Knowledge-Based Task Analysis for Human-Computer Systems

M. D. Wilson, P. J. Barnard, T. R .G. Green and A. MacLean

1987

Authors' addresses:

M. D. Wilson
Informatics Division
Rutherford Appleton Laboratory
Chilton
Didcot
Oxon OX11 0QX, U.K.

P. J. Barnard, T. R. G. Green
MRC Applied Psychology Unit
15 Chaucer Road
Cambridge CB2 2EF, U.K.

A. MacLean
Xerox EuroParc
Ravenscroft House
61 Regent Street
Cambridge

INTRODUCTION

The analysis of tasks is a fundamental and important process in many areas of applied behavioural science. Task analysis offers methods for exploring relationships between the properties of systems and user performance. Traditionally (eg. see Miller, 1962), the analyst takes descriptions of the cues that should be perceived and the actions that should be performed, and maps these onto behavioural units; but working with computers presents novel problems. It is the user's conceptual skills, not the perceptual motor skills of a previous generation of technology, that must now be automated. Successful task execution now depends critically on the user's knowledge of the system, its properties, capabilities, and requirements. Units of behaviour can no longer usefully be viewed in isolation.

We shall review recent progress towards incorporating knowledge requirements into task analysis, comparing eleven forms of task analysis. All have been developed with the aim of describing knowledge intensive tasks in HCI; but naturally enough, different techniques address different aspects in differing degrees of detail, and some techniques have been more fully developed than others. The major characteristics we shall stress are the following.

As Morton et al. (1979) point out in their Block Interaction Model, many different types ('blocks') of knowledge influence the user's representation of a current problem or task. One major division separates the knowledge held by the ideal user from non-ideal knowledge, such as distorted or inaccurate versions of the ideal knowledge, or input from other sources such as analogy with other systems or inferences from natural language.

User Centered Task Dynamics

We shall distinguish between analysis methods that explicitly describe the user's goals - and possibly the higher level intentions as well - and more global methods which evaluate the knowledge required to account for the user's actions without specifying any particular steps, or which concentrate on comparing the overall structure of differing methods to accomplish tasks without descending to detail.

We shall also note attempts to describe short term transitions in user's mental representations, such as which machine mode the user believes is currently in force; and long term transitions, including learning and associated changes in representations.

Cognitive Limitations on Processing

Analyses can either present functional descriptions of processing and knowledge representation, or can go further and specify limitations of the human information processing mechanism, such as constraints on memory capacities or limitations on perception, or parameters for process times.

Use of the Technique

Each section will describe how easy it is to employ an analysis technique; what knowledge the analyst would have acquire to use the technique, and what limits and cautions the analyst should be aware of when using a technique.

The eleven techniques that we shall review can be grouped under four headings, as follows.

Group 1: analyses of knowledge content in real world tasks:

i) Task Strategies approach (Miller, 1974);

ii) Task Analysis for Knowledge Descriptions (Johnson et al., 1984);

iii) Command Language Grammar (Moran, 1981);.

Group 2: analyses designed to predict difficulties from interface specifications:

iv) External-Internal Task Mapping Analysis (Moran, 1983);

v) Task Action Grammar (Payne and Green, 1986);

vi) The GOMS family of models (Card et al., 1983);

vii) The User-Device model (Kieras and Polson, 1985).

Group 3: analyses of users' conceptual structures:

viii) Task Analysis for Information Structure Description (Wilson et al., 1985);

ix) Analysis of Menu Systems (Young and Hull, 1983).

Group 4: analyses of cognitive activities

x) Decomposition of mental activity (Norman, 1986);

xi) Cognitive Task Analysis (Barnard, 1987).

In order to focus upon their broader properties, the descriptions of individual approaches will necessarily be brief. Likewise, several prominent analyses have not been included for lack of space (eg. Reisner, 1981; Duncan, 1974; Bullen and Bennett, 1983; Sasso, 1985; Rasmussen, 1986). To avoid overlap, we have also abbreviated the descriptions of Task-Action Grammar, the GOMS family, and the User-Device model, since these are described at length in the chapter by Green, Schiele and Payne in this volume, and have confined ourselves to discussing the four characteristics listed above.

GROUP 1

In this group of methods, the focus is on analyses of the knowledge content in real world tasks.

i) The Task Strategies Approach

Miller (1974) created a descriptive and analytic terminology to represent the generalised information processing functions of a highly skilled operator. The technique was developed prior to many recent developments in HCI, but it is a useful point of departure because it was an early attempt to bring cognitive factors into task analysis. In this approach, there is no specific representation of knowledge or of cognitive resources.

For the purpose of analysis "a task consists of a series of goal-directed transactions controlled by one or more 'programs' that guide the operations by a human operator of a prescribed set of tools through a set of completely or partially predicted environmental states" (Miller, 1973, p 11). The spirit of the analysis is captured in an example: "even a piano mover should scan and detect a marble on the stairs, interpret its potential significance, and devise a foot-moving strategy that will avoid its untoward possibilities" (Miller, 1973, p 3). A task is described in terms of 25 task functions which refer to possible cognitive actions (eg. detect, transmit plan) or entities (eg. a "message", a "goal image" or "short term memory buffer"). A complete list is given in Fleishman & Quaintance (1984). The analysis requires four stages:

identifying key aspects of the environment (eg. goals and stressors)
identifying what needs to be learned
naming the task functions

Methods are specified for both identifying and teaching work strategies, which can be either behavioural strategies to maximise the efficiency of the operator as a resource, or task strategies to cope more effectively with the uncertainties of the environment.

Miller's approach has been used in a variety of military contexts, from rifle maintenance to the operation of office equipment. Tasks can readily be decomposed into identifiable stages, and it is possible to compare the complexity of different methods for performing a task by counting the task functions required. However, the analysis is not appropriate for describing general models.

Knowledge

The elemental unit of knowledge within this technique is 'a message'. The knowledge of a person performing a task is not the focus of the task analysis, and consequently it offers little to aid either the segregation of knowledge into any domains or any formalism for the specification of knowledge elements. The first of the four stages of the analysis requires a specification of the task or domain content, but no structure is imposed on this nor is a mechanism offered for deriving it. There is also no distinction in the analysis between an ideal set of knowledge and other knowledge.

User Centered Task Dynamics

The analysis captures most aspects of the user centered dynamics of a task. Goals are explicitly incorporated into the analysis, and there are several functions which operate on them. Although there is no specific function to account for motivational goals, a twenty-sixth function explicitly for motivation has been suggested (Fleishman and Quaintance, 1984). Since the analysis accounts for the processing of information, it explicitly addresses the short term dynamics of task performance. Long term transitions and user learning are explicitly acknowledged by the possible proceduralisation of functions as sequences are learned and by the inclusion of work strategies to account for highly skilled performance.

Cognitive Limitations on Processing and Use of the technique

There are several attempts in the analysis to provide cognitively salient limitations on processing. The short term memory buffer and a "compute" task function are good examples. However, values are not specified for these limitations.

Use of Technique

To use this approach an analyst must follow the four stages described eaarlier. Tools are not supplied in the approach to support the first three of these stages. Definitions and descriptions are provided which will guide the

The explicit entity that an analyst is given in this technique is the set of 25 functions, and the definitions that permit their use. The specifications of the functions are detailed, but there is no indicated method for the selection of the uniquely appropriate function. This is mostly due to the overlap in the scope of several functions, which results in alternative ways of describing the same task sequence. A consequence of this is that an analyst would have to be psychologically knowledgeable in order to partition information in the world, and to select the appropriate functions based on a separate psychological view of task performance.

In fact, the first stage of the analysis advocated by Miller could well be augmented by the kind of technique outlined in the next section.

ii) Task Analysis for Knowledge based Descriptions (TAKD)

Task analysis for knowledge based descriptions (Johnson et al,1984; 1985) aims to make explicit the knowledge requirements for a particular world task. Whereas the task strategies approach represents a task as a series of domain independent functions, here the task is characterised by domain specific knowledge. As with the task strategies approach, TAKD involves four main stages:

generate a task description
identify required knowledge in terms of objects and actions
classify these into generic actions and objects
express the task in a knowledge representation grammar (KRG)

The authors stress the use of as many sources of information as possible in the first stage of the analysis - structured interviews, direct observations of real and structured tasks, and the analysis of protocols collected during and after completion of a task by trainees, instructors and experienced users.

An initial task description generated after the first step will be a sequential plan of statements. For example, from an analysis of a joiner's order for a staircase as part of the design of a computer based ordering system (from Johnson, 1985), two steps in the task description were:

- receive order for 15 open-plan, piranha pine staircases of design pattern p1375; - check stock-list to see if supply can be met from stock (answer no);

the subsequent dictionary of identified generic objects and actions would include : CHECK: (inspect, query) SELECT: (identify, choose) TIMBER: (piranha pine; standard pine) DESIGN: (open-plan; closed) FINISH: (varnish, paint)

from which valid KRG expressions would be constructed, for example:

SELECT/ a TIMBER/ for a DESIGN/with a FINISH which could be translated as: "choose a piece of piranha pine, for an open plan design no. p1375, with a polished finish".

TAKD has been used to design a syllabus for teaching information technology skills (Johnson et al, 1984) and for the generation of designs for computer programs (Johnson, 1985).

Knowledge

The principal objective of TAKD is to produce a specification of the knowledge required to use a system, or the knowledge that a system would have to include to perform a task. Although the methodology of TAKD will require the initial collection of both ideal and non-ideal knowledge, by the time the final specifications are produced, non-ideal knowledge will have been filtered out by the process of generification, and the KRG creation.

The method of representing knowledge elements in this technique, by action/object associations, is a relatively weak formalism since it does not distinguish between subject and object, direct and indirect objects, or instruments and agents. The power of this method is increased by the use of a knowledge representation grammar, but it is not clear that the grammar used on one occasion would

be appropriate on another, any more than the knowledge suitable for one domain would be germane to another.

User Centered Task Dynamics and Cognitive Limitations on Processing

TAKD has no representation of either task dynamics or cognitive limitations.

Use of technique

TAKD has certain weaknesses in identifying so-called generic actions and objects and in combining them into a knowledge representation grammar.
For example, from an analysis of general information technology skills (Johnson et al, 1984), 14 generic actions and 22 generic objects were derived that would underlie a syllabus structure. The recombination of these items into the KRG statements is very difficult as they overlap in scope, and their structural relationships are not made explicit by the simple dictionary structure. Consequently, it is not clear whether "type x" should be described as 'insert' with a 'textual input device' (eg. key-board) or "insert" an include relationships between the objects, this confusion would not arise.

For the non-specialist, therefore, TAKD is only likely to be usable when the 'generic' actions and objects are closely related to standard English terms.

iii) Command Language Grammar (CLG)

The Command Language Grammar was originated by Moran (1978; 1981) as a design tool to 'separate out the conceptual model of a system from its p. 5). CLG hierarchically decomposes a system's function into its objects, methods and operations. These are described in a set of definable levels which progressively specify a task in more detail.

The grammar consists of three components, each divided into two levels which are further subdivided. The Task Level describes of the user's major intention. Below this are Semantic and Syntactic Levels which focus on the objects and actions the user employs to accomplish the task. The Semantic level contains a conceptual model composed of objects and operations that may be performed on them, and semantic methods for accomplishing the tasks of the previous level. The Syntactic Level describes the command language structure, discriminating between commands, arguments, descriptors and command contexts. At the Interaction Level below this, the dialogue must be mapped onto a sequence of physical actions, eg. key presses. The lowest two levels are the Spatial Layout Level, where the physical layout of the input/output devices are specified, and the Device Level, where the remaining physical features are defined.

The important feature of CLG is that its several levels of description are designed to correspond to the levels of representation held by users. CLG maintains that users need not represent all knowledge at all levels - some knowledge (especially at lower levels, but also higher) will be held as procedures whilst other knowledge will be declaratively represented. CLG implies that users can operate efficiently with only the Interaction Level methods. However, users who have not come to represent the higher levels will probably be at a loss when something goes wrong. For example, they will not possess the appropriate concepts to make sense of an error situation. Conversely, users may know their objective but may not know what commands to use to reach it; that is, they know the semantic method, but not the syntactic method.

CLG has been used as a design tool to develop the structure of computer programs (Moran, 1981) and as a method for evaluating interfaces (Davis, 1983), for which it was only moderately successful.

Knowledge

Non-ideal knowledge is not an important part of CLG and its inclusion depends on the analyst, so inconsistencies between interface terms or differences between interface terms and natural language terms may well go unobserved.

CLG offers a symbolic notation, and a grammar for describing knowledge which permits the relationships between knowledge to be expressed. This is a more powerful formalism than the use of object/action pairs which only permit a relationship between two items. The limitation on the use of the formalism is in the identification of the knowledge to be represented. Once knowledge has been specified, relationships and conflicts can be captured by the grammar.

User Centered Task Dynamics and Cognitive Limitations on Processing

The CLG grammar rules are not a performance theory and their structure While the different levels for the representation of knowledge in CLG have psychological credibility, it is unfortunate that no mechanism is provided to select which knowledge exists at which level.

Use of the Technique

CLG guides designers by placing an order on the decisions they must make. It also enables the designer to maintain consistency throughout this process, and suggests the stages at which reduction in the elements should take place, but the analyst/designer must invent each new element. This is a slow and laborious process. Without a tool such as CLG the process results in many inconsistencies and the design must be checked and changed as conflicts and errors are discovered by chance. However, CLG is a cumbersome tool if used only to explicate consistency; other methods for this are more easily managed - for example, TAG, discussed below. CLG can only aid the designer in making design and trade off decisions if the rules of how to design user interfaces are included in it as originally planned (Moran, 1978).

GROUP 2

The next four techniques focus on capturing knowledge of systems from designs and specifications to assess complexity, learnability, transfer or performance times.

iv) External-Internal Task Mapping Analysis (ETIT)

The purpose of the External-Internal Task Mapping Analysis (Moran, 1983) is to assess the complexity of learning a system for a naive user and the potential transfer of knowledge from one system to another.

The user comes to a system with a task to perform which is defined for ETIT in terms of the external task space (eg. reports, chapters) and not in terms of the internal task space (eg. directories, files, editing commands). The complexity of the relationship between these two spaces will reflect the difficulty found in using the system and especially in learning to use it.

ETIT compares representations of these two spaces, assuming the user's knowledge of a system's properties is either complete or none. For a text editing task, the external task space can be taken as the set of core editing tasks defined by Roberts and Moran (1983). They reduced 212 tasks which text editors could potentially perform to "the minimal subset of an editor's functions" (Roberts, 1979, p8). That is, the 37 tasks which all of a sample of editors actually performed. Each task (eg. Remove-Word) consists of a function term and a task term. This external task space for editing tasks

was built from eight editing functions and five types of text entities. The internal task space for a display editor (Moran, 1983) may have only one entity - a character string - and three functions - Insert, Cut and Paste. To map from the external to the internal task space requires a set of ten rules. One of these would translate all text entities into strings, another would map Remove-Word from the external space into Cut-String in the internal space of the editor commands. In contrast to this straightforward mapping, a line editor (Moran, 1983) may require 15 mapping rules which would be more complex than those for the display editor. Therefore the complexity of the mapping rules reflects an increased complexity in using one system over another.

Transfer from one system to another can be assessed as the number of common rules between the systems. For example, every display editor rule exists for the line editor, therefore the transfer would be easy. The converse is not true. This analysis assumes that learning will not necessarily be facilitated by any form of similarity between commands or command sequences in different systems. Rather, the key factor is the similarity in mapping rules from the internal to external task spaces for the two systems.

ETIT has only been used for the demonstration examples of a line and text editor summarised above from Moran (1983), and in Douglas's (1983) thesis to account for several aspects of the Roberts and Moran (1983) learning data for different text editors. Outside the area of text editing there are no available applications, and there are no available specifications of the external task space other than that of Roberts and Moran (1983).

Knowledge

The internal task space captures the ideal knowledge of the system. The external task space captures knowledge of the world version of the task domain. Some aspects of knowledge of other devices and systems can be identified by the comparison of the ETIT mapping rules for two systems. This comparison will not show how this knowledge will interfere with the ideal knowledge of any system under consideration, although it will show where the knowledge overlaps. Knowledge of natural language is not caught by the analysis unless this is incorporated in either the external or internal task space.

The level of description of elements of knowledge used in the example external task space is of actions and objects. This matches the simple 'function and one object' command languages used on the systems investigated. For other external task spaces other descriptions may have to be derived.

User Centered Task Dynamics and Cognitive Limitations on Processing

This analysis only captures goals to the extent that they represent nodes in the external or internal task space. There are no dynamic properties to this analysis and therefore no cognitive limitations on it. ETIT does not address the problem of interference or negative transfer effects.

Use of the Technique

To use ETIT the analyst must code a system specification into an internal task space and code domain knowledge into an external task space, and then produce mapping rules between these spaces. The construction of the internal task space from a system specification can be performed by a non-specialist.

The development of a representation of tasks in the real world as an external task space is more problematic. ETIT is presented as two example analyses using the same external task space provided by Roberts and Moran (1983). The mechanism used for establishing this external task space appears to

establish the common internal task space of a set of computer systems designed to perform the same task. To accurately reflect the performance of individuals, the external task space ought to be an individual's conceptual model of the task which would be susceptible to influences from that individual's general knowledge. There is no simple method of assessing this, therefore one must use a representation of the task abstractly presented in the world. Whether the external task space produced by any method is sufficiently close to a user's conceptual model for an analysts purposes can only be judged by empirical investigation. Consequently, although an internal task space may be easy to construct, the correct method for constructing an external task space is uncertain.

v) Task Action Grammar (TAG)

knowledge of the mappings from tasks to actions, and to predict learnability by capturing all the generalities that the use may be aware of. Green et al. give a detailed account of TAG in this volume.

An analysis of a computer version of a task into a TAG requires its decomposition into "simple tasks". A simple task is "any task that a user can routinely perform" or which can be accomplished with no problem solving component or control structure; they may be at a very low level, for novices, or for more experienced users they may be compiled (Anderson, 1983) into larger groupings. For example, "move cursor one character upward" may require several actions, but after practice it would equate to a single command and therefore to one "simple task". Higher levels of planning, requiring several "simple tasks"' are modelled by a separate planning component, left unspecified.

TAG offers two mechanisms for capturing the generalities within command languages. Firstly, simple tasks may be defined as having the features of another simple task, but with one or more specifically different. For example, 'move cursor one character down', could be described as having the features of 'move cursor one character upward' with the 'direction' specifically set to 'down'. The second mechanism allows TAG to capture different forms of general semantic knowledge in separate rules. For example, when the command term "UP" is the token used in the command language, it is not an arbitrary symbol - it draws on knowledge of natural language.
TAG is able to express the notion that a command language token is based on the presumed natural language knowledge of the user. Rules of this sort (which can apply to other domains besides natural language) give TAG a powerful mechanism to capture information relevant to the interaction which are not specified in the language itself.

TAG has been used (Payne, 1985) to describe the languages used in several experiments (eg. Carroll, 1982) and its predictions are consistent with the experimental findings. It has also been used to describe various systems such as MacDraw and Multiplan which are described in the chapter by Green et al, in this volume.

Knowledge

Like ETIT, TAG captures other knowledge in addition to ideal system knowledge. However, TAG goes further, in showing where non-ideal knowledge interferes with system knowledge. One of the benefits of TAG is its ability to capture influences from natural language which ETIT and CLG could not. TAG can therefore indicate potential user errors due to the choice of command names or command ordering. For example, abbreviating the commands UP, NEXT and DELETE to their first letters (U, N, D) may unfortunately lead users to enter 'D' when they want to move "down" because of the relationship between up and down in natural language. Knowledge of other systems or of world domain versions of tasks could also be entered into TAG representations to show effects of interference from these knowledge sources. As yet, however, none of the presented examples of TAGs include these components.

One problem is that the choice of features may be unstable. This is because the set of features that defines a simple task depends on the contrast one set of directional features suitable for contrasts between movement tasks which vary in direction and magnitude will be inappropriate for comparing a variation between functions such as 'move' and 'copy'. This mechanism has the advantage that no absolute set of all the features need be specified, but it has the disadvantage that the feature set for a command must be changed or increased to capture new contrasts. Unless the analyst has thought of all possible contrasts which may occur in the analysis, errors due to unsystematic changes to the feature sets are likely.

User Centered Task Dynamics and Cognitive Limitations on Processing

TAG claims to be a 'cognitive competence' model, not a performance model. It does not explicitly account for any dynamics of the user's representation or processing. User goals can be equated with 'simple tasks' in TAG, but there is no mechanism to chain 'simple tasks' and there is no precise definition of what a 'simple task' is or how a system's language should be segmented into them. There is also no attempt to capture higher level goals or motivations.

As a competence model TAG proposes a more powerful view of users' competence than is usually adopted. Most models are presented as though users were unable to perceive the 'family resemblances' that TAG uses to give a structuring to the command language.

Use of the Technique

The TAG analyst must re-code an ideal representation of a computer interface dialogue into a task dictionary and rule schemata. Other information must also be coded into the same form, and the analyst must attempt to minimise the number of schemata required for the TAG description.

The re-coding of a well-specified system description into a TAG description is comparatively simple, but the selection of what other (non-ideal) information to encode relies on the analyst's intuition and observation. This can lead to difficulties in comparing two interfaces.

Complexity metrics can only be comparative between TAGs for different systems where the same non-system knowledge is included in all analyses. If there were specified psychological constraints in the model there might be a simple route for deriving a complexity metric by which to judge the rules derived by the analysis. Without these the scope of the predictions from any analysis are limited to comparative judgments of the complexity of different systems.

vi) The GOMS Family of Models

The purpose of the GOMS analysis (Card, Moran and Newell, 1980; 1983) is to generate useful engineering models to predict the time to complete tasks. Different members of the GOMS family incorporate different grains of analysis. A technique of sensitivity analysis (Card et al, 1983) can be used to assess the grain of analysis which is most appropriate for describing observed the GOMS model itself, and the Keystroke model. Further details, together with examples, are given in the chapter by Green et al. in the present volume.

The GOMS model provides a simple view of mental processes in terms of 'goals', operators, and 'selection rules' for choosing between alternative methods. The task to be analysed is decomposed into successively smaller sub-tasks until the level of 'unit tasks' is reached. This is the level that drives the mechanisms in GOMS and for which the user is assumed to know particular methods (eg.

a "line feed" method for locating a line); in the task of editing, the unit task can be equated with a particular correction on a manuscript. Within the task a user is driven by a series of goals. The top level goal (eg. EDIT-MANUSCRIPT) would create a sub-goal to perform each 'unit-task' (eg. EDIT-UNIT-TASK) which would be repeated until no more unit tasks remain to be performed.

The Keystroke Level model is based on the GOMS model but its role is purely to predict "the time an expert user will take to execute the task using the system, providing he uses the method without error" (Card et al, 1983, p 260). The user's task is again divided into 'unit tasks' but instead of being decomposed into goals and methods, the time to perform each is directly the sum of three components: an 'acquisition time' (computed at 1.8 seconds); a performance time which is the sum of the times to execute the keystrokes in the commands (dependent on the user's typing speed); and times for mental operations (computed at 1.4 seconds). The model incorporates a set of rules for the application of mental operators such as, adding times for each 'cognitive unit' (approximately a command string).

The Keystroke Level model is only intended to give quantitative predictions of the performance time for error free expert performance. In doing this, it assumes the shortest sequence of commands is used to perform a task, although evidence suggests that even experts do not always use the technique that is either fastest (eg. MacLean et al, 1985) or has fewest keystrokes (Embley and Nagy, 1982). Additionally, Roberts and Moran (1982), found that major errors (those that took more than a few seconds to correct) occupied between 4 and 22 percent of testing; Allen and Sczerbo (1983) report that the average time spent correcting errors was 20.1 percent of the time not involved in errors. Considering this, it is remarkable that the Keystroke Level model appears to generate fairly good predictions.

Roberts and Moran (1983) present a comparison of nine text editors using the model and performance tests. Although the values generally follow the predictions, the editor predicted to be fastest by the model was only found to be sixth fastest in practice. The model is reported as being accurate to a standard error of 21% over a variety of different tasks and systems (Card et al, 1983, p 297). The model's predictions should be adjusted by a factor of 1.4 (1.1 to 2.0) to fit true performance (Roberts and Moran, 1982).

Knowledge

These analyses seek to predict performance times assuming ideal system ideal system knowledge is contained within the system specification, but in addition the models use individual typing speed estimates, determined from performance, and in the case of GOMS individual selection rules are also determined from performance.

User Centered Task Dynamics

GOMS incorporates task goals in the analysis, but not motivational goals, which are assumed to be constant. Since the technique is limited to predictions for expert users, the goals and sub-goals are derived from stored (learnt) plans, rather than being generated through problem solving. The analysis is also greatly limited by being unable to account for learning, since it assumes that the user has optimal system knowledge. GOMS can, however, describe the short term dynamics of the interaction to some degree, depending on the grain of the operators used. The more the processing operators used are based on the human information processing system, the more exact the account of the dynamics of processing.

The only aspects of the dynamics of processing which the Keystroke Level model attempts to capture are the times for the action and mental operators. This is reasonable since its sole purpose is to

predict the times for expert error free performance on a system. However, it has been suggested (Allen and Scerbo, 1983) that the error margin that exists for these predictions is due to inadequacies in the rules for the application of mental operators.

Cognitive Limitations on Processing

The very limited degree to which this analysis involves any psychological process model can be assessed from the amount of psychological reasoning behind it. The analysis is based on two basic principles of psychology. Firstly, that people act so as to attain their goals through rational action given the structure of the task, and secondly, that problem solving activity can be described in terms of a set of knowledge states; operators for changing states; and control knowledge for applying knowledge.

Since "Operators are elementary ... information processing acts, whose execution is necessary to change any aspects of the user's memory ..." (Card, 1978, p 58) the model has the potential for employing psychologically salient operators, to represent short term processing operations (eg. the perception of salient cues). Card et al (1983) present a model incorporating such operators (the Model Human Information Processor) which is used to derive action performance times, but these operators are not included in the example analyses using GOMS.

Use of the Technique

The Keystroke Level model is a concise engineering formula which is obviously predictive of approximate performance times from a design description and is also comprehensible to the non-specialist. The only parts of the model which are not explicit are the definitions of the 'unit-task' and the 'cognitive unit'. The approximation of these units to system commands is workable, units of representation chosen in the model are those of the system (optimal methods and keystrokes) rather than psychologically motivated units of representation. It could be avoided if the model were more strongly motivated by a cognitive theory which specified cognitively salient units, or if the units were based on user performance. However, it is unclear that a sufficiently detailed cognitive theory exists to make predictions, and if assessments were made of user performance, time predictions would be redundant and the model's applicability would be severely reduced.

The usability of the GOMS analysis is far more constrained by the problem of defining the 'unit task'. For applications where there is an established body of previous computer versions of the domain application, the approximation to a command sequence may be acceptable. In other applications, the analyst must rely heavily on his own intuition in dividing the task into units. It would also be difficult for a non-psychologist to estimate the correct range for the operators in an application. In many cases, it may be necessary for any analyst to attempt several grains of operator before accepting one as appropriate. It is also necessary for any potential users of the GOMS family of tools to decide the extent to which predictions of times for optimal error free performance may in fact be useful for their situation.

vii) The User-Device Model

Kieras and Polson (1985) present a two-component approach to assess the effects of transfer of knowledge from one system to another, the complexity of devices, learning and performance times, and error frequencies. This approach is described at length elsewhere in this volume, in the chapter by Green, Schiele and Payne, and will only be outlined here.

The first component is a production system model of the user's how-to-do-it knowledge of system

use, based on the GOMS representation. The user's goal structure is derived using a mechanism similar to that in GOMS, with the addition that active goals are represented in working memory. In this model, one measure of user complexity is the depth of the goal stack during operation - a more complex method for achieving a task will require a deeper goal stack than a simple method. Another measure of complexity is the number of productions required to represent a method for achieving a task. Performance time predictions can also be made from the number of productions fired.

The second constituent of the approach is a Generalised Transition Network (GTN) model of the device - that is the computer interface, showing the system's possible states, the possible actions the user can take in that state, and a connection to the next state that will result from that action. Whereas the production system model captures the how-to-do-it knowledge this model can be viewed as capturing how-it-works knowledge. This representation is comparable to other formal grammars of devices, with the advantage that it is visually more appealing.

The description of a device and a task using these formalisms offers the opportunity for running computer simulations of task performance. A simulation has been presented for the IBM Displaywriter (Kieras and Polson, 1985) which appears to predict the learning rate and performance time for the

Production rules are assumed to be single units that are learnt on an 'all or none' basis, and rules already learnt for one 'task' or 'method' can be incorporated when a new one is learnt. Therefore if two methods (eg how to delete a word, and how to delete a character)have some production rules in common, learning one method will make it easier to learn the other. In the same way, knowledge which the user possesses prior to learning a particular device is 'device independent' and therefore has no learning cost; 'device dependent' information does carry a learning cost. There is no formalism for assessing the device independence of knowledge, although if the knowledge is employed in a domain outside that of the device then it is assumed to be device independent.

Knowledge

This technique incorporates a representation of the ideal knowledge of the system, but not any non-ideal knowledge. The technique does permit the comparison of ideal knowledge of different systems. Although not yet implemented, this approach is potentially capable of capturing other 'device independent' knowledge such as that from a domain version of a task. The technique contrasts with ETIT where transfer of knowledge between systems is assessed, not by the overlap in representations of methods from different systems, but in the mapping rules from those methods to a third user representation. ETIT attempts to incorporate knowledge of a task outside any system whereas this technique only incorporates knowledge of systems.

User Centered Task Dynamics

The production system component of the technique has many of the properties and drawbacks of the GOMS approach on which it is based. The generalised transition network, however, presents one element of task dynamics which none of the other techniques discussed in this chapter do. It presents what information is on the screen at any time. This permits the analyst to assess the consistency of use of system prompts etc. When this component of the approach is linked to the production system, it will offer a starting point for assessing the perceptual and attentional aspects of an interaction. At present, although rules can contain conditionals on the perceptual cues required for a method, no mechanism is provided to account for the effort of discriminating these salient cues from other objects in the environment.

Cognitive Limitations on Processing

Kieras and Polson's model places few limitations on cognitive processing abilities. When modelling experts, working memory is used as an infinitely large, non-decaying goal stack. Novice-level users are assumed to test all goals that are in working memory, to explicitly attend to feedback information and verify each step they take, so production rules to model novices are written in a constrained style. It is not clear how these views relate to other views of working memory in the human information are applied. Constraints on these aspects could be added to the tool in the future, but are not at present included.

Use of the Technique

The analyst would have to encode a system specification into two formalisms, one for each component of the model. Both these formalisms are complex to use, and production systems are generally cumbersome to construct and maintain.

Programming style determines the number of production rules used to represent a method. The analyst must therefore take care to maintain a strict consistency of style so that a count of productions can be used as a complexity metric. Various metrics of performance and transfer are available with this technique and the choice of the 'best' one for any circumstances cannot be guided until more experimental data is available (but see Polson, 1987). If an assessment was to involve rules of different flavours (eg. rules for natural language or the work domain version of a task) then a more complex metric than mere rule counting would have to be employed. This metric would have to incorporate some psychologically based balance between the different rule types.

GROUP 3

The next two techniques are empirically based analyses of the knowledge actually possessed by users, not the idealised knowledge presented in design specifications.

viii) Information Structure Description

Wilson et al (1985) used a task analysis to classify user errors, so that the knowledge elements or 'information structures' supporting actual user performance could be derived. A second purpose was to describe long term transitions in learning as knowledge structures evolve.

This approach makes use of two levels of task description. One level describes the overall process of controlling a dialogue sequence in terms of attempts to satisfy task goals. The other specifies particular "information structures" or "meta information structures" that make up the knowledge called upon to achieve those goals. Performance protocols are analysed with respect to these two levels.

The process of controlling a dialogue sequence is described as a simple goal structure, consisting of a major goal (eg. edit-manuscript) subdivided into individual goals approximately equivalent to individual editing instructions. It is assumed that for each goal the user makes an attempt to achieve it and tests to see whether the attempt succeeded: an attempt-test cycle.

The attempt-test cycle for the example study involved four stages: one of attempt specification (a mental event) and three of action with contingent tests on that action - establishing a command context, the performance of a command specific procedure, and termination of a command sequence. The definiton of these three latter stages was derived from a syntactic analysis of the particular menu

interface examined.

At each stage the test could either be passed, and the attempt continued as specified; failed resulting in a local correction (e.g. deleting the last character typed) or, failed resulting in the specification of a new attempt or goal. There were also special cases for passing a test and the user assuming the attempt was completed so a new goal was prematurely attempted. This analysis permits a complete classification of user performance (both correct and erroneous), to arcs connecting the four stages of the attempt-test cycle, that is, a description that is sensitive to the device specification.

When tasks are performed in an optimal fashion it can only be assumed that users possess ideal task knowledge. When errors arise, non-ideal knowledge can be inferred or a performance breakdown assumed. The second level of the analysis involves a specification of the user knowledge required to support the analysed performance. Two classes of information structure are incorporated in this analysis: Meta-Information Structures, or general rules of system performance, and Information Structures, or specific fragments of performable methods. Information structures for an expert user are similar to the methods (or task-action mappings) used in GOMS, however for other users these will account for the partial knowledge they hold

The changes that take place during a user's learning are viewed as changes in a repertoire of these information structures. Early in learning, the repertoire contains a small number of both types, some of which may accurately and completely specify appropriate task-action mappings, or general characteristics of system operation. Other members will be inaccurate, incomplete or contradictory with each other. As practice continues, inaccurate or incomplete information structures may become inaccessible with the addition of new members. Expert status may be achieved when the content of knowledge within the repertoire stabilises, and is in accord with, the system requirements for the tasks that that user needs to undertake.

As the process of learning progresses, information structures are derived from more general meta-information structures which are themselves also generated. An illustrative model has been presented that could account for this process but it will not be described here.

Particular information structures and meta-information structures can be inferred from the analysis of user performance provided by the first half of the technique. This enables the system knowledge held by users at different stages of learning to be characterised without merely assuming it to be none for a novice and 'complete' for an expert. This allows the designer to be aware of user difficulties and to support performance at different stages of knowledge development, and for different types of user.

Knowledge.

This technique attempts to capture the knowledge involved in actual error prone performance. Consequently, it captures both ideal system knowledge and the non-ideal knowledge which a user may have acquired. The information structures and meta-information structures derived from performance can contain influences from natural language, the task domain and other devices and systems.

The information structure terminology provided by this technique is powerful enough to capture consistencies and structured relationships between knowledge at various levels. The information structures are written as informal English phrases embedded within a bracketing notation. Mismatches between the knowledge assumed for actual system use and idealised system use emerge in the context of the bracketing of sequences which actually occur.

User Centered Task Dynamics

The first level of the analysis captures the goal structure of a task in a tree. The top level of the tree in the example analysis did not capture high level motivations, but these could be represented in the same way as other goals if required. All actions by the user and changes in the system state are specified in this technique. Therefore the task model developed in the example analysis permits the capture and structuring of all short term transitions both in the system and in the user's mental representations. Since the system state is described and the user's representation, the technique permits the mismatches between between the two to be identified.

The area where this analysis differs from most of the others, is that it captures not only the short term transitions but also the long term transitions in the user's representations. The description of a repertoire of information structures which changes as knowledge develops permits the analyst to investigate the interaction of the various knowledge sources for maturing user populations.

Cognitive Limitations on Processing

This analysis does not assert that the inferred information structures have absolute psychological saliency. These structures are taken to represent the 'content' of knowledge held by users but not necessarily the 'form' in which that knowledge is represented. The analysis characterises the set of information structures that might need to be generated to support the task-action mappings performed by users. Beyond these qualities of the representations, there are no other information processing constraints included in the analysis on the processing of user knowledge.

Use of the Technique

To use this technique analysts must describe the optimal method to achieve a task. Then they must specify the stages of the attempt-test cycle for that task. They can then assign user actions to arcs in that cycle. Having identified the sites of major deviations from the optimal method, analysts must hypothsise the knowledge held by users which led them to these deviations. Where performance is studied for a sufficient period, the analysis can be used to identify the changes that take place in a repertoire of knowledge as learning progresses.

The technique has been described for a single system, and much of its application is heuristic rather than formalised. The major drawback with using this analysis is the time and effort required to assign individual user actions to the possible arcs of the fully specified attempt-test cycle. The original analysis was performed to give example mismatches between actual user knowledge and idealised knowledge in order to motivate further experimental research. For these purposes the significant effort required is often worthwhile (e.g. Wilson, Barnard and MacLean,1985). However, for more practical system evaluations, problems can be identified with much less effort (eg. Hammond et al, 1983) and progress may depend upon automating the more sophisticated scoring schemes such as that deployed with this technique.

ix) An Analysis of Menu Systems

Few of the previous analyses attempt to account for problem solving during performance. The next method of task analysis (Young and Hull, 1982; 1983) explicitly addresses real-time problem solving during a search for target information, and illustrates how the performance of users can be based on the effects of their deep mental models (Carroll, 1984) - both of the system and of the domain of the information presented on-screen.

(Prestel). In this system each page, or frame, provides information and a menu leading to other

frames. The user's task, at each step, is to select the appropriate item from the menu; that is, to match a menu item with a hypothesis about the route leading eventually to the target information. Young has analysed examples of user's performance on the menu search task and has suggested an implicit relation between the overall topic of a menu frame and the individual options on its menu, and the way these relations can constrain the way that users can process the frame.

Each menu presents a categorisation of a domain into subsets. Different decision strategies in order to handle them correctly. Young attempted to develop a taxonomy of 'categorisation structures' for menu systems giving the prototypical decision strategy for each. Examples of these structures include partitioning the topic into non-overlapping subsets; the 'N + Other' structure, with a series of specific choices followed by a catch-all category; categorisation by multiple bases or dimensions, forcing the user to first pick the appropriate dimension, then to choose within it; overlapping categories; and structures based on analogies to other sources of information, such as newspaper lay-out. Each of these requires a different user strategy.

In relation to this partial taxonomy Young presented a tentative process model for menu selection. In this 'the decision making is opportunistic, driven strongly by the particulars of each choice, so that the decision "method" emerges as a product of the interplay between the context, the user's query, and the details of the frame itself' (Young and Hull, 1983).

Knowledge

This technique recognises that the knowledge used to construct a hypothesis or understand a menu frame could originate from any source available to the user - not just the 'idealised' knowledge of how to find the target. Thus it captures aspects of the general knowledge brought to bear by users which other analyses have not considered.

User Centered Task Dynamics and Cognitive Limitations on Processing

Young's process model is restricted to cases where the user's main goal is always constant and neither higher motivational goals nor lower goal nestings need be considered. The process model presented does not attempt to include any cognitive limitations on processing.

Use of the Technique

This analysis is simply an example of one analysis of a particular task which results in a partial taxonomy of menu structures. It serves as an illustration and a reminder of the shortfall of many other analysis techniques, but as it stands it does not generalise further.

GROUP 4

occurs during task performance. These complement the analyses outlined in the previous sections which emphasised the knowledge requirements of tasks.

x) Decomposition of mental activity

Norman (1984, 1986) describes an analysis based upon the decomposition of a task into different mental activities. The description of mental activities is intended to help analysts understand the cognitive consequences of design features. The analysis requires mapping from system variables to

psychological ones.

The user approaches a system with a set of goals to achieve, yet these goals must be realised through physical actions at the terminal. The gap between goal formation and action is what Norman calls "the gulf of execution". Bridging that gulf requires three types of mental activity: forming an intention to act; specifying an appropriate action sequence; and executing that action sequence.

Once an action sequence has been executed, there is a corresponding "gulf of evaluation" because the user must relate the system's response to the original goal. This is again bridged by three complementary types of activity: perceiving the physical properties of the system state; interpreting them; and evaluating those interpretations in relation to the original goal.

Taking these stages together with the initial formation of a goal gives seven stages of mental activity:

1 Establishing the Goal.
2 Forming the Intention.
3 Specifying the Action Sequence.
4 Executing the Action.
5 Perceiving the System State.
6 Interpreting the State.
7 Evaluating the System State with respect to the Goals and Intentions.

In real tasks the stages need not necessarily occur in strict sequence and some may even be omitted. The definition of stages represents an approximate theory of action which itself is undergoing evolution (the number and definition of stages may change). In practice, the analysis "must be used in ways appropriate to the situation" (Norman,1986, p. 42). The system designer can tackle substantial gulfs of execution or evaluation by making the system more like the user's psychological needs. Alternatively, the user may have to create plans and and carry out action sequences that meet the requirements imposed by the physical system. These action sequences can, of course, be developed through experience and supported by training until they feel almost part of the system.

Different aspects of system design support different stages of activity, and designers can use this analysis to assist their choices. In many cases, design decisions are really trade-offs between supporting one stage or another. Menus, for example, can assist the stages of intention formation menus may slow down the stage of action execution. In contrast, command languages may support rapid execution of action but create problems in action specification and so on.

Such an analysis helps us to understand the benefits that arise from direct manipulation interfaces or from providing a consistent and explicit system image. Menus and pointing devices combined with immediate visual feedback can help to reduce the gulfs of execution and evaluation. Likewise, a consistent model can help support the user during phases of activity that involve a strong problem-solving component.

Knowledge and User Centered Task Dynamics

This analysis makes frequent reference to the use of knowledge, both correct and incorrect, during the various stages of activity. No formalism is offered for specifying or analysing that knowledge. Rather, analysts must draw their own inferences about how and when particular forms of knowledge are relevant.

Similarly, the approach incorporates an analysis of goals during task performance, and also allows for the construction of plans during performance - in fact, it emphasises plan construction. As with

the case of knowledge, no explicit mechanisms are provided to guide more detailed analyses or for assessing their relation to higher motivations.

By simply identifying attributes of the gulf between the state of the user and the state of a system, the approach allows the analyst to be aware of where learning would be required by the user. Once again, the technique does not specifically address how longer term changes in user representations might come about.

Cognitive Limitations on Processing

The seven stages suggested are all forms of cognitive processing that are required to map a mental representation onto a physical system. In this respect, the approach provides a means of thinking about the likely complexity of that mapping. In its current form the analysis does not seek to specify constraints on mental representations, mental capacity or mental processing in exact form. As an "engineering approach" it makes use of approximate representations drawn more generally from the science base of psychology - such as known attributes of cognitive skills (eg. perceptual search, the motor control of typing), or of underlying processing constraints (eg. the approximate capacity of human short term memory).

Use of the Technique

Users of this analysis technique need to divide the user's tasks into the seven stages and see what support the system gives to each stage. The mappings from the system to the users' model must be as direct as possible, another. Approximate tools for assessing the value of these trade offs are also suggested but they are outside the scope of this review. At present, the technique is best viewed as a tool for thought rather than a rigorous form of task analysis. Since the technique clearly requires a basic knowledge of cognitive psychology, users of the technique will need a working knowledge of cognitive skills and capabilities relevant to the tasks and systems being analysed. Although not as explicit as some of the analyses presented in earlier sections, it does provide a way of representing what a user might actually being doing. In this respect, human factors specialists have commented that this form of analysis makes contact with their immediate concerns more readily than more formal approaches (eg. Whiteside & Wixon, 1987).

xi) Cognitive Task Analysis (CTA)

Like Norman's approach, Cognitive Task Analysis (Barnard, 1987) focuses on the nature of mental activity rather than on tasks themselves. The general idea is that a theoretically based decomposition of cognitive resources can be used to describe mental activity associated with dialogue tasks.

Following the Interacting Cognitive Subsystems (ICS) approach to human information processing (Barnard, 1985), it is assumed that the cognitive system is divided into subsystems, each of which processes information in a particular mental code (eg. propositional). Each subsystem has associated with it a local 'image record' which stores representations in its own code. A subsystem also contains translation processes that recode its input into the codes that are used by other subsystems or by specific effectors.

Sensory subsystems (e. g. Visual, Acoustic) generate codes that can be processed by representational subsystems, which in turn handle higher level descriptions of linguistic and visual structure, as well as meaning and its implications. Effector subsystems (eg. articulatory, limb) translate the representational codes into codes for controlling particular effectors (eg. hands, speech musculature).

The description of mental activity contains four components: the mental processes required; the procedural knowledge embodied in those processes; the contents of relevant memory records that may be accessed; and the way in which the cognitive mechanism will be dynamically controlled during task execution.

The first component, the configuration of processes, does no more than describe what is required for a given phase of cognitive activity. In reading, for example, a visual representation is translated into an object encoding, which is subsequently recoded through a representation of linguistic form into a propositional representation of the meaning of the information and so on. The set of processes required defines the configuration for reading. A different set of processes would define the configuration for say typing text.

The second component, procedural knowledge, describes the properties of each individual process within the configuration. For example, this component out a particular mental recoding required by the task. So, linguistic processes cannot automatically recode novel command words (eg. BLARK) into a propositional representation of their meaning.

The third component, record contents, specifies the properties of memory records that are likely to be accessed during task execution. This includes a representation of the task to-be-performed. In some circumstances users may rely on a semantic form of encoding (eg. a propositional representation). Under these circumstances they may confuse elements with similar meaning (such as "Display" and "Show") and have trouble resolving the sequencing of dialogue constituents. In other circumstances they may rely on a representation of linguistic form which specifies sequencing but which is open to confusions between constituents that sound similar (such as the filenames "Nine.txt" and "Mine.txt").

The first three components describe properties of processes and representations. The fourth component, dynamic control, describes the flow of information among processes. This component specifies both the nature and extent of the transactions among processes that occur during task execution. Thus, the generation of a dialogue sequence may require resolving the identities of constituents and their order. This may be achieved by inferential activity or by recovering specific memory records or by some combination of both.

The description of dynamic control is arrived at by applying rules which interrelate the components. These rules are derived from empirical phenomena of system use (eg. Barnard, 1987). For example, where the knowledge required for a task is incompletely proceduralised, or where record contents have a high degree of order uncertainty associated with them, then the complexity of dynamic control will increase and may take on a different form.

The complete four component description makes up a cognitive task model and the attributes of such models are used to predict user behaviour at novice, intermediate, and expert levels. The complexity of dynamic control required by a task is, for example, assumed to relate to overall ease of learning and performance.

This form of task analysis is based upon a theoretical decomposition of cognitive resources and heuristic principles which describe their functioning. Both the decomposition and the principles can be explicitly represented in the knowledge base of an expert system. Hence, the process of analysis and prediction can be automated (see Barnard et al., 1987). To date, illustrative principles have been derived to deal with a limited range of issues associated with command names, structures and menu dialogues.

Knowledge

In Cognitive Task Analysis knowledge is described in two forms - the form required for mental

recodings (procedural knowledge) and the form required to represent task specific concepts and sequences (record contents). The analysis can therefore take into account a broad range of user knowledge, including knowledge of other systems, natural language, and domain versions of tasks. The important feature of this analysis is that knowledge is described in an approximate form in a way that relates directly to mental of order uncertainty via inference). It seeks to represent the use of knowledge in performance rather than providing a formalism for representing the structure of required system knowledge. Task action grammars, in contrast, attempt to represent the structure rather than the use of knowledge.

User Centered Task Dynamics

The CTA is capable of characterising most aspects of task dynamics. Goals are implicitly specified in this model as information structures held in propositional memory records. Short term aspects of task dynamics are captured in terms of phases of cognitive activity and their contents. Longer term changes that take place during skill acquisition are dealt with by building different models for novice, intermediate and expert users.

Cognitive Limitations on Processing

CTA is first and foremost a means of making relationships explicit between approximate knowledge representations and cognitive limitations on their mental processing. The characteristics and limitations are specified in terms of the properties of mental codes; restricted capabilities for coordinating and controlling processes which handle those codes; and more specific limitations such as recency and description effects in memory retrieval. The approach essentially provides a language in which such constraints can be specified. The language refers to processes and coded mental representations which can be described in terms of their attributes. In its present form, only a limited range of attributes and constraints are actually utilised. They can, however, be added to as further analyses of user performance provide additional empirical justification for extending that range.

Use of the Technique

CTA can be used in two ways by different kinds of analysts. One class of analyst needs to analyse tasks in detail and establish the principles which interrelate the four components of the task model (cf. Barnard, 1987). Clearly, this type of analyst needs both to be a specialist in the cognitive sciences and to have a detailed working knowledge of the model-building process. The products of this class of usage are principles of cognitive task modelling; mappings from those principles to particular classes of application or system; and mappings on to properties of user behaviour. These principles and mappings can then be incorporated in the knowledge base of an expert system, as has been shown by Barnard et al., 1987.

The second class of analyst is the user of the resulting expert system. One of the attractive features of this approach is that the expert system users do not need detailed knowledge of the model building process. They simply need to be in a position to answer the particular queries for information that the expert system needs in order to build its model (Barnard et al, 1987). In its present form, the expert system user does need a working knowledge of cognitive psychology in order to answer all the queries in a behavioural analyst rather than a system designer.

The authors see this approach as the most extreme position yet reached in a trend that will affect other models. Understanding human behaviour requires detailed consideration of many different processes. The underlying models therefore become increasingly detailed and accessible only to

specialists. While one solution is to look for powerful approximations (such as the GOMS approach), our belief is that it will be more successful to develop intelligent design aids, such as the expert system tool that is under development via this technique.

CONCLUSIONS

At the start of this chapter we observed that the interaction of people with computers offers novel problems for task analysis, and in the course of this chapter we have reviewed the various attempts at solving these new problems. The eleven techniques of task analysis summarised in this chapter are not simple rivals, all trying to achieve the same results in the same ways with varying degrees of success. Although, for expository purposes, they have been grouped into four groups according to the starting points they take for their analysis, the techniques differ also in their focus, their output, and their usability to the non-specialist. There is no single "best" technique. As Olson (1987) has pointed out, we have fragments of the kinds of representations required to meet the full needs of researchers and practitioners.

Users of task analysis do, nevertheless, have to choose among the fragments to get the form of representation that best suits their particular requirements. But is any method acceptable at present? Despite the wide variety, there are obvious and important problems that none of these techniques satisfactorily address at present, so we shall start this final section by reviewing some shortcomings and conclude with some remarks concerning future prospects.

Shortcomings of Present Techniques

None of the techniques discussed provides very much in the way of a treatment of cognitive strategies or individual differences. This is likely to be an important area of development since there is a body of research (eg. Egan and Gomez, 1985; Greene et al, 1987) which suggests that there are different learning styles between individuals as well as differences in various cognitive abilities. TAG, for example, explicitly excludes any mechanism to describe how users choose their strategy to achieve their goals. In contrast, GOMS provides at least some basis for strategic variation via selection rules based on actual user preferences incorporating empirical data. There is a clear need to develop analyses which can more readily cope with individual and strategic variation.

Similarly, user motivation - including the social motivations to work is ill-represented as are the constraints of office and organisational life (but see Olson, 1987; Malone, 1987). Studies of office automation have demonstrated that the attitudes of workers to the introduction of technology affect how that technology is used (eg. Long et al., 1983; Macaulay et al., 1985), yet only one of the analyses considers higher-level motivations in any significant degree.

Assessments of reliability and validity are another weak spot. Where predictions are offered there is little data as yet on the general validity. In addition, most of these techniques have only been used by their originators. How would they fare in other hands? Would other people produce the same analyses, in the analysis method used by Kieras and Polson, where the number of production rules is one of the metrics of complexity. Here the style in which the rules are written determines their number. Unless the style can be acquired by other analysts, the methodology will be difficult to standardize.

Finally, different techniques require different analytical skills. Some techniques are expected to be easy for designers to use, eg. the Keystroke Level Model; but others, notably the Young/Hull analysis and Wilson et al, effectively require training in cognitive science. This makes it hard to transfer methodologies from the research community to the design community, who are the potential users of these techniques. One solution may be to automate them as executable formalisations, as Kieras and Polson have done and as the authors of TAG are currently doing, or as automated decision aids, as is being done with CTA. The use of intelligent interfaces to these techniques (see

Barnard et al, 1987) may offer the best hope of making the more complex techniques directly available to designers rather than indirectly through behavioural specialists.

Pragmatic Issues

Choice of a technique must also be influenced by such pragmatic issues as ease of use, and sometimes a balance must be struck between pragmatic and conceptual advantages. For instance, there are two techniques aiming at knowledge specifications, TAKD and CLG. Of these, TAKD is more general and less cumbersome; but it also less formalised, and unlike CLG it does not include specific components to help in the structuring of design specifications.

Similarly, within the group of four techniques aiming at complexity predictions, increase in conceptual power must be balanced against predictive tightness. At one end, GOMS uses only the knowledge in the design, and produces absolute estimates of performance time - "It will take 3.45 seconds for an skilled user to perform this task using this system". Kieras and Polson increase the conceptual power by including knowledge of other systems, to estimate times for transfer of learning as well; ETIT tries to capture world knowledge of the task and makes estimates of comparative complexity of learning systems. At the far end, TAG introduces knowledge of natural language and of family resemblances between commands, but is limited to estimates of relative difficulty in learning systems - "This system will be harder to learn than that".

Choice of Focus

The techniques we have surveyed have, broadly speaking, three kinds of focus: ideal knowledge representation, user-centered task dynamics with less than ideal knowledge, and cognitive activities. We are now in a position to offer some observations on these possibilities.

A focus on knowledge representation seems to lead to techniques which are relatively quick to apply but are limited in predictive range. The simplest a system design. This clearly restricts predictions to error-free optimal performance. Introducing other types of knowledge requires more time and effort to verify the knowledge, and its completeness is always doubtful, so that increasing generality also leads to increased uncertainty. These approaches use models of performance which collapse over short term transitions in processing so that they cannot predict which particular errors will be made, only that one system will be more complex than another and hence that ’more’ errors will be made or that ’more time’ will be spent learning it.

However, these estimates of performance time and judgments of complexity can be used as soon as a design is available; they do not have to be based on a running system, and they are comparatively quick to use, so they can be applied early in the design cycle and their output can be used to guide design decisions before the systems are programmed (cf. Card et al., 1983).

A focus on user-centred task dynamics, associated in this survey with the work of Miller, Young & Hull, and Wilson et al., gives a more detailed understanding of why particular errors occur. As such these techniques can potentially provide more directive input concerning design attributes. However, they are time consuming, since they are based on actual behaviour, and they also require a running system, a simulation, or a comparable product for data collection. (Note that even if a prototype is used, it must still present a very accurate version of the intended user interface.) The time and effort required during the design cycle may not be worthwhile (Hammond et al., 1983).

A focus on the constraints of the human information processing system is given by the work of Norman and of Barnard. At present these are best described as conceptual frameworks which require

considerable development to produce mature design aids.

Prospects

In the last analysis, user-system behaviour must be determined by a combination of task requirements and the ways in which the human information processing system copes with them. Accordingly, it seems vital to develop integrated approaches that can encompass different forms of knowledge requirements as well as information processing activity (Barnard, 1987): and if the techniques are to be useful outside the laboratory, mature ways of transferring the required skills and knowledge to the design community or to others who wish to use the techniques.

For the knowledge based techniques to move towards integration, they must identify sources of knowledge which are likely to interfere with ideal system knowledge during task performance. Then they must specify this in a way that does not rely on experts' abilities to identify relevant information for each analysis. Having done this, they require a mechanism to describe the user's mental processing that takes place during performance so that they can predict which errors will occur as well as offer comparative judgements of complexity.

the starting point point required by its techniques, and the form of its output. To these must be added the more pragmatic concerns of the requirements for particular skills, knowledge and effort. Little enough attention has been paid to these pragmatic considerations to date. Before the discipline can be called mature, we shall need to study these pragmatic issues in far more depth.

References

Allen, R. B., & Scerbo, M. W. (1983). Details of command-language keystrokes. ACM Transactions on Office Information Systems, 1, 159-178.

Anderson, J. R. (1983). The Architecture of Cognition. Mass: Harvard University Press.

Baddeley, A. D. (1983). Working memory. Philosophical Transactions of the Royal Society, Series B, 302, 311-324

Barnard, P. J. (1985). Interacting cognitive subsystems: a psycholinguistic approach to short-term memory. In A. Ellis (ed.), Progress in the Psychology of Language, Vol II., chap 6, Hillsdale, N. J. : Lawrence Erlbaum Associates, 197-258.

Barnard, P. J. (1987). Cognitive resources and the learning of human-computer dialogues. In J. M. Carroll (ed.), Interfacing Thought: Cognitive Aspects of Human-Computer Interaction, chap 6, Cambridge, Mass. : MIT Press, 112-158.

Barnard, P. J., Wilson, M. D. and MacLean, A. (1987). Approximate Modelling of Cognitive Activity: towards an expert system design aid. In: Proceedings of CHI + GI 1987 (Toronto April 5-9), New York: ACM, 21-26.

Bullen, C. V. and Bennett, J. L. (1983). Office workstation use by administrative managers and professionals. IBM Research Report RJ 3890.

Card, S. K. (1978). Studies in the psychology of computer text editing systems. Report no: SSL-78-1. Palo Alto, Calif. : Xerox Corp.

Card, S. K., Moran, T. P. & Newell, A. (1980). Computer text-editing: an information processing

analysis of a routine cognitive skill. Cognitive Psychology, 12, 32-74.

Card, S. K., Moran, T. P. & Newell, A. (1983). The psychology of human-computer interaction. Hillside, N. J. : Lawrence Erlbaum Associates.

Carroll, J. M. (1982). Learning, using and designing command paradigms. Human Learning, 1, 31-62.

Carroll, J. M. (1984). Mental Models and software human factors: an overview. IBM Watson Research Centre, Research Report RC 10616 (#47016).

Davis, R. (1983). Task analysis and user errors: a methodology for assessing interactions. Int. J. Man-Machine Studies, 19, 561-574.

skill acquisition. PhD. dissertation, Stanford University.

Duncan, K. D. (1974). Analytical techniques in training design. In Edwards, E. & Lees, F. P. (eds.) The Human Operator in Process Control. London: Taylor & Francis.

Egan, D. E. and Gomez, L. M. (1985). Assaying, isolating, and accommodating individual differences in learning of complex skill. In R. F. Dillon, (ed.) Individual Differences in Cognition, vol 2, London: Academic Press, 174-217.

Embley, D. W., & Nagy, G. (1982). Can we expect to improve text editing performance? In: Human Factors in Computer Systems, (Gaithersburg) , New York: ACM 152-156.

Fleishman, E. A., & Quaintance, M. K. (1984). Taxonomies of Human Performance. San Diego, Calif. : Academic Press.

Green, T. R. G., Schiele, F. & Payne, S. J. (1986). Formalisable models of user knowledge in HCI. This volume.

Greene, S. L., Cannata, P. E., Gomez, L. M. and Devlin, S. J. (1987). A novel interface for database query: no IF's, AND's or OR's. In: Proceedings of Bellcore Database Symposium (Sept 15-17, 1987), Bellcore Special Report.

Hammond, N., MacLean, A., Hinton, G., Long, J., Barnard, P. and Clark, I. A. (1983). Novice use of an interactive graph plotting system. IBM Hursley Human Factors Report #HF083. IBM UK Laboratories Ltd., Hursley Park, Winchester, UK.

Johnson, P. (1985). Towards a task model of messaging: an example of the application of TAKD to user interface design. In P. Johnson and S. Cook (eds.), People and Computers: Designing the Interface. Cambridge: Cambridge University Press, 46-62.

Johnson, P., Diaper, D. & Long, J. B. (1984). Tasks, skills and knowledge: task analysis for knowledge based descriptions. In B. Shackel (ed.), Human-Computer Interaction - INTERACT '84. Amsterdam: North Holland, 499-503.

Johnson, P., Diaper, D. & Long, J. B. (1985). Task analysis in interactive system design and evaluation. Paper presented to the IFAC/IFIP conference on Design and Evaluation Techniques for Man-Machine Systems, Italy.

Kieras, D. E. & Polson, P. G. (1985). An approach to the formal analysis of user complexity. Int. J. of Man-Machine Studies, 22, 365-394.

Long, J., Hammond, N., Barnard, P. and Morton, J. (1983). Introducing the interactive computer at work: the user's views. Behaviour and Information Technology, 2(1), 39-106.

Maclean, A., Barnard, P. J. & Wilson, M. D. (1985). Evaluating the human interface of a data entry system: user choice and performance measures yield different tradeoff functions. In P. Johnson and S. Cook (eds.), People and Computers: Designing the Interface. Cambridge: Cambridge University

Macaulay, L. A., Fowler, C. J. H. & Porteous, R. (1985). What do clerical workers think about computers? In P. Johnson and S. Cook (eds.), People and Computers: Designing the Interface. Cambridge: Cambridge University Press, 290-298.

Mack, R., Lewis, C. and Carroll, J. (1983). Learning to use word processors: problems and prospects. ACM Transactions on Office Information Systems, 1, 254-271.

Malone, T. W. (1987). Computer support for organisations: towards an organisational science. In J. M. Carroll (ed) Interfacing Thought: Cognitive Aspects of Human-Computer Interaction. chap 11, Cambridge Mass: MIT Press, 294-324.

Miller, R. B. (1962). Task description and analysis. In R. M. Gagne, Psychological Principles in System Development. New York: Holt, Rinehart, & Winston.

Miller, R. B. (1967). Task taxonomy: science or technology? In W. T. Singleton, R. S. Easterby & D. C. Whitfield (eds), The Human Operator in Complex Systems. London: Taylor & Francis.

Miller, R. B. (1973). Development of a taxonomy of human performance: Design of a systems task vocabulary. JSAS Catalog of Selected Documents in Psychology, 3, 29-30 (Ms. No. 327).

Miller, R. B. (1974). A method for determining task strategies (Tech. Rep. AFHRL_TR_74_26). Washington, D. C. : American Institutes for Research.

Moran, T. P. (1978). Introduction to the Command Language Grammar. Report no: SSL-78-3. Palo Alto, Calif. : Xerox Corp.

Moran, T. P. (1981). The Command Language Grammar, a representation for the user interface of interactive computer systems. Int. J. Man-Machine Studies, 15(1), 3-50.

Moran, T. P. (1983). Getting into a system: external-internal task mapping analysis. In: CHI '83 Conference on Human Factors in Computing Systems (Boston). New York: ACM, 45-49.

Morton, J., Barnard P. J., Hammond N. V., & Long J. B. (1979). Interacting with the computer: a framework. In E. J. Boutmy & A. Danthine (eds.), Teleinformatics 79. Amsterdam: North- Holland, 201-208.

Norman, D. A. (1984). Stages and levels in human-computer interaction. Int. J. Man-Machine Studies, 21, 365-375.

Norman, D. A. (1986). Cognitive engineering. In D. A. Norman and S. W. Draper (eds), User Centered System Design, Hillsdale, N. J. : Lawrence Erlbaum Associates, 31-62.

Olson, J. R. (1987). Cognitive analysis of people's use of software. In: J. M. Carroll (ed), Interfacing Thought: Cognitive Aspects of Human Computer

Payne, S. (1984). Task-action grammars. In B. Shackel (ed.), Human-Computer Interaction - Interact '84, Amsterdam: North Holland, 527-532.

Payne, S. (1985). Task-action grammars: the mental representation of task languages in human-computer interaction. Unpublished PhD. Dissertation, University of Sheffield, U. K.

Payne, S. & Green, T. R. G. (1986) Task-action grammars: a model of the mental representation of task languages. Human Computer Interaction, 2
(2) 93-133.

Polson, P. (1987). A quantitative theory of human-computer interaction. In: J. M. Carroll (ed), Interfacing Thought: Cognitive Aspects of Human-Computer Interaction, chap 8. Cambridge Mass: MIT Press, 184-235.

Polson, P. G. & Kieras, D. E. (1985). A quantitative model of the learning and performance of text editing knowledge. In Proceedings of CHI '85 Human Factors in Computing Systems (San Francisco), New York: ACM, 207-212.

Rasmussen, J. (1986). Information Processing and Human-Machine Interaction. Amsterdam: North Holland.

Reisner (1981). Formal grammar and design of an interactive system. IEEE Trans. Software Engineering, 7, 229-240.

Roberts, T. L. (1979). Evaluation of computer text editors. PhD. Dissertation, Stanford Univ., Stanford, Calif.

Roberts, T. L. & Moran, T. P. (1982). Evaluation of text editors. In Proceedings of Human Factors in Computer Systems (Gaithersburg), New York: ACM, 136-140.

Roberts, T. L. & Moran, T. P. (1983). The evaluation of text editors: methodology and empirical results. Comm. ACM, 26 (4), 265-283.

Sasso, W. A. (1985). A comparison for two potential bases for office analysis: function and organizational unit. Dissertation from the Graduate School of Business Administration, the University of Michigan.

Whiteside, J. and Wixon, D. (1987). Discussion: improving human-computer Interaction - a quest for cognitive science. In: J. M. Carroll (ed), Interfacing Thought: Cognitive Aspects of Human-Computer Interaction, Cambridge, Mass: MIT Press, 353-365.

Wilson, M. D., Barnard, P. J. & MacLean, A. (1985). User learning of core command sequences in a menu system. IBM Hursley Human Factors Report #HF114. IBM United Kingdom Laboratories Ltd., Hursley Park, Winchester, UK.

Young, R. M. & Hull, A. (1982). Cognitive aspects of the selection of viewdata options by casual users. In M. B. Williams (ed.), Pathways to the Information Society. Amsterdam: North Holland, 571-576. Young, R. M. & Hull, A. (1983). Categorisation structures in hierarchical menus. In: Proceedings of the the Tenth International Symposium on Human Factors in Telecommunications, Helsinki, 111-118.