Approximate Modelling of Cognitive Activity with an Expert System: A Theory-Based Strategy for Developing an Interactive Design Tool*

P. BARNARD, M. WILSON AND A. MACLEAN
MRC Applied Psychology Unit, 15 Chaucer Road, Cambridge CB2 2EF

This paper outlines an approach to 'user modelling'. In the approach, constructs from theoretical psychology are used to decompose the representational and processing resources of human cognition. This decomposition supports a form of 'cognitive task analysis' through which user performance can be related to the underlying functioning of their cognitive mechanism. Such functional relationships have been formalised and embodied in an expert system. This builds approximate models which describe cognitive activity associated with the execution of dialogue exchanges in human–computer interactions. Attributes of these 'cognitive task models' are used to derive likely properties of user performance. This paper describes two examples of working knowledge bases and discusses their properties.

Received May 1988

1. INTRODUCTION

It has often been emphasised that the study of human-computer interactions is essentially a multi-disciplinary enterprise. Many different disciplines can, and do, contribute to the evolution and enhancement of user interfaces. As a technology that can support and extend our capabilities for representing and manipulating information, the behavioural and cognitive sciences are an obvious source of potential insight concerning the nature and problems of user–system interactions. Indeed, these disciplines turned their attention to researching human–computer interactions at a very early stage in the history of interface design and development.[8] Most of those concerned had some broader aims concerning the potential of behavioural concepts and methods within the broader multi-disciplinary context.

Many concentrated on the more traditional aims of 'human factors'. The measurement of user performance with an appropriately sampled population of users could help choose among alternative interfaces on a firmer foundation than decisions based solely on designer 'intuition'.[12] Others were concerned primarily with using behavioural evidence to formulate systematic guidelines for interface design.[18] Various forms of conceptual analysis offered potentially more powerful alternatives to traditional emphases on empiricism and guidelines. These included the use of formal methods to represent interface designs in a manner that would reveal their overall complexity;[17] the use of psychological assumptions to predict how long it would take users to perform tasks,[7] or learn how to perform them.[10] Yet others argued that a deep psychological understanding of cognitive issues could help create new forms of interface design that would help to avoid problems encountered with earlier designs.[11]

These more analytic approaches to researching human–computer interactions each express some view of the relationship between user cognition and performance in complex tasks. They also tend to be part of longer-term efforts to develop applied theories, tools and techniques, rather than a part of shorter-term efforts to evaluate specific products. Rather than simply accumulate empirical facts about behaviour with particular interfaces some broader synthesis is sought. Two key questions underlie much of that research. How should we understand and model the user?; and how should that understanding or modelling be applied in the process of system development? These questions touch on a wide range of theoretical and practical issues in both cognitive and computer science. As such, it is not particularly surprising that there are almost as many ways of elaborating the questions and framing their answers as there are active research groups and practitioners.

In spite of this diversity, three themes run through much of the literature on user modelling. The first concerns approximation. User perception, cognition and action are clearly knowledge-intensive and complex. Likewise, system design is itself a complex and intricate process, involving many factors and trade-offs. To render user considerations manageable and compatible with the real demands of system design and development, approximation is desirable in the analysis of user behaviour and any modelling based upon it.[7]

The second theme concerns the explicitness of approximations. For analysis and modelling techniques to be transferred from research to design communities, the elements of the analysis and the principles on which it is based must be made explicit. Without this, there is little hope that approximate models can be applied systematically and consistently. Obviously, to be of continuing value they must also capture and predict important behavioural consequences of design decisions in an accurate way. This requires knowledge of how users actually behave with real or experimental systems. Thus the third theme is that such models must have a firm foundation in the empirical phenomena of system use.

This paper outlines a particular form of approximate user modelling and offers one class of answer to the question about how that modelling can be applied within the process of system design and development. It

* This paper is a considerably extended version of material originally published as 'Approximate modelling of cognitive activity: towards an expert system design aid', which appeared in *Proceedings CHI+GI'87, Human Factors in Computing Systems and Graphics Interface*, edited J. Carroll and P. Tanner, pp. 21–26. ACM, New York.

therefore represents a contribution from the more analytic perspective with a relatively long-term objective. At its heart is the idea that theoretical concepts and principles from cognitive psychology can be codified to describe the mental activity that occurs during task execution. Principles can be justified from empirical research and the codification made explicit in an expert system. This enables a rule system to make performance predictions by reasoning about cognitive representations and their processing.

In its conceptual content, this approach will differ markedly from models that are made explicit through approximate calculations concerning user actions;[7] or through extensive simulation of the how-to-do-it knowledge on which those actions rely.[10] First, explicitness will be achieved by formalising rules for building **descriptive** models of cognitive activity together with rules for associating those descriptive models with textual descriptions of the probable properties of user behaviour. Secondly, the descriptive models will seek to integrate over properties of a system design and properties of user cognition. The models will describe not only what a user must **know** in order to perform a task but also **how that knowledge is likely to be manipulated within the human information-processing mechanism** as it seeks to control and co-ordinate behaviour during task execution.

The approach will also have different consequences for its potential application. The expertise required to construct descriptions of cognitive activity will effectively be built into the knowledge base of an expert system. This offers the longer-term prospect of design aids which do not require their users to have extensive how-to-do-it knowledge of user psychology or the modelling process itself. Rather, a behavioural analyst can approach the system with some design issue in mind. The system will elicit from the analyst just that information it needs to build a model and will output a textual report.

In this way many of the difficulties associated with accessing, understanding and applying guidelines can be circumvented.[1, 14, 19] Unlike simulation methodologies, system specifications will not have to be embodied in the underlying codification. These specifications will be treated as **data** for the model-building rules to interpret. As such, it should be possible for the behavioural analyst to explore varying designs for their potential behavioural consequences on a time scale compatible with development cycles.

In the following sections, some preliminary exercises are described. These exercises are of strictly limited scope. However, it is hoped that they will illustrate the longer-term possibilities for an expert system design aid based upon, and deriving predictive power from, cognitive theory.

2. A THEORETICAL FRAMEWORK OF COGNITIVE RESOURCES

In order to describe cognitive activity, it is necessary to have a theoretical base which specifies the underlying resources of human information processing. That utilised here is the framework of Interacting Cognitive Subsystems developed by Barnard.[2] This defines a modular architecture for human information processing in which subsystems operate in specific domains of processing (see Fig. 1). There are subsystems for sensory (Acoustic, Visual) and effector (Limb, Articulatory) processing. There are also representational subsystems which specialise in processing higher-order encodings of linguistic (Morphonolexical [MPL]) and visual form (Object), as well as those which represent and process meaning and its implications (Propositional and Implicational).

Subsystems have the same internal organisation. This involves three kinds of resource. First, each subsystem includes an Image Record. This preserves a non-decaying record of all representations input to the subsystem. These are laid down by the second type of resource, a COPY process. The third type of resource involves processes which actively re-code information from one kind of mental code to another. Thus, within the propositional subsystem, the PROP → MPL process takes a propositional representation and translates it into a surface-structure representation of linguistic form. This process, of course, forms a key stage in normal language production.

The operation of this architecture is constrained. Its detailed properties and principles of operation have been outlined elsewhere in the context of laboratory psychology,[2] and in the context of dialogues for human–computer interaction.[3] The objective here is to create approximate, rather than exact models of cognitive activity. For this purpose, the functioning of resources needs to be captured at a 'molecular' rather than 'atomic' level of description. Given basic definitions of individual resources, that description can be organised around four concepts.

It is assumed that overt behaviour calls upon **configurations** of resources. A configuration effectively specifies how information flows from one subsystem to another by identifying the particular classes of mental transformation involved. Thus, reading text on a VDU (see Fig. 1) requires a sequence of mental processes to translate from sensation to visual form (VIS → OBJ); from visual to linguistic form (OBJ → MPL); from linguistic form to meaning (MPL → PROP); as well as any consequential inferential processing (PROP → IMPLIC). Another subset would be required to process non-linguistic 'icons'. Yet others would be required for keystroking text or controlling a mouse. The performance expected of the complete system will naturally depend upon the precise **configuration** of resources called into play. The architectural framework shown in Fig. 1 provides a basic definition of the set of processing resources from which configurations can be selected according to certain information-processing restrictions. So, for example, any given mental process can only do one thing at a time. It cannot simultaneously be a part of two configurations.

Within configurations, a translation is carried out by each process. Hence, cognitive performance will also be constrained by the **procedural knowledge** required to perform that translation. With well-practised skills, such as natural-language comprehension, the knowledge required to effect a particular mental transformation would be fully proceduralised. However, without appropriate **procedural knowledge**, a process may be unable to perform automatic re-coding of representations required for a specific task. So, for example, the PROP → MPL process takes as input a semantic specification of the meaning of an utterance, and outputs a sequence of lexical items in an appropriate grammatical form. Were

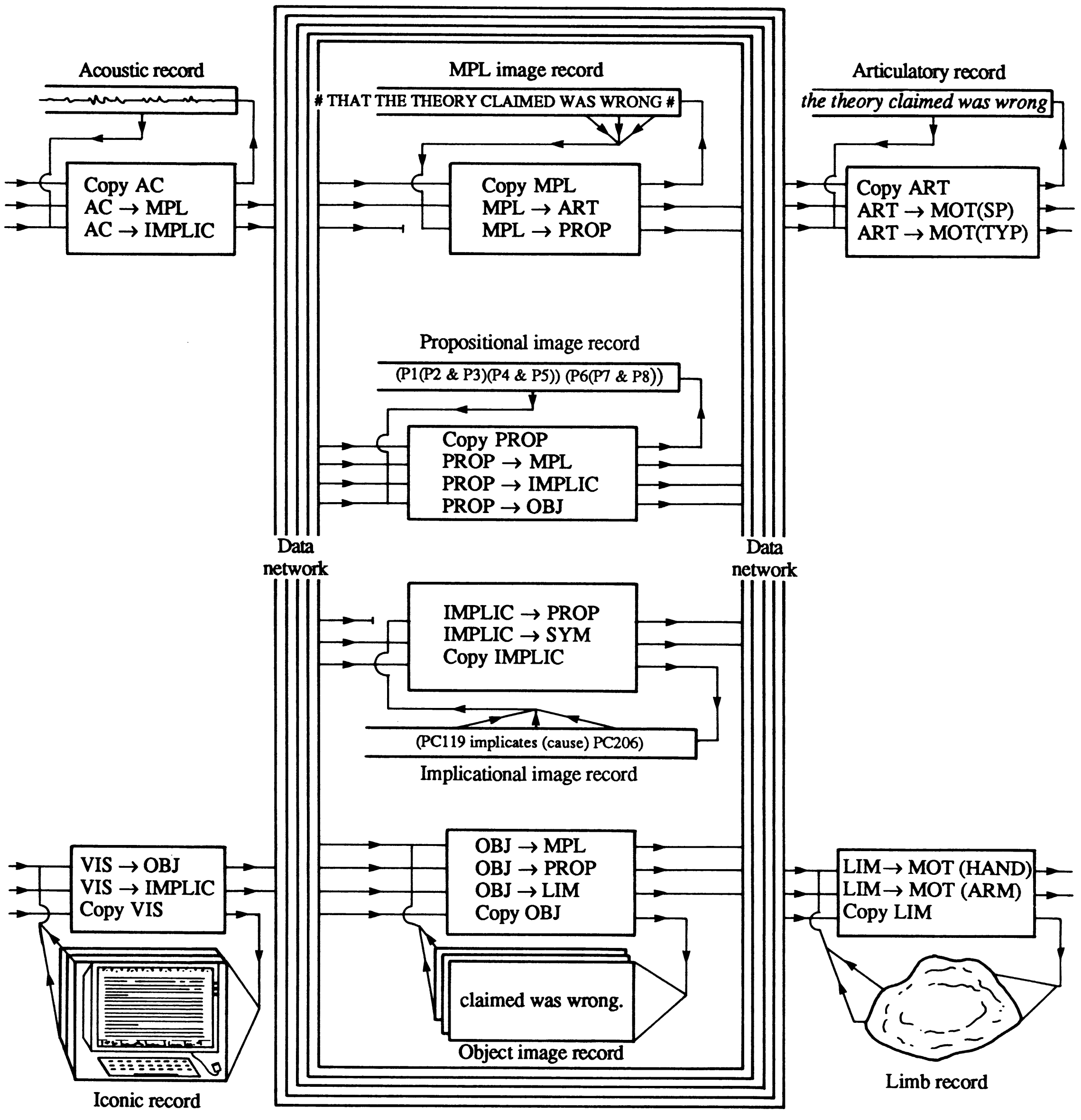


Figure 1. An architecture for human information processing (from Ref. 3).

the individual to be a novice speaker of a second language, say French, both the form and content of their utterances would be constrained by the knowledge that had so far become proceduralised within the PROP→ MPL process. So, for example, that novice speaker might have a representation of the meaning corresponding to a 'ring' but be unable to generate the appropriate French word (*bague*). Within a well-known version of BASIC, users may know that they want to exit to do something else, but they may not know, or be able to infer, that the appropriate command is 'SYSTEM'.

As with the learning of a foreign language, so the learning of the kinds of dialogue employed in human–computer interaction will typically require users to acquire new forms of procedural knowledge, often with specific attributes or characteristics. The **presence or absence of procedural knowledge** as well as its **specific attributes** will influence the precise pattern of activity within the overall cognitive mechanism, and hence the overt behaviour exhibited by a user.

During task execution, processing activity within the cognitive mechanism does not only require appropriate procedural knowledge. It may also require the recovery and use of declarative representations stored in image records. Individual records may, for example, represent the sequence of commands and parameters required to send electronic mail to a colleague. Likewise, they may preserve representations of the contents of text files, their names and the directories in which they are located. In the course of task execution many such records may need to be accessed and their contents recovered. Records preserved in the various subsystems of cognition will have different form and content. Speech-based representations will have different properties from representations of visual objects and so on across the full range of cognitive subsystems. The characteristics of these **record contents** and their recoverability constitute a further class of constraint likely to influence user performance.

Finally, all resources within the framework of interacting cognitive subsystems fulfil specific functions. There are no general-purpose resources such as 'a central executive' or 'working memory'. The entire system is considered as a distributed network that is self-controlling via representations passed from one subsystem to another. To generate and control overt behaviour, individual resources must act together in a co-ordinated way. The concept of **dynamic control** involves characterising the passage of representations among subsystems. With simple, well-learned tasks **dynamic control** may be straightforward. With complex or novel tasks, many transactions among subsystems and their image records may be required to co-ordinate overt action. These transactions may include multiple access to image records and the creative combination of their contents.

The fundamental assumption is that a description of cognitive activity can systematically be related to a description of overt behaviour. This will be some complex function (F_n) of the four classes of constraint:

Properties of Behaviour $\Leftarrow F_n$ (Process Configuration and Procedural Knowledge and Record Contents and Dynamic Control).

This forms a basis for cognitive task analysis through which a theoretical decomposition of cognitive resources can be related to performance. Cognitive task analysis examines the representational and mental processing demands of using particular interface features and relates these to patterns of user performance obtained in experimental research. The aim of the analysis is to infer rules interrelating properties of: process configurations, procedural knowledge, record contents and the dynamic control of the complete cognitive mechanism. Several examples are dealt with at some length in Barnard.[3] Armed with systematic principles that are specified in terms of abstract properties of human information processing, it becomes possible to build approximate models that describe cognitive activity in terms of these abstract properties. With the addition of rules that relate these models to characteristics of user performance, the underlying principles can be generalised to novel task and interface settings, The whole modelling process should thus have predictive properties.

3. DEVELOPING A FAMILY OF COGNITIVE TASK MODELS

The complexities of human cognition cannot be expected to yield to a very simple-minded analysis of this form. The function outlined above relates primarily to approximation over activity that occurs within the very short-term dynamics of human cognition. The time frame over which principles at that level might apply can be expected to be very short. Interactions with a computer require the cognitive mechanism to be co-ordinated to control extended sequences of behaviour. Any unified sequence of behaviour is likely to involve a number of types or phases of cognitive activity.[15] So the correction of material during text editing may require reading existing sentences, pointing to and selecting a 'delete' command using a mouse, and the keystroking of new text. Each phase will recruit a different configuration of processing resources. These phases will require different but interrelated descriptions.

Alternative configurations may also be recruited to support the same overt actions. Novices engaged in creative composition may use one configuration to control keystroking and expert copy typists another. Similarly, for a novice, the descriptions of **procedural knowledge** and of **record contents** will be different from those for an experienced user. As learning progresses users will not only have potential access to a larger range of relevant memory records, but their response patterns will increasingly acquire the characteristics of fully proceduralised knowledge. Thus, in order for the modelling approach to have significant scope, it becomes necessary to approximate over the short-term changes involved in the co-ordinated control of sequences of action and over the longer-term changes that occur as a result of learning and experience. It is therefore appropriate to consider families of interconnected models for the different phases of activities and for different levels of user expertise (see Fig. 2). This general idea is obviously extensible to include both alternative strategies for coping with a task, and individual differences within a particular class of user population. Variation in the descriptive models should nevertheless be related systematically to varying attributes of user behaviour.

The building of such models was tackeld by performing cognitive task analyses on experimental evidence.[3] This evidence focused upon variations in the style, structure and content of user–system dialogues. Stable performance effects were interpreted in terms of patterns of cognitive activity that could have given rise to them. The exemplars examined covered the use of command language and menu system interactions. As foreshadowed earlier, candidate principles were defined as heuristics relating **process configurations, procedural knowledge, record contents** and the **dynamic control** of the complete cognitive mechanism. The full range of heuristics covered both the model-building process and the use of the models to produce performance predictions.

Knowledge bases embodying these principles have now been developed within an expert system shell. Example principles and heuristics are elaborated in later

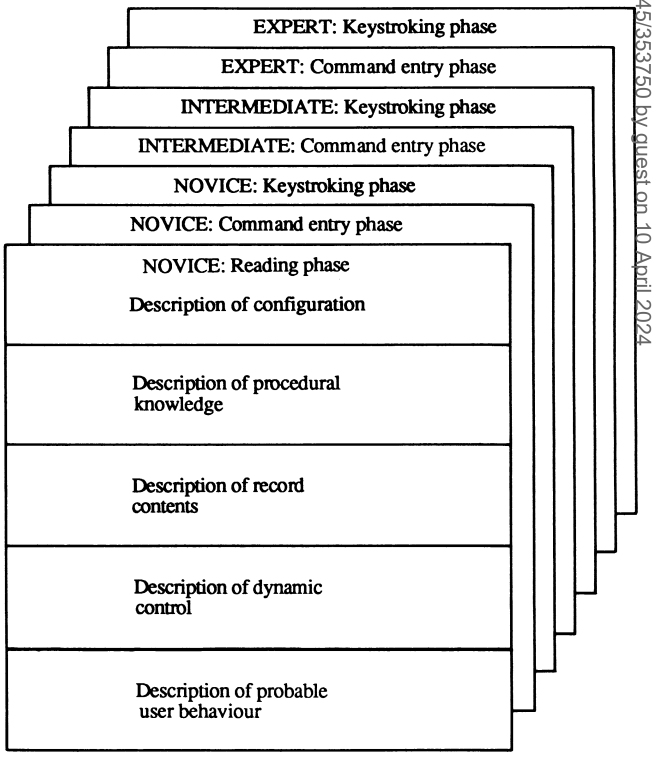

EXPERT: Keystroking phase
EXPERT: Command entry phase
INTERMEDIATE: Keystroking phase
INTERMEDIATE: Command entry phase
NOVICE: Keystroking phase
NOVICE: Command entry phase
NOVICE: Reading phase
Description of configuration
Description of procedural knowledge
Description of record contents
Description of dynamic control
Description of probable user behaviour

Figure 2. A family of cognitive task models (from Ref. 6).

sections. These sections will illustrate the application of the approach to the naming of commands and for organising commands into extended sequences. The following section provides an overview of the general characteristics of the expert system and the way in which rules, facts and answers to questions interrelate to govern the construction of cognitive task models.

4. AN OVERVIEW OF THE EXPERT SYSTEM IMPLEMENTATION

The knowledge bases have been developed within Expertech's XI/Plus shell. This is a commercially available system which runs on the IBM PC. A basic description of the shell language will be presented as a background for understanding subsequent examples.

The shell is composed of four major parts: a database (working memory), a goal agenda, a knowledge base and the inference engine. The knowledge base uses production rules to represent knowledge. A production rule has the form: **if ⟨condition⟩ then ⟨action⟩** statement. When the conditions are true, the actions are performed. The order of search of production rules is controlled by the goal agenda. This is established by statements in the knowledge base and can be modified by the actions of production rules. The achievement of the goals on the agenda can be controlled by the inference engine using both backward- and forward-chaining logic.

Actions and conditions in production rules can either operate on items in the database, operate on the goal agenda, act as commands to the shell to load new knowledge bases, or produce output reports. Statements in the database have the form **⟨identifier⟩ ⟨relation⟩ ⟨value⟩**. For example, in the case of **number of commands = 20**, 'number of commands' is an identifier, '=' is a relation, and '20' is the value. Relation can be numerical or logical. The logical relations used in the examples cover hierarchical class relations (**is a**), set membership (**includes**), and simple variable assignment (**is**).

Control of backward-chaining logic is performed by placing goals on the agenda and examining that agenda to see if they have been achieved. Items are placed on the agenda by the 'check' command, and goals are tested by a 'done' statement. There are a limited set of system commands available to permit the resetting of 'identifiers', to change items in the knowledge base, and the loading of new knowledge bases ('command load ⟨knowledge-base-name⟩'). In addition, other rules output reports to the user (e.g. **report ⟨report text⟩**). These reports can be directed to a file or an output device (e.g. the screen).

For a rule condition to be satisfied, it must match a statement in the database. This can come about in three ways. First, the item can appear in the database as a consequence of a production rule firing. Secondly, it may simply be declared as a fact in a knowledge base. For example, a particular fact might specify that 'the natural language of the user population is English'. The third method by which a condition can be satisfied is by the shell asking the user a question.

Although the shell has the ability to generate questions, those used below were all explicitly defined in the knowledge bases. Individual questions are presented to the user in panels on the display together with a menu of answers. The user can select answers with the cursor keys. The possible answers may include the option 'unknown'. If this option is selected, a default will be assumed by the system. The shell can accept one or more answers, depending on the type of identifier being queried. A single-value identifier will accept only one answer, a multi-value identifier will accept as many as the user wishes to select. A help key will deliver the user extra screens of information on the content of the questions and how to answer them. Users can also request information on 'why' a question is being asked. The shell will then present them with the rule whose conditions it is trying to fulfil. The user can repeatedly ask 'why' and be presented with successive rules.

The process of building a cognitive task model is governed by control rules. These determine how specific goals get placed on the agenda. In order to build two interrelated models for cognitive activity associated with the entry of command names covering a phase in which information on a display had to be understood, and a phase in which the command name had to be entered, an appropriate control rule would have the following form:

if query names
then check comprehension configuration
and check entry configuration
and check comprehension procedures
and check entry procedures
and check record contents
and check dynamic control

The control rules also test that these major goals have been satisfied. When the task model stabilises and no further rules apply, a separate knowledge base is then loaded. This contains reporting rules that map conditions in the cognitive task model held in working memory on to textual descriptions of the probable properties of user behaviour. These then appear as one or more panels on the screen.

The shell will chain backward from the major goals controlling model development. It tries to fulfil the conditions of rules whose actions may satisfy them. In doing this, the system will ask the user questions and assert information in the database. When data are asserted, the system will chain forward to any rules using these data as conditions. Rules in the knowledge base will add subsidiary goals to the agenda during this inference process which must, in turn, be satisfied.

The process of 'consulting' the tool can involve four major stages. First, a knowledge base must be loaded and a particular query initiated by selecting an item (e.g. names) from a menu. Secondly, the system will ask the analyst a series of questions which elicit relevant design features of a projected system. Thirdly, a report will be presented on the probable properties of user behaviour for that particular scenario. Fourthly, the analyst can then modify the task model to explore alternative design concepts. This modification can be done in two ways. The 'what if' capability can be recruited to alter specific statements in the database. Alternatively, the analyst can query specific identifiers and change them. Under these circumstances the system will re-ask relevant questions build a new model and issue a fresh report.

Within this general framework several knowledge bases have now been developed and their properties explored. In the first instance the intention was to keep the knowledge bases small and manageable. Each knowledge base was therefore designed to focus upon the properties of particular heuristics for descriptive modelling of cognitive activity. In the first illustration given below the heuristics primarily concern the role of **procedural knowledge** and its consequences for the dynamic control of cognitive activity. In the second example the heuristics primarily concern similar consequences of attributes of **record contents**.

Example 1. Proceduralisation and dynamic control

One knowledge base focused on the construction of cognitive task models for data on the learning of command names for text editing.[9] These data showed that command names which bore a semantically specific relationship to the underlying operation they performed (e.g. delete) were learned relatively efficiently. Abbreviations of these names based on a three-letter contraction rule (e.g. dlt) gave rise to significant performance costs. Command names that were semantically unrelated to their function in context (e.g. parole) induced a comparable level of performance cost in terms of time, errors and unrequired dialogue transactions. Novel pseudowords (e.g. ragole) of the kind often invented to cover novel concepts (cf. BYTE) and random three-letter consonant strings (e.g. fnm) incurred even greater performance costs in learning and use.

Cognitive task analysis suggested a number of modelling heuristics that could help to characterise this form of result. Briefly, the argument runs as follows. In order to acquire the vocabulary of command names that are semantically related to the meaning of the underlying operations they invoke, users are faced with a demand to update their knowledge of what those words mean in this computing context. They should nevertheless be in a position to generate the word from its meaning (PROP $\rightarrow$ MPL). Once they have generated this known word, the procedural knowledge should also be in place to generate a representation of the sequence of letters required to control keystroking (e.g. MPL $\rightarrow$ ART for a typist with moderate experience). Thus, given a representation of the intention to get rid of a lexical item (stored in an image record), they should be able to keystroke the command name relatively easily. This basic idea can then be extended to cover the general case.

(*a*) Where all processes within a configuration embody appropriate proceduralised knowledge, a phase can be co-ordinated from the contents of a single image record. Dynamic control will involve relatively few transactions among the subsystems of cognition.

With the vocabulary of abbreviations, users would not initially possess appropriate procedural knowledge for generating and articulating the abbreviated form (MPL $\rightarrow$ ART). They should, however, know how to generate the full command word (PROP $\rightarrow$ MPL). In contrast, users of the vocabulary of the semantically unrelated names would not be able automatically to generate the appropriate word from a representation of the meaning of the operation to be performed. In the absence of appropriate procedural knowledge, the user would have to access memory records, draw inferences or consult the help system. In any event, these demands would entail additional exchanges among the subsystems of cognition to control behaviour. Again in the general case we may state the following.

(*b*) Where any one process does not embody appropriate proceduralised knowledge, dynamic control will involve extra information-processing transactions among subsystems. Additional access to image records may, for example, be required to control output.

With the vocabularies of pseudowords and consonant strings, there would be demands to develop new procedural knowledge in mental processes required to generate form from meaning (PROP $\rightarrow$ MPL) and in the process controlling the articulation of keystroke activity (MPL $\rightarrow$ ART) with even more extreme consequences for the control of cognitive activity.

(*c*) Where two (or more) processes do not embody appropriate procedural knowledge, dynamic control will involve an even greater number of transactions among subsystems.

From these statements of principle, it is a relatively straightforward step to encode them in a form appropriate to the knowledge base of an expert system. Rules relating the contents of the models to user performance obviously need to be included. In the above examples, it is helpful to think of extra transactions among the subsystems of cognition as contributing to 'cognitive workload', and thereby increasing performance times and the probability of errors. For an implementation of these and other related heuristics for establishing the relevant task conditions and characterising user performance, Fig. 3(*a*) illustrates a consultation with the expert system and Fig. 3(*b*) provides a schematic representation of the evolution of a cognitive task model.

The first major goal on the agenda is to infer the appropriate configurations of processes for the task. In this example, the configurations relate to two phases: understanding the system state and issuing a command (the entry configuration). The knowledge base contains several rules which specify these on the basis of the physical properties of the system and the experience of the user population. After a series of questions (e.g. the first five questions in Fig. 3(*a*)), the system can define the appropriate configurations. These are established as two sets representing the processes involved in the different phases of the task. These configurations are, of course, the same for all the conditions in the Grudin and Barnard[9] experiment. One of the relevant rules has the following form (the 'because' part of the rule is passive and provides explanation if the analyst asks why a question is being posed).

```
   if input includes keystroke
   and keystroke entry includes string entry
   and typing experience is not expert typists
   then entry configuration includes prop_implic_n
   and entry configuration includes implic_prop_n
   and entry configuration includes prop_mpl
   and entry configuration includes mpl_art
   and entry configuration includes art_mot_t
(because) this rule establishes the entry configuration for
   (and) command strings typed into a system by a non-
                                                   expert
```

The configuration for the entry phase starts with users inferring (PROP $\rightarrow$ IMPLIC & IMPLIC $\rightarrow$ PROP) the operation to be performed (e.g. deletion). Subsequent processing includes the translation (PROP $\rightarrow$ MPL) from a propositional representation of that operation to its

(*a*)

1 How does the system communicate with users?
- speech
- **vdu**
- unknown

2 How does the user communicate commands to the system?
- **keystroke**
- direct manipulation
- speech
- unknown

3 What form of cuing does the system automatically offer the user during each dialogue transaction?
- **command prompts**
- argument prompts
- word menus
- pictorial menus
- none

4 What form do the keystroke commands take? (CNTRL characters. PF and cursor keys involve binding.)
- **string entry**
- key binding
- unknown

5 Are the users?
- expert typists
- **some experience with typing**
- no experience with typing
- unknown

6 Can it be assumed that users will be able easily to infer the nature of the operations they need to perform?
- **yes**
- no
- unknown

7 Would the exact command string to be entered be in a user's normal productive vocabulary?
- all command words
- three-quarters command words
- half command words
- a quarter command words
- **no command words**
- unknown

8 Are the meanings of the full command words in natural language clearly related to the underlying operations they invoke in the system?
- yes
- **no**
- unknown

9 Does the command to be entered conform to normal rules for word construction?
- **yes**
- no
- unknown

10 Do the letters of the command string to be entered abbreviate or otherwise represent concepts or words in the task domain?
- by abstractable ruler
- **no obvious relationship**
- unknown

(*b*)

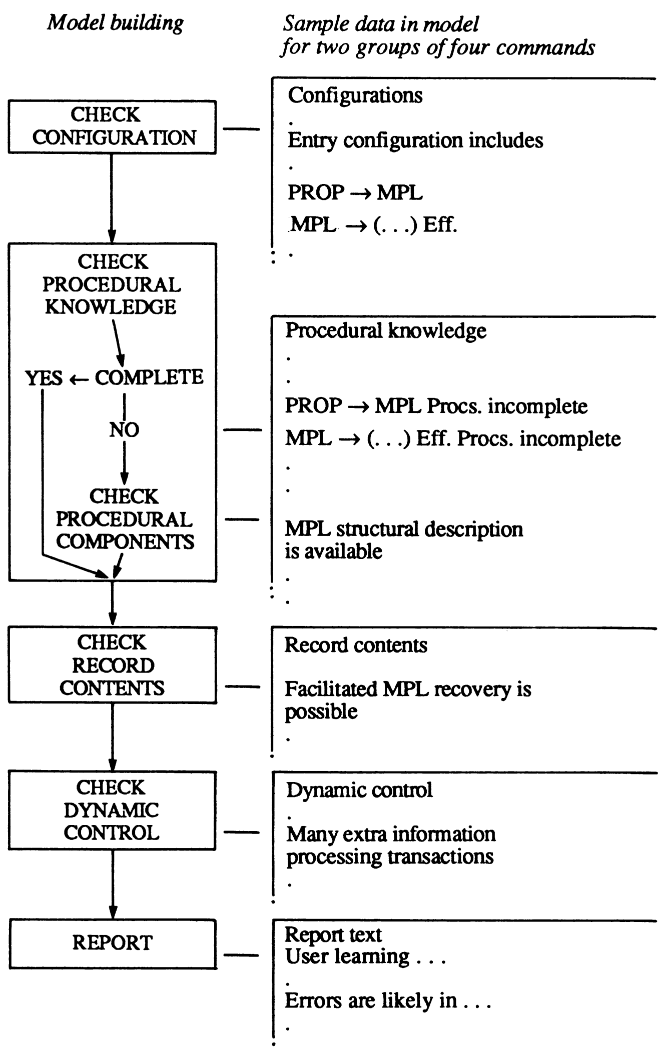

Model building
Sample data in model for two groups of four commands
CHECK CONFIGURATION
CHECK PROCEDURAL KNOWLEDGE
YES ← COMPLETE
NO
CHECK PROCEDURAL COMPONENTS
CHECK RECORD CONTENTS
CHECK DYNAMIC CONTROL
REPORT
Configurations
Entry configuration includes
PROP → MPL
MPL → (...) Eff.
Procedural knowledge
PROP → MPL Procs. incomplete
MPL → (...) Eff. Procs. incomplete
MPL structural description is available
Record contents
Facilitated MPL recovery is possible
Dynamic control
Many extra information processing transactions
Report text
User learning ...
Errors are likely in ...

Figure 3(*a*). An example dialogue for a command name set enquiry. The answers in bold are those appropriate for the pseudoword condition from the Grudin and Barnard study.[9] Figure 3(*b*). A schematic representation of the evolution of a task model for pseudowords.

lexical realisation (e.g. this might be the pseudoword 'ragole'). This in turn is converted into an articulatory code (MPL→ART) from which a motor code for generating a string of characters is derived (ART→MOTt).

Once the relevant configuration is defined, further rules invoke a series of questions designed to elicit information about the procedural knowledge embodied in each stage (e.g. questions 6–8 in Fig. 3(*a*). One such rule is as follows:

```
if the semantics for command words is yes
and ambiguity of command words is none
then entry procedures includes prop_mpl
```

```
(because) this is a condition on the full proceduralisation of
(and)     the PROP → MPL process in the propositional
                                              subsystem
```

If the conditions of these rules are fully met (e.g. the meanings of command terms are known and unambiguous) then the process under scrutiny will be marked as fully proceduralised (e.g. entry procedures includes PROP → MPL).

For the other answers proceduralisation is marked as incomplete. A range of additional rules and questions are then invoked to establish other more detailed attributes that could influence dynamic control and record access (e.g. the presence or absence of linguistic structure through questions 9 and 10 in Fig. 3(*a*)). For example, pseudowords have structural descriptions which conform to the rules of linguistic form (MPL), whereas consonant strings do not. A further heuristic assumes that a structural description will facilitate retrieval from the relevant image record. This in turn assists learning and performance. Thus, while the models built for novice performance with pseudowords and consonant strings will be very similar, the attributes describing access to record contents will start to diverge in models for intermediate users.

As the data structures for procedural knowledge and record contents evolve, the conditions for the heuristics concerning the description of the dynamic control of the whole mechanism start to be satisfied. Hence, rules of the following form will be able to fire.

```
if entry configuration includes implic_prop_n
and record contents includes will support implic_prop_n
                                          output cycle
and dynamic control includes control from single image
                                                record
then command reset extent of dynamic control
and extent of dynamic control is relatively automatic
                                                output
(because) when the system is fully proceduralised and runs
(and)     off a single image record, the dynamic control is
(and)     relatively automatic for any particular configura-
                                                  tion
```

Once no further rules concerning dynamic control apply, the task model will have obtained its final form. For Grudin and Barnard's alternative vocabularies they will contain the following key attributes. In the case of the 'specific' vocabulary the system should infer that appropriate procedural knowledge exists for both PROP → MPL and MPL → ART re-codings. Under these conditions a rule (the one illustrated above) fires to describe dynamic control as involving relatively automatic output. Few transactions among subsystems will be required to co-ordinate and control command entry.

For two other sets, a data structure marks one process as incomplete proceduralised (MPL → ART for abbreviations; PROP → MPL for semantically unrelated words). Under these conditions a different rule fires to imply that dynamic control will involve extra information-processing transactions among the subsystems of cognition. With pseudowords and consonant strings both processes (PROP → MPL and MPL → ART) are marked as incompletely proceduralised (as in Fig. 3(*b*)), and it is inferred that dynamic control will involve many extra transactions among subsystems.

The behavioural consequences of particular conditions within stabilised task models are specified in reporting rules. These are represented in a separate knowledge base, and link conditions in the task model to textual descriptions of likely attributes of user behaviour. Thus, where the task model describes dynamic control as involving few transactions among subsystems, a report describing relatively efficient learning and performance will be generated. Where additional transactions are detected in the description of dynamic control the system reports a higher level of difficulty, and where many such transactions are required an even greater level of difficulty. Where more specific conditions are detected (e.g. concerning access to particular forms of memory record) the system also advises the analyst of the likelihood of specific types of error and the kinds of help likely to be required. A full report is generated by outputting the text associated with any reporting rules whose conditions are satisfied. One such reporting rule is as follows.

```
if a_report is wanted
and extent of dynamic control is several information-
                                  processing mappings
and record contents includes facilitated recovery from
                                     MPL image record
then report 'Intermediate performance with this kind of'
and report 'dialogue is likely to remain problematic. It is'
and report 'highly probable that users (particularly'
and report 'intermittent ones) will continue to confuse'
and report 'which command name goes with which
    operation.'
and report 'Consultations of help facilities to resolve'
and report 'ambiguity are likely to remain frequent.'
and report 'Constructional errors in the formation of'
and report "command" strings are likely to reduce'
and report 'markedly with experience.'
(because) inference from Grudin and Barnard data for
(and) pronounceable pseudowords
```

Any given report is likely to be pieced together from the output of a number of such rules. However, for the present illustrative purpose, the important point is that the output is mediated by an abstract description of conditions within the cognitive task model and not directly from a description of interface properties. It is this abstract characterisation that provides the essential theoretical base for generalising to novel circumstances.

Example 2. Record contents and dynamic control

A second knowledge base for building task models was complementary to the one outlined above. It focuses on heuristics concerning the use of **record contents** rather than those for **procedural knowledge**. The illustration given here will not go into as much depth as that given in the previous section, and implemented rules will be omitted from the presentation. The purpose is to show how the same ideas can be applied to the analysis of dialogue structures. In this case the dialogue structures concern extended sequences of commands.

In the context of an electronic mailing task, Barnard *et al.*[5] compared two organisations for strictly ordered sequences of commands. In one version, sequences were

2 × 4 grouped organisation (incoming messages):
*Establish the time at which the message arrived
*Establish the originator reference of the sender
*Establish the register number for keeping a copy
*Establish the mail point of the recipient

*Mark the message with its time of arrival
*Acknowledge receipt of the message (to sender)
*Keep a copy of the message in the register
*Forward the message to the recipient's mail point

4 × 2 paired organisation (outgoing messages)
*Establish the company reference of the sender
*Attach the company reference to the message

*Establish the price of sending the message
*Debit the department sending the message

*Establish the file number for keeping a copy
*Keep a copy of the message in the fire

*Establish the destination code for the recipient
*Forward the message to the destination code

Figure 4. The two alternative types of sequence from Ref. 5.

organised into two groups of four commands. Commands fulfilling a common function (e.g. functions to 'establish' relevant data or functions that actually operated on the 'message' itself) were carried out together. In the alternative arrangement, the same commands were organised into sequences of four groups of two operations. Here, commands with a common referent were carried out together (establish destination code/forward to destination code). The full task involved handling 'incoming' and 'outgoing' mail by drawing upon twelve basic functions. Fig. 4 shows examples of the alternative sequence structures.

Sequence learning was found to be easier when commands were organised into four groups of two operations interlinked by a common referent. In spite of its overall superiority, a particular class of error was more frequent with the organisation based upon four pairs of operations. These errors involved confusions between semantically related command names (e.g. display/show).

Cognitive task analysis (again, see Barnard,[3] for a more detailed treatment) suggested that these findings could be handled by heuristics governing the representation and recovery of item and order information from Propositional and MPL image records. It is assumed that users abstract and represent the structure of the sequence in the form of a **'common task record'**. This is preserved in the image record of the Propositional subsystem. To execute the task users need to access this record and recover its contents. However, as a semantic form of encoding, propositional representations do not specify lexical form or order. These must be imposed during PROP $\rightarrow$ MPL recovery and re-coding. If there is a great deal of uncertainty or ambiguity concerning operations or their order, additional transactions among subsystems will be required to sort them out. These may involve inferential processing (PROP $\rightarrow$ IMPLIC) or the explicit recovery of mental representations of surface structure which do preserve lexical identity and order (MPL record).

With the organisation involving two groups of four command operations, the 'establish' commands must be carried out before the 'action' commands, since the latter group require information from the former group of operations. However, from a pragmatic perspective the commands within a group could, in principle, be carried out in any order. There is thus order uncertainty associated with the sequencing within a group but not between the two groups. In addition, superordinate descriptions of the two groups have a relatively abstract character – referring primarily to general classes of command operation.

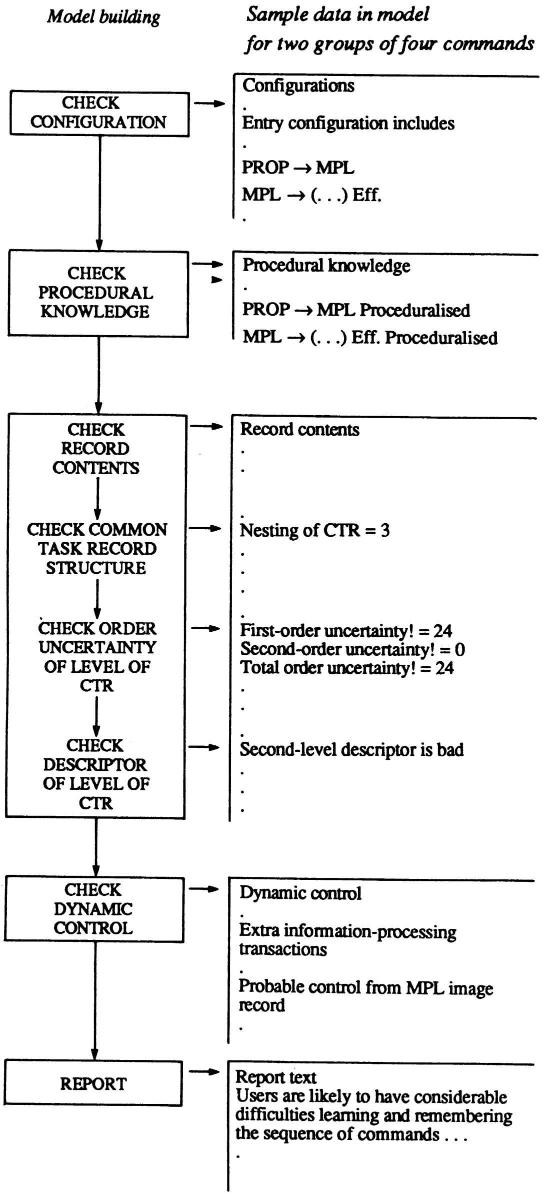
By contrast, the organisation based upon four pairs of two operations has different properties. First, the order of the two operations within a pair is effectively constrained, since the first member of the pair produces information used by the second. In principle, the pairs of operations could themselves be output in any order. In this case the uncertainty associated with ordering four pairs of two operations is lower than the uncertainty associated with the ordering of four elements within each of two groups. In addition, superordinate descriptions of four pairs of two operations are related to more concrete concepts in the task domain (e.g. company reference).

On the assumption that the recovery of information from memory records is made more problematic by increasing order uncertainty and by the presence of an abstract, rather than a more concrete, description of the particular groupings involved (effectively item uncertainty), the overall advantage of the paired organisation can readily be understood. In addition, the differential occurrence of semantic confusions can be accounted for by assuming that high levels of item and order uncertainty give rise to a relatively superficial strategy for learning the command sequence: users simply try to remember the sequence of names in a speech-based code (MPL). Where item and order uncertainties are low, users are more likely to rely on a semantic form of encoding (PROP). In effect, for the two different types of sequence organisation, mental-processing activity would have a different probabilistic distribution. Since the semantics of words like 'display' and 'show' are confusable, errors are likely to result in memory retrieval. In contrast, 'display' and 'show' are not phonologically confusable in a speech-based code, and hence the frequency of these particular confusions would be expected to decrease markedly under conditions where users adopt the more superficial learning strategy.

Attributes such as order uncertainty are also likely to be influenced by other factors. In simple list memory, for example, there are phenomena known as the primacy and recency effects. In memory retrieval, people are likely to recall the first and last items in a sequence with considerably greater accuracy than those items embedded in the centre of a list. These list-end effects also function to reduce order uncertainty in memory retrieval. However, in a directly analogous manner to the preceding section, it should not be difficult to grasp how this type of behavioural evidence can be translated into more general principles. These govern the attributes of the record contents component of the cognitive task models and influence the dynamic control of processing activity during their subsequent retrieval. As with the discussion of procedural knowledge, the principles are readily expressible in the form of general if–then rules for incorporation in a knowledge base.

In the particular expert system implementation covering the data outlined above, the configuration is again set for keystroking commands. It is also assumed that proceduralised knowledge for the entry of individual names is equivalent for both forms of sequence. The key rules concern properties of the common task record. Initially, rules fire to request basic information about the number of commands and their grouping. On the basis of the answers, the common task record is described as a hierarchical structure. This specifies both low-level nodes representing individual operations and superordinate nodes that describe their conceptual structure. The system then seeks to elaborate this description. As with the previous example, a schematic representation of the evolution of a task model is given in Fig. 5.

Using questions similar to those given earlier in Fig. 3(*a*), the system requests information about any pragmatic constraints concerning the ordering of groups and the ordering of operations within a group. On the basis of the answers, rules fire to specify attributes of order uncertainty. With the two groups of four commands, order uncertainty is high. The operations within groups could potentially have been performed in any order, but the two groups are logically ordered (establish information before acting on the mail). The converse applied with four groups of two commands. There were logical constraints within groups but only weak constraints on the order of the four groups. The system 'knows' about primacy/recency effects in retrieval from human memory and corrects for them. Here, order transpositions are likely to be confined to the two central pairs of operations. In consequence, a data structure is set that specifies a low level of order uncertainty for the four groups of two operations.



Model building
Sample data in model for two groups of four commands
CHECK CONFIGURATION
Configurations
Entry configuration includes
PROP → MPL
MPL → (...) Eff.
CHECK PROCEDURAL KNOWLEDGE
Procedural knowledge
PROP → MPL Proceduralised
MPL → (...) Eff. Proceduralised
CHECK RECORD CONTENTS
Record contents
CHECK COMMON TASK RECORD STRUCTURE
Nesting of CTR = 3
CHECK ORDER UNCERTAINTY OF LEVEL OF CTR
First-order uncertainty! = 24
Second-order uncertainty! = 0
Total order uncertainty! = 24
CHECK DESCRIPTOR OF LEVEL OF CTR
Second-level descriptor is bad
CHECK DYNAMIC CONTROL
Dynamic control
Extra information-processing transactions
Probable control from MPL image record
REPORT
Report text
Users are likely to have considerable difficulties learning and remembering the sequence of commands...

Figure 5. A schematic representation of the evolution of a task model for two groups of four commands issued in strict sequence.

The system then requests further information about the hierarchical description of the **common task record**. If superordinate nodes are poor descriptors of the command operations subsumed under them, item retrieval is likely to be problematic. If superordinate nodes are good descriptors, item retrieval will be facilitated. The rule base specifies that an abstract class of system function is a poorer descriptor than concrete reference to concepts in the task domain.

For the four groups of two commands, data structures in the task model thus specify minimal order ambiguity and good descriptors for retrieving operations. A separate rule reacts to these conditions by specifying that dynamic control can be co-ordinated from the propositional image record. Relatively few transactions will be required among subsystems to resolve item and order uncertainty. Reporting rules will then respond to these conditions, for example sequence learning should be efficient. However, since output is controlled from the semantic descriptions of the Propositional image record, command names with similar semantics will tend to be confused.

For the two groups of four operations, data structures will specify higher levels of order uncertainty and poorer hierarchical descriptors. Under these circumstances output cannot simply be co-ordinated from the propositional image record. Additional transactions among subsystems will be required. These may include frequent reference to the MPL image record to resolve command identities and their order. Words like 'display' and 'show' are semantically but not phonologically confusable. Accordingly, reporting rules will output a description of greater difficulties in sequence learning but a lowered likelihood of confusion errors between semantically related command names.

5. DISCUSSION

The approach described here took as its point of departure a theoretically motivated decomposition of the representational and processing resources of human cognition. This decomposition was then used as a basis for describing the cognitive activity that occurs during user–system dialogue. The description was 'molecular' rather than 'atomic' and involved characterising relationships among four concepts: process configurations; procedural knowledge; record contents; and dynamic control. Principles were motivated on the basis of empirical research on user–system dialogue. These were then converted into explicit heuristics and embodied in an expert system. The resulting rule sets will, for limited domains, build explicit models which specify and describe cognitive activity. These descriptions are then used by reporting rules to furnish textual descriptions of the probable properties of user behaviour.

As concept demonstrators, the knowledge bases were primarily designed to replicate the reasoning underlying the explanation of particular experimental phenomena. However, it is clear that restricted knowledge bases of this form have generalising properties. For example, to generate data structures concerning order uncertainty in sequences, the rules operated on an abstract description of the number of items and their constituent structure. These rules will fire in response to sequence descriptions other than those incorporated in the experiment that motivated them. Likewise, the general principles that concern relationships between dynamic control and proceduralisation of knowledge within a configuration have a general form. Thus, if 'icons' were substituted for lexical commands, the same rules could be applied, but over a different configuration of resources. Indeed, the knowledge base for command names was systematically extended to include a number of additional assumptions concerning the relevant configuration of processes and also concerning abstract and concrete icons. High-level rules within the system fired appropriately and invoked very general reporting rules concerning overall levels of difficulty that were originally incorporated for the lexical commands.

The expert system is itself capable of going some way down the generalising path. It can, for example, generate its own questions when it requires information. However, in doing so they draw on the language in which the rules are written. Thus, in order to maintain transparency, generalisation is better supported by incorporating additional questions and reporting rules. In certain instances this has been done by making very straightforward additional assumptions that extend a particular empirical result. In the case of order uncertainty, greater levels than those studied are assumed to be more difficult to resolve, while lower levels are assumed to be easier to resolve.

An important part in the process of approximation is played by the analyst answering questions. With the rule base for command names, the behavioural analyst must judge (or otherwise measure) the extent of name-operation ambiguity within the overall set. Likewise, within the rule base for structured sequences of command, the analyst must judge whether particular groups are based, for example, on abstract or concrete relationships, etc. This constitutes another means by which generalisation can occur and a potential limitation. Generalisation can occur because the analyst can describe novel command sets or structures in terms of their overall attributes rather than their identities. Obviously, if analysts fail to agree the reports they obtain will also vary.

It is partly for this reason that the supposed user of the expert system has been described throughout this report as a 'behavioural analyst' rather than a system designer. Although many of the questions could be easily answered by a software specialist, others require a behavioural perspective to be adopted. Hence the target user population for any future tool is, for the present, best confined to those who have some expertise in cognitive science. The introductory points concerning the properties of the tool nevertheless stand. The analysts do not need detailed how-to-do-it knowledge of the model-building process itself. Furthermore, with a sufficiently rich rule base, use of this kind of design aid would be compatible with the time constraints on development cycles. Having answered a basic set of questions about the design under consideration, the 'what if' facilities can be used to explore interactively how behavioural consequences might change with alternative design features.

Obviously, any real tool based on these concepts would require considerably greater scope than the knowledge bases described here. The knowledge base for command-name entry covered only rather gross distinctions between types of name set. More detailed principles governing factors such as the overall level of ambiguity in name-operation mapping within sets of realistic names and alternative abbreviation schemes are being developed.[4] These are currently being incorporated into the knowledge bases. However, for truly broad scope to be achieved many fundamental issues need to be addressed and resolved. In particular, principles are required concerning the cumulative influences on cognitive activity of a fuller range of features that make up a complete interface design.

Three points are nevertheless worth noting concerning the potential scope of this approach. First, the underlying theoretical framework specifies an overall set of resources which are utilised in the co-ordination and control of perception, cognition and action. The configurations involved in controlling human–computer dialogues draw upon the same underlying processing resources as would be drawn on in understanding or producing natural language or even driving a car. Since the resources are defined in terms of 'mental codes' rather than elements of 'tasks' or 'systems', the framework has considerable general applicability. In essence, the underlying theory provides a general language within which it is possible to describe the cognitive activity associated with a broad range of tasks.

Secondly, the rules that generate the approximate models make use of this general language. This contrasts with several influential approaches that focus more or less exclusively on the representations of tasks and dialogue requirements. Many of these recruit grammatical or simulation techniques to describe the consistency or complexity of dialogue designs.[10, 13, 16, 17] In deriving their predictions they need to model dialogue requirements in depth. In doing so they tend to make only restricted reference to cognitive limitations. The present approach adopts a different trade-off. Knowledge requirements are captured in less detailed form but are integrated with limitations on the operation of the human information-processing mechanism. As such, both forms of consideration fall within the descriptive scope of the modelling technique.

Finally, the technique also involved capturing general relationships within the human information-processing mechanism. These concerned its process configurations, procedural knowledge, record contents and their overall dynamic control. Since the general heuristics are modular, they can be added to and modified as new empirical evidence for additional rules becomes available. Likewise, as new classes of system design are created, the relevant sets of questions and reporting rules can be updated so that the general heuristics can be applied in novel contexts. The approach can thus be systematically extended to bring an increasing range of design issues within its scope.

6. CONCLUSION

As an enterprise in knowledge engineering, this approach is obviously in its infancy. The illustrations given above and other more complex exercises for menu dialogue styles have demonstrated that rule bases can be used to construct approximation models of cognitive activity. Questions concerning predictive validity must await further developments. However, the general form of the approach permits the knowledge bases to be refined and extended in a cumulative manner. As they are enriched, less implicit approximation by the analyst should be required. The next stage of development will involve combining knowledge bases to explore more complex interactions between rules and to improve the descriptive language in which they are expressed.

REFERENCES

1. P. Barnard, Applying the products of research on interactive dialogues, In *Man–machine Interaction: Proceedings of the Joint (IBM/University of Newcastle Seminar*, edited M. J. Elphick, pp. 11–20. University of Newcastle-upon-Tyne (1984).
2. P. Barnard, Interacting cognitive subsystems: a psycholinguistic approach to short term memory. In *Progress in the Psychology of Language*, vol. 2, ch. 6, edited A. Ellis, pp. 197–258. Lawrence Erlbaum, London (1985).
3. P. Barnard, Cognitive resources and the learning of human–computer dialogues. In Carroll, J. M. (Ed.), *Interfacing Thought: Cognitive Aspects of Human Computer Interaction*, ch. 6, edited J. M. Carroll, pp. 112–158. Cambridge, MA. MIT Press (1987).
4. P. Barnard, J. Grudin and A. MacLean (In Press). Developing a science base for the naming of computer commands. To appear in *Cognitive Ergonomics and Human–Computer Interaction*, edited J. Long and A. W. Whitefield. Cambridge University Press (1989).
5. P. Barnard, A. MacLean and N. Hammond, User representations of ordered sequences of command operations. In *Proceedings of Interact '84: First IFIP Conference on Human–Computer Interaction*, vol. 1, edited B. Shackel, pp. 434–438 IEE, London (1984).
6. P. Barnard, M. Wilson and A. MacLean, Approximate modelling of cognitive activity: towards an expert system design aid. In *Proc. CHI+GI '87 Human Factors in Computing Systems and Graphics Interface*, edited J. Carroll and P. Tanner, pp. 21–26. ACM, New York.
7. S. K. Card, T. P. Moran and A. Newell, *The Psychology of Human–Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates (1983).
8. B. R. Gaines, From ergonomics to the fifth generation: 30 years of human–computer interaction studies. In *Human–Computer Interaction – Interact '84*, edited B. Shackel, pp. 3–7. North-Holland, Amsterdam (1984).
9. J. Grudin and P. Barnard, The cognitive demands of learning and representing command names for text-editing. *Human Factors* **26**, 407–422 (1984).
10. D. E. Kieras and P. G. Polson, An approach to formal analysis of user complexity. *International Journal of Man–Machine Studies* **22**, 365–394 (1985).
11. T. K. Landauer, Psychological research as a mother of invention. In *Proc. CHI '85, Human Factors in Computing Systems*, p. 44. ACM, New York (1985).
12. H. Ledgard, J. Whiteside, A. Singer and W. Seymour, The natural language of interactive systems. *Communications of the ACM* **23**, 556–563 (1980).
13. T. P. Moran, The command language grammar, a representation for the user interface of interactive computer systems. *International Journal of Man–Machine Studies* **15**, 3–50 (1981).
14. A. Newell and S. K. Card, The prospects for psychological science in human–computer interaction. *Human Computer Interaction* **1**, 209–242 (1985).
15. D. Norman, Cognitive engineering. In *User Centered System Design*, edited D. A. Norman and J. W. Draper, pp. 31–62. Erlbaum, Hillsdale, NJ (1986).
16. S. Payne and T. R. G. Green, The user's perception of the interaction language: a two-level model. In *Proc. CHI '83, Human Factors in Computing Systems, Boston*, pp. 202–206, ACM, New York (1983).
17. P. Reisner, Further developments towards using formal grammar as a design tool. In *Proc. Gaithersburg Conference, Human Factors in Computer Systems*, pp. 304–308. ACM, New York (1982).
18. S. L. Smith and J. N. Mosier, *Guidelines for Designing User-interface Software*. Report 7 MTR-10090, Esd-Tr-86-278, Mitre Corporation, Bedford, MA (1986).
19. M. Wilson, P. Barnard and A. MacLean, Using an expert system tool to convey HCI information. In *People and Computers: Designing for Usability*, edited M. D. Harrison and A. F. Monk, pp. 482–497. Cambridge University Press (1986).