Adaptive Systems: A Solution to Usability Problems

DAVID BENYON
Computing Department,
Open University,
Milton Keynes,
MK7 6AA, U.K.
E-mail: d.r.benyon@open.ac.uk

(Received 12 February 1991; in final form 12 February 1993)

Abstract. Improving the usability of computer systems is perhaps the most important goal of human-computer interaction research. Current approaches to usability engineering tend to focus on simply improving the interface. An alternative is to build intelligence into the system. However, in order to do this a more comprehensive analysis is required and systems must be designed so that they can be made adaptive. This paper examines the implications for systems analysis, design and usability specification if adaptive systems are to be a realistic solution to usability problems.

Key words: Usability, adaptive systems, analysis, adaptive system architecture

1. Introduction

Whiteside et al. define usability engineering as 'a collection of techniques to support management of resources in the development of user interfaces and computer function' (Whiteside et al., 1988). They see users' experience of the usability of systems as the ultimate test of system quality and stress that usability, functionality and system architecture are intrinsically linked.

The general approach to usability engineering is for user interface experts to study the usability of an existing or prototype system and to recommend improvements. Appropriate methods for such an analysis include experts criticising the interface (heuristic evaluation), formal laboratory-based usability testing, the application of HCI guidelines and structured 'cognitive walkthroughs'. Usability problems are identified and classified and solutions proposed. These possible solutions are obtained from a variety of sources such as experience, protocol analysis and brainstorming sessions.

In some cases a small change to the interface may be an effective solution to a usability problem, but in others more radical alternatives are required. In particular, it is increasingly possible to include some measure of intelligence – in the form of an automatic adaptive capability – in the system. This paper considers the circumstances under which the designer may wish to use this option and provides guidelines as to the developments of traditional analysis and design techniques which are required in such cases. The importance of a complete analysis is illustrated by means of a small example which also demonstrates how the recommended adaptive system architecture is used to specify the adaptive system's capabilities.

This paper does not address the usability of adaptive and intelligent systems themselves. Criticisms of adaptive systems include considerations of feasibility (e.g., Thimbleby, 1990), excessive cost, problems of 'hunting' (Edmonds, 1987) and issues of consistency. However, none of these is *inherently* damning to the notion of a system having an adaptive capability. The desirability of the adaptive system solution to a usability problem is an issue which demands experimental evidence, consideration of general HCI guidelines and evaluation against other design options. This can only be achieved given the specific circumstances and the purpose of the system.

2. Usability Requirements

Shackel (1990) recognizes that there are four components of any work situation: user, task, system and environment. In the case of human-computer interaction, the system is the computer system. Environment includes physical aspects such as appropriate heating, lighting, equipment layout, operating circumstances and so on as well as psychological aspects such as the provision of help and training and socio-political features such as the organizational environment in which the interaction takes place. Usability is concerned with achieving a harmony between these components.

Shackel proposes that usability can be seen in terms of four operational criteria; effectiveness, learnability, flexibility and attitude. Effectiveness is specified with respect to the performance (as measured by a characteristic of the interaction such as the time taken to complete the task or the number of errors made) of a range of tasks by some percentage of the users within some proportion of the environments in which the system will operate. Learnability is defined in terms of the time taken to learn (to some specified level of competence) given a specified amount of training. Learning also includes the time taken to relearn the system if details are forgotten. Flexibility covers the amount of variation in the tasks and/or environments which can be accommodated by the design. Attitude concerns the acceptable levels of human cost (tiredness, effort, etc.) which are required so that users are satisfied enough to continue to use and to enhance their use of the system.

Whiteside et al. do not identify usability criteria so precisely, preferring to deal in general with usability attributes which should be determined, along with functional and data requirements, during the requirements specification stage of system development. However, both approaches emphasize that usability requirements should be definable and measurable. This aspect of usability identifies a trend which is reflected in recent work on usability metrics. Thus a good usability attribute will define, precisely, what the task is (the level of difficulty etc.), how it is to be measured and the expected level of performance.

For example, a database system may have the following usability requirement

> In this system retrieval tasks with a level of complexity of a single SQL type 'select . . . from . . . where . . . ' statement should be accomplished in an average of less than 25 seconds (s) by 80% of the users. Over twelve tasks less than 10% of users should make more than two errors.

The problem with this approach to usability specification is that it does not go far enough. In particular it does not consider explicitly the variety which is inherent in many systems. Most systems will be used by heterogeneous users who will be interacting with the system in a number of different environments. These considerations need to be made explicit in the definition of usability criteria.

3. Variability in Systems Design

The problems of dealing with the 'average' user of a system was illustrated in an experiment which was conducted recently (Jennings and Benyon, in press). During a formative evaluation of a database system, a menu interface could only achieve an average retrieval time of 33 seconds. However, it achieved this with less than two errors in twelve tasks. A command interface to the database system was able to achieve a response time of 29 seconds for all subjects, but 25% of subjects made more than 2 errors on the twelve tasks. However, the other 75% of subjects achieved an average response time of 24 seconds with less than 1 error on the twelve tasks.

In this sort of situation, designers are faced with the problem that a single interface cannot be designed which meets the usability requirement (as given above). Moreover, there is a large difference in the performance of a significant 25% of the users. Making just the menu interface available is clearly a severe restriction on 75% of users (since it takes an average of 9 seconds per query longer for them) and may lead to other usability problems. In such a case, designers have a number of options. They may decide to review the usability requirements, to redesign the interface, to provide additional help or user training or to make both interfaces available.

A similar situation arises with respect to variety of operating environments. A specific system may be used in an office environment where there is ample local expertise, or where face-to-face help can be given if required. The same system may be used by a lone user where such support is not available. This variety of environments needs explicit consideration in defining usability criteria.

4. Options for the Designer

The job of the system designer is to achieve a harmony between users, tasks, environments and system. Any of these may be altered in order to improve the relationship. Users can be educated and trained. The environments can be enriched. Tasks can be changed or the system can be redesigned. However, systems which are adapted to interact with other systems simply through a fixed design will always be ill-adapted to interact with systems which are outside the scope of the design limits. For example, a system which is designed to be used after a given amount of user training will be ill-adapted to interact with a person who has not received that training.

One way to facilitate interaction between two systems which are otherwise unable to engage in a successful interaction is to use an adapter. An adapter is a third system which mediates between the two ill-adapted systems. This may be a software system – a 'front-end' – or a human intermediary such as a librarian who helps users conduct literature searches.

A larger number of successful interactions can be afforded by providing a designed system with a customising or tailoring facility. Customisation requires one system to alter the state or behaviour of another system.

The final way to support successful interaction between two systems is if one or both systems has some mechanism which can automatically select alternative behaviours. These are usually referred to as adaptive systems. Adaptive systems have advantages over fixed design systems in that they can engage in a larger number of interactions without needing to employ an adapter, or requiring action by another system to change their state or behaviour. However, they incur the cost of having to maintain more complex representations; of themselves, of the systems with which they can interact and of the interaction itself. Humans are adaptive systems. Computer systems can be made adaptive.

Within the broad category of adaptive systems, Browne et al. (1990) identify four levels of adaptation based on the complexity of the representations maintained by the system and the ability of the system to utilize those representations. 'Simple' adaptive systems use a 'hard wired' stimulus-response mechanism. 'Self-Regulating' systems monitor the effects of the adaptation on the subsequent interaction and evaluate this through trial and error. 'Self-mediating' systems monitor the effect on a model of the interaction (as opposed to monitoring the effect of the change in behaviour on the actual interaction). Hence possible adaptations can be tried out in theory before being put into practice. 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. Adaptive mechanisms (for example, 'predictive interfaces') may also employ statistical inference mechanisms and domain knowledge to anticipate the behaviour of the other system (Greenberg et al., 1991).

Browne et al. (1990) also argue that the levels of adaptivity 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. There is little to be gained by having a self-modifying capability if the context of the interaction is never going to change.

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 and the environments. Adaptive systems are a serious solution to usability problems where a degree of variety is present. In this sense, adaptive systems are pragmatic. Any system may require an adaptive capability depending on the variety with which it has to deal. Some systems, such as Natural Language (NL) systems, or Intelligent Tutoring Systems (ITS) must be essentially different for each individual user. The functional requirements of such systems determine that they must be adaptive. Other systems may require only a small part of the system to be adaptive and then only to a small number of user groups or environmental differences.

5. Architecture for Adaptive Systems

The designer has to consider the range of ways in which systems can be made adaptive and to select a capability appropriate to the usability problem at hand. The most basic architecture of an adaptive system is illustrated in Figure 1. An adaptive system requires three models, or representations. The

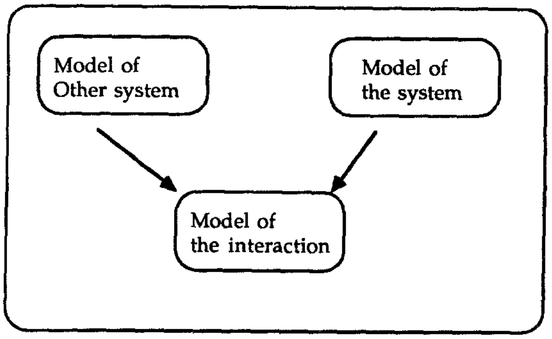

Fig. 1. General model of an adaptive system.

sophistication of the adaptive mechanism depends on the quality of these models.

The model of the system describes the characteristics which can be altered i.e. the aspects of the system which are adaptable. A system model which represents only physical aspects of the interaction such as screen displays, dialogue content or the effects of function keys will only be able to adapt at this level. A system model which represents the logical structure or functioning of the system will be capable of adapting at a logical level. For example, HAM-ANS (Morik, 1989) deals with a logical description of hotel rooms whereas the adaptive menu system (Greenberg and Witten, 1985) only adapts the physical arrangement of menu items. A further level of adaptivity may be obtained if the model of the system describes the system at the task level i.e. what the system can be used for. A complex system may be kept functionally simple for a new user if it can adapt at the task level. These three levels of description are necessary because each reveals some generalisation which would otherwise not be available. The philosophical arguments for this view can be found in Dennett (1987) and Pylyshyn (1984) amongst others.

The model of the other system describes the attributes of the system which the adaptive system can adapt to. This is the adaptive system's 'notional world'. In the case of two interacting computer systems, this model is maintained at the three levels described previously. In the case of a system adapting to a human, the model of the other system is the *user model* and may represent characteristics of individual users such as their cognitive characteristics, personal profile and/or domain knowledge. Hence HAM-ANS employs a model of the 'world' of its users in terms of profile data such as their job. Intelligent tutoring systems usually see the world in terms of the users domain knowledge (i.e. a *student model*).

The models of the interacting systems only define what adaptations are possible – how the system can change and what it can adapt to. The interaction model describes the actual adaptations which the system makes. It also describes the inferences which can be made about the other system from the interaction and may include evaluation mechanisms which measure actual performance against some definition of purpose (as represented in the model of the system). The inference, adaptation and evaluation mechanisms constitute the system's *interaction knowledge base.* In addition to this, any adaptive system must maintain a record of the interaction – the dialogue record. The grain and length of this record is another significant factor in determining the capability of the adaptive system. The interaction model thus consists of a dialogue record and an interaction knowledge base.

The architecture for an adaptive system which adapts to human users is illustrated schematically in Figure 2. Although only one domain, user and interaction model is shown in the Figure, any given system may have several such models if it has to adapt to, or make inferences from a number of other systems. For example, in Cohen and Jones's system (Cohen and Jones, 1989) which helps parents and psychologists understand the learning difficulties of a student, there is a need for two user models; one representing the 'patient' and the other the 'agent' (Spark Jones, 1989). Similarly there may be a need for more than one domain model. There is no requirement for every adaptive system to possess all the components illustrated in Figure 2. However, the Figure does provide a reference model which allows a comparison of different systems.

Referring systems to the components in Figure 2 allows us to classify systems according to the amount of the architecture which they explicitly implement and the complexity of the models which they employ. Thus the focus of a predictive interface such as the reactive keyboard (Greenberg et al., 1991), the adaptive menu system (Greenberg and Witten, 1985) or adaptive manual (Mason, 1986) is on the inference and adaptation mechanisms and the dialogue record. In these systems there is often no attempt to abstract long-term user characteristics (although in one application of the reactive keyboard a long-term user model is maintained) and the domain model is usually represented only implicitly in the system code. The simple stimulus-response adaptive systems contain an adaptation mechanism which selects the output on the basis of an analysis of the dialogue record. They may include an explicit, embedded user model which provides further conditions to control the firing of the adaptation rules. Self-regulating adaptive systems require inference and evaluation mechanisms since they have to learn from interacting with the environment. They have to be able to abstract from the dialogue

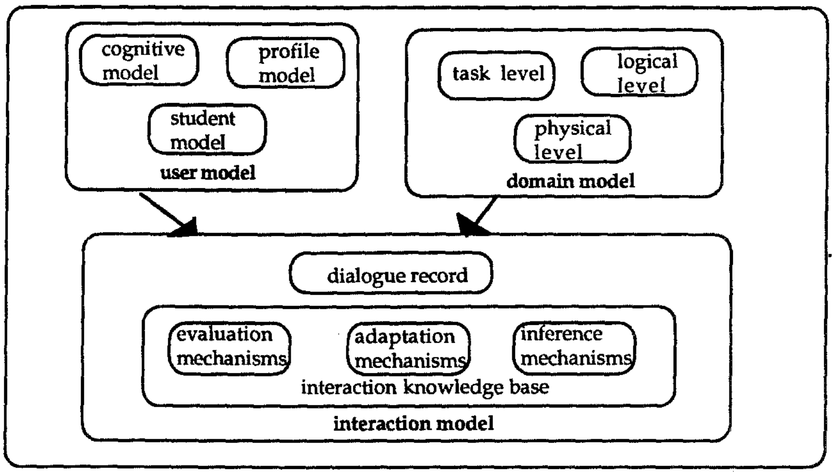

Fig. 2. Schematic outline of adaptive human-computer system.

record and capture a logical or task level interpretation of the interaction. An explicit domain model is vital in such systems. Self-mediating systems require a more sophisticated model of the interaction and of the other system. The domain model has to include an explicit statement of the reasoning which underlies the adaptive capabilities, so that the system can try out alternative adaptations in theory before trying them in practice. Self-modifying systems can change these models and hence have meta-rules which allow the system to add, modify or delete existing aspects of the interaction knowledge-base. Systems which can amend the user and domain models represent another level of sophistication.

6. The Need for Better Analysis and Design

Systems analysis is the process of understanding the problems of any current system and establishing the requirements for any replacement system. Systems analysis inevitably involves some design and needs to be seen in the context of an iterative approach to systems development.

If systems analysis is to provide the designer with enough data to formulate usability requirements successfully – and to allow the designer to consider an adaptive system solution to usability problems – then traditional methods of analysis which focus simply on functional and data requirements are inadequate. An enhanced concept of systems analysis should be seen as consisting of five interrelated and interdependent activities;

- *functional analysis* aims to establish the main functions of the system.
- *data analysis* is concerned with understanding and representing the meaning and structure of data in the application. Data analysis and functional analysis go hand in hand to describe the information processing capabilities of the system (Benyon, 1990).
- *task knowledge analysis* focuses on the cognitive characteristics required of the users by the system e.g. the search strategy required, cognitive loading, the assumed mental model etc. This analysis is device dependent (Benyon, 1992) and hence requires some design to have been completed before it can be undertaken.
- *user analysis* determines the scope of the user population which the system is to respond to. It is concerned with obtaining attributes of users which are relevant to the application such as the required intellectual ability, cognitive processing ability and prerequisite knowledge required. The anticipated user population will be analysed and categorised according to aspects of the application derived from task, functional, data and environment analysis.
- *environment analysis* which covers the environments within which the system is to operate. This includes physical aspects of the environment and 'softer' features such the amount and type of user support which is required.

Although formal methods exist for data and functional analysis and several cognitive task analysis methods are emerging (Diaper, 1988; Barnard, 1987) there are few *formal* techniques for conducting user and environment analysis. Checklist approaches can be found in (Catterall et al., 1991; and Macaulay et al., 1990) and various application specific user modelling techniques are described in (Kobsa and Wahlster, 1989), but generic, analytic techniques are still wanting. One contribution which work on user modelling may make to this area is to provide a framework within which user analysis can be more thoughtfully conducted.

The process of analysis results in the specification of system requirements. Once these have been obtained, the application can be specified at the three levels identified in the domain model and the mappings between them. The application should be represented in terms of structure and functions at each of these levels. A number of notations such as Entity-relationship diagrams (Benyon, 1990) and dataflow diagrams (DeMarco, 1979) are available for this purpose.

The application described in this manner is suitable for implementation as the domain model in an adaptive system.

- The task level describes the (external) tasks, or goals which the system is to support.
- The logical level describes the logical functioning and structure of the application.
- Physical design is concerned with the layout of screens and the construction of icons, etc. (the representational aspects) and with the operational aspects.
- The task/logical mapping describes the cognitive processing required by users as they translate their goals into the system's functionality.
- The logical/physical mapping describes the consistency, learnability and other aspects of the actual system use.

At some point in the system development process, the designer may decide that the system requires an adaptive capability. Designers should make such a decision on the basis of the analysis which has been conducted. For example, an automatic meeting scheduling system may include the requirement that individual user preferences for dates and times of meetings are taken into consideration. Such a requirement would demand an adaptive mechanism. A particularly important part of system design is the transition from a logical to a physical design (when functions are allocated either to the human or to the machine). A decision to be made by designers is that if a function is allocated to the user, it will require them to face additional cognitive processing, but if it is allocated to the machine, then the system will require an adaptive capability. This decision is informed by cognitive task analysis which considers the mental load imposed on the user by possible designs. For example, the simple adaptive mechanisms now being provided by modern spell-checkers illustrates how the task of correcting spelling mistakes is increasingly allocated to the machine rather than the user.

Once the adaptive capability option has been identified, the adaptive system must be developed in parallel with the application. 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 needs to be adapted and what it needs to adapt to. 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.

7. Example

This is an extended version of the example in Benyon and Murray (1993). It illustrates in terms of the models presented in the earlier parts of this paper, the processes which were undertaken and decisions which were arrived at during the development of an exemplar adaptive system (Jennings et al., 1991; Jennings and Benyon, in press). It is a small example which is intended to be illustrative of the approach and to provide a suitable vehicle for demonstrating the content of the representations which are required for any adaptive system.

The outline specification of the system may be taken as follows:

The purpose of the system is provide access to data about students and staff in a university. The system is to easily usable, with no prior training (a 'walk-up-and-use' system). It is to be used by managers and administrative staff who will require the system only very infrequently and also by clerical staff who will use the system regularly.

7.1. Analysis

During the analysis and design of the basic system the following analyses were made:

Data Analysis	There would be two files; one containing staff details and the other student details.
Functional Analysis	The system should support simple queries of the form Select <attributes> From <file> Where <selection criteria>.
User Analysis	Users vary in terms of how often they use the system and in their previous experience of computer systems.
Task Knowledge Analysis	The user will have to enter the required file, attributes and selection criteria.
Environment Analysis	Owing to the large number of possible sites for access to the database, there is unlikely to be much human or documentary support available for users.

Considering the user and task knowledge analysis suggests that two interfaces are required. HCI guidelines suggest that regular users are likely to prefer a user-determined dialogue (e.g. the structured query language, SQL). Infrequent users and new users are more likely to prefer a system-determined dialogue such as a menu interface. The menu interface also reduces memory load as it assists users in navigating through the database and relieves them of the need to remember SQL syntax, file and attribute names. Hence two interfaces should be provided, an SQL type interface and a menu type interface.

The result of the environment analysis meant that a help system had to be developed. This process produced a design which provided a separate help system for the SQL type interface and the requirement that all information would be available on the screens in the menu type interface.

The system was then prototyped and evaluated with a number of users. During the evaluation several user characteristics were examined to determine what affect they may have on the interaction. It was known, from the user analysis, that users would vary in terms of their frequency of use of the proposed systems and their previous experience. Hence performance using the system was correlated with users level of experience. Additionally, a measure of users' spatial ability was taken. Previous research into individual differences in cognition (Dillon and Schmeck, 1983; Dillon, 1985; and Sternberg, 1985) and individual differences in HCI (Egan, 1988; see also Benyon, 1993) suggested that spatial ability (i.e. an individual's position on a standard psychometric test which involved the mental rotation of objects) was an important factor in determining performance in information retrieval tasks (Egan, 1988). It was felt important in this case that such cognitive differences should be examined.

The following additional requirements were identified as a result of the further analysis.

Task Knowledge Analysis	Revealed that a group of users had trouble using the command language system and that this was related to the users spatial ability and their level of experience with command languages. Users who had *both* a low spatial ability *and* no experience of command languages performed less effectively (on average they made more errors and took longer to complete a task) than the other users (those with high spatial ability or those with low spatial ability and high or low experience).
Functional Analysis	There is a need to edit commands to save having to re-type long queries.

The command interface was much quicker to use (average task completion time 24 s) than the menu interface (average task completion time 33 s) for 75% of users.

A simple line editor was developed for the SQL type interface and the provision to go backwards through the menu screens was provided for the menu interface.

As a result of this analysis, the following system requirements were determined.

7.2. System Requirements

System requirements may be described in terms of functional, data and usability requirements.

7.2.1. Functional Requirements

The system supports three tasks; 1. Process Query, 2. Change criteria, 3. Get help. These map onto the logical functions (task 1 maps to logical functions 1.1, 1.2, etc.) and physical functions as shown in Table I.

7.2.2. Data Requirements

Data on staff and students.
Data for help system.
Data on previous query.

7.2.3. Usability Requirements

Table II, adapted from the ideas presented in (Whiteside et al., 1991; and Shackel, 1990), describes the usability attributes relevant to this application, given the original requirements that the system should be a 'walk-up-and-use' system. The 'Effectiveness' and 'Attitude' attributes were derived from prototyping the system. The 'Learnability' and 'Flexibility' attributes are included for completeness even though they did not form part of this study.

7.3. Meeting the Requirements

Apart from the details of physical design decisions (such as the actual layout of screens, the file design, the implementation of the system, etc.), these requirements raise a major *logical* design decision. Since there is a requirement for two interfaces, a necessary task which needs to be performed is to select the menu or command interface. Should this task be allocated to the system

TABLE I
Logical/physical mapping of system functions

Logical Level	Physical Level	
	Menu interface	Command language interface
1.1 Select File	select option 1 or 2 from screen s1 press <enter>	Enter Select <attributes> from <file> where <selection criteria>
1.2 Specify required attributes	select option 1 or 2 from screen s2 press <enter>	
1.3 Specify selection criteria	select option 1, 2, 3 or 4 from screen s3 press <enter>	
1.4 Get data	press <enter>.	press <enter>
2.1 Edit File	P = get previous screen F = go to first screen	b moves cursor backwards f moves cursor forwards
2.2 Edit attributes		
2.3 Edit selection criteria		
3. Get help	all help information to be included on the displayed screens	type hlp <enter> use arrows to scroll type quit <enter> to return to database system

or should the user make the selection? The experimental results[1] suggested that users with a low score on the spatial ability test ('low spatial ability') and no command language experience would perform badly using the command interface and hence should be provided with the menu interface. However, since users cannot really be expected to know what their spatial ability is they are not in a position to make an appropriate selection. Furthermore, the only way of determining users' spatial ability scores is to test them. But because the spatial ability test takes 20 minutes to administer, this would clearly cause other usability problems. The decision to allocate the selection of an appropriate interface to the system (an 'interface selection agent') raises another usability issue. If the system suddenly changes the interface, users may be

[1] These results are discussed in Jennings et al. (1991) and Jennings and Benyon (in press).

TABLE II
Usability requirements

Attribute	Measuring Concept	Measuring Method	Planned Level
Learnability	Ability to enter required query successfully	Time taken	less than 2 minutes
Effectiveness–1	number of errors	proportion of errors to task	For all users <= 1 error in 12 tasks
Effectiveness-2	processing speed	average time taken to complete task (ATCT)	For all users – average task completion time < 25 secs on command interface, < 35 secs using menu interface
Attitude–1	Attitude questionnaire	overall score	score > 80%
Flexibility–1	environmental changes	No measure required system is already flexible.	
Flexibility–2	task changes	No flexibility with respect to tasks.	

TABLE III
Additional usability requirements

Attribute	Measuring Concept	Measuring Method	Planned Level
Effectiveness–3	selection of interface	interface selection agent	system selects interface
Attitude–2	feeling of control	ask users	users may override system's suggestion of interface

perturbed.

Hence, taking these considerations into account, some further usability requirements arise (Table III).

7.4. Design of the Adaptive System

The purpose of the adaptive system is simply to select the more appropriate interface for the users in order that the system can meet its usability requirements i.e. to make fewer than 1 error/12 tasks and to maintain the average task completion time (ATCT) at < 25 seconds using the command interface and < 35 seconds using the menu interface.

The adaptive system only needs to know enough about the domain in order to provide the required functionality. In this case, the experimental evidence suggested that the number of errors made by users using the command interface correlated significantly with a combination of two user characteristics; the level of their spatial ability and command language experience. In turn,

TABLE IV
Usability requirements

Level	Description	Attribute Name	Values
Task	A Task is a successful completion of a query	tasks	{1...N}
Logical	An error is defined as; an incorrect formulation of a query (including specifying attributes in the wrong place, etc.) a missing or incorrect operator (such as <, > etc.) an inappropriate command (e.g. typing 'select....' when in the help system)	errors	{1...N}
Logical	The average task completion time is calculated as the total time to complete a block of 12 tasks divided by 12.	ATCT	{1....N} seconds
Physical	An interface is a coherent style of interaction	interface	{menu, command}

command language experience is affected by the frequency of computer use. Hence the relevant user characteristics required for the user model component of the adaptive system are spatial ability, command language experience and the frequency of system use. The relevant aspects of the interaction which the adaptive system needs are knowledge of the number of tasks the user had completed using the command language interface and the number of errors which they had made.

The adaptive system component of the application can be described in terms of the architecture outlined in section 5.

7.4.1. Domain Model

The domain model in this case consists of four attributes, one at the task level, two at the logical level and one at the physical level of description (Table IV).

TABLE V
User model for the adaptive system

Model	Attribute Name	How Obtained	Values	Initial values
Cognitive	spatial ability	inferred from the interaction (see inference rule 1)	{high, low}	high
Profile	command experience	1. by asking user (mandatory) 2. inference rule 1	{high, low, none}	null[2]
Profile	Frequency of computer use	by asking user (mandatory)	{frequent, occasional}	null
Student	Tasks	from dialogue record	{1...N}	null
Student	ATCT	from dialogue record	{1...N} seconds	null
Student	Errors	from dialogue record	{1...N}	null
Student	Interface	from dialogue record	{menu, command}	command

[2] null indicates the absence of a value.

7.4.2. User Model

The user model contains data on the user's personal profile, cognitive characteristics and the student model (Table V). The student model inherits all the attributes from the domain model and is updated from the dialogue record. Initially it is assumed that users have a high spatial ability and will be presented with the command interface. Mandatory attributes have to be provided before the user can use the database. An important aspect of defining the user model is to establish how the particular data will be obtained.

7.4.3. Interaction Model

The interaction model defines the data which the system will obtain from the interaction (the dialogue record) and the inferences, adaptations and evaluations which the system can make. The inference rules exploit the data in the dialogue record to update the user model. The adaptation rules exploit the data in the user model in order to effect the changes. In this case the components are:

Dialogue Record The dialogue record records details of the number of errors made and the number of tasks completed. It updates the student model.

It consists of the data items; (Number of tasks, Number of errors).

Interaction Knowledge Base
The Interaction Knowledge base describes the inferences, adaptations and evaluations which the system can make.

Inference Rule 1
if interface = command and errors > 1 and tasks = 12
then spatial ability = low and command experience = none

Adaptation Rule 1
if spatial ability = high
then interface = command

Adaptation Rule 2
if spatial ability = low and command experience = low and computing = frequent
then interface = command

Adaptation Rule 3
if spatial ability = low and command experience = none and frequency of computer use = occasional
then interface = menu

7.5. Implementation

The system was implemented using the knowledge-engineering environment KEE^{TM}. Users are implemented as objects which can exist in the different 'worlds' of the two interfaces. The final implementation used a co-operative adaptation strategy in order to comply with the usability attribute Attitude–2. When the system detected that a user was experiencing difficulties with the command interface (inference rule 1), the user was advised that a menu interface was available. At this point the user could inspect their user model and amend it if required. Thus users were kept fully informed of the basis for the computer's recommendations.

7.6. Improving the System

The above design illustrates how the proposed architecture and enhanced analysis and design may be used to develop adaptive systems. The system

TABLE VI
Adding an evaluation rule

```
Evaluation rule 1
if ATCT > 25 and interface = command
or if ATCT > 35 and interface = menu
then Print evaluation report
```

has explicit user, domain and interaction models which gives the system a highly flexible design. The fact that this system does not actually need a user model as it stands does not detract from the utility of the approach. In this example the domain *independent* characteristic of spatial ability has been inferred from the interaction and may subsequently be used by other adaptive systems. The adaptive system learns characteristics of the system with which it is interacting.

As it stands, the system is a fairly simple, rule-based adaptive system. However, it is straight forward to see how this system can be developed into a higher level adaptive system.

The move to a self-regulating system requires the system to monitor its own success. To do this an evaluation mechanism may be added. This simply alerts the designer to the fact that the system is not achieving one of its objectives (Table VI).

If the system is to be a self-mediating system then it needs to include an explicit statement of its own rationale, i.e. it needs to evaluate the adaptations against a *model* of the interaction. This model can be realised in a number of ways, but perhaps the clearest is to include specific propositions on which the design rationale is based. Two propositions could be included in the (logical level) of the domain model. Proposition P1 represents the belief that users make fewer errors using the menu interface. Proposition P2 represents the belief that the user uses the interface which is offered by the system. Another evaluation rule (evaluation rule 2) can be included in the interaction knowledge base which makes the justification for the selection of the menu interface explicit (Table VII).

A self-modifying system has access to its knowledge and has the ability to amend the representations – of user, of domain or of the interaction – if necessary. For example, at present the system only adapts if the user's command experience is none. If it transpires that users with a low command experience

TABLE VII
Adding justification for the adaptation to the system

proposition P1 := Errors with Interface = menu < errors with interface = Command
proposition P2 := If interface = X, THEN user uses Interface = X
Evaluation rule 2
If spatial ability = low and command experience = none and frequency of computer use = occasional
then (since P1 and P2) interface = menu

TABLE VIII
Illustrating adding a rule to the interaction knowledge base

Evaluation rule 3
If interface = command and errors> 1 and tasks = 12
and command experience = low and spatial ability = low
THEN ADD Adaptation rule
if spatial ability = low and command experience = low and computer use = occasional
then interface = menu

are experiencing difficulties with the command interface, the system could add a new rule (Table VIII).

8. Conclusions

Usability is concerned with achieving a harmony between users, tasks, environments and the system. It will be improved if designers pay attention to the options which are available. One of these is to develop adaptive systems.

Adaptive systems can improve usability if it can be shown that without the adaptive capability, the system would have performed less effectively. For example, if we can determine characteristics of users (of which they themselves may be unaware) which affect the interaction and if these can be measured reliably and used efficiently to alter the interaction then usability can be improved. These factors may be factual knowledge which a user may understand badly or misunderstand completely (e.g. some concept such as the concept of a paragraph in a word processing package or some function such as *rm* in Unix), cognitive factors such as the user's level of spatial ability, their preferred learning style or field dependency. Alternatively, they may be personal 'profile' characteristics such as previous experience, age, gender or job requirement. It is likely that in many cases, the relevant user characteristics will be a combination of these. In the example given here, a formative evaluation of a database system revealed that three user characteristics – two profile (frequency of computer use and command language experience) and one cognitive (spatial ability) – were significant in affecting the ability of users to employ successfully a particular interface.

The development of adaptive systems must be seen within the framework of developing human-computer 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 NL 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. Using the architecture described in this paper provides a reference model by which alternative systems and designs can be compared.

As with usability criteria, it is important that the scope of the adaptive mechanism of any system is carefully and precisely specified. Using the architecture described here, this can be achieved. Browne et al. (1990) recommend the use of adaptive system metrics alongside usability metrics. This paper has presented a comparable formulation which places adaptive systems in the context of usability. Usability metrics identify the objective of the adaptive system (in the example, usability attribute Effectiveness–3) and the impact on implementation (Effectiveness–2). The adaptation mechanism (Adaptation rules) identifies what triggers the adaptive capability. The assumption on which the adaptive mechanism is based is specified in the system requirements and the domain model identifies the recommendation underlying the adaptive mechanism. The generality of the system is defined by the characteristics contained in the user model since the user model identifies all the characteristics of the users which have been considered for the adaptive mechanism.

There are many reasons for not having an adaptive capability in a system, but it is short-sighted of designers to reject this option until it has been considered alongside other design options. The recognition that an adaptive capability may be desirable leads to an improved systems analysis and design. Adaptive systems as solutions to usability issues are here to stay.

References

Barnard, P.: 1987, 'Interacting Cognitive Subsystems'. In: J. M. Carroll (ed.): *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction.* MIT Press.

Benyon, D. R.: 1990, *Information and Data Modelling*, Blackwell Scientific Publications, Oxford.

Benyon, D. R.: 1992, 'The Role of Task Analysis in Systems Design'. *Interacting with Computers*, **4**(1), 102–121.

Benyon, D. R.: 1993, 'Accommodating Individual Differences through an Adaptive User Interface'. In: M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski (eds.): *Adaptive User Interfaces – Results and Prospects.* Amsterdam: North-Holland.

Benyon D. R. and D. M. Murray: 1993, 'Applying User Modelling to Human-Computer Interaction Design'. *Artificial Intelligence Review* **6**, 43–69.

Browne, D. P., P. A. Totterdell, and M. A. Norman: 1990, *Adaptive User Interfaces.* London: Academic Press.

Catterall, B. J., B. C. Taylor, and M. D. Galer: 1991, 'The HUFIT Planning, Analysis and Specification Toolset: Human Factors as a Normal Part of the I.T. Product Design Processing'. In: J. Karat (ed.): *Taking Software Design Seriously.* London: Academic Press.

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

DeMarco, T.: 1979, *Structured Analysis, System Specification.* Prentice-Hall.

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

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

Dillon, R. F.: 1985, *Individual Differences in Cognition Volume 2.* London: Academic Press.

Dillon, R. F. and R. R. Schmeck: 1983, *Individual Differences in Cognition Volume 1.* London: Academic Press.

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: M. Helander (ed.): *Handbook of Human-Computer Interaction.* Elsevier-Science, 1988.

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

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

Jennings, F. and D. R. Benyon: (in press) 'Database Systems. Different Interfaces for Different Users'. Available as Computing Department Report. Open University.

Jennings, F., D. R. Benyon, and D. M. Murray: 1991, 'Adapting Systems to Individual Differences in Cognitive Style'. *Acta Psychologica* **78** (1–3), 243–256.

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

Macaulay, L. A., C. J. H. Fowler, M. A. R. Kirby, and A. F. T. Hutt: 1990, 'USTM: A New Approach to Requirements Specification'. *Interacting with Computers* **2** (1), 92–108.

Mason, M. V.: 1986, 'Adaptive Command Prompting in an On-Line Documentation System'. *International Journal of Man-Machine Studies* **25**, 33–51.

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

Pylyshyn, Z. W.: 1984, *Computation and Cognition.* Cambridge, Ma.: MIT press.

Shackel, B.: 1990, 'Human Factors and Usability'. In: J. Preece and L. Keller (eds.): *Human-Computer Interaction.* Hemel Hempstead, UK: Prentice Hall.

Spark Jones, K.: 1988, 'Realism about User Modelling'. In: A. Kobsa and W. Wahlster (eds.): *User Models in Dialog Systems.* Berlin: Springer-Verlag.

Sternberg, R. J.: 1985, *Human Abilities. An Information Processing Approach.* New York: Freeman and Co.

Thimbleby, H.: 1990, *User Interface Design.* Wokingham: Addison Wesley.

Whiteside, J., J. Bennett, and K. Holtzblatt: 1988, 'Usability Engineering: Our Experience and Evolution'. In: M. Helander (ed.): *Handbook of Human-Computer Interaction.* Elsevier-Science.

Author's Vita

David Benyon is a Lecturer in Computing at the Open University, U.K. He received his B.A. (Hons.) in Mathematics and Politics from the University of Essex in 1974 and his M.Sc. in Computing and Cognition from the University of Warwick in 1983. His main research interests are in human-computer interaction (HCI), particularly in the application of knowledge-based techniques to HCI, and information systems design. He has obtained two grants from the National Physical Laboratory (NPL), U.K. for work on user modelling and adaptive systems where he worked closely with Dianne Murray. The work reported here arose out of the NPL funded research.