Artificial Intelligence Review 7, 199–225, 1993.

Applying user modeling to human-computer interaction design

DAVID BENYON* AND DIANNE MURRAY+

* *Computing Department, Open University, Milton Keynes, MK7 6AA, UK*
\+ *Social and Computer Sciences Research Group, Dept. of Sociology, University of Surrey, Guildford, UK*

Abstract. Since the early 1980's, intelligent and adaptive systems have appeared and have been written about in a variety of guises. Although there are many differences between such systems as adaptive user interfaces, intelligent help systems and explanation systems, there are many similarities as well. The most notable of these is that they all require user models to be maintained and exploited as part of their designs. Furthermore, they share the same high level architecture. This paper discusses the use of models in human-computer interaction design and offers a common architecture for these adaptive systems. A methodology for the development of these systems is presented.

Key Words: adaptive systems, intelligent interfaces, user models, domain models, adaptive system architecture, development methodology

1. INTRODUCTION

Both experience and research tells us that computer systems can be difficult to learn and once learnt, may be easily forgotten. Even when practised users learn one system well, the software industry's predilection for development of new features means much interface functionality is continually changing through regular use. There is an additional problem of users changing their perception of and proficiency with different software systems. The range of skills, knowledge and preferences of system users means that any computer system which offers a fixed interface will be better suited to some users than to others.

One of the most common ways to discriminate between users is to focus on their experience and frequency of computer use. Carroll (1991) argues that an effective way of dealing with system complexity for the novice user is to provide a functionally simple system. Intermittent or discretionary computer users have to master different application packages as the need arises, and seldom have any real choice in the purchase, selection or use conditions of the software which their management decides upon. Discretionary users are a particularly important class as they often have to be encouraged to make use of a system and will generally benefit from easy-to-use, intuitive and functionally simple interfaces. Another familiar class of user is the committed or so-called 'expert' user who may choose to employ different packages for different activities.

The difficulties involved in designing an interface which will deal effectively with individual preferences and experience, whilst minimising user frustration, learning effort and transfer errors is widely recognised as a stubborn HCI problem. However, many systems could potentially have a range of different features to match the diversity of user populations. Moreover, it is possible to provide these different features automatically by exploiting artificial intelligence and user modelling techniques to build intelligent or adaptive capabilities into the interface. Systems known as 'adaptive systems' or 'intelligent (user) interfaces' seek to do just this.

In this paper, we describe how user models can be exploited in order to enhance human-computer interaction. We present an overview of the types of system which have exploited a user modelling approach, provide a unifying architecture for these systems and outline a methodology for adaptive system development. Before this, however, we briefly look at the potential problems of adaptive interfaces.

2. ARE ADAPTIVE INTERFACES DESIRABLE?

There are, of course, many systems which cater for different physical environments and tasks and which attempt to match individual requirements in a number of ways. The simplest approach is to build completely different systems for different markets. An example is the variety of word processors which are used by populations so diverse such as school-children, business professionals, authors and typesetters.

Another design solution is to allow individualisation by means of 'customising' facilities to amend and shape major parts of system behaviour and responses. Users can define personal command sets, change or determine interface characteristics and write their own 'macros'. These facilities are present in many of the computer systems we now use but the drawback is that, in order to adequately exploit customisation, the user must learn yet another set of activities and functions which imposes additional mental load or which may interfere with the completion of other tasks. The user, or a surrogate, must become a quasi-expert in operations ranging from manipulation of the screen display (e.g. shaping windows, selection of colour and background patterns) to running saved profiles which initiate start-up processes to the coding of function keys and macros. Some routines will only have to be set once and may be intuitively obvious or be in keeping with the 'look and feel' of a particular interface, but others may involve learning a specialised command set or finding out how a specific terminal or piece of equipment operates at a low physical level. The user may have to master complex syntactical and semantic constructs in order to make use of the system whilst the specification of macros is akin to a programming task, forcing users to engage in a specialist activity for which they may have little or no training.

Adaptive user interfaces relieve users of the burden of personalising the system and so may overcome the limitations of customisation as a design

solution, but they present other potential problems. We have been conditioned by habit and perceived good human-factors practice to expect to meet computer interfaces which are static and (more-or-less) predictable. We fear that a system which suddenly or autonomously changes is going to be confusing, unsettling and inconsistent. Consistency is an important goal of HCI research and adaptive systems appear to go against this maxim.

The problem of 'hunting' has been raised with respect to adaptive systems (Innocent 1982, Edmonds 1987). This can happen when the response which a fully adaptive system makes to its user is amended to meet the changed circumstances or interpreted needs of that individual user, just at the time when the user has adapted his or her response patterns or actions to that previously demanded by the interface. The user then has to learn a new interface. In the case of adaptive systems, the pertinent problem is one of intervention at an appropriate point, identifying the natural and suitable breaks in an interaction at which adaptation can be effective and efficient with respect to the tracking of user task needs and the interpretation of user actions. A major problem to be resolved is that of whether adaptation should be controlled solely by the system, or whether it should be a co-operative process, achieved by mutual recognition of the need and a shared agreement on implementation.

There is a significant cost associated with the implementation of adaptive systems, both in the early design stages and in the subsequent coding, and the question naturally arises as to whether this cost can be justified. The answer is likely to be 'yes' if adaptive systems significantly improve usability and the quality of interaction. Furthermore, the inclusion of an adaptive capability may not be such a large overhead if it arises as a natural consequence of better attention and metrics being applied to interactive system design. One of the objectives of the work reported here has been to develop cost-effective software support for adaptive systems development.

Another argument against the implementation of adaptive systems is that of 'feasibility'. Not only has it been asked whether systems can actually incorporate enough suitable knowledge about an individual user in order to make adaptive responses to that person a viable proposition, but the basis upon which user characteristics can be inferred has been questioned (Thimbleby 1990, Pullinger 1989).

Interaction with computer systems can be a frustrating, time-consuming and ultimately unrewarding experience with many user misunderstandings and difficulties because of inflexibility of system design. On the other hand, human-human interaction shows excellent reactive, adaptive and co-operative capabilities with an ability to recognise limits and provide 'satisficing' responses. As computer systems evolve in complexity to become self-determining agents and dialogue partners, aspects of human-computer interaction will require a similar degree of flexibility and accommodation to the external world and to the needs of the other dialogue partner. Adaptive interfaces and other adaptive systems offer the potential to deal with these issues if they can overcome the problems mentioned.

Adaptive systems are systems which can automatically alter aspects of their

functionality and/or interface in order to accommodate the differing needs of individuals or groups of users and the changing needs of users over time (Benyon, Innocent and Murray 1987). It would seem that some systems have to be adaptive if they are to serve their purpose. For example, an explanation system or an advice giving system has to provide explanations and advice appropriate to the user's individual circumstances and the task. At the other end of the spectrum, particularly simple systems do not have to be adaptive and indeed there may be distinct advantages from having non-adaptive systems. In the middle, however, lies a host of systems which may benefit from an adaptive capability, depending on their intended purpose and the envisaged user groups.

It is not the argument presented here that systems should adapt to everyone. We may decide that systems should not be expected to adapt to complete novices. People should invest some effort in understanding the system if only for the sake of safety and security. Similarly we may take the decision that systems should not automatically adapt to the individual nuances of behaviour exhibited by expert users. Experts can be legitimately expected to learn a customising system so that they can tailor the system to their own needs. Once again we have a host of intermediate users for whom the interaction is difficult enough without imposing a tangential load of a customising system and who want to use the system but perhaps only infrequently and intermittently. Moreover it is this large group of users whom we want to encourage to use technology and who are put off by its complexities and inflexibility.

3. ADAPTIVE USER INTERFACES

Since the early 1980's, intelligent and adaptive systems have appeared and have been written about in a variety of guises. One early workshop on the topic of 'Intelligent Front Ends' (Bundy 1983) brought together some 24 research projects in the area and demonstrated just how wide ranging and — as subsequently became apparent — how naive many of the assumptions were about the viability of such constructions. One of the first fully-detailed descriptions of the requirements of an adaptive interface was provided by Innocent (Innocent 1982) who specified the need for a self-adaptive user interface (SAUI). Building on the idea of a SAUI, Benyon developed the design for an adaptive system entitled MONITOR (Benyon 1984). A prototype system was implemented which included an explicit representation of individual users and an explicit representation of the domain. This was qualitatively different from other 'adaptive' systems at the time which provided adaptation according to various mechanical or statistical means (e.g., Furnas 1985, Greenberg and Witten 1985).

The essential difference is that the latter systems held a very limited and *implicit* model of the user. This took the form of a record of command usage or data access and provided an automatic response to the frequency of such usage. For example, the inference was that the more a user used a command the more

they would be likely to use it in the future and the higher up the preferred command list it would appear (Greenberg and Witten 1985). There was no attempt to infer or represent any other information about the user, nor to maintain a long-term representation of user characteristics. The model which was employed, moreover, remained implicit rather than explicit. Although these approaches can be extremely effective, particularly when coupled with a programming by example approach to inferring user intentions and a number of successful systems have been developed (Greenberg, Darragh, Maulsby and Witten 1991), they are necessarily limited because the user model is localised to the application and unavailable for scrutiny.

The adaptive system concept could also be seen in the domain of intelligent tutoring (see Elsom-Cook, this volume) and in the provisions of help and other user support. With the application of computer as coach, assistant or teacher, one use identified for intelligent systems is in the provision of context-dependent 'active' help, fashioned to particular stuations and user difficulties or experiences (Fischer, Lemke and Schwab 1986, Hansen, Holgaard and Smith 1988). On-line help systems track the user's context and incorporate assistant strategies and a set of action plans in order to intervene when most appropriate or when the user appears to be having difficulty. They are a highly relevant and active area for research into adaptive interface systems.

Such systems must have a model or a set of domain specific knowledge in addition to a general 'recogniser' and planner. They have many commonalities with ITS since a diagnostic strategy is required to provide the most appropriate help for that user in that particular situation, with that individual history and error-making behaviour and misconceptions. These types of intelligent system have been categorised as 'intelligent support systems' (Fischer *et al.* 1986, Fischer, Morch and McCall 1989, Fischer 1989). Similar work on the Unix Consultant system (UC) (Wilensky, Arens and Chin 1984, Chin 1986, 1989) and on other interventionist interfaces to Unix systems (Quilici, Dyer and Flowers 1986) illustrated the need for a full understanding of user dialogue and for a model of the user in order to effectively provide a directed and context-sensitive help facility. These systems have the aim of providing assistance to natural language queries on how to best meet the user's stated goals, whether such users be experienced or novices. Related work is in the provision of user support in accessing and customising the Unix manual system (Mason 1986, Jerrams-Smith 1985).

Intelligent help has further developed into 'critiquing systems' (Fischer 1989), where users must be competent in the subject domain being critiqued, rather than being tutees or learners. A critic system provides learning on demand and needs into infer the user's context and world view, goals and plans. ITS and help systems can be placed on a dimension from 'active learning' and exploration through to pure information dissemination but critiquing systems occupy the middle ground because the information imparted to the user is requested and structured, rather than being open-ended. The critiquing systems which have been built so far have been largely on powerful workstations in very

specific domains. One area is critical analysis of programming style (Moore and Swartout 1988) such as LISP-Critic (Fischer 1987). Another is in design environments such as CRACK (Fischer 1989).

A similar but slightly different strand has become apparent in recent research into the provision of an explanatory facility for the behaviour of the system. Applications can be seen in describing complex devices such as telephones (Paris 1989), educational diagnosis (Cohen and Jones 1989) and general advice such as facial skin care (Saratinos and Johnson 1991). The need for adaptivity and a representation of the user (or receiver) of the explanation has been recognised (Kass and Finin 1989, Carroll and McKendree 1988) as being central to the provision of good explanations.

Chignell and Hancock (1988) see an intelligent interface as an intermediary, the function of which is to encode and translate information between two interacting agents (for example, a human and a computer) who possess an incomplete understanding of the other's knowledge. They argue that intelligent interfaces are needed when there is an excessive semantic distance between the user's and the machine's language for 'sufficiently' complex tasks, as training surrogates or where the task can be shifted between human and machine. The role played by a human search intermediary in information retrieval tasks in such an example. Branjik, Guida and Tasso (1990) describe a similar application with their IR-NL system which helps and advises users of bibliographic databases to obtain a sensible set of search criteria. A slightly different approach, but one which again emphasises the need for an explicit user model is the Adaptive Intelligent Dialogues (AID) project (Browne, Totterdell and Norman 1990).

Recent interest in computer supported co-operative work (CSCW) and in intelligent agents represent a further development of the adaptive system concept. Co-operative systems require models of all the systems and humans participating in the interaction (Seel 1990). More complex systems are 'agent' based — capable of voluntary, rational action carried out in order to achieve goals and holding a representation or 'belief' in the state of the world. They come to hold these beliefs through existing data and by deriving new belief from interaction with external sources and as a result of internal reasoning. Intelligent computer systems are deemed to be agents capable of conversing and discoursing about the events and beliefs of the world which they share with their users. Kiss (Kiss 1988) argues that intelligent dialogue agents are autonomous, co-operative (but permitting conflict), continuously active and reflexive in that they are capable of self-understanding and self-modification. Multiple agent systems incorporating surrogates to filter routine tasks and provide personalised 'theatrical metaphor' interfaces are cited as the wave of the future (Negroponte 1989) and simple agent-based interaction is a prerequisite of co-operative dialogue, between human or computer as in the interface agents which provide 'expertise, skill and labour' described by Laurel (Laurel 1990). Agents are adaptive systems, but systems which are specialised and know about only a very small part of the world.

4. AN ARCHITECTURE FOR ADAPTIVE SYSTEMS

The above survey illustrates how, in the last ten years, many researchers from different backgrounds have recognized the potential which user models have in improving aspects of the interaction. Although the system categories in which user modelling and adaptivity have been deployed are quite diverse and work has taken place from different disciplines, with the researchers employing different techniques and specialised language to explain their concepts, they seem to share an underlying architecture. In this section we present a description of the components which all adaptive systems must have. Clearly the complexity of the system and the requirements of the application have an impact on the detail contained in each component. The advantage of developing a generalised architecture for adaptive systems is that it enables researchers to talk the same language, to compare different systems and to develop appropriate representation techniques.

The systems with which we are concerned are adaptive systems in that they automatically alter aspects of the system to suit the requirements of individual or groups of user — or more generally to suit the needs of other agents in the system. All have to infer characteristics of the other agent from the interaction. All utilize models of individual or groups of user. Work on active help, intelligent tutoring, natural language, explanation-based and co-operative agent systems contribute to the confusion because they describe systems from distinct perspectives. We need to consider how adaptive systems are designed and how the adaptive functionality can be incorporated into existing architectures. A conceptual structure for adaptive systems is shown in Fig. 1. It consists of three models; the user model, the domain model and the interaction model each of which is explained in detail in the following sections.

4.1 *The user model*

A user model has to be an integral part of any adaptive system. The term as used here means a representation of the knowledge and preferences which the system 'believes' that a user (which may be an individual, a group of people or non-human agents) possesses. Wahlster and Kobsa (1989) stress that a user model is a knowledge source which is separable by the system from the rest of its knowledge and contains explicit assumptions about the user. Finin (1989) argues that a user model is 'knowledge about the user, either explicitly or implicitly encoded, which is used by the system to improve the interaction.' (p. 412)

A user model is different from both the actual knowledge possessed by a user and knowledge employed by system designers. System users have some knowledge of the system which is represented in their mental models of the system — the user's model (Norman 1986). System designers form mental models of users (the designers' models) and use these to guide their designs. Every system thus embodies an implicit model of the user (which may be as

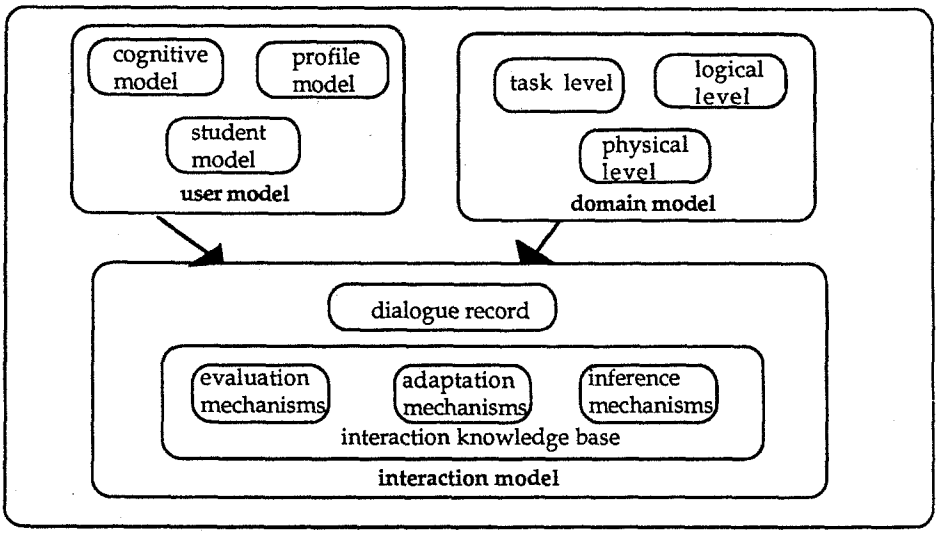

Fig. 1. Overall architecture for an adaptive system.

basic as the assumption made about the physical capabilities of users, such as being able to type or use a mouse). This data though is not accessible to the system and cannot be considered a user model. We must also distinguish between the data which a system may have about a user (such as the data stored in a personnel database) and a user model. Personal data of which the system is unaware and which it cannot exploit in its interactions does not constitute a user model (Wahlster 1989).

The user model component of an adaptive system is a representation of the characteristics of the users, or more generally the agents, with which the system can interact. The user model is used to provide adaptivity either by intervention or by co-operative agreement with a user. User models may be highly pragmatic in that they represent only what is required in order to facilitate the required adaptation. Other user models may be oriented towards a realistic description of the user. Murray (1987) refers to these as 'Embedded User Models' (EUMs).

For example, we expect some adaptive interfaces will have to deal with fundamental cognitive characteristics such as a user's preferences for particular styles of display, basic cognitive capabilities such as spatial ability and preferred learning styles (van der Veer 1990). In these systems long-term, psychologically valid models are vital. Cognitive models are also important to intelligent support systems and explanation systems where the requirements of the system demand that help or explanations take into account users' existing knowledge and the intention which they have in asking for advice.

Knowledge for the user model can be acquired implicitly by making inferences about users from their interaction, by carrying out some form of test, or from assigning users to generic user categories usually called 'stereotypes' (Rich 1983, 1989). Other mechanisms may be employed to infer user characteristics.

In terms of our adaptive system architecture, it is the interaction model which is primarily responsible for inferring knowledge about users. Explicit acquisition may be achieved through some co-operative behaviour such as asking relevant questions. Alternatively, the user model may be fed with previously stored information, held on whatever medium may be appropriate or transportable. The use of user records, profiles or scores in a personal data store such as a 'smart' card is one mechanism to allow full-scale use of adaptive systems with explicitly acquired data (Murray 1989).

User models may contain one or more of three types of knowledge of the user. Firstly, the user model may hold data about what the system believes the user believes about the domain. Because of the similarity of this data to that held by intelligent tutoring systems, we refer to this portion of the user model as the *student model.* The student model contains knowledge which the system believes that a user has about the domain within which the adaptive system is to function. Student model data is domain dependent data and may be kept at one or more of three levels: the task level, the logical level and the physical level.

The *task level* describes user goals in the domain. For example, an intelligent support system may need to infer that a user is trying to display the contents of a directory and not list the content of a file in response to a query such as 'how do I use *ls* to display my directory'. A second level of description of user knowledge of the domain is the *logical level.* Here the system records what it believes the user understands about the logical functioning and the logical concepts embodied by the domain. For example, a user having trouble reformatting a paragraph in a word processor may have misunderstood the logical concept of a paragraph rather than misconstruing the effect of a particular system function. Finally the system records the user's (inferred) knowledge at the *physical level.* It may be that anaphoric references (for example) can be effectively inferred by using some heuristic concerning the most recent use of a referent.

There is a strong justification for these three levels of description as will become clear in Section 4.2 and much of the effectiveness of an adaptive system depends on how comprehensively the levels, and the mappings between the levels, are represented in the domain model. However, it is unlikely that any system will be able consistently to infer the level appropriate for the user without some form of co-operative metacommunication. At each of these levels the user model should record the user knowledge and the user's erroneous beliefs in the form of an 'overlay' or 'perturbation' model.

What separates the domain specific knowledge of the student model from the other components of the user model is simply that the student model represents users' beliefs about the domain in which the adaptive system is effective. Domain *in*dependent data may be considered either as fundamental psychological data or as profile data. Psychological data is concerned with essential cognitive and affective traits of users and is held in the *psychological model* component of the user model. There is an ever increasing body of experimental evidence which confirms that users differ in cognitive skills and personality traits which significantly affect the quality of certain interaction styles and user

requirements (van der Veer 1990, Egan 1988, Jennings and Benyon in press). We believe that this data is sufficiently different from other individual differences that it deserves to be separated into a recognisable component of the user model. Moreover, these characteristics of users are particularly resistant to change and hence represent user attributes which are particularly important for adaptive systems. If users find it difficult or impossible to change aspects of their make-up, these are exactly the characteristics to which the system should adapt. Our own research in this area (Benyon and Murray 1988, Benyon, Murray and Milan 1987, Benyon, Innocent and Murray 1987, Murray 1988, Jennings, Benyon and Murray 1991, Jennings and Benyon in press) has been looking at a number of cognitive style attributes and adaptation to spatial ability appears feasible and appropriate in some domains. Spatial ability is a characteristic which has been identified by others (Vicente and Williges 1988, Vicente, Hayes and Williges 1987, Egan 1988) as affecting the quality of interaction, particularly where users have to navigate through the conceptual space of file structures or system modes.

Data concerning the background, interests and general knowledge of users is held in a *user profile* component of the user model. This data is not psychological in nature, but may interact with cognitive characteristics in a number of ways. For example, users with poor spatial ability may be able to deal effectively with an interface style if they have a certain level of experience using that style (Jennings and Benyon in press). Knowledge of generic applications is stored in the user profile as is much of the stereotype-inherited data such as being a business traveller (Morik 1989) or feminist (Rich 1983).

As an additional consideration, it is worth noting that there are many moral and legal problems associated with such types of user model and these can only become more acute as their accuracy improves (Pullinger 1989, Rivers 1989). There is the issue of how open and available the model should be to the user whom it represents and the amount of influence that a user can have in viewing and changing the representation. We do not pursue these issues in this paper but mention them to show that there are fundamental moral issues at stake when developing and designing interactive systems with a degree of intelligence and autonomy and that designers would do well to be aware of and to be wary of the potential for harm and misuse (Kobsa 1990).

4.2 *The domain model*

The domain model defines the aspects of the application which can be adapted or which are otherwise required for the operation of the adaptive system. Other terms which have been used for this concept include application model, system model, device model and task model. It is 'owned' by, and embedded in the adaptive system. The object of the domain model is the application which is to have the adaptive capability. For example, the domain model of UC is a representative of certain aspects of Unix.

The domain model will serve a number of purposes. Firstly, it forms the basis

of all the inferences and predictions which can be made from the user-system interaction. It is important therefore that the model is at an appropriate level of abstraction to allow the required inferences to be made. Secondly, the domain model defines the aspects of the system which are adaptive by describing alternative representations of domain features. Thirdly, the domain model holds the characteristics of the application which are measurable, so that they can be evaluated for effectiveness against the required criteria. The final use of the domain model is to form the basis of the student model component of the user model. The system needs to record what it 'believes' the user believes about certain aspects of the application. The domain model must describe the system so that it can store data about the user's understanding of the various concepts and functions in the application.

The domain model consists of one or more abstractions of the system. It is a description of the application which contains facts about the domain, i.e. the objects, their attributes and the relationship between objects. The domain model is the designer's definition of the aspects of the application relevant to the needs of the adaptive system. A central question in constructing a domain model is deciding what level of description should be represented. For example, the domain model may represent the fact that in Unix the command *rm* is used to remove files as

- to remove a file called foo type 'rm foo'
- rm is a command which removes files
- If you want to make more space for new files, you can use rm.

These are all quite different levels of abstraction. The first gives a physical, 'how-to-do-it' description. The second provides a logical perspective. The third describes the purpose of *rm*.

This three-level domain model reflects the levels of description found elsewhere in computer science. Issues of representations of systems and the separability of these descriptions are central to the development of user interface management systems (UIMS) and related software such as user interface design environments (UIDE). Although the concerns of UIMS are quite different from those here (UIMS are concerned with how the interaction can be managed so that it may allow for multi-threaded dialogues whereas we are concerned with finding a suitable abstraction of the system to enable adaptivity), it is important to understand the relationship between the two.

The 'Seeheim' model (Green 1985) of an interface processor is still the foundation of most discussion of UIMS. This model describes the interface in terms of three components; an application model (or application interface) which describes the application, a dialogue controller which deals with the processing of input and output and a presentation manager which is concerned with the actual displays. Cockton's approach to defining UIMS (Cockton 1987) includes reference to a conceptual model of the application. This provides the basis from which both the semantic, functional description (the non-interactive core of the application) and the user interface (UI) can be derived. A linking mechanism is introduced to communicate between these components. The UI

corresponds to the dialogue and presentation components of the Seeheim model. Myers (Myers 1989) also recommends merging the dialogue and presentation layers of the Seeheim model.

Coutaz (Coutaz 1987) takes a similar approach introducing the notion of multiple, hierarchically arranged dialogue agents. These dialogue agents react to events. Her model of an interactive system then consists of a presentation component, an abstraction component and a control which links the other two together and communicates with other agents. The top level of the hierarchy describes the application and the presentation. Lower levels are implementations of specific aspects of the system.

The separation of presentation from abstraction is a theme which runs through most of the articles in the IEEE Software issue on Interface Systems (IEEE Software 1989). It is illustrated in Fig. 2. Foley *et al.*'s (Foley, Kim, Kovacevic and Murray 1989) description of a UIDE makes clear their preference for an explicit conceptual design. Hurley and Silbert's 'Interaction model' (Hurley and Silbert 1989) describes the interface between the UI and the Application. The two levels of description are apparent as is the need for an underlying conceptual model of the whole system. Fischer (Fischer 1989) identifies the need for a representation of the problem domain and the communication proceses.

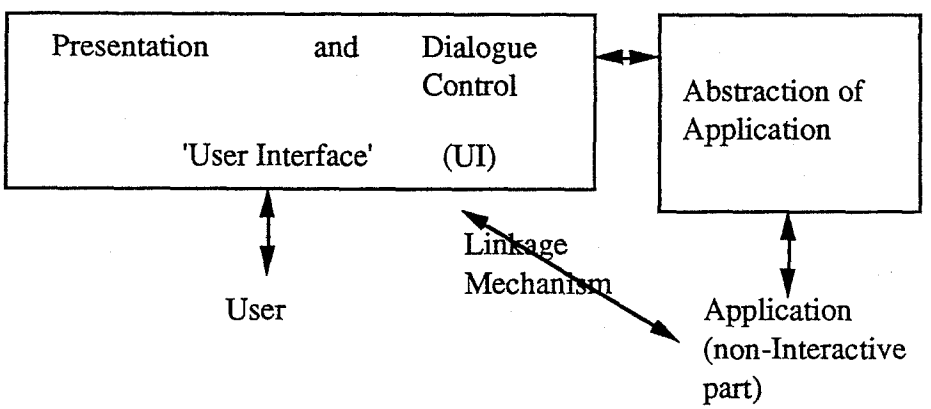

Fig. 2. General view of levels of description in user interface software.

The literature on software for user interface development gives us one view of how to model computer applications and the separation of the presentation level (concerning the graphics used, the windowing mechanisms, fonts, dialogue style, screen layouts and so on) from an abstraction of the functionality of the application appears central. These two components are linked through some mechanism which maps aspects of the presentation onto functionality and vice-versa. However, an important difference between the purpose of our domain model and the representations arising from UIMS research is that we seek an abstraction of the application at various levels in order to facilitate adaptations

to people. There should, therefore, be some correspondence between the domain model and human cognitive structures.

One of the areas to consider a realistic cognitive representative of computer systems is the work on Task Analysis techniques (Wilson, Barnard, Green and Maclean 1988, Diaper 1988, Payne and Green 1989, Bösser 1987). These are formalisms which attempt to represent some aspect of the application. For example Cognitive Task Analysis (Barnard 1987) is concerned with the nature of mental activity required to perform a task. Other techniques focus on the prediction of difficulties from interface specifications (ETIT Moran 1983), a measure of complexity in order to provide necessary education, such as Task Action Grammar, (TAG, Green, Schiele and Payne 1988, Payne and Green 1989), or cognitive complexity theory (Kieras and Polson 1985), performance e.g. GOMS (Card, Moran and Newell 1983), learnability (TAG), or the knowledge actually possessed by users (Wilson, Barnard and Maclean 1985, Young and Hall 1982, Johnson 1989).

The model of human-computer interaction which underlies task analysis (Benyon 1992) recognizes that the user has something which s/he wishes to achieve outside the computer system (e.g. produce a letter). This goal is then translated into a number of tasks which have to be performed using a specified device such as a particular word processor (e.g. start the word processor, edit the letter, print it). Tasks are decomposed into a hierarchy of simpler subtasks. These can be further decomposed through several levels until some 'simple task' (Green and Payne 1989) or 'unit task' (Card *et al.* 1983) is reached. These simple tasks are those which can be characterised by having no problem solving or control structure component. Simple tasks may be different for experts and novices.

This model reflects the abstraction/presentation dichotomy of the UIMS approach, but also includes the *goal* or *external task* level of description. Goals are concerned with what the system can be used for. A central concern of HCI is how goals can be mapped onto the functions which are available in the application. These three levels can be seen in Rasmussen's consideration of mental models and HCI (Rasmussen 1986, 1987) and are also apparent in the philosophical arguments of Pylyshyn (Pylyshyn 1984) and Dennett (Dennett 1989) and in the work of Newell (Newell 1982).

Thus, bringing together these descriptions, we may consider that a cognitively valid domain model should capture descriptions of the application at three levels which we shall refer to as the

- Task level
- Logical level
- Physical level.

At each of these levels, the structure (the objects and relationships which exist) and the processing of the application need to be described. The task level is important because the user needs to be aware of the system purpose. A task level description of UC should reveal that the purpose of UC is to provide advice, but it will not actually perform any Unix actions. Unless users understand this level of description, they will be left frustrated and wondering why

UC did not actually display who is using the system following this interaction (from Chin 1989)

User: Who is on the system?
UC: To find out who is on the system, type 'who'.

The logical level of the domain model emphasises that in order to achieve some purpose, certain functions *have* (logically) to be performed and certain objects *have* to exist. It describes how something works. An important issue in user-system interaction is whether some data is to be supplied, or whether a particular function is to be carried out by the system or by the user. Logically something has to be done in order to fulfill a purpose. Whether it is done by human or machine is a design decision.

The physical level describes how to do something. It is a causal description concerned with how simple tasks are sequenced and how objects and displays are laid out. It is concerned with the presentation of the system, dialogue control and physical actions.

In addition to describing the system at these three levels, the domain model must describe the mappings between them. For example, a task maps to one or more logical methods (sequences of functions, or plans) of accomplishing that task. These functions are mapped onto the required physical actions. Descriptions may then be provided with or without the mappings. A description in physical actions describes simply how to do something. Including the mapping to logical functions or to tasks describes why the actions are appropriate.

As with the user model, the domain model does not contain everything about the application. The domain model represents the aspects of the application which are to be used in providing the adaptive capabilities.

4.3 *The interaction model*

The third component of an adaptive system is its representation of the actual and designed interaction between user and application. We refer to this as the interaction model. The interaction model contains everything which is concerned with the relationship which exists between the representation of the users (the user model) and the representation of the application (the domain model).

Although the domain model and the user model describe the domain and users respectively, it is only when a user actually uses the system that things can be inferred, or the system can adapt. For our purposes, we may define an interaction as a user making use of the system at a level which can be monitored. From the data thus gathered, the system can make inferences about the user's beliefs, plans and/or goals, long-term characteristics, such as cognitive traits, or profile data, such as previous experience. The system may tailor its behaviour to the needs of a particular interaction or, given suitably 'reflective' mechanisms, the system may evaluate its inferences and adaptations and adjust aspects of its own organization or behaviour.

There are two main aspects to the interaction model;

- capturing the appropriate raw data

- representing the inferences, adaptations and evaulations which may occur

Raw data is obtained through maintaining a dialogue history or dialogue record (DR) which records aspects of the individual user's observed behaviour. The mechansims by which inferences, adaptations and evaluations can be accomplished are represented in an Interaction Knowledge Base (IKB).

The dialogue record is simply a trace of the interaction at a given level of abstraction. It is kept for as long as is required according to the needs of the adaptive system and is then deleted. The dialogue record may contain details such as:

- the sequence of keystrokes made
- mouse clicks and mouse movements
- timing information such as the time between commands or the total time to complete a task
- system messages and other system behaviour

The dialogue record is an abstraction of the interaction so far as it does not capture everything which takes place. Facial expressions and other gestures are not yet part of the dialogue record, nor is it possible to record any non-interactive activities (such as reading a book) which a user may undertake during the interaction. However, as the variety of input devices continues to increase with the introduction of video recordings of interactions, tracking of eye-movements, etc. so the dialogue record may become more subtle.

Although the dialogue record contains only low level data, it is surprising how much can be inferred if the knowledge necessary is available (i.e. provided by a human or stored in a domain model). For example, using a typical WIMP editor the comman sequence 'position and cut' indicates a possible error since there is no 'drag' (or 'highlight') operation which indicates the data to be cut. User errors, habits and preferences can also be quickly inferred.

The second part of the interaction model is a description of the 'stereotyped' interaction; the Interaction Knowledge Base (IKB). This describes the inferences that can be made from the dialogue, the evaluations of the interaction which are possible and the changes (adaptations) which the system can accomplish.

The user model and domain model define what *can* be inferred. The IKB actually does the inferencing by combining the various domain model concepts to infer user characteristics or by combining user model concepts to adapt the system. The IKB represents the relationship between domain and user characteristics. It provides the interpretation of the dialogue record.

The interaction model is a vital part of an adaptive system. The individual interactions are maintained in the dialogue record and will be maintained for as long as is required by the data requirements of the interaction knowledge base. For example, in the IR-NLI system (Branjik *et al.* 1990) individual 'session histories' are analysed at the start of each interaction. The complete dialogue record, or a summary of it, has to be maintained between sessions in this system.

An important design decision which the developer of adaptive systems has to make is the level of abstraction which is required for the dialogue record, the

individual user data and the interaction knowledge-base. In our approach, the interaction model must be constrained so that all atrributes used in the representation are defined either in the domain model or in the user model as appropriate. Automatic mechanisms are required for this through software support.

The discussion above may have suggested that an adaptive system requires only one user, domain and interaction model, but this will not always be the case. For example, in a given system there may be several user roles. The focus of the system may be *about* a person. This is the case in Cohen and Jones's system (Cohen and Jones 1989) which helps parents and psychologists understand the learning difficulties of a student. It is also the issue raised by Spark Jones (Spark Jones 1989) who considers the need for an Agent (the person who conducts the interaction) and Patient (the person who is the subject of the system) user roles. In general there may be a large number of user models representing the individual agents in a multi-agent co-operative system.

Similarly there may be more than one domain model. For example, a system may have a model of the domain of books and another model in the domain of a particular company. The interaction can demand that these two domains are considered in a particular interaction. In the Alvey AID project (Browne *et al.* 1990) one of the systems developed would have required one domain model of the background system (Telecom Gold) and one for the help system.

The interaction model as discussed seems deceptively simple, but again this will not always be the case. The adaptive, inference and evaluation mechanisms contained in the IKB may be extremely complex and do not have to consist of simple rules. For example, in a natural language dialogue which is attempting to infer a user's focus of attention, the inference mechanisms will need a model of the syntax, semantics and pragmatics of dialogue in general. The IKB will need access to a theory of language understanding and generation. The system will also need to store and refer to its own reasoning which produced the recommendations. Although this arrangement is seen by some as what a discourse model should be (e.g. Wahlster 1988), others (e.g. Morik 1988) emphasise the difference between the interaction-independent theory of language and the interaction-dependent nature of the actual dialogue. This reflects the distinction which we have drawn between the dialogue record and the mechanisms in the IKB which use the data.

It follows from this consideration of natural language systems as adaptive systems that we can expect tutoring systems to include a model of teaching in their interaction model, help systems to include a model of help giving and explanation system to include a model of explanation giving. In general, the interaction model will represent the strategies and theory of the particular type of system. The IKB thus embodies what Spark Jones refers to as the 'System model' which contains the system's strategies and plans (Spark Jones 1989).

As a final note, it is important to understand that this is a conceptual framework and it does not prescribe any particular implementation. What actually goes into each model will be a design decision. For example, the relationship between a particular class of book and a particular class of user

could be represented as an attribute of the book object in the domain model or it could be represented as a inference mechanism in the interaction model. Such design options will always be available. What this framework does do is to focus attention on the different kinds of knowledge which are required by the different models.

5. METHODOLOGY FOR ADAPTIVE SYSTEM DEVELOPMENT

Whether or not a system should have an adaptive capability is essentially a human-computer interaction (HCI) activity. Alternatives to adaptive systems will often exist for system developers including training users in the use of a system, the use of other support mechanisms and customisation facilities. The suitability of each of these solutions will depend on the nature of the system, the environments in which it will be used and the characteristics of its users. Adaptive systems are one solution to usability problems (Benyon, in press). However, we do expect that incorporating more knowledge into the interface and providing systems with an adpative capability will become an increasingly attractive option for designers.

As an HCI activity, developing adaptive systems shares problems with all HCI development. HCI problems are typically ill-structured and HCI design is characterised by 'design instability' (Fischer 1989). Requirements cannot be fixed and used as the basis for deriving a formal design equivalent and so designers must take an 'alternating waves' approach to analysis and design rather than a strict top-down approach. Evaluation of alternative designs becomes central and effective software support tools are vital (Hartson and Hix 1989).

Adaptive systems differ from other interactive systems in that they are characterised by *design variety.* Rather than the designer trying to obtain a single solution to a problem, the designer specifies a number of solutions and matches those with the variety and the changeability of users, environments and tasks. At some point in the system development process, the designer may decide that the system requires an adaptive capability. This may be to meet a specified user requirement, it may arise from the analysis or it may happen during the transition from a logical to physical design. Once this option has been identified, the adaptive system must be developed in parallel with the application and using the same development approach.

The specification of the adaptive system part of the application requires the specification of the domain, user and interaction models. This in turn requires the designer to focus on what features of the system are to be adaptive, what characteristics of the users it will adapt to and how the adaptations will be accomplished. It may be that the whole system is to be adaptive. In this case, the domain model and the system design are the same thing. However, usually only a part of an application will require an adaptive capability and so it is this part which must be isolated and specified in the domain model.

One important issue in adaptive system development is ensuring that the

adaptivity can be controlled and measured and that the reasons for having an adaptive mechanism are known and clear. We believe that the architecture and methodology presented here contribute to this. However others (Browne, Totterdell and Norman 1990) suggest that specific adaptivity metrics are necessary in order to control the development process. We see adaptivity as a usability issue which can be accommodated by usability metrics.

Browne *et al.* also recognise four levels of adaptive system. (*Simple*) *adaptive systems* are characterised by their ability to produce a change in output in response to a change in input. Thus they possess a dialogue record and some adaptation mechanism, but do not have to have explicit user and domain models (though refining the mechanisms is made more difficult if they do not). Predictive interfaces (Greenberg *et al.* 1991) and some natural language systems fall into this category. They have a generally limited variety of behaviour because the adaptive mechanism is 'hard wired'. These are the stimulus-response systems, or simple rule-based systems.

Self-Regulating systems monitor effects of the adaptation on the subsequent interaction and evaluate this through trial and error. A mechanism is required which maintains a history or at least provides immediate feedback. This evaluation mechanism then selects from a range of possible outputs for any given input. These systems require inference mechanisms. They have to be able to abstract from the dialogue record and capture a logical task level interpretation of the interaction. Similarly the system must now include a representation of its own purpose at the task level of its domain model.

Whereas self-regulating systems monitor the effect of the change in behaviour on the actual interaction, *self-mediating systems* monitor the effect on a model of the interaction. Hence possible adaptions can be tried out in theory before being put into practice. Self-mediating systems thus require a model of the other system with which they are interacting (in order to estimate the change of behaviour which will result from the systems own adaptive change). They also need evaluation mechanisms in order to determine how effective a possible adaptation may be.

In all the other adaptive systems the models are static. *Self-modifying systems* are capable of changing these representations. This allows self-modifying systems to reason about the interaction. These are meta-adaptive systems in that they can change the user, interaction and domain models.

Browne *et al.* point out that these levels reflect a change of intention moving from a designer specifying and testing the mechanisms in a (simple) adaptive system to the system itself dealing with the design and evaluation of its mechanisms in a self-modifying system. Moving up the levels also incurs an increasing cost which may not be justified. It is important that designers consider what capabilities are most appropriate for the particular application at hand. There is little to be gained by having a self-modifying capability, for example, if the context of the interaction is never going to change.

5.1 *Example*

As an illustrative example of the methodology in action, consider the following case.[1] A database system was to be developed with the purpose of providing access to data about students and staff in a university. The system was to be easily usable, with no prior training by two main user groups; managers and administrative staff who would require the system only very infrequently and also by clerical staff who would use the system regularly.

During the systems analysis phase, two possible interface designs were developed — a command language interface similar to SQL and a menu interface — and experimentally evaluated with representative users. 80% of the users performed better (faster and more accurately) using the command interface, but 20% performed significantly better using the menu interface. All users were tested on a number of characteristics, some profile and some cognitive, and it was discovered that the 20% who had performed better using the menu interface had a low score on a spatial ability test, had low experience with command interfaces and were infrequent computer users.

The designer now has a dilemma. Should a single interface be provided, should users be able to select one of the available interfaces or should the designer adopt an adaptive system solution to the interface design? There is a large group of users (20%) who require one sort of interface and another group who require another. Being infrequent computer users, the users who require the more constrained interface should not be expected to customise the system. Training in the command language is not a viable option as syntax would soon be forgotten. Thus in this case, there seems to be good reason to consider the adaptive system approach.

Before making a final decision, however, the designer should consider whether data is available from the dialogue record to facilitate the required adaptation and what mechanisms could be employed to make the relevant inferences and adaptations. An analysis of the experimental data revealed that the number of errors made when using the command interface correlated with the users' spatial ability and command experience — every user making more than one error in twelve tasks was in the low spatial and low experience class. Errors could be easily detected from the dialogue and could be used to infer appropriate characteristics of users. The system was to be adaptive insofar as it would change the interface style in response to these characteristics of the users.

In this case, then, the adaptive system option seems feasible and desirable. Data for the user model can be unobtrusively inferred from the interaction and the mechanisms can be easily programmed. The domain model consists of four attributes, one at the task level (tasks completed), two at the logical level (errors made and average task completion time) and one at the physical level of description (the interface being used).

The user model contains data on the user's personal profile (experience with command interfaces and frequency of computer use), cognitive characteristics (spatial ability) and knowledge of the domain (the student model). This inherits

all the attributes from the domain model and is updated from the dialogue record.

The interaction model consists of the dialogue record which records details of the number of errors made and the number of tasks completed and the Interaction Knowledge-base which describes the inferences (the level of spatial ability is inferred from the number of errors and number of tasks) and adaptations (providing the command or menu interface) which the system can make. In this case no evaluations are undertaken since the system was not required to be self-regulatory.

Each of the models developed is appropriate for its purpose both in terms of the feasibility of collecting data and in terms of the characteristics represented.

5.2 *Tool support*

In our experience, a serious bottle-neck in the development of adaptive systems has been the difficulty in extracting and modifying the IKB. The Adaptive Systems Development Environment (ASDE, Benyon, Murray and Jennings 1990) is a designer's tool-kit which tailors the general-purpose nature of an AI Toolkit to the specific needs of an adaptive system developer and which exploits the UIMS facilities of the underlying system at run-time.

The ASDE is similar to the concept of an expert system shell or user modelling shell (Branjik *et al.* 1990, Finin 1989). During the development phase of building an adaptive system, the ASDE has to play two separate but related roles. In the first instance the developer employs the ASDE to specify the characteristics of the target adaptive system and its users, their interaction and the mechanisms which will guide the inferencing, adaptations and evaluations which are to take place (see Section 4.3). These features are represented in the adaptive system's knowledge-base. Once the structure of a particular adaptive system has been established, the developer uses the ASDE to specify the values of relevant attributes of individual or classes of user and the inferences, adaptations and evaluations which are to be made from these values.

The user interacts directly with the target system which then exploits its knowledge of users and the domain in order to adapt the system appropriately. The interaction of the ASDE with an adaptive system is illustrated schematically in Fig. 3.

The ASDE reflects the architecture of adaptive systems described in Section 4. It consists of a domain model which describes the structure and functioning of the domain of the target (adaptive) system and provides the structure of an overlay model which forms the basis of the student model. In the user model, each user (individual or class) is represented in the adaptive system as a system object. The attributes are represented as slots in the objects and hence all members of that class of user automatically inherit all the attributes of the class. The facilities offered by inheritance mechanisms are particularly appropriate for the user model. Initially, the designer specifies the characteristics of classes, or stereotypes of user. When individuals use the target system they are allocated to one or more stereotypes on the basis of the knowledge which the system has

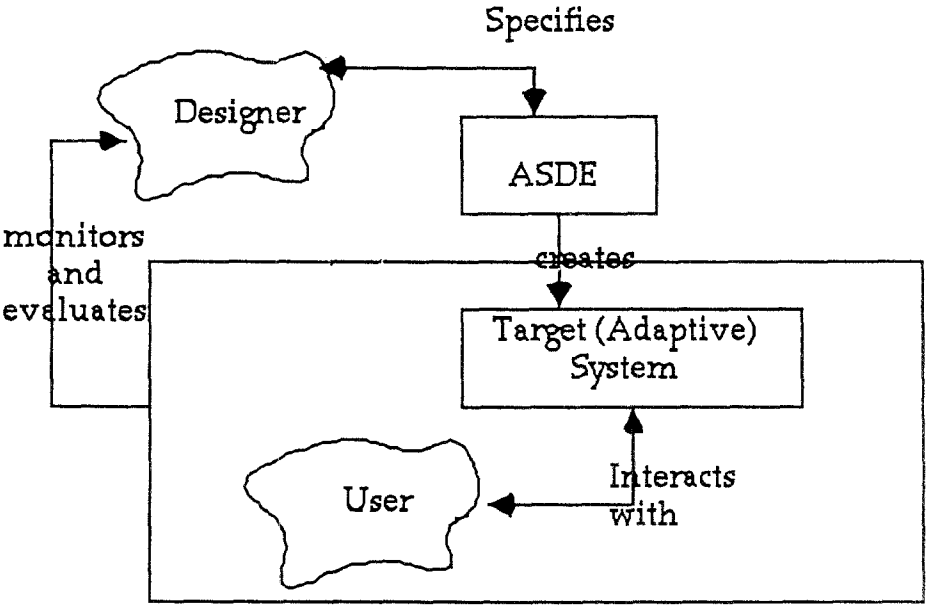

Fig. 3. Relationship between the ASDE and an adaptive system.

acquired about the user. The individual then becomes a member of that class and inherits the characteristics of the class. Conflicting classifications can be handled through the conflict resolution mechanisms provided by the AI Toolkit. As the system learns more about the user, so the user model becomes increasingly individual. The IKB is defined using the knowledge representation facilities offered by the AI Toolkit.

One of the most important aspects here is that the model of each user is explicit and can be displayed and edited if necessary. This facility is vital both in respect of the privacy of individuals and their rights under data protection legislation, and in order to maintain the accuracy of the model. Making the model explicit and separable from the other components of the adaptive system facilitates changing the mechanisms as details of the interaction are understood.

6. DISCUSSION

Building intelligence into the interface is a viable option for designers of interactive systems particularly if the benefits can be shown to outweigh the costs. Up until now, however, it has been a rather ad hoc affair. The contribution which we make here is to place such decisions on a more stable footing. Developers of adaptive systems should follow the methodology outlined here and describe their designs in terms of the models identified. Consideration of the level of adaptivity required is also important (Section 5).

The development of adaptive systems must be seen within the framework of developing interactive systems. The need for an adaptive system emerges as a design decision during the development of an interactive system. In some cases (e.g. a natural language interface) this design decision may occur very early in the development process, but in other systems it may not appear until later. The purpose and functions of the adaptive system must be carefully formulated and understood before they are expressed within the adaptive system architecture. We expect that as adaptive systems become more widely used, guidelines will be formulated to assist the designer in making a decision on when and where to exploit an adaptive system option. Principles emerging from our own work suggest guidelines such as;

- 'adapt the system where users are least adaptable'
- 'adapt to features which have the largest impact on the interaction'
- 'adapt to the needs of intermittent and discretionary users'.

The adaptive system developer needs to consider the opportunities for adaptivity provided by the framework of the three models and their relationships identified in Section 4. Developing the user model forces the designer to focus on the psychological, profile and domain knowledge which users will require. The domain has to be described at the task, logical and physical levels. Specifying the interaction model requires the designer to consider what data is available from the interaction (the dialogue record) and the inferences, adaptations and evaluations which can be made from the dialogue record within the context of the domain and user models.

Although the 'bandwidth' of the dialogue record is necessarily rather narrow (sequences of tokens passed across the interface and attributes of that sequence such as timing information), this data can be put to powerful use when coupled with an explicit domain model and the potential of re-use of user data through long-term user models. For example, from the dialogue record we can identify the use of individual commands and command sequences which allows the system to infer goals from the physical/logical/task mappings contained in the domain model. Unusual activities and inefficient behaviour can be identified by comparing individual user activities with a normative user model and the physical/logical mappings in the domain model. Exploiting psychological data alongside a three level domain model allows the system to adapt at the task level (e.g. by avoiding high-risk tasks such as formatting discs for nervous users), at the logical level (e.g. by making different functions available for users with particular cognitive skills) and at the physical level (e.g. changing the interface style). Profile data can be used in relating tasks to jobs, restricting the tasks the systems can do or altering features such as command keys to suit previous computer experience. Student model data can be used to introduce new tasks when a need is observed or when a level of proficiency is reached, introduce new functions or short cuts or to provide simple how-to-do-it knowledge if an important concept is missing.

The methodology presented in this paper and the necessity for adequate tool-support for design provides a firmer grounding for adaptive interface development. To develop adaptive systems, the designer needs to:

- identify the need for an adaptive capability, the features of the system which are to be adaptive and the variety of the users and environments in which the system is to operate
- identify the scope of the adaptive mechanisms and define methods for measuring the scope
- consider how data can be obtained for the user model — explicitly through asking the user or implicitly through inferring characteristics from the dialogue
- consider what data can be obtained from the dialogue record and whether the degree of detail required for the IKB is available
- evaluate and refine the IKB

ACKNOWLEDGEMENTS

Much of the work reported here was sponsored by the National Physical Laboratory, Teddington, UK under extra-mural research agreements NPL 86-0436 and NPL 82-0486.

NOTE

[1] This is based on our experimental work which is reported fully in Jennings, Benyon and Murray, 1991 and Jennings and Benyon, in press)

REFERENCES

Barnard, P. (1987). Cognitive Resources and the Learning of Human-Computer Dialogues. In Carrol, J. M. (Ed.), *Interfacing Though: Cognitive Aspects of Human-Computer Interaction.* MIT Press, Cambridge, Mass.

Benyon, D. R. (1984). Monitor: A Self-adaptive User Interface. In B. Shackel (Ed.), *Proc. INTERACT '84, First IFIP Conference on Human-Computer Interaction.* Elsevier Science Publishers B.V., Amsterdam.

Benyon, D. R. (1992). The Role of Task Analysis in Systems Design. *Interacting with Computers* **4**(1).

Benyon, D. R. (in press). Adaptive Systems; A Solution to Usability Problems. To appear in *User Modelling and User Adapted Interaction.*

Benyon, D. R., Innocent P. R. and Murray, D. M. (1987). System Adaptivity and the Modelling of Stereotypes. In Shackel, B. and Bullinger, H.-J. (Eds.), *Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction.* Elsevier Science Publishers B.V., Amsterdam.

Benyon D. R., Murray, D. M. and Milan, S. (1987). Modelling Users' Cognitive Abilities in an Adaptive System. In Rasmussen, J. and Zunde, P. (Eds.), *Proc. 5th Symposium EFISS,* Risø National Laboratory, Denmark, November 1987. Plenum Publishing, New York.

Benyon, D. R. and Murray, D. M. (1988). Experience with Adaptive Interfaces. *The Computer Journal* **31**(5).

Benyon, D. R., Jennings, F. and Murray, D. M. (1990). An Adaptive System Developer's Toolkit. In Diaper, D. *et al.* (Eds.), *Proc. INTERACT '90, Third IFIP Conference on Human-Computer Interaction.* Elsevier Science Publishers B.V., Amsterdam.

Bösser, T. (1987). *Learning in Man-Computer Interaction.* Springer-Verlag, Berlin.

Branjik, G., Guida, G. and Tasso, C. (1990). User Modelling in Expert Man-Machine Interfaces: A Case Study in Information Retrieval. *IEEE Trans. Systems, Man and Cybernetics* **20**(1).

Browne, D. P., Toterdell, P. A. and Norman, M. A. (1990). *Adaptive User Interfaces.* Academic Press, London.

Bundy, A. (Ed.) (1983). *Alvey 1983 Intelligent Front End Workshop* 26—27 Sept 1983 Cosener's House, Abingdon, England. DTI, London.

Card, S., Moran, A. P. and Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Carroll, J. M. and McKendree, J. (1987). Interface Design Issues for Advice-Giving Systems. *Communications of the ACM* **30**(1).

Carroll, J. M. (1991). *The Nurnberg Funnel.* MIT Press, Cambridge, Mass.

Chignell, M. H. and Hancock, P. A. (1988). Intelligent Interfaces. In Helander, M. (Ed.), *Handbook of Human-Computer Interaction.* Elsevier Science Publishers B.V., Amsterdam.

Chin, D. N. (1986). User Modelling in UC, the Unix Consultant. In Mantei, M. and Orbiton, P. (Eds.), *Proc. CHI '86, Human Factors in Computing Systems.* ACM, New York.

Chin, D. N. (1989). KNOME: Modelling what the User Knows in UC. In Kobsa, A. and Wahlster, W. (1989) (Eds.), *User Models in Dialog Systems.* Springer-Verlag, Berlin.

Cockton, G. (1987). A New Model for Separable Interactive Systems. In Shackel, B. and Bullinger, H.-J. (Eds.), *Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction.* Elsevier Science Publshers B.V., Amsterdam.

Cockton, G. (Ed.) (1990). *Engineering for Human-Computer Interaction.* Elsevier Science Publishers B.V., Amsterdam.

Cohen, R. and Jones, M. (1988). Incorporating User Models into Expert Systems for Educational Diagnosis. In Kobsa, A. and Wahlster, W. (1989) (Eds.), *User Models in Dialog Systems.* Springer-Verlag, Berlin.

Coutaz, J. (1987). PAC: An Object Orientated Model for Implementing User Interface. In Bullinger, H. J. and Shackel, B. (Eds.), *Human-Computer Interaction, Proceedings of INTERACT '87.* Elsevier Science Publishers B.V., Amsterdam.

Demazeau, Y. and Muller, J. P. (Eds.) (1991). Decentralized Artifical Intelligence, Vol. 2. *Proceedings of 2nd European Workshop on Modelling Autonomous Agents in a Multi-Agent World.* Elsevier Science Publishers B.V., Amsterdam.

Dennett, D. (1989). *The Intentional Stance.* MIT Press, Cambridge, Mass.

Diaper, D. (1989). *Task Analysis for Human-Computer Interaction.* Ellis Horwood, Chichester.

Edmonds, E. A. (1981). Adaptive Man-Computer Dialogues. In Coombs, M. J. and Alty, J. L. (Eds.), *Computing Skills and the User Interface.* Academic Press, London, pp. 389—426.

Edmonds, E. A. (1987). Adaptation, Response and Knowledge. *Knowledge-Based Systems* **1**(1), Editorial.

Egan, D. E. (1988). Individual Differences in Human-Computer Interaction. In Helander, M. (Ed.), *Handbook of Human-Computer Interaction.* Elsevier Science Publishers B.V., Amsterdam.

Finin, T. W. (1989). GUMS — A General User Modelling Shell. In Wahlster, W. and Kobsa, A. (Eds.), op. cit.

Fischer, G. (1987). Making Computers More Useful. In Salvendy, G. (Ed.), *Cognitive Engineering in the Design of Human-Computer Interaction and Expert-Systems.* Elsevier Science Publishers B.V., Amsterdam.

Fischer, G., Lemke, A. C. and Schwab, T. (1986). Knowledge-based Help Systems. In Mantei, M. and Orbiton, P. (Eds.), *Proc. CHI '86, Human Factors in Computing Systems.* ACM, New York.

Fischer, G., Morch, A. and McCall, R. (1989). Design Environments for Constructive and Argumentative Design. In *Proceedings CHI '89, Human Factors in Computing Systems.* ACM, New York.

Fischer, G. (1989). HCI Software: Lessons Learned, Challenges Ahead. *IEEE Software,* January 1989.

Foley, J., Kim, Won Chul, Kovacevic, S. and Murray, K. (1989). Defining Interfaces at a High Level of Abstraction. *IEEE Software,* January.

Green, M. (1985) Report on Dialgoue Specification Tools in UIMS. In Pfaff, G. E. (Ed.), *User Interface Management Systems.* Springer Verlag, Heidelberg.

Green, T. R. G., Schiele, F. and Payne, S. J. (1988). Formalisable Models of User Knowledge in Human-Computer Interaction. In van der Veer, C. C., Green, T. R. G., Hoc, J. M. and Murray D. M. (Eds.), *Working with Computers: Theory versus Outcome.* Academic Press, London.

Greenberg, S. and Witten, I. H. (1985). Adaptive Personalized Interfaces — A Question of Viability. *Behaviour and Information Technology* **4**(1).

Greenberg, S., Darragh, I., Maulsby, D. and Witten, I. H. (1991). *Predictive Interfaces. What Will They Think of Next?* presented at CHI '91 (unpublished).

Hancock, P. A. and Chignell, M. H. (Eds.) (1989). *Intelligent Interfaces; Theory, Research and Design.* North-Holland, New York.

Hansen, S. S., Holgaard, L. and Smith, M. (1988). EUROHELP: Intelligent Help Systems for Information Processing Systems. In *Proc. 5th Annual ESPRIT Conference,* Brussels, November 1988. Kluwer Academic Publishers, Amsterdam.

Hartson, H. R. and Hix, D. (1989). Toward Empirically Derived Methodologies and Tools for HCI Development. *International Journal of Man Machine Studies* **31**, 477—494.

Hurley, W. D. and Silbert, J. L. (1989). Modelling User Interface Application Interactions. *IEEE Software,* January.

IEEE Software (1989), January.

Innocent, P. R. (1982). A Self-Adaptive User Interface. *International Journal of Man Machine Studies* **16**(3), 287—300.

Jennings F., Benyon, D. R. and Murray, D. M. (1991). Adapting Systems to Individual Differences in Cognitive Style. *Acta Psychologia* **78**(1—3), December.

Jennings, F. and Benyon, D. R. (in press) Database Systems: Different Interfaces for Different Users. In *Behaviour and Information Technology* (forthcoming).

Jerrams-Smith, J. (1985). SUSI — A Smart User-System Interface. In Johnson, P. and Cook, S. (Eds.), *People and Computers: Designing the Interface.* Cambridge University Press, Cambridge.

Johnson, P., Johnson, H., Waddington, R. and Shouls, A. (1988?). Task-related Knowledge Structures: Analysis, Modelling and Application. In Jones, D. M. and Winder, R. (Eds.), *People and Computers IV: From Research to Implementation.* Cambridge University Press, Cambridge.

Johnson, P. (1989). Supporting System Design by Analyzing Current Task Knowledge. In Diaper, D. (Ed.), *Task Analysis for Human-Computer Interaction.* Ellis-Horwood, Chichester.

Kass, R. (1989). Student Modelling in Intelligent Tutoring Systems. In Kobsa, A. and Wahlster, W. (1989), *User Models in Dialog Systems.* Springer-Verlag, Berlin.

Kass, R. and Finin, T. (1988). The Need for User Models in Generating Expert System Explanations. *International Journal of Expert Systems* **1**(4).

Kass, R. and Finin, T. (1989). The Role of User Models in Cooperative Interactive Systems. *International Journal of Intelligent Systems* **4**(1).

Kay, A. (1989). User Interface: A Personal View. In Laurel, B. (Ed.), op.cit.

Kay, J. (1991). UM: A Toolkit for User Modelling. *User Modelling and User-Adapted Interaction* **1**(?).

Kieras, D. and Polson, P. G. (1985). An Approach to the Formal Analysis of User Complexity. *International Journal of Man Machine Studies* **22**(?).

Kiss, G. (1986). *High Level Dialogues in MMI,* Final Report on the Alvey Survey Project. Department of Trade and Industry Information Engineering Directorate, London.

Kobsa, A. (1987). *A Taxonomy of Beliefs and Goals for User Modelling in Dialog Systems,* Memo Nr. 28. Universitat des Saarlandes, Saarbrucken.

Kobsa, A. (1988). *A Bibliography of the Field of User Modelling in Artificial Intelligence Dialog Systems,* Memo Nr. 23. Universitat des Saarlandes, Saarbrucken.

Kobsa, A. and Wahlster, W. (1989). *User Models in Dialog Systems.* Springer-Verlag, Berlin.

Laurel, B. (1990). Interface Agents. In Laurel, B. (Ed.), *The Art of Human-Computer Interface Design.* Addison Wesley, Wokingham.

Lehner, P. E. (1987). Cognitive Factors in User/Expert-System Interaction. *Human Factors* **29**(1).

Mason, M. V. (1986). Adaptive Command Prompting in an On-line Documentation System. *International Journal of Man-Machine Studies* **25**(?).

Moore, J. and Swartout, W. R. (1988). Planning and Reacting. *Proc. AAAI Workshop on Text Planning and Generation*, August 25, 1988. St. Paul, Minnesota.

Moran, T. P. (1981). Command Language Grammar: A Representation of the User Interface of Interactive Computer Systems. *International Journal of Man-Machine Studies* **15**(3).

Moran, T. P. (1983). Getting into a System: External-internal Task Mapping Analysis. In Smith, R. N. and Pew, R. W. (Eds.), *Proceedings CHI '83: Human Factors in Computing Systems.* ACM Press.

Morik, K. (1989). User Models and Conversational Settings: Modelling the User's Wants. In Kobsa, A. and Wahlster, W. (1989). *User Models in Dialog Systems.* Springer-Verlag, Berlin.

Morik, K. (1988). Discourse Models, Dialog Memories and User Models. *Computational Linguistics* **14**(13), 95—97.

Murray, D. M. (1987). Embedded User Models. In Shackel, B. and Bullinger, H.-J. (Eds.), *Proc. INTERACT '87, Second IFIP Conference on Human-Computer Interaction.* Elsevier Science Publishers B.V., Amsterdam.

Murray, D. M. (1988). Building a User Modelling Shell. In Zunde, P. (Ed.), *Proc. 6th Symposium EFISS*, Georgia Tech., Atlanta, Georgia, USA, October 1988. Plenum Publishing, New York.

Murray, D. M. (1989). Modelling for Adaptivity. *Proceedings of 8th Interdisciplinary Workshop, Informatics and Psychology*, Scharding, Austria, May 1989. North Holland, Amsterdam.

Myers, B. (1989). User Interface Tools: Introduction and Survey. *IEEE Software*, January.

Negroponte, N. (1989). Beyond the Desktop Metaphor. *International Journal of HCI* **1**(1).

Newell, A. (1982). *The Knowledge Level Artificial Intelligence* **18**(1), 87—127.

Norman, D. (1986). Cognitive Engineering. In Norman, D. and Draper, S. (Eds.), *User Centred System Design.* Lawerence Erlbaum Assoc., Hillsdale, New Jersey.

Paris, C. L. (1989). The Use of Explicit User Models in a Generation System. In Kobsa, A. and Wahlster, W. (1989), *User Models in Dialog Systems.* Springer-Verlag, Berlin.

Payne, S. J. and Green, T. R. G. (1989). Task-Action Grammar: The Model and Its Developments. In Diaper, D. (Ed.), *Task Analysis for Human-Computer Interaction.* Ellis-Horwood, pp. 75—107.

Pullinger, D. (1989). Moral Judgements in Designing Better Systems. In *Interacting with Computers* **1**(1).

Pylyshyn, Z. W. (1984). *Computation and Cognition.* MIT Press, Cambridge, Mass.

Quilici, A., Dyer, M. and Flowers, M. (1986). AQUA: An Intelligent Unix Advisor. In Steels, L. (Ed.), *Proc. 7th ECAI*, Brighton 1986.

Rasmussen, J. (1986). *Information Processing and Human-Maching Interaction.* Elsevier North-Holland, Amsterdam.

Rasmussen, J. (1987). Mental Models and Their Implications for Design. In *Austrian Computer Society 6th Workshop on Informatics and Psychology*, June 1987.

Rich, E. (1983). Users Are Individuals: Individualising User Models. *International Journal of Man Machine Studies* **18**(3), 199—214.

Rich, E. (1989). Stereotypes and User Modelling. In Kobsa, A. and Wahlster, W. (1989), *User Models in Dialog Systems.* Springer-Verlag, Berlin.

Rivers, R. (1989). Embedded User Models; Where Next? In *Interacting with Computers* **1**(1).

Sarantinos, E. and Johnston, P. (1991). Explanation Dialogues: A Theory of How Experts Provide Explanations of Novices and Partial Experts. Unpublished.

Seel, N. (1990). From Here to Agent Theory. *AISB Quarterly*, no. 72, Spring.

Spark Jones, K. (1988). Realism about User Modelling. In Kobsa, A. and Wahlster, W. (Eds.), op. cit.

Steels, L. (1987). The Deepening of Expert Systems. *AICOM*, No. 1, 9—16.

Thimbleby, H. (1990b). *User Interface Design.* Addison Wesley, Wokingham.

van der Veer, G. C. (1990). *Human-Computer Interaction. Learning, Individual Differences and Design Recommendations.* Offsetdrukkerij Haveka B.V., Alblasserdam.

Vicente, K. J. and Williges, R. C. (1987). Assaying and Isolating Individual Differences in Searching a Hierarchical File System. *Human Factors* **29**, 349—359.

Wahlster, W. (1988). Distinguishing User Models from Discourse Models. *Computational Linguistics* **14**(3), 101—103.

Wahlster W. and Kobsa, A. (1987). Dialogue-based User Models. *Proc. IEEE* **74**(4).

Wilensky, R., Arens, Y. and Chin, D. (1984). Talking to Unix in English: An Overview of UC. *Communications of the ACM* **27**(6).

Wilson, M. D., Barnard, P. J., Green, T. R. G. and Maclean, A. (1988). Task Analyses in Human-Computer Interaction. In van der Veer, C. C., Green, T. R. G., Hoc, J. M. and Murray, D. M. (Eds.), *Working with Computer Theory versus Outcome.* Academic Press, London.

Wilson, M. D., Barnard, P. J. and Maclean, A. (1985). User Learning of Core Command Sequences in a Menu System. *IBM Hursley Human Factors Report, HF114.* IBM Hursley Park, Winchester, UK.

Young, R. M. and Hull, A. (1982). Categorisation Structures in Hierarchical Menus. In *Proceedings of 10th International Symposium on Human Factors in Telecommunications.* Helsinki, pp. 111—118.