, and programmers struggled with it. We conclude that a different form of dialog is needed, such as asking the domain expert whether a given set of actions must be done in order.

In general, subjects performed well correcting the Wizard's suggestions. Providing examples of task structure helped tremendously: where examples were perfunctory, our instructions failed to help users. The gradual refinement process broke the task description problem into transformations that users could manage and that a system could feasibly automate.

For our implementation, we chose a subset of the functionality simulated in the Wizard of Oz experiment. We adopted a purely user-driven approach so that we could test whether this was adequate before we attempted the implementation of a complex system.

THE USER-TASK ELICITATION TOOL (U-TEL)

Figure 1 illustrates the user-task elicitation tool that we implemented. The upper left window contains the text of the user's initial task description. Below it is an outline editor in which the user organizes the description into steps and subtask hierarchies. To indicate sequencing, the user selects a subtask node and chooses one of the following options from a popup menu:

- Steps must be done in sequence
- Steps may be done in any order
- Subtask may be repeated

Note that the "repeated" annotation may be given in addition to one of the other two. Other forms of control flow, such as interleaving, were not provided because we felt that users might not understand them.

The three windows at the right contain lists of domain terms for objects, actions, and actors. Each list is hierarchical, but the tool does not indicate whether a hierarchy stands for taxonomy ("is-a") or containment ("part whole"). During preliminary design studies, we found that some users wanted a hierarchy, but that in

practice they did not distinguish between these two semantics. Our policy is to avoid imposing knowledge-engineering distinctions upon domain experts, so we let them use the hierarchy however they wish, and even inconsistently. To classify a word or phrase, the user selects it in the text or outline view and chooses “thing”, “action”, or “user” from a popup menu. All occurrences of that text in both views are highlighted accordingly. The user may classify a given piece of text under more than one category.

At the end of the session, the system can store the results of the elicitation into a declarative knowledge representation understandable by the MOBI-D tools. This representation is dictated by the MOBI-D interface modeling language and it stores the elements of the elicitation into the appropriate components of an interface model. Thus, the outline becomes a user-task model; the list of objects a domain model; and the list of users, a user model. These models, of course, are just skeletons and need refinement and specification, via MOBI-D tools, by a knowledge engineer or interface designer.

USER STUDIES ON U-TEL

To evaluate U-TEL, we conducted user studies with a balanced group of eight subjects. The subjects included males, females, programmers and non-programmers. Each subject received basic training in the use of U-TEL and then completed an outline for each of two standard tasks. We subdivided each task into steps similar to those of the Wizard-of-Oz simulations. We then measured the time for completion of each step, assessed the difficulty of each step on a discrete scale, and recorded the verbal comments from each user.

Remarkably, programmers and non-programmers needed similar amounts of time to finish the tasks. Whereas programmers were able to quickly classify terms (as opposed to non-programmers), they spent considerably more time in refining the outline than non-programmers, thus balancing the total time needed. All subjects agreed on the level of difficulty of each step and none marked any step as difficult. Comments from users indicated the need for providing various drag-and-drop operations in the outline window, as well as identified the need for using tree hierarchies for the lists of objects, things, and users (these lists were flat originally).

ELICITING A USER-TASK MODEL FOR A MEDICAL INTERFACE

The example shown in Figure 1 is for a user interface to visualize temporal data in the medical domain. The typical use of the elicitation tool is to obtain user-task outlines from more than one domain expert. In this case, we ran sessions with two physicians. One has programming experience and is an expert in temporal data abstractions. The second one is an expert in the field of diabetes (which requires considerable temporal data visualization) but has no programming experience. The example shown is for the expert with programming experience.

In this case, the programmer expert was able to produce a useful and detailed outline of the user-task model. However, the terminology used by this expert was very abstract and resulted in a labeling of interface elements that was not comfortable to end users. The non-programmer expert, however, while not producing as detailed an outline, did supply a terminology that was much more acceptable to end users. The job of reconciling these examples and selecting the best features of each fell on the interface designer. This illustrates how the user elicitation tool helps the interface design process by adding a structured channel of communication between domain experts and interface designers.

FURTHER WORK

There are two areas of potential improvements to U-TEL: intelligent assistance and graphical editing capabilities. We plan to add some of the intelligent parsing and advice functions tested in our Wizard-of Oz studies. We will do so in an incremental fashion and as dictated by our continuing evaluation by end users. We are also planning to address one fundamental limitation, that is the lack of facilities to describe a task diagrammatically instead of by text outline.

CONCLUSION

We have developed a methodology for eliciting models of user tasks from domain experts and implemented tool support for that methodology. Our user-task elicitation tool allows domain experts to refine a textual description of a task into a structured outline. The results of the elicitation sessions can be used directly in a model-based interface development environment to build interface models from a user-centered point of view.

Our tool structures and supports the difficult step of defining user-task models while increasing the usefulness and acceptability of model-based systems.

ACKNOWLEDGMENTS

The work on MOBI-D is supported by DARPA under contract N66001-96-C-8525. We thank Eric Cheng, Jacob Eisenstein, Kjetil Larsen, and Justin Min for their work on the development of MOBI-D.

REFERENCES

1. Hix, D., Hartson, H. *Developing User Interfaces. Ensuring Usability through Product and Process*. New York: John Wiley & Sons, 1993.

2. Puerta, A.R. *A Model-Based Interface Development Environment*, IEEE Software, July/August 1997, pp. 40-47.

3. Puerta, A.R., Eriksson, H., Gennari, J.H., Musen, M.A. *Beyond Data Models for Automated User Interface Generation*, in Proc. of HCI'94, pp. 353-366.

4. Szekely, P., Luo, P., Neches, R. *Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design*, in Proceedings of CHI’92, pp. 507-514.

5. Wilson, S., Johnson, P., Kelly, C., Cunningham, J., Markopoulos, P. *Beyond Hacking: a Model Based Approach to User Interface Design*, in Proc. of HCI'93, pp. 217- 231.