Beyond Hacking: a Model Based Approach to User Interface Design

S. Wilson†, P. Johnson†, C. Kelly‡, J. Cunningham* and P. Markopoulos†,

†Department of Computer Science, Queen Mary and Westfield College, University of London, Mile End Road, London E1 4NS

Email: {steph, pete, markop}@dcs.qmw.ac.uk

‡British Aerospace plc, Sowerby Research Centre, PO Box 5, Filton, Bristol BS12 7QW

Email: kelly@src.bae.co.uk

**British Maritime Technology Ltd, Orlando House, 1 Waldegrave Road, Teddington, Middlesex TW11 8LZ*

Email: jlc@bmtech.uucp

This paper discusses the role of models in the design of user interfaces, with particular emphasis on integration across different modelling stages. We are concerned with bridging the gap between psychologically motivated modelling approaches to HCI and implementation oriented interaction models, to produce a task-informed user interface design process. An early version of a UIDE which provides support for the role played by models in the design process is described and exemplified through models taken from a case study. We conclude with an assessment of our experiences and a discussion of how the work will proceed.

Keywords: User Interface Design, Task Modelling, User Modelling, UIDE

1. Introduction

Designers of interactive systems are creative within a design space constrained by requirements originating from widely varying perspectives. From a user-centred perspective come considerations of tasks and general user characteristics, from the implementation platform come hardware and software technology constraints and from the application domain and client organisations originate yet more requirements. Despite the fact that user interface design methodologies encompass these issues, the various requirements are not made explicit within the one design support environment. As a consequence, it is difficult to ensure that the user-centred requirements addressed in the early stages of design are carried forward into the later stages.

Adept is a novel design environment for prototyping user interfaces which addresses these issues by encouraging the designer to construct explicit models of the users and their tasks. The aims of the work on the Adept project were twofold: first, to investigate how knowledge of users' tasks can inform the design of a system to support those tasks and, second, to provide an integrated, model-based environment to support user interface design and development.

The following section motivates our work through a discussion on the role of models and tools in the design of user interfaces. These ideas have been illustrated recently in a software demonstrator implemented for the Adept project which is described in the third and fourth sections of this paper. The fifth section discusses the contribution and status of this demonstrator and assesses how far it has gone towards our goal of an integrated, task-oriented, user interface design environment.

2. User Interface Design

2.1 Tools and Models in User Interface Design

User interface design is difficult and designers require effective models and tools to enable the design process. Models serve to make explicit the results of analyses and of design, while tools serve to support the creation of models and new designs. In spite of the plethora of work which has been reported in each of these areas, there remains a gulf between tool-based and model-based approaches to user interface design.

User interface management systems (UIMSs) represented some of the earliest tools to focus specifically on the development of the interaction component of an interactive system (see (Hix, 1990), (Myers, 1989) for an overview of the subject). Their aim was to promote rapid prototyping of the user interface by separating the user interface software from the application software and by partially automating the design process. The general scheme was that the designer parameterised a generic UIMS with a model of the desired interaction, often concentrating on the dialogue, and let the UIMS generate the user interface and assume responsibility for run-time management of the interaction. However, at the best, these systems offered the designer only the crudest of tools with which to design the interaction model. Recognising that effective support for the design stages is more likely to encourage and ease iterations through prototypes, modern user interface design environments (UIDEs) place less emphasis on managing the interaction and more emphasis on supporting the designer. One of the more sophisticated UIDEs is Garnet (Myers et al., 1990) which provides the designer with a rich set of primitives and components, together with a collection of graphical editors and interface builders. However, in common with other UIDEs, Garnet supports only one aspect of the design life cycle: the construction and prototyping of the user interface software, while neglecting requirements, specification and evaluation. It is surely time that advances were made towards supporting the entire user interface design process within an integrated environment.

There also exists a significant body of research on the role of models in HCI. Different models serve a purpose at different stages of the design process (Wilson et al., 1992). For

example, user models describe the users' abilities and beliefs, task models express the knowledge required or procedures used to perform some task, dialogue models describe the users' interaction with a system, architectural models provide a paradigm for the construction of interactive systems, formal models provide a specification of the interaction or of the software, etc. However, without tool support many of these techniques are cumbersome and have limited practical use. Furthermore, it is too often the case that those models concerned with the user are not made explicit and therefore rely on the designer's intuition to influence the design. All models which contribute to the design should be made explicit and, moreover, should be made explicit within the integrated design environment mentioned above.

2.2 Task Based Design

One focus of the Adept work is the notion that user interface design should be task-centred. A model of the users' existing knowledge about a task, termed the 'initial task model', provides the starting point for the design of an artefact to support that task. This should generally be accompanied by a statement of design goals expressing either general aims or recognising problems in the existing task which the design of the artefact is to address. For example, aspects of the existing task may be error prone or inefficient and it would be a design goal to rectify such problems. Design goals may be applied at two levels. First, the task itself may be subject to some redesign in accordance with the design goals and, second, given that alternative artefact designs could support the task expressed in a task model, the design goals serve as a mechanism for differentiating between the design solutions. It is of no consequence if there does not already exist an artefact that supports the task directly, although there will generally be a greater distance between existing and designed tasks in this case.

The user acquires new task knowledge as a consequence of using the newly designed artefact; we term this the 'resultant task model'. Clearly, there is no guarantee that the resultant task model will be the same as the designed task model which was used in the creation of the artefact. The resultant task model may, in some future design scenario, function as the initial task model for the design of another artefact. This is similar in philosophy to the notion of task evolution presented in (Carroll, Kellogg & Rossen, 1991) whereby a designed artefact opens up new possibilities for human tasks, which in turn place requirements for new artefacts — the so-called 'task-artefact cycle'.

The remainder of this paper describes the Adept software demonstrator which illustrates how users' existing task knowledge can contribute to the design of an interactive system to support that task and shows how integrated tool support can be provided for the design process. In so doing, we hope to approach the design of user interfaces through a sound modelling basis rather than arbitrary hacking.

3. The Adept Design Environment

The previous section motivated the Adept work, culminating in two crucial points. The first is that all the models used in the design process, irrespective of whether they result from analyses or from design, should be made explicit and available within an integrated

environment which facilitates the transfer of knowledge from one modelling stage to another. The second point is that user interface design should be user centred and therefore should take account of the users' characteristics and the tasks which they wish to carry out. Adept achieves this by encompassing design from task analysis, through specification, to the generation of a run-time system, via a number of modelling stages. This is developed in more depth below. First, the context is set by summarising the models upon which the design environment is based and outlining the associated design philosophy.

A number of models are central to the Adept demonstrator: task models, user models, abstract interface models and concrete interface models. These models support different stages of the user interface design life cycle. Task models in Adept represent either the knowledge possessed by users and recruited in performing some existing task or the knowledge which will be required to perform a designed task. User models provide a perspective on more general user characteristics which are not captured by the task models: they typically model levels of expertise and types of knowledge system of the user group. Abstract interface models (AIMs) provide a high-level description (or specification) of the interface and, finally, concrete interface models (CIMs) provide a low-level description of the interface: an instantiation of a CIM is an implementation.

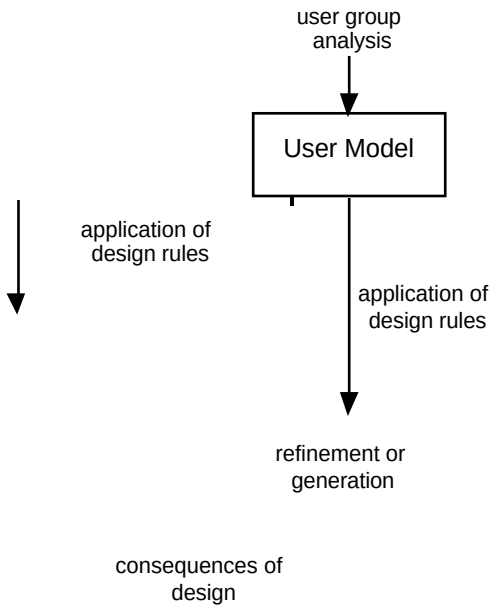

Figure 1 (modified from (Johnson et al., 1993)) shows these models and summarises how they are utilised in the process of user interface design; section 4 provides further details. The various tools in the Adept demonstrator support the design processes either through manipulation of the models or by facilitating transitions from one modelling stage to another. Those processes represented by solid lines in Figure 1 are supported directly by the tools of the Adept demonstrator, the remainder require further development. The discussion here focuses on the models rather than the tools which as individual entities (e.g. task model editor, graphical interface builder) are not unique to Adept and are therefore of less interest to the research. The description is illustrated with example models developed in a case study carried out with a group of radiographers from the London Hospital at Whitechapel. A task analysis was performed with the radiographers for the task of taking an X-Ray examination to be used in clinical diagnosis.

3.1 An Integrated Design Process

In any user-centred design philosophy, the starting point for the design of the interface component of a system is an analysis of the prospective users of that system. There are two distinct aspects to this: the first is a task analysis and the second is a more general analysis of the anticipated user group. The results of these analyses are expressed as task models and user models respectively.

In Adept, the initial task model elicited by the task analysis procedure provides the basis for the interface design, while the information in the user model is used to influence the design and to select among alternative solutions in the design space. Therefore, it could be said that at least one design goal is always applied to user interfaces prototyped with the Adept demonstrator: to select design alternatives which are best suited to the user group as characterised by the user model. Or, in other words, to promote user-centred design.

As mentioned in section 2.2, the initial task model does not generally constitute a design model; in most cases the task is subject to some redesign, directed towards satisfying the design goals. The new task is represented by the box labelled "Designed Task Model" in Figure 1. The following design stage involves moving from redesigning the task to designing (at a conceptual level) an interface to support that task, expressed as an AIM. The AIM specifies what the interface will do, but not how it will be achieved. The final stage of the development is the progression from the AIM specification to implementation as a CIM. It is at this stage in the Adept demonstrator that the user model exerts its influence, although there is no implicit constraint in the user model which dictates that its influence is limited to this area. The additional user characteristics expressed in the user model provide guidance in discriminating between alternative CIMs which satisfy the AIM specification.

The preceding discussion suggests a top-down approach to design; nonetheless, it is not our intention to be prescriptive about the design process and therefore designers are free to work at whatever level of abstraction they find appropriate. Ultimately, the goal should be to produce a design environment which is sufficiently integrated that the consequence of design decisions made at any modelling level can be perceived at any other level (either automatically or under designer control). So, for example, the consequence of deleting

some component at the implementation level (e.g. a numeric slider) would percolate upwards so that its repercussions for the task could be observed and analysed.

4. The Adept Demonstrator

4.1 The Task Model

To achieve a user task-based approach to user interface design, a task modelling technique is required which can feed into the creation of new designs. Many existing task modelling techniques (e.g. TAL (Reisner, 1981), TAG (Payne & Green, 1986)) were intended primarily as mechanisms for analysing existing designs by predicting user performance, rather than as modelling techniques which can feed into new designs. Exceptions do exist, for example, TAKD (Diaper, 1989) and Task Knowledge Structures (TKS) (Johnson & Johnson, 1991) are more suited towards expressing users' existing task knowledge. In TKS a goal sub-structure component describes the users' goals and sub-goals; a procedural component describes how the goals will be achieved in terms of sequences of actions; and a taxonomic sub-structure component identifies task objects, their attributes and the actions that are performed upon those objects.

It is clear that components of a model such as TKS have direct relevance to the design of a user interface and therefore can feed forward into a design. There are three areas where this design relevance is most obvious: sequencing information in the task model can feed into the design of interaction sequences in the user interface; task objects may be realised as interaction objects in the interface and task actions may be realised as actions upon the appropriate interaction object.

Adept has adopted a modified version of TKS: temporal relations are an essential part of our task modelling approach and have been influenced by related work on the use of formal languages to express task models and models of user interfaces (Johnson, Markopoulos & Johnson, 1993). Temporal relations are specified between the goals, sub-goals, procedures and actions of the task model. Initially, we have considered four temporal relationships between task components (these will be extended): sequence, interleaved, parallel and choice. They can be described informally as follows, where A and B are goal, procedure or action components of a task model:

Sequence	(A >> B)	The components are performed in strict temporal sequence
Interleaved	(A ||| B)	The components are performed simultaneously, with the user alternating activities between them
Parallel	(A || B)	The components are performed concurrently
Choice	(A [] B)	One of the two task components is performed (the user has a choice in the execution of the task)

An interleaved type is particularly useful for describing temporal relationships in direct manipulation interfaces where there are often a number of tasks on offer simultaneously to a user. Consider the situation where a user has a word processing application open in one window and an electronic mail application in another. Both of these tasks are available simultaneously, but at any instant the user may be engaged in at most one of the tasks — the user interleaves activities by switching between windows. These temporal extensions

to TKS are similar to those included in the enhanced version of UAN described by Hartson & Gray (1992) who discuss one advantage of an interleave operator as being that it provides a more concise expression of such asynchronous behaviour than would be possible using a state transition network where every possible state and state change must be recorded explicitly. The notion of interleavability allows the possible transitions between asynchronous tasks to be expressed implicitly.

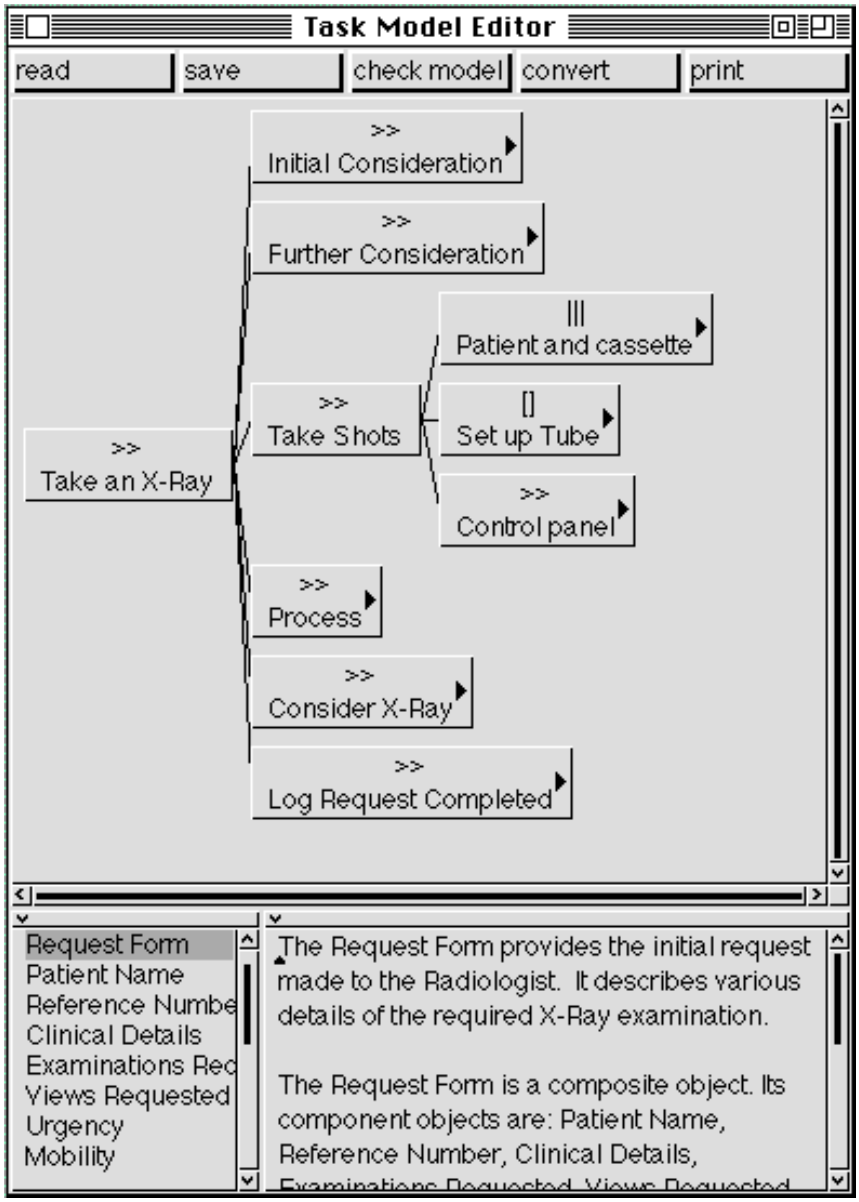

Figure 2. A task model and editor

The approach adopted by Hartson & Gray is not dissimilar to our own, but there are distinctions: although Hartson & Gray use the term 'task model' when referring to their notation, it is in fact used to provide a detailed description or specification of the interaction design at a level of detail comparable to the abstract interface model in Adept. Moreover, the extended UAN work does not have the same focus as Adept on a complete

methodology for user interface design commencing from a representation of the users' existing task knowledge.

Figure 2 shows the current version of the Adept task model editor which supports the designer in editing and browsing task models. The upper view in the editor contains a graphical presentation of the goals, sub-goals, procedures and actions, while the lower views display the objects of the taxonomic sub-structure and their descriptions. This tool supports syntax directed editing with a menu based interface, automatic graphical layout and hiding of information under user control. Task models are represented internally as abstract syntax trees, facilitating alternative presentations to the user, one of which is an equivalent textual representation.

This editor imposes a restriction not inherent to the underlying model that the same temporal relation must exist between all sub-components of a given component. An alternative view is that all components are considered to be of a certain 'type' which expresses the temporal relationships between its sub-components. For example, if a goal is denoted to be of type 'sequence', then all its sub-goals have been identified as being performed in a strict temporal sequence. The editor shown in Figure 2 contains the initial task model for the radiography task and a small portion of the goal sub-structure component is visible in the upper view. The top-level goal for the task is "Take an X-ray"; this is a sequential goal (denoted by the >> symbol) and therefore its six component sub-goals are performed in sequence, from "Initial consideration of the X-ray request" through to "Logging that the request has been completed". This restriction significantly reduces clutter in the display of the task model and simplifies the editing operations. Where necessary, dummy nodes may be introduced to allow more complex relationships to be expressed.

4.2 Designing a Task Model

One particularly novel aspect of Adept is that we have made explicit the notion of redesigning the task. This allows us to acknowledge the important, indeed sometimes critical, contribution of the users' existing task knowledge, while recognising that new artefacts are often designed with the specific intention of changing that task. It would indeed be a constrained design scenario which did not allow the designer the creative freedom to introduce new design ideas. However, in redesigning the task, Adept encourages new ideas to be expressed in terms of the task to be performed rather than in terms of low-level details of screen design which obscure from the designer the potential consequences of the changes for the users.

The task model editor supports the user interface designer in constructing the initial task model and in redesigning the task in accordance with design goals. There are a number of activities involved in designing the task, one of which is to decide which of the sub-tasks in the initial task model should be supported by the new artefact. The initial task model describes a complete task from the users' perspective, but the artefact to be designed may only be intended to support some portion of that task. For example, we made the (fairly arbitrary) decision in the radiography case study to support mainly the sub-task of setting up the control panel and to disregard sub-tasks such as positioning the camera and developing the film. Other sub-tasks were deleted because they involved physical actions

(e.g. collecting the patient) or mental processes on the part of the task performer. It is not only the goals representing the sub-tasks which may be deleted: the same is true of both procedures and actions. Moreover, new goals/procedures/actions may be added and the task may be 'restructured'. Another aspect of the redesign process is introducing flexibility into the task description: the initial task models tend to be very sequential in nature and have a deep hierarchic structure. In general, a less moded interface design will result if appropriate temporal relations in the designed task are designated to be interleaved rather than sequential, so that the new artefact facilitates alternating between activities. Clearly, this is important in the design of direct manipulation interfaces. The designed task is sometimes referred to as the "computer task model" because it represents the task which the interactive system will support.

The explicit nature of both the designed task model and the AIM discussed in section 4.4 offers the opportunity to explore design alternatives at an early stage in the development life-cycle. This could be achieved through either analysis or animation of the models.

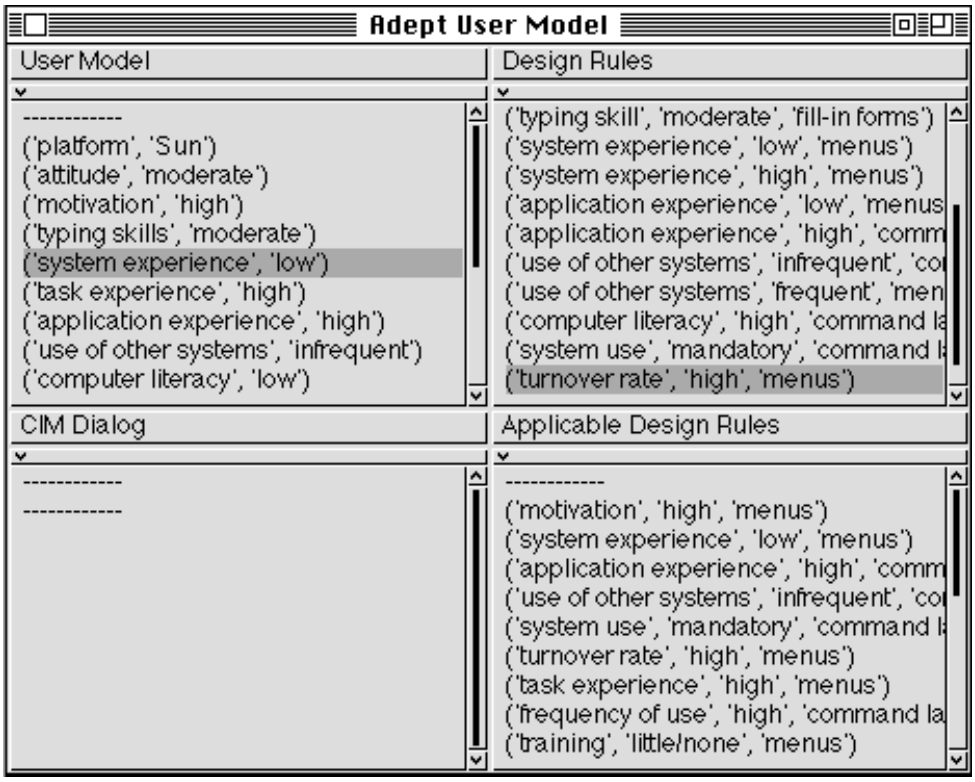

Figure 3. User model tool

4.3 The User Model

The user modelling component of Adept provides information about the users additional to that contained in the task model: it captures general user characteristics. A detailed description of its form and function is given in (Kelly & Colgan, 1992).

The user model constitutes a picture of the typical user, modelled in terms of their abilities, knowledge and styles of information processing. It is hierarchical in structure and includes models of the generic user, task independent and task dependent characteristics. These characteristics are expressed as simple facts held in a knowledge base. For example, three facts taken from the radiography user model state that the radiographers have low system experience, use other systems infrequently and have high task experience, as follows:

```
('system experience' 'low')
('use of other systems' 'infrequent')
('task experience' 'high')
```

The interface designer instantiates a new knowledge base of user model facts for each group of users (e.g. the radiographers) to specify their task dependent and task independent characteristics (the generic user level is intended to contain information applicable to all user groups). This activity is supported by the user model tool for the Adept demonstrator, shown in Figure 3, where the user model facts are displayed in the upper left view.

The user model shapes the user interface development by constraining the set of possible options in the design space. This is achieved through another knowledge base incorporated within the user model tool, displayed in the upper right view of Figure 3. This knowledge base incorporates design rules which associate specific user characteristics with usability knowledge and user interface design guidelines. For example, the design rules below state that if the users have moderate typing skills, then fill-in forms might be an appropriate form of interaction, and if their system experience is low then menus might be appropriate:

```
('typing skills' 'moderate' 'fill-in forms')
('system experience' 'low' 'menus')
```

Matching the design rules against an instantiated knowledge base of user model facts results in that subset of the design rules which should be applied to the design of an artefact for that user group performing the given task(s). Therefore, the ('system experience' 'low' 'menus') rule would be among those applicable to the radiography task. The applicable design rules are displayed in the lower right view of the tool.

4.4 The Abstract Interface Model

The abstract interface model (AIM) is an important intermediate modelling stage in the transition from task model to implementation. The AIM provides a high-level description of what the interface is to do, without specifying how it is to be achieved and is therefore a useful expression of the design at the conceptual level. The designer can edit and elaborate the AIM using the editors and browsers of Adept; an AIM from the radiography task model is shown in Figure 4.

At present, the AIM has a simple structure. It contains two major types of components: groups and abstract interaction objects (AIOs). A group is essentially a collection of other components; it can include other groups, leading to a nested hierarchical structure. The group at the root of this tree represents the complete interface. AIOs are abstract descriptions of the interaction objects in a user interface and form the leaves of the tree which is the AIM. At the AIM level, details of interactive behaviour are unimportant — it is only the purpose of an interaction which is significant. AIOs are therefore classified in

terms of the type of input which they describe, for example, a numeric AIO specifies that the user should provide a numeric input, but makes no statement concerning whether this interaction will be through a slider object, or type-in text field, etc. Clearly, numerous alternative implementations at the CIM level could satisfy the description given in an AIO.

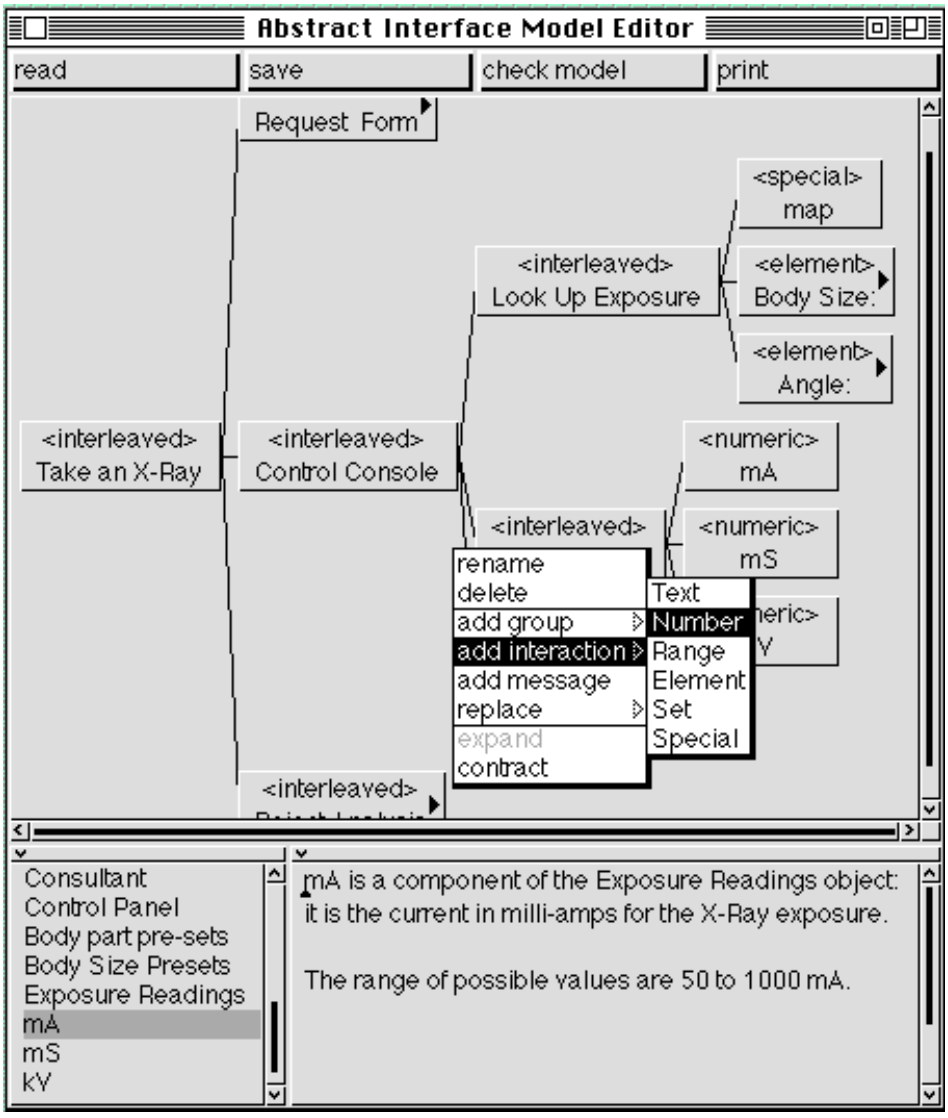

Figure 4. Abstract interface model and editor

As in the task model, temporal relations may be specified between the components of the AIM. The combination of this temporal information and the hierarchical structure of the AIM groups is important: embedded within it is the sequencing information, equivalent to the dialogue model of a UIMS. The tree could be translated into other dialogue specification notations, for example, a grammar where the AIOs formed the terminal symbols. The AIM structure also emphasises the relationship between the AIOs and the temporal aspects of the interaction and is used as a guideline for laying out screens at the CIM level. This indicates the close relationship between presentation and dialogue often ignored in UIMSs.

4.5 Designing an Abstract Interface Model

At some stage, the user interface design process must move from task oriented concepts to interface oriented concepts. This is represented by the transition from designed task model to AIM. Our view on this is that while the designed task model describes the task (in terms of its task knowledge requirements) which will be carried out with the designed artefact from the users' perspective, the AIM describes the artefact which will support that task, from the computers' perspective. This is a 'sideways' move involving consideration of the form of interface appropriate to support the designed task.

In the Adept demonstrator, a simple automated process assists the designer in moving from a designed task to an AIM by attempting to produce an equivalent AIM in which the goals and procedures of the task model are replaced with groups representing the dialogue structure in the AIM. The most interesting aspect of the AIM design is the destiny of task actions. It is at the level of actions in the task model that the 'real' activities of the task are carried out and ultimately these actions should be supported by interaction objects in the user interface. At the level of the AIM, the designer needs to decide what form of interaction best supports the task action. For example, the radiography task contained actions such as 'Set the current' and 'Set the voltage' for the X-ray examination. In both cases, the action is 'Set', while current and voltage are the objects. These task actions were replaced by numeric AIOs in the AIM, illustrating how the choice of AIO is often influenced by the description of the objects which were involved in the task action. Other relevant properties of the AIOs can also be specified.

As mentioned previously, we do not have a regimented view of the interface design process, rather we view it as a continuous process. The designers' preference dictates what proportion of the design is carried out at the task level and what at the AIM level. Therefore, design activities continue at the AIM level with interface concepts similar to those which occurred with task-oriented concepts: the AIM may be restructured or flexibility may be introduced as part of the design process to meet the stated design goals.

4.6 The Concrete Interface Model

The concrete interface model (CIM) is a platform independent description of the interface design at a detailed level of interaction objects, their behaviour, screen layout and sequencing; it is a low-level description of the user interface from which a run-time system may be produced by a compiler or interpreter tool. Within the demonstrator, an interpreter provides this functionality. Furthermore, a translator tool in the demonstrator produces an equivalent, platform dependent, implementation from the CIM which makes use of the Open Look™ widget set. Adept is not committed to any particular implementation platform and therefore other platforms could be accommodated by writing alternative translators (obvious choices would be a Motif widget set or the Garnet gadgets). The interface design can be fine tuned at the CIM level using conventional graphical editors.

™Open Look is a trademark of AT&T

4.7 Generating a Concrete Interface Model

The CIM could be developed from the AIM through a refinement process carried out by the designer. However, as an experiment in intelligent design automation, the demonstrator incorporates a generator tool to provide this functionality. The generator fulfils a number of functions, for example, it selects one or more CIM interaction objects as needed for every AIO in the AIM and instantiates them appropriately (e.g. a slider or type-in text field might be selected in the CIM for a numeric AIO in the AIM). It also performs complex calculations to generate layouts which are ergonomically desirable and which satisfy the temporal constraints on the interaction given in the AIM. Sequencing information is also incorporated in the CIM and, where necessary, new interaction objects are created to control context switches between stages of the dialogue.

Many alternative user interfaces could satisfy the conceptual level design given in the AIM: the generator is guided in deciding between design alternatives both by in-built heuristics and by input of user centred design principles from the user model tool. Most of these design constraints guide layout and presentation. In so doing, an implicit design goal being applied at this level is that the user interface should be suitable for the user group characterised by the user model facts. The interaction between the CIM generator and the user model tool takes the form of question/answer dialogues which are carried out automatically during the processing of an AIM. The CIM generator poses questions and the user model tool attempts to reply from its set of applicable design rules. This dialogue is displayed in the lower left view of the user model tool shown in Figure 3. On occasion the situation arises where the CIM generator poses a question which the user model tool cannot answer. There are two alternative solutions: the normal resolution is that the question is passed on to the interface designer who is provided with a dialogue box in which to answer. The long-term solution is that appropriate rules should be added to the user model expressing the recommended approach.

5. Discussion

Adept has successfully demonstrated a novel approach to user interface design. It espouses an integrated, model-based view of user interface design within which the human factors contribution is made explicit. One large design problem and a number of smaller ones have been tackled. Our aims are similar to those expressed in (Neches et al., 1993) who describe MASTERMIND, a step towards their vision of "integration and continuity across the entire life cycle of the user interface". They envisage a knowledge-based, design-time and run-time environment. They anticipate various advantages to an environment which incorporates explicit models, including the provision of semi-automated design critics and early exploration of design alternatives (properly based on the task and user models). While Adept has focused on the task and user models, work to date on MASTERMIND has focused on the intelligent design tools and 'generic model' — a high-level specification of the interface, similar in philosophy to the AIM in Adept. Work on MASTERMIND commenced from a base of existing tools which were to be integrated through shared models, while Adept set out to design new tools to support the requisite models.

The Adept demonstrator has served its purpose in illustrating the feasibility of our ideas, however its scope is limited, for example, it supports only the creation of interfaces comprised from a set of pre-defined interaction objects. Another development phase is planned which will improve the tools and enrich the models of the design environment. In particular, the information in the taxonomic sub-structure of the task model is under utilised. We envisage elaborating the object descriptions to include details of attributes and relationships and then using this information as the basis for the design and implementation of new, task-specific, interaction objects. This would facilitate the transfer of knowledge from the task objects to the interface design and will also involve enhancing the AIM and replacing the CIM with a full architectural model along the lines of PAC (Coutaz, 1989). The user model also demands further work to enhance its knowledge bases and to investigate more sophisticated application of the design rules. The improved Adept UIDE will then be subject to a more formal evaluation than has been carried out with the demonstrator. Of particular interest would be the evaluation of an interface design produced through our task based design methodology against a design for the same system produced using more conventional design techniques.

We would not claim that the Adept models suffice to capture all information relevant to interface design. This is clearly not the case: task models and user models are important representations of the users and their task knowledge but other contributing factors include the social and physical environment (ISO, 1991) and the application domain. An environment such as Adept could also include explicit models of these factors and their design relevance, where this can be ascertained.

While the discussions herein have focused on design, other advantages are accrued from the explicit nature of the models in Adept, combined with the fact that a 'history' of models is held within the system. One potential benefit is concerned with the evaluation phase of user interface design which has not been covered here. The history should provide a basis for evaluating design alternatives through comparison and analysis of the models, for example, the initial task model could be compared with alternative designed task models. A more speculative benefit is that a study of this history could yield a picture of the design process as the models progress from task model through to implementation which in turn should help us to understand how we can best support the design process. Studies of design practice could be enhanced by adding support for a design rationale notation (e.g. (MacLean et al., 1991)) to the Adept tools, thus encouraging the explicit documentation of decisions in the design space with respect to the design goals. This could be further supplemented with observation and interview techniques. In spite of the fact that the user model tool contains a knowledge base of design rules, much of the Adept environment is unintelligent: tools and models support designers but ultimately leave most design decisions in their hands. By examining design practice, we can gain design knowledge and add it to our system. Such knowledge bases would then support either further automation of the design process or the provision of intelligent design guidance; of particular interest are design constraints which operate between the task model and AIM levels.

Clearly, the system we envisage (which might be termed a 'user interface modelling and design environment') is a large one. Adept has gone some way to look at the models which will form the heart of such a system; this work must now be extended and developed in a number of directions to produce a sophisticated, model-based design environment for

interactive systems. In this way some of the many diverse strands of user interface design can be drawn together and, hopefully, better design practice will result.

Acknowledgements

We are grateful to Mario Wolczko of the University of Manchester for his Grapher 'goodie' which was used in the implementation of the task model and AIM editors. Lynne Colgan of British Aerospace was responsible for the design and implementation of the user model tool. The tools

The Adept Project (Advanced Design Environments for Prototyping with Task Models) is a collaboration between Queen Mary and Westfield College (University of London), British Aerospace Plc, British Maritime Technology Ltd and MJC^2. It is led by Queen Mary and Westfield College and is funded by the DTI and SERC (Grant No. IED 4/1/1573).

References

J.M. Carroll, W.A. Kellogg & M.B. Rossen (1991), "The task-artifact cycle", in J.M. Carroll (ed.), Designing Interaction: Psychology at the Human-Computer Interface, Cambridge University Press, pp. 74-102.

J. Coutaz (1989), "Architecture Models for Interactive Software", Proc. ECOOP '89, S. Cook (ed.), Cambridge University Press.

D. Diaper (1990), "Task Analysis for Knowledge Descriptions (TAKD): The Method and an Example", in Task Analysis for Human-Computer Interaction, D. Diaper (ed.), Ellis-Horwood, pp. 108-159.

H. Rex Hartson & Philip D. Gray (1992), "Temporal Aspects of Tasks in the User Action Notation", Human-Computer Interaction, vol. 7, no. 1, pp. 1-45.

Deborah Hix (1990), "Generations of User-Interface Management Systems", IEEE Software, vol. 7, no. 5, pp. 77-88 Sept.

ISO 9241 (1990), "ISO 9241 Ergonomic Requirements for Office Work with Visual Display Terminals" Parts 5&6, Environmental Requirements, Committee Draft.

H. Johnson & P. Johnson (1991), "Task knowledge structures: Psychological basis and integration into system design", Acta Psychologica, 78, pp. 3-26.

P. Johnson, P. Markopoulos & H. Johnson (1993), "Task Knowledge Structures: A specification of user task models and interaction dialogues", to appear in M. Tauber (ed.), Task Analysis for System Design.

Peter Johnson, Stephanie Wilson, Panos Markopoulos & James Pycock (1993), "ADEPT — Advanced Design Environment for Prototyping with Task Models", Demonstration Abstract, Proceedings Interchi 1993, ACM, p. 56.

Chris Kelly & Lynne Colgan (1992), "User Modelling and User Interface Design", in People and Computers VII, Proceedings of the HCI'92 Conference, Cambridge University Press, pp. 227-239.

Allan MacLean, Richard M. Young, Victoria M.E. Belotti & Thomas P. Moran (1991), "Questions, Options and Criteria: Elements of Design Space Analysis", Human-Computer Interaction, vol. 6, nos. 3&4, pp. 201-250.

Brad A. Myers (1989), "User interface tools: Introduction and Survey", IEEE Software, vol. 6, no. 1, pp. 15-23. Jan.

Brad A. Myers, Dario A. Guise, Roger B. Dannenberg, Brad Vander Zanden, David S. Kosbie, Ed Pervin, Andrew Mickish & Philippe Marchal (1990), "Comprehensive Support for Graphical Highly-Interactive User Interfaces: The Garnet User Interface Development Environment", IEEE Computer, vol. 23, no. 11, pp. 71-85.

R. Neches, J. Foley, P. Szekely, P. Sukaviriya, P. Luo, S, Kovacevic & S. Hudson (1993), "Knowledgeable Development Environments using Shared Design Models", Proceedings of 1993 International Workshop on Intelligent User Interface, ACM press, pp. 63-70.

S.J. Payne & T.R.G. Green [1986], "Task-Action Grammars: A Model of the Mental Representation of Task Languages", Human-Computer Interaction, vol. 2, pp. 93-133.

P. Reisner [1981], "Formal Grammars and human factors design of an interactive graphics system", IEEE Transactions on Software Engineering, vol. 45.

S. Wilson, P. Markopoulos, J. Pycock & P. Johnson [1992], "Modelling Perspectives in User Interface Design", Proceedings of East-West Conference on Human-Computer Interaction, pp. 210-216.