

# **Bridging the Generation Gap: From Work Tasks to User Interface Designs**

*Stephanie Wilson and Peter Johnson*

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

*Phone: +44-171 975 52{31, 24} – Fax: +44-181 980 6533*

*E-mail: Peter.Johnson@dcs.qmw.ac.uk*

*WWW: <http://www.dcs.qmw.ac.uk/~pete>*

## **Abstract**

Task and model-based design techniques support the design of interactive systems by focusing on the use of integrated modelling notations to support design at various levels of abstraction. However, they are less concerned with examining the nature of the design activities that progress the design from one level of abstraction to another. This paper examines the distinctions between task and model-based approaches. Further, it discusses the role of design activities in such approaches, based on experience with one task-based technique, and the resulting implications for tool support and design guidelines. The discussion is contextualised by examples drawn from a number of case studies where designers applied a task-based approach to solve one particular design problem: that of developing an airline flight query and booking system.

## **Keywords**

Automatic generation, design guidelines, model-based design, task-based design, task models,

## **Introduction**

Current interest in task and model-based approaches to design signifies a trend towards placing greater emphasis on what an interactive system should do and how people might use it rather than how the system itself works. Designers are encouraged to conceptualise designs at a higher level of abstraction than is the case when working with standard prototyping tools; in particular, they are encouraged to focus on the behaviour and structure of the user interface rather than on specific details of low-level interaction objects. This interest is reflected in papers presented at the DSV-IS workshops [DSV-IS94, DSV-IS95].

Task and model-based approaches to design have many features in common. Most notably, they both focus on the use of models to represent the various sorts of in-

formation that contribute to the design of interactive systems. For example, there are models of users' tasks, domain objects and actions, user characteristics, dialogue and interface behaviour, style guidelines, etc. (see also [Puerta96]). The models are expressed using formal and/or semi-formal notations, and relations may be defined between the different models. Secondly, both approaches discuss issues pertaining to the use of the models in design activities (e.g., analysis, evaluation, generation, verification, etc.), some of which result in the creation of one model from another. Thirdly, tools of various sorts have been developed to support the design approaches and their modelling activities; some of these tools have aimed to automate the design activities, while others have aimed to assist or support designers in their work.

Broadly speaking, the task and model-based techniques are distinguished by their ability to model aspects of usage of proposed systems: model-based approaches tend not to model how a system might be used by users in accomplishing their work tasks. This distinction is reflected in the extent to which the approaches have focused on either the design process or the design support tools. We would suggest that, to date, task-based techniques have displayed greater interest in the former, while model-based approaches have been more concerned with the latter. Figure 1 compares the two approaches.

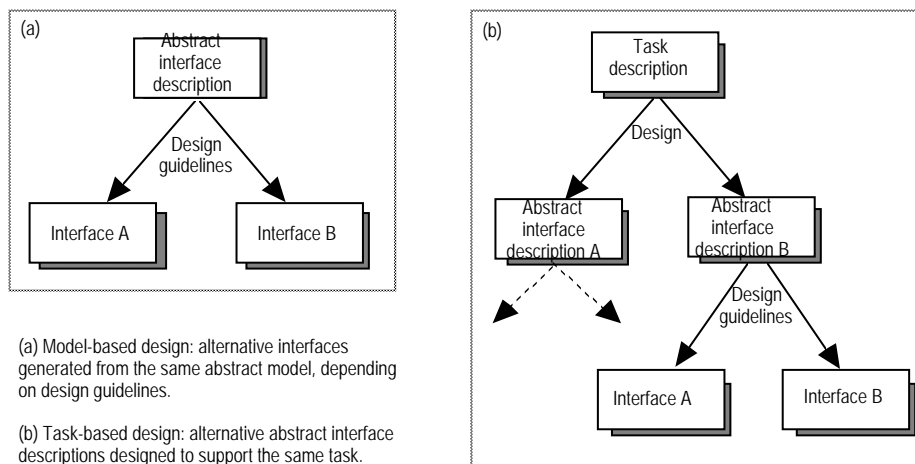


Figure 1. Comparing model-based design and task-based design

Model-based approaches such as UIDE [Foley91, Foley94], HUMANOID [Szekely-93] and MECANO [Puerta94b] were developed in an attempt to provide the designer with better facilities for constructing user interface software; they aim to improve interface design by changing the level of abstraction at which it is done and by improving or automating the tools with which it is done. These techniques incorporate models that allow the designer to express the proposed design at a high level of abstraction, focusing on the behaviour of the interface. Automatic tools then generate executable interfaces from these abstract models, usually under the guidance of other information such as style guides or user models. The abstract

model is in effect a design solution, albeit an abstract design solution. As such, these techniques limit their interest in the design process to those processes that occur in the transition between abstract and concrete (or executable) design solutions. Model-based techniques are not user centred *per se*; they support the designer more in the construction than in the design of usable systems, imposing no constraints on how the abstract design solution is produced.

Task-based techniques such as ADEPT [Wilson93] and MUSE [Lim94] aim to improve design primarily by improving the usability and suitability of the design for supporting the users' work. These techniques focus on the process of creating design solutions: they advocate developing design solutions from information about the users' tasks, thus increasing confidence that the system is compatible with the task it is intended to support. The tool support for task-based design has tended to be weak, focusing on either editor tools to support task modelling notations or lower-level generator tools similar to those of the model-based approaches.

The modelling structure in task-based approaches provides a context for interface design: it offers a framework within which designers can practice their craft. While much has been reported about the models in these approaches, about notations to describe them and ways of checking or proving properties about them, considerably less has been said about exactly how the models should be used to develop designs.

It is only at the level of graphical user interface design, the level addressed by model-based approaches, that a body of wisdom has been distilled from the collective experience of the HCI community over the last decade to guide the design process (e.g., [Smith86, Hix93, Vanderdonck95c]). This knowledge is most commonly expressed as design guidelines.

Other than this, little practical guidance has been offered to the design practitioner to assist in the application of these techniques, although some steps in this direction have been taken in the context of scenario-based design (see [Rosson95] for example.) This gives rise to questions such as what is it to develop a user interface from a task description? What design decisions are involved? What makes one design choice better or worse than another? What constitutes a good or bad interface to support a particular task? In the first instance, it raises the issue of developing practical guidelines to support the task-based design of interactive systems; in the longer term it raises the issue of developing task-based design principles.

This paper reflects on these issues in the context of our experience with one task-based approach to design, ADEPT, and discusses the wider implications of these experiences for tools to support task-based design. We examine firstly the activities involved in producing a task model from a number of task analyses; secondly, the design decisions that take place in moving from a model of existing work to envisioning the tasks that will be supported by a future system; and, thirdly, the design decisions that take place in moving from envisioned tasks to the design of a system to support those tasks.

The remainder of this paper is structured as follows. Section 1 provides some essential background information; it highlights the main features of a task-based approach to design and presents an overview of the ADEPT approach in view of these features. Sections 2, 3 and 4 discuss the creative processes that progress the design from one modelling activity to another in this design paradigm, and take a first step towards drawing out some guidelines to support these processes. The discussion is illustrated with examples taken from a number of case studies where groups of designers were asked to solve a particular design problem. In each case study the designers were asked to develop a system that would support the task of airline flight querying and booking.

This example task is particularly topical in view of the recent advent of on-line flight schedules and booking systems that are accessible by the general public. The designers applied the ADEPT approach using either pen and paper techniques or the prototype suite of tools developed to support the approach. Section 5 examines the implications for tool support and the last section concludes the paper with some reflections on the current state of the art and the future challenges for research and practice in this area.

## **1 Task-Based Design and ADEPT**

Task analysis is today accepted within the HCI community as making an important contribution to interactive system design practice. Although its inclusion in user-centred design approaches has been advocated for some time, it is only recently that we have seen methods which offer a tighter integration of the task analysis activities with subsequent design activities, thereby supporting greater use of task information in creating a design.

Task-based design emphasises the importance of designers developing an understanding of users' existing work tasks, the requirements for changing those tasks and the consequences that new designs may have for tasks. This places people and their tasks at the starting point of the design process, meaning that activities such as prototyping are no longer simply a matter of trial and error, where an initial design is gradually improved by a series of design iterations, but are informed from the outset by information about the tasks that the system is to support. This is particularly important in view of the fact that prototyping often fails because designers do not have the opportunity to iterate from the early prototypes due to time constraints and external pressures (e.g. from management or customers). Furthermore, the task descriptions can provide a focus for the generation of design ideas, helping to ensure that novel ideas are motivated by a user-task perspective.

Figure 2 summarises our view of what might be described as a minimal task-based design process, focusing on the models involved in the process. It starts with an analysis of the users' existing tasks, the results of which are expressed as the 'Existing task model'. It then progresses, via a process of design, to a description of the tasks it is proposed that the user will perform with the new system, known as the

'Envisioned task model'. The process concludes with the detailed design of an interactive system to support the envisioned tasks, termed the 'Interface model'. Clearly, this simplistic overview does not include evaluation activities, nor does it show the iterative nature of the design activities.

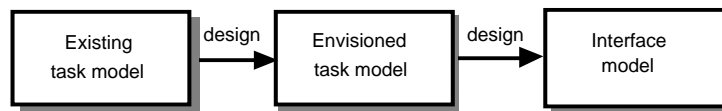


Figure 2. Overview of task-based design

Each of the components shown in figure 2 may be elaborated to reveal further models and processes. For example, all of the models and processes involved in a typical model-based approach such as UIDE would be encapsulated within the component labelled 'Interface model'. Likewise, a task model might consist of a description of the task goals and a separate description of task objects.

A further important point is that not all task-based design approaches make this clear distinction between existing and envisioned tasks. There are a number of different approaches to task-based design and only some of these take an analysis of existing tasks as a starting point. Others have no description of existing tasks but do have some form of description of the tasks to be performed with the system or of the methods involved in using the system.

A number of task-based design approaches have been reported which broadly conform to the overview in figure 2 (for example, [Lim94, de Haan94, de Bruin94, Bodart95a]), although they have set out with various aims. For example, to integrate human factors techniques with software engineering methods or to provide formal descriptions of user interfaces at various levels of abstraction with a view to verifying properties of the system. In our work on the ADEPT project [Wilson93, Johnson95], we set out to investigate how descriptions of users' tasks should influence and guide the design of systems to support those tasks, and to show how tool support might assist the designer in following such an approach.

An overview of task-based design in ADEPT is provided in figure 3 (again omitting details of evaluative or iterative design processes). We take a work-task to be a meaningful unit of work that a person undertakes in a given domain in the process of achieving their work goals. Hence, the approach starts with an analysis of the users' existing work tasks and continues with the development of a description of the envisioned tasks as an early design activity.

The envisioned task model is not a description of the methods for using the system, but a description of how work goals can be achieved for which, as yet, no system may have been designed. The existing task model forms part of the description of the problem space for the design, while the envisioned task model forms part of the proposed solution space for the design.

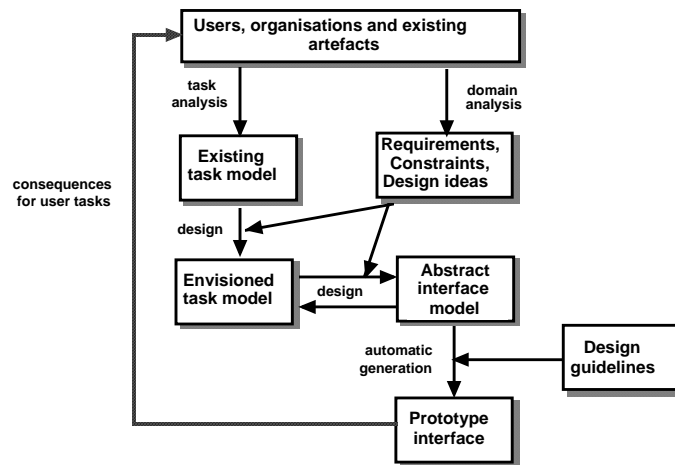


Figure 3. Task-based design in ADEPT

Other aspects of the problem domain are also captured and recorded in the form of requirements, design constraints and design ideas. These are in no sense finalised at the point where the analysis activities are completed, but can be supplemented, elaborated and modified as the design progresses. This additional information from the problem domain will influence the design choices that are made at each step in the process, including the design decisions made during the creation of the envisioned task model.

Having created an initial vision of the tasks that users will perform with the new system, the process continues with the development of an 'abstract interface model'. This is a high-level description of an interface to support the task, expressed in terms of abstract interaction objects, groupings of these objects and dialogue information.

Further design decisions at the level of the abstract interface model may have consequences for the envisioned task; these are reflected in the diagram by the backward arrow. The final stage in the process is the progression from abstract interface model to prototype interface — a low-level, executable form of the proposed design. Prototype design tools have been provided in ADEPT to support all stages of the design process, but only this final step is automated under the influence of a set of modifiable design guidelines. Once the prototype interface has been produced by the automatic generator tool, the designer may choose to modify it using an interface builder tool.

Again, this is a simplistic overview of the complexities of the design process and it would be naive to believe that design always proceeds in this orderly, top-down fashion. Design is not a simple top-down process, but frequently involves bottom-up activities as various studies have reported (e.g., [Hartson89]). Design modifications or decisions made at the level of the more concrete models during bottom-up design activity may have consequences for more abstract models.

ADEPT and other task-based design approaches offer a guiding structure for the design of interactive systems, the structure being provided by the series of explicit models to be produced at various points during the process.

However, as alluded to in the previous section, these approaches do not offer the designer guidance in how to progress the design from one modelling stage to another, except at the final stage of the process where existing user interface design guidelines are influential. While this has the advantage of not constraining or restricting how design is done, it has the disadvantage that designers are being asked to design within a new paradigm but yet are offered no practical guidance as to how this should be achieved. The following three sections of this paper examine this issue.

## **2 Analysing and Modelling Existing User Tasks**

In this section we consider how designers perform task analyses within the context of task-based design, and how the results of task analyses may be combined to produce a coherent model for use in subsequent design activities (summarised later in figure 6). The method of task analysis used in ADEPT follows our earlier work on task analysis [Johnson91a], and emphasises the importance of modelling how users perform tasks at present and their current knowledge of the domain and tasks.

There are a number of data collection, analysis and modelling techniques that may be employed in performing a task analysis. For example, data collection methods include direct observation of workers in the workplace, interviews, questionnaires, demonstrations and techniques that encourage workers to produce their own descriptions of their work. Some techniques are more or less suitable for use in different situations.

For example, direct observation may be difficult in hazardous or safety critical situations (since the presence of an observer may be hazardous to the observer or may increase the probability of the observed worker making a serious mistake), or in tasks that are highly cognitive in nature, for example in translating a document, where there may be very little directly observable behaviour. In contrast, direct observation of tasks involving much overt activity will provide a rich source of data. Detailed discussions of the various data collection techniques are given in [Johnson92a] and [Diaper89]. Some heuristics for selecting data collection techniques are given below:

1. Always use more than one data collection technique since any technique will only give partial information about a task.
2. Direct and indirect observation techniques are well suited for identifying patterns of behaviour, temporal aspects of tasks, behaviours and procedural aspects of tasks, but are poorly suited to predominantly cognitive tasks. The analyst needs to be aware that observations are time consuming, cannot be used in

isolation, and that interpretation of observations can involve a degree of inference on the part of the analyst.

3. Interviews provide a useful technique for identifying general rules, background knowledge, conditions and constraints upon tasks, the goal structure of a task and dependencies between tasks, but are poor at identifying temporal and procedural aspects of tasks. The analyst should be aware that people are better at remembering conditions given actions, rather than actions given conditions.
4. Questionnaires are best used to obtain shallow descriptions of task properties, and are useful to identify objects and attributes of a domain and their structural (class and component) properties, but are poor at providing detailed task information or information about context sensitive task behaviours.

Each of the techniques has more specific guidelines for their use in task analysis. One important point is that the analysis should focus on identifying characteristics of specific tasks rather than asking users to generalise across many tasks. The analysis should seek to identify all the variations and individual differences in each task as well as general characteristics across many tasks.

This is achieved by analysing many different users performing each task, resulting in a task description for each user on each task. These individual task descriptions carry all the individual differences regarding how users achieve a given goal. (The tasks are described in terms of the users' goals, sub-goals, procedures and actions, together with a description of the objects used in performing the actions. See [Johnson91b] for details.)

In our case studies, designers were asked to carry out an analysis for the task of querying and booking a flight. A number of subjects were used in the analysis and in each case data was collected about the last occasion the subject had booked a flight — a specific task. This highlighted many individual differences in the way the task was performed, all of which should be taken into account during the design process.

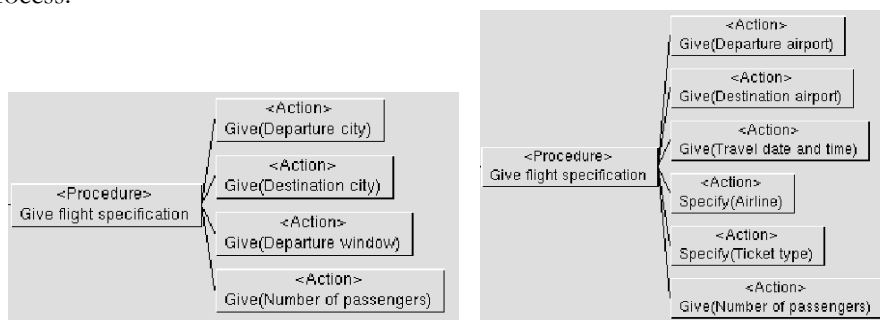


Figure 4. Alternative task descriptions for giving travel details

A trivial example of this is shown in figure 4 for one component of the overall task: giving details of the desired journey to a travel agent. In the first scenario the subject does not specify which airport they wish to fly from (presumably any local



airport is acceptable to this individual), and they specify that they wish to depart within a certain time interval (the 'Departure Window'). In the second scenario the subject names specific departure and destination airports, specifies a departure time, a preferred airline and ticket type.

In order to use the information in design, the analyst must produce a composite task model from each task description for the same goal. The composite task model should include all the different ways of performing the task. Developing the composite task model involves identifying all the alternate ways of achieving the same goal, resolving any conflicting descriptions (e.g., where the same course of action appears to lead to different sub-goal states) and identifying all the optional and compulsory aspects of a task (the optional ones will be indicated by a high degree of variance and low occurrence across each of the specific task descriptions, while the compulsory ones will be indicated by a low variance and high occurrence across each of the specific task descriptions).

In addition, in developing the composite task model the analyst should identify different objects used in the different specific tasks and any differences in the relevance of their attributes to the task. The analyst should also identify typical examples of any object where there are a number of different examples of the same object across the different task descriptions (for example, in booking an airline flight there may be many different examples of timetables, some may be atypical in that they exclude information on time differences, while others may be atypical in that they include information on in-flight meals for each journey).

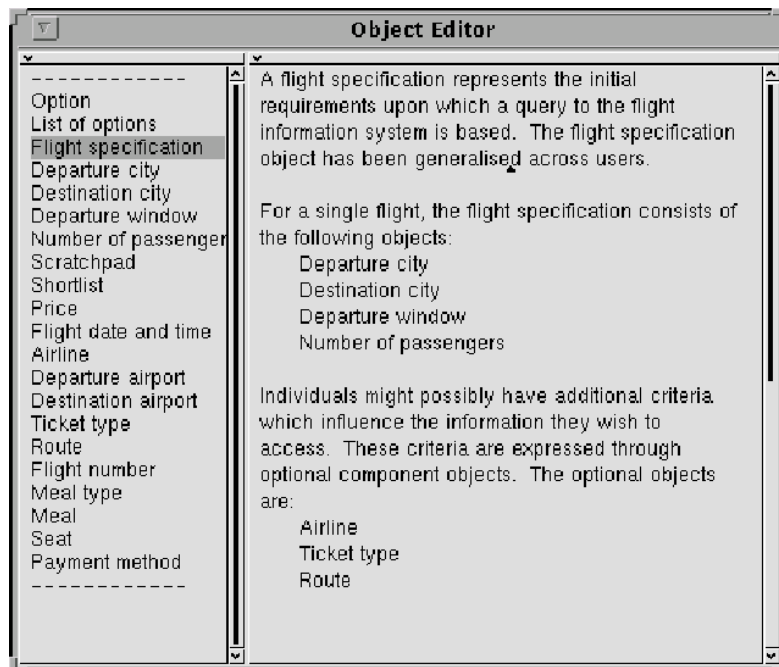


Figure 5. A composite object description

An example of a composite object description, from the ADEPT object browser, is given in figure 5 where the composite flight specification object consists of a number of compulsory and optional sub-objects.

Having produced a composite task model for all the relevant tasks in the domain of work, the analyst should now consider characteristics across tasks. This aspect of the analysis identifies commonalities of behaviours, common patterns of behaviours and common objects across the various tasks. In addition, this can identify constraints and dependencies across tasks. For example, in the domain of international travel, passenger information may be used by travel agents for invoicing the traveller and by the airline for advertising new products to potential customers. Similarly, patterns of behaviour such as the pattern by which the travel agent requests the traveller's destination and departure dates may correspond to the pattern that the agent must use to enter information into a flight enquiry system.

<b>The Development of Existing Task Models</b>	
Specific Task Models:	Identify characteristics of specific tasks in the first instance. Analyse many different users performing each task. Identify all variations and individual differences in tasks. Produce a task description for each user on each task.
Composite Task Models:	From each task description for the same goal, produce a composite task model which includes all the different ways of achieving the goal Identify all the different ways of achieving the same goal. Resolve conflicting descriptions (e.g. where the same course of action appears to lead to two or more different goals). Identify optional aspects of a task (i.e. where there is a high degree of variance and a low occurrence across the specific task models). Identify compulsory aspects of a task (i.e. where there is a low degree of variance and a high occurrence across the specific task models). Identify commonalities of behaviour, patterns of behaviour and common objects across the different tasks. Identify constraints and dependencies across tasks. Identify the different objects and typical instances of objects where there are a number of different examples of the same object across the different tasks.

Figure 6. Guidelines for developing extant task models.

### 3 Envisioning Future User Tasks

In a true task-based design approach, the first real design activities occur with the consideration of how existing work tasks may be changed or enhanced and the form that the work tasks will take in the future. In a general design situation the work might be changed in many ways, such as reorganising the structure of the workforce, rescheduling working patterns, relocating the work etc. However, in the context of interactive system design and human-computer interaction it is only those aspects of work that could be changed by the design and introduction of an interactive computer system that are the focus of concern. What we term the 'En-

envisioned task model' is a model of the anticipated nature of work which would come about as a result of designing an interactive computer system.

The envisioned task model is developed from the existing task model, the requirements and the overall design problem statement of the design situation. The overall design problem might be (as in our case studies) "to increase the quality and efficiency with which air-travellers can book their flights, without increasing the costs in terms of training time or number of staff required to carry out these tasks".

The requirements might include constraints on the possible design solutions, such as the system must be integrated with existing computer systems and must enable users to transfer between the existing and new systems with minimum retraining. Working with these requirements, problem statements and the existing task model, the designers must identify where they might introduce a new interactive system to improve the quality and efficiency of booking flights.

A first step in this process is to identify any tasks which could be either avoided or carried by a new system on behalf of the users (where this is seen as desirable). Additionally, the designers should identify any tasks that are not to be carried out by the new system and which therefore must be carried out by the users, and any tasks which will involve the users interacting with the new system. In doing this, they have begun to define the scope of the new system design and where it impacts the work domain.

Having defined a potential scope for the new system, a number of other considerations influence the development of the envisioned task model from the existing task model. These include identifying where sequences of activity can be made easier to perform, perhaps by removing unnecessary constraints between activities, making it possible to carry out activities in parallel or in an interleaved fashion where previously only sequential activities were possible.

For example, in seeking to improve the booking of flights, in the existing task model it is only possible to make enquiries of specific flights (i.e., of particular dates of travel and particular destinations), and this forces the user to make repeated queries whenever they want to know what flights might be available during a given 'window' of time for departure and return. One possible design solution would be to allow the user to make enquiries on more than one departure date and more than one return date within a single query. This would have the effect of replacing a series of actions with a new, more efficient action. Further design considerations centre around the objects that the user interacts with. One design option is to create new objects that compose or combine many individual objects. By creating such new objects the designer is attempting to make it possible to carry out actions on those objects which will be more effective, and to bring together into a single composite object those attributes of several objects that are all relevant to a particular aspect of the task. For example, in the domain of air travel the user of a flight booking system often needs to be able to retain a list of flight options that are available at given dates, times, prices and routings. This information is often

distributed around several information objects rather than held in a single object, making it difficult for the user to carry out actions that involve all the information. By creating a new object that brings together all the information into a single object, say the "option-list", it not only makes that information readily available to the users, it also makes it possible for them to perform actions directly on it, such as redefining the departure dates, or enquiring about the availability of all items on the option-list.

In developing the envisioned task model from the existing task model, the intention is to attempt to improve the work situation. One important aspect of work that must be considered is the safety and security issues. Often safety and security are embedded in the procedures of the work practice. For example, strict patterns of behaviour and sequences of actions are performed to ensure that an unsafe or insecure state does not occur. Since such embedding occurs it is possible to reinforce, and in some cases automate, the safety/security procedures in the new system.

<b>The Development of an Envisioned Task Model</b>	
Influences:	The envisioned task model is developed from: the requirements, the problem statement and the existing task model.
Scoping the design:	<ul style="list-style-type: none"> <li>Identify any tasks that can be avoided or that are unnecessary.</li> <li>Identify any tasks that can be carried out completely by the computer.</li> <li>Identify any tasks that can only be carried out by the user.</li> <li>Identify where users and the computer will need to interact to carry out a task.</li> </ul>
Improving the work:	<ul style="list-style-type: none"> <li>Identify where sequences of activity can be made easier to perform, e.g. by removing unnecessary constraints between activities, making it possible to interleave activities and/or carry out activities in parallel.</li> <li>Create more powerful objects by composing and combining individual objects, making it possible to carry out actions on those composed objects.</li> <li>Bring together information that is distributed across several objects but all required at the same point in a task.</li> <li>Ensure that safety and security procedures are supported.</li> </ul>

*Figure 7. Guidelines for developing envisioned task models*

However, it is also possible to make a previously safe/secure system become unsafe/insecure by changing the temporal dependencies between actions, or by changing the point in time at which particular information is displayed. It is therefore important to recognise that changes made to increase the efficiency of the work may inadvertently affect the quality of the work. Figure 7 summarises these guidelines for developing the envisioned task model.

These design deliberations lead to the development of an envisioned task model which provides a definition of where a new system is going to fit into the workplace, what tasks it will support, where users will interact with it, for what purposes they will interact with it and how it will improve or otherwise change the quality and efficiency of the work.

It does not specify how any interaction is to occur or how any information display will appear. It does provide the constraints that any design of interaction or display will have to meet: it provides the starting point for the development of the user interface.

## **4 Creating an Interface Design**

In creating an interface design to support a particular task, the question that arises is how should the envisioned task description inform the design of the interface? This progression from task to interface design will, in the first instance, be considered here as a single step, as might be the case when using paper-based tools, or when using a paper-based task model in combination with a rapid prototyping tool.

Later, in section 4.4, we will discuss how this progression is actually supported by existing task-based design tools. This activity starts once the designer has created a vision of what the users' future tasks might be and has validated this vision with users. Various factors then contribute to the development of an interface design, notably:

- Task descriptions (including task decomposition information, action and object descriptions, sequencing information).
- Requirements (including functional and usability requirements for the new system).
- Design ideas (which may be prompted by the task descriptions and the requirements).
- Design constraints (including hardware, software and organisational constraints that may render certain design options infeasible or too costly).
- Design guidelines (including layout rules, style guides, colour and typography guidelines, etc.).

The focus here is on the first of these. A multitude of different interface designs might be produced, each of which would vary in its fitness to support the users' tasks.

In a task-based design approach, task information is the primary determinant of the content, behaviour and structure of the user interface. Other factors such as requirements, design ideas and design constraints influence design choices.

The task models contain several different types of information which are used in different ways to guide the interface development: task decomposition information, action and object descriptions and sequencing information. These are discussed below and summarised in figure 10.

## 4.1 Decomposition Information

Task decomposition refers to the goal / subgoal structure identified initially in the task analysis and subsequently reflected in the envisioned task model. For example, figure 8 shows the top-level decomposition for the flight booking task. It involves the traveller making some initial decisions about their travel dates and then repeatedly getting travel options from agents and either booking a flight or perhaps refining their flight specification because there were no suitable options.

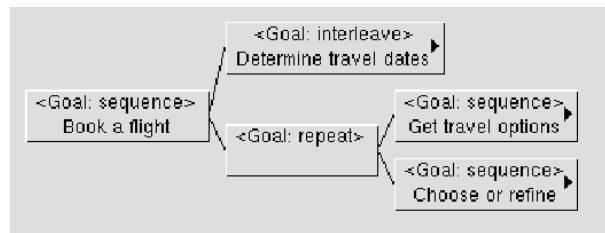


Figure 8. Top-level goals and sub-goals for the flight booking task

This decomposition of a high-level task goal into subgoals, and eventually into the procedures and actions that the user performs to achieve the goal, should be reflected in the overall structure of the interface. Essentially, the decomposition information should be reflected in the 'grouping' of components in the user interface, i.e. components of the interface that are intended to support closely related parts of the task should be grouped together. This grouping of components should be strongest at the lowest level of decomposition: the actions that the user performs to achieve some goal should be closely related.

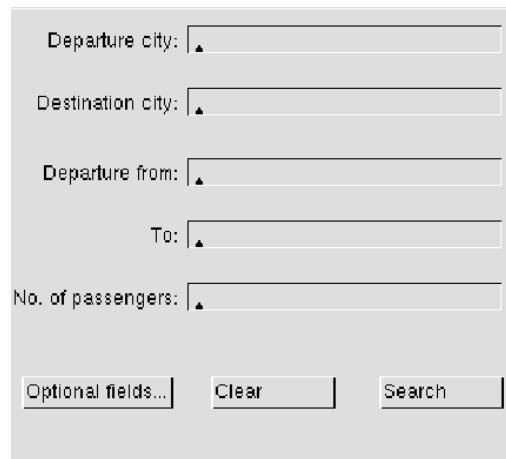
For example, figure 4 showed some specific models for the "Give flight specification" component of the task. An interface designed to support either of these scenarios should group together the interaction components intended to support the various actions that make up the flight specification task. Grouping interface components may mean placing them in close spatial proximity on the screen, or in close temporal proximity in the dialogue structure.

## 4.2 Action and Object Information

The task model includes information about the actions that users perform to achieve their goals and about the objects involved in the actions. This information also guides the development of the interface; in particular, it influences the components that will actually appear in the interface and the ways in which those components can be manipulated.

Broadly speaking, actions in the task model are indicative of commands that the user will issue to the system, while objects suggest the information to be manipulated by the commands or to be displayed on the screen. The action-object groupings therefore indicate information that can be manipulated in particular ways. In terms of choosing interface components, simple task objects and the actions ap-

plied to them can be supported by the sorts of widgets found in standard user interface toolkits. For example, the task of formulating an initial request to an on-line flight booking system involves a number of actions where information (represented as informational objects) is given to the system. Actions such as specifying preferred airports or dates of travel can be supported by simple widgets such as type-in text fields.



The image shows a flight booking form with the following fields and buttons:

- Departure city:
- Destination city:
- Departure from:
- To:
- No. of passengers:
- Optional fields...
- Clear
- Search

Figure 9. Examples of interface widgets to support simple task action-object groupings

Figure 9 gives a simple example of widgets that might be selected to support the actions and objects of the flight specification task. More complex task objects either require specialised widgets or can be supported by a group of standard widgets. For example, specialised widgets could allow the user to select departure and destination airports from a clickable world map or to select a preferred seat from an outline representation of the aircraft.

### 4.3 Sequencing Information

The final aspect of the task description that influences the development of the user interface is sequencing information. As can be seen in figure 8, the ADEPT task models include detailed information about the temporal ordering of task activities. If users perform their tasks in a certain order, clearly the systems designed to support the tasks should support the same task sequencing. In other words, the dialogue structure of the interactive system should be developed in line with the task sequencing information.

Our experience has suggested that while it is critical that the system should not violate the task sequencing constraints (i.e., it should not force the users to perform their tasks in a different order), it can relax the constraints in situations where safety conditions will not be violated, allowing users to perform tasks either in the sequence they are currently performed or allowing them to develop new strategies for achieving their task goals.

<b>Using Task Information to Guide the Development of Interface Designs</b>	
Task decomposition:	<p>Reflect the goal, sub-goal and action decomposition in the overall structure of the interface.</p> <p>Group interface components that support closely related parts of the task.</p> <p>Let the lowest level of task decomposition (i.e. the actions) be the strongest determinant of task structure.</p> <p>Group interface components by placing them in close spatial proximity on the screen, or in close temporal proximity in the dialogue structure.</p>
Task actions and objects:	<p>Use task actions and objects to determine the components that will actually appear in the interface and the ways in which those components can be manipulated.</p> <p>Use actions to suggest commands.</p> <p>Use objects to suggest information to be manipulated and/or displayed.</p> <p>Use action-object groupings to indicate information that can be manipulated in particular ways.</p> <p>Support simple objects, and the actions applied to them, by the sorts of widgets found in standard user interface toolkits.</p> <p>Support complex task objects by either specialised widgets or by a group of standard widgets.</p>
Sequencing:	<p>Let sequencing information in the task model be the major determinant of the dialogue structure of the interactive system.</p> <p>Do not violate task sequencing in the interface design.</p> <p>If desirable, relax sequencing constraints in situations where safety conditions will not be violated</p>

*Figure 10. Guidelines for developing user interfaces to support tasks*

#### **4.4 Tool Support for Interface Design**

Most task-based design approaches support the transition from envisioned task to interface design via a number of intermediate steps. As figure 3 showed, this is a two-stage process in the case of ADEPT: from envisioned task to abstract interface model and then from abstract interface model to executable prototype (see [Wilson93] for details). There are several motivations for this.

Firstly, a high-level description of the user interface, such as that provided by an abstract interface model, allows the designer to reason at a level of abstraction removed from implementation details, focusing on the behaviour of the interface rather than the interaction details.

Secondly, it facilitates taking account of existing user interface design guidelines. The use of task information discussed above primarily governs the transition from envisioned task to abstract interface model, while the further transition to implementation is governed by a different set of rules. In stark contrast to the paucity of information available to guide the transition from envisioned task to abstract interface model, there is a whole body of guidelines covering issues at the level of screen and dialogue design. Thirdly, it is easier to provide tool support for the process when it is decomposed into a number of sub-activities, each with its own con-



cerns and associated guidelines. It should be noted that this discussion has focused on the progression from envisioned task to prototype interface in the context of design; it has not been concerned with examining the nature of the relationship that exists between a final description of the envisioned task and the final interface. Others have specified this as a refinement relationship. However, it is not reasonable to suppose that the designer will formulate and express a complete design at the level of the envisioned task at the first attempt. Rather, we can expect that further design decisions may be made at the level of the interface description which have consequences for the users' tasks.

## **5 Implications for Tool Support**

As mentioned earlier, there are relatively few usable tools available at present to support task-based design. However, tool support is clearly an issue when designers are confronted with large scale design problems where it would be difficult, if not impossible, to manage the various models and their relationships on paper in a correct and consistent manner. Those tools that are available to support the earlier stages of the design process (task modelling and abstract interface modelling) tend to take the form of editors. Editor tools offer designers a high degree of flexibility.

They impose no restrictions on the set of task or abstract interface models that may be described, nor do they constrain or guide the designer in exploring design alternatives or in making the design decisions involved in progressing from one model to the next. In fact, their main contributions to task-based design support are to ensure that the task information is available in an integrated environment, to ease manipulation and management of the information and to ensure that the models are syntactically correct.

While there are relatively few guidelines relating to the actual activities of design in these approaches, there are rather more guidelines concerned with producing the final software system. In other words, there are guidelines that offer suggestions as to appropriate and inappropriate features of user interfaces. These guidelines can be applied at the transition from abstract interface model to executable interface, and cover many issues such as selection of interaction objects, layout, use of colour and platform-specific style guides. This is the stage of the design process that currently offers the greatest potential for automation, as is evident from the tool support provided for model-based design. Existing tools have taken advantage of these guidelines, although too much automation can come at the expense of insufficient flexibility.

This paper has reported some initial work on providing designers with practical guidance in adopting a task-based approach to design. We are hopeful that further research in this direction could result in the development of task-based design guidelines which, in turn, would offer a basis for enhanced tool support. In this context, we are talking about offering guidance and support to the designer rather than encoding rigid guidelines to which the designer must adhere or which would

be applied automatically. It is clearly premature even to consider automating these essentially creative design activities, otherwise we would unduly and inappropriately constrain the design activities. In the longer term, it remains an open question as to how far it will ever be appropriate to automate these activities: design is by its very nature a creative process and removing creativity from the process can only result in a lack of innovation and a deskilling of designers. However, we can assist designers by removing tedious and mundane jobs, and by providing appropriate support to facilitate their creative activities.

The discussion in this paper has intentionally focused on design activities, but evaluation activities are also important in task-based design. Guidelines can help in providing support for evaluation activities; it becomes feasible to assess where good practice guidelines have been followed and where the design deviates from the guidelines. For example, guidelines governing the transition from envisioned task to interface design embody some notion of what it is for an interface to support a task and could therefore provide the basis for an assessment of the task fit of the interface design.

## **Conclusion**

This paper has highlighted some important features of task and model-based approaches to design and has contrasted the two techniques. To date, there has been little evidence of the uptake of these techniques in design practice. This might be accounted for by a number of factors such as the immaturity of the techniques and the prototype status of the design support tools (where they exist at all).

Further, in the case of task-based design, we believe that it is unrealistic to expect designers to design within a modelling framework without offering practical guidance as to how design should be carried out in this context. These task and model-based techniques can only hope to move out of the research community when they begin to address issues beyond those of the form of the models they employ. This paper has offered some insight into the design activities that occur in a task-based approach to design, based on actual experience with such an approach. These results represent a tentative first step towards the development of task-based design guidelines; further work in this direction remains a challenge for the HCI design community.

## **Acknowledgements**

The ADEPT project was funded by DTI and SERC, grant no. IED 4/1/1573. Our current research is funded by the EPSRC, grant no. GR/K19211. We are grateful to the Amodeus project for providing the original idea for the design problem used in this paper and to the participants at our tutorials on task-based design for their inspiration and novel solutions to the design problem. Thanks also to the anonymous CADUP'96 reviewers for their detailed and helpful comments.

## References

- [Bodart95a] Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Provot, I., Sacré, B., Vanderdonckt, J., *Towards a Systematic Building of Software Architectures: the TRIDENT Methodological Guide*, in Proceedings of 2<sup>nd</sup> Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'95 (Château de Bonas, 7-9 June 1995), R. Bastide and Ph. Palanque (Eds.), Eurographics Series, Springer-Verlag, Vienna, 1995, pp. 262-278. <http://www.info.fundp.ac.be/cgi-bin/pubspec-paper?RP-95-019>
- [de Bruin94a] de Bruin, H., Bouwman, P., van den Bos, J., *A Task Oriented Methodology for the Development of Interactive Systems as used in DIGIS*, in Proceedings of the 15<sup>th</sup> Interdisciplinary Workshop on Informatics and Psychology, Interdisciplinary Approaches to System Analysis and Design (Schaerding, 1994).
- [de Haan94] de Haan, G., *An ETAG based approach to the design of user interfaces*, in Proceedings of the 15<sup>th</sup> Interdisciplinary Workshop on Informatics and Psychology, Interdisciplinary Approaches to System Analysis and Design (Schaerding, 1994).
- [Diaper89] Diaper, D., *Task observation for HCI*, in «Task Analysis for HCI», D. Diaper (Ed.), Ellis Horwood, Chichester, 1989.
- [DSV-IS94] Proceedings of 1<sup>st</sup> Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'94 (Bocca di Magra, 8-10 June 1994), F. Paternó (Ed.), Focus on Computer Graphics Series, Springer-Verlag, Berlin, 1995.
- [DSV-IS95] Proceedings of 2<sup>nd</sup> Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'95 (Château de Bonas, 7-9 June 1995), R. Bastide and Ph. Palanque (Eds.), Eurographics Series, Springer-Verlag, Vienna, 1995.
- [Foley91] Foley, J.D., Kim, W.C., Kovacevic, S., Murray, K., *UIDE - An Intelligent User Interface Design Environment*, in «Intelligent User Interfaces», J.W. Sullivan, S.W. Tyler (Eds.), Addison Wesley, ACM Press, 1991, pp. 339-384.
- [Foley94] Foley, J.D., *History, Results and Bibliography of the User Interface Design Environment (UIDE), an Early Model-based Systems for User Interface Design and Implementation*, in Proceedings of 1<sup>st</sup> Eurographics Workshop on Design, Specification, Verification of Interactive Systems DSV-IS'94 (Bocca di Magra, 8-10 June 1994), F. Paternó (Ed.), Focus on Computer Graphics Series, Springer-Verlag, Berlin, 1995, pp. 3-14.
- [Hartson89] Hartson, H.R., Hix, D., *Toward Empirically Derived Methodologies and Tools for Human-Computer Interface Development*, International Journal of Man-Machine Studies, Vol. 31, 1989, pp. 477-494.

- [Hix93] Hix, D., Hartson, H.D., *Developing User Interfaces - Ensuring Usability Through Product and Process*, John Wiley & Sons, New York, 1993.
- [Johnson91a] Johnson, H., Johnson, P., *Task Knowledge Structures: Psychological basis and integration into system design*, Acta Psychologica, Vol. 78, 1991, pp. 3-26. <ftp://ftp.dcs.qmw.ac.uk/publications/91-JohnsonH-1.ps.gz>
- [Johnson92a] Johnson, J.A., *Selectors: Going Beyond User Interface Widgets*, in Proceedings of the Conference on Human Factors in Computing Systems CHI'92 « Striking a balance » (Monterey, 3-7 May 1992), P. Bauersfeld, J. Bennett, G. Lynch (Eds.), ACM Press, New York, 1992, pp. 273-279.
- [Johnson95] Johnson, P., Johnson, H., Wilson, S., *Rapid Prototyping of User Interfaces Driven by Task Models*, in « Scenario-Based Design: Envisioning Work and Technology in System Development », J. Carroll (Ed.), John Wiley & Sons, London, 1995, pp. 209-246.
- [Lim94a] Lim, K.Y., Long, J., *The MUSE Method for Usability Engineering*, Cambridge University Press, Cambridge, 1994.
- [Puerta94b] Puerta, A.R., Eriksson, H., Gennari, J.H., Musen, M.A., *Beyond Data Models for Automated User Interface Generation*, in Proceedings of British Conference on Human-Computer Interaction HCI'94 « People and Computers IX » (Glasgow, 23-26 August 1994), G. Cockton, S.W. Draper, G.R.S. Weir (Eds.), Cambridge University Press, Cambridge, 1994, pp. 353-366. [http://www-ksl.stanford.edu/KSL\\_Abstracts/KSL-93-62.html](http://www-ksl.stanford.edu/KSL_Abstracts/KSL-93-62.html)
- [Puerta96a] Puerta, A.R., *The MECANO Project: Enabling User-Task Automation During Interface Development*, in Proceedings of AAAI'96 Spring Symposium on Acquisition, Learning & Demonstration: Automating Tasks for Users (Stanford, March 1996), AAAI Press, pp. 117-121.
- [Rosson95] Rosson, M.B., Carroll, J.M., *Integrating Task and Software Development for Object-Oriented Applications*, in Proceedings of the Conference on Human Factors in Computing Systems CHI'95 « Mosaic of Creativity » (Denver, 7-11 May 1995), I.R. Katz, R. Mack, L. Marks, M.B. Rosson, J. Nielsen (Eds.), ACM Press, New York, 1995, pp. 377-384.
- [Smith86] Smith, S.L., Mosier, J.N., *Design Guidelines for the User Interface Software*, Technical Report ESD-TR-86-278 (NTIS No. AD A177198), U.S. Air Force Electronic Systems Division, Hanscom Air Force Base, Massachusetts, 1986.
- [Szekely93] Szekely, P., Luo, P., Neches, R., *Beyond Interface Builders: Model-Based Interface Tools*, in Proceedings of the Conference on Human Factors in Computing Systems INTERCHI'93 « Bridges Between Worlds » (Amsterdam, 24-29 April 1993), S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, T. White (Eds.), ACM Press, New York, 1993, pp. 383-390. <http://www.isi.edu/isd/Interchi-be-yond.ps>

[Vanderdonckt95c] Vanderdonckt, J., *Tools for Working with Guidelines*, Tutorial #12 notes, 6<sup>th</sup> International Conference on Human-Computer Interaction HCI International'95 (Yokohama, 10 July 1995), 1995.

[Wilson93] Wilson, S., Johnson, P., Kelly, C., Cunningham, J., Markopoulos, P., *Beyond hacking: a model based approach to user interface design*, in Proceedings of British Conference on Human-Computer Interaction HCI'92 « People and Computers VIII », J.L. Alty, D. Diaper, S. Guest (Eds.), Cambridge University Press, Cambridge, 1993, pp. 217- 231. <ftp://ftp.dcs.qmw.ac.uk/publications/93-WilsonS-1.ps.gz>