

Task Knowledge Structures: Psychological basis and integration into system design.

Hilary Johnson and Peter Johnson

Department of Computer Science
Queen Mary and Westfield College
University of London.
Mile End Road
London
E1 4NS

Johnson, H. & Johnson, P. (1991) Task Knowledge Structures: Psychological basis and integration into system design. *Acta Psychologica*, 78 pp 3-26.

Abstract

A major goal of HCI is to assist software designers to construct useful and usable computer systems. One way to approach this problem is to provide software designers with information about the knowledge users utilize in performing tasks. This paper first outlines the psychological basis for our approach to task modelling, Task Knowledge Structures (TKS) and then considers how to achieve the ultimate objective in developing TKS, i.e to integrate task models into system design by providing information about users and tasks at appropriate stages in design. To facilitate the integration of TA into system design, a survey of designers was conducted to identify where, when and how task data can be represented to aid integration into system design and the designers' requirements for tools to support task model and system design integration. The results from the survey are then summarised and the designers' requirements for TA tools are outlined, followed by a discussion as to how these requirements might be met. Finally, brief details of design specifications for the proposed tools are presented.

Introduction.

The main concern of researchers in human computer interaction (HCI) is to develop models, methods and techniques to assist software designers to construct computer systems which people find useful and usable. One way to approach this goal is to assume that knowing something about how users approach and carry out tasks will aid software designers when making design decisions which will ultimately affect computer system usefulness and usability. As a result task analysis has emerged as an important aid to early design in HCI. It provides an information source from which design decisions can be made and a basis for evaluating designed systems. Task analysis is an empirical method which can produce a complete and explicit model of tasks in a domain, and of how people carry out those tasks. It focuses design on users' tasks and goals, and the methods for achieving those goals. Design focused on these aspects of a user's task domain can result in improved, more usable systems. This is because rather than leaving design for usability to luck and intuition, task analysis methods can be utilized to

inform the designer about those factors concerning users' tasks that can influence usability in advance of the designers making inappropriate design decisions.

Rosson et al (1988) have argued that designers consider obtaining information about users and tasks as a major contributor to the generation of design ideas. They distinguish between design idea generation and design development and suggest that design idea generation is facilitated by task/user analysis tools and techniques, while design development is facilitated through the use of specification and prototyping tools. By carrying out task analyses, users and tasks will be taken into account in the formation of the design idea. If this is the case then interface and system design will take place from an informed position on the part of the designer, resulting in the need for less iterations.

Although task analysis is the investigation of what people do when they carry out tasks, an approach to task analysis involves more than simply observing how people perform tasks. It also involves developing a theory of tasks, techniques of data collection, a method of analysing tasks and a representational framework for constructing task models. This paper divides into two halves respectively, reflecting our two main objectives, which are first, to outline in detail the psychological basis for our theory of tasks as this underpins both the TKS methodology and the design of the task analysis tools. This is covered in the first part of the paper. The second half of the paper is concerned with our second objective which is to produce tools to support designers undertaking task analyses and consequently integrating task data into system design. Although the composition of the two halves of the paper may appear distinct, it is important to note that the psychological basis for TKS governs both the nature of the data collected and modelled and the specifications for the TA tools.

In the second half of the paper we first briefly summarise a detailed but small-scale survey undertaken to identify both design practices and designers' requirements for task analysis tools. The final part of this section briefly describes our initial thoughts about

how we might attempt to design the TA toolset and to see if the designers rather general requirements can be met. For a description of the data collection and analysis methods and constructing task models, see Johnson et al (1988a). Our approach to task analysis known as Task Knowledge Structures (TKS) has been developed from earlier work on Task Analysis for Knowledge Descriptions (TAKD), Johnson, Diaper and Long (1984). It is to be contrasted with task analysis techniques which are not concerned with knowledge, such as ability profiling (Fleishman and Quaintance, 1984), Hierarchical Task Analysis (Annett and Duncan, 1967), and other techniques which have an evaluative role in assessing the complexity of task performance but have no explicit method of task analysis.

Information about how users plan and carry out goal-directed behaviour is one aspect of users and tasks that can be identified and analysed. Probably the most often cited approach to TA in HCI is the GOMS approach of Card et al (1983). The underlying philosophy of GOMS is that tasks can be thought of in terms of goal structures, operations, methods and selection rules. Kieras and Polson (1985) extend the GOMS approach by arguing that production rules can be used to model these goals, operations, methods and selection rules. The work of Kieras and Polson (1985), Payne and Green (1986) and Card, Moran and Newell (1983), are good examples of current evaluative approaches in HCI which incorporate methods of either predicting the time taken to carry out a task or the degree of difficulty of using an interactive computer system, and assume some form of task model. Each of these approaches is capable, in varying degrees, of making recommendations about how proposed system designs can be used in terms of the ease with which users could perform given tasks.

There are two important features to these approaches which distinguish them from TKS; first, they are designed purely as evaluative tools, and consequently are not concerned with design generation. By this we mean that these approaches assume that decisions about what tasks the system should support have been made elsewhere, and also that one or

more design solutions have already been proposed. Second, they focus on the evaluation and prediction of user performance and do not detail any particular method of task analysis. In contrast, TKS is concerned to identify the knowledge requirements of tasks and is aimed at assisting in the generation of design solutions although TKS may also form part of an evaluation methodology.

It should be noted that the range and complexity of tasks with which we have been concerned in the development of TKS have been beyond the level of simple keyboard tasks, neither is the approach restricted to physical tasks. We are concerned with tasks as complex and rich as designing the room layout of houses, producing graphs, tables and multi-media documents, producing group documents, running meetings and controlling sophisticated building surveillance equipment.

I. Psychological basis for Task Knowledge Structures (TKS)

In previous papers, most notably in Johnson and Johnson (1987) and Johnson et al (1988b) we have briefly outlined our theoretical assumptions, derived from research in cognitive psychology and empirically supported, which underpin Task Knowledge Structures. Three assumptions have been made, concerning treating tasks as concepts to be represented in memory, the structure of entities in tasks and the differential status of those entities in terms of their representativeness or typicality. In this paper we delve a little deeper into the basis and evidence for these assumptions.

To reiterate our first assumption, we believe that TKSs could be represented as concepts, for instance the TKS “design a house room layout” could be represented as a concept. These concepts would be stored in human memory in the same way as are other concepts. The basis for this assumption is that various types of knowledge are represented and structured in memory, for skilled performance (Anderson, 1976), motor control (Johnson, 1982), language acquisition and comprehension (Karmiloff-Smith, 1986a), hierarchical planning (Cromer,

1986), problem solving (Karmiloff-Smith, 1986b), categorisation (Johnson, 1983) and so on, and it seems unlikely that task knowledge is the exception given the goal driven nature of some of the previously referenced work. Additionally, evidence that knowledge about events is represented in memory comes from Schank (1982) who assumes that knowledge of frequently occurring events is structured into meaningful units in memory, as either scripts, or memory organisation packets. Minsky (1975) also assumes that knowledge about entities is represented in memory in the form of schemas or frames. Further support for our assumption that tasks are represented as cohesive units in memory comes from the work of Galambos (1986) who conducted a series of experiments which show that people recognise and use structures of events, such as the order, the sequence and the importance of activities within the event sequence to understand, explain and make predictions about these events. Graesser and Clark (1985) also provide support for the notion that task knowledge is represented in memory. They assume that general knowledge structures which represent goals to causal and enabling states, plans for achieving goals, intermediate states and alternative solutions or paths are represented in a conceptual knowledge structure that is used to interpret events. Finally, Barsalou (1982; 1983; 1987) in discussions of the graded structure of categories, (i.e. from good to poor examples) has argued that there are a number of different types of concept, from common taxonomic categories to adhoc and goal-derived categories. In our view the idea of goal-derived categories corresponds with our assumption that there are concepts directly related to tasks which have specific goals that can be utilized in task performance.

Although we might cite the work of Barsalou as an indication that there is some psychological validity for TKSs, we do not share his views about the data-driven nature of categorisation being entirely in the manner of similarity-comparison processes in working memory. Barsalou (1987) puts forward the proposition that conceptual knowledge is not stored in long term memory. There are two implications from this proposition; first, there are no knowledge structures related to concepts which are permanently stored in memory which can be accessed and processed on different occasions. Second, as we have already

stated, categorisation is treated in a data-driven manner, not involving long term memory. Therefore, stimuli is interpreted and processed in working memory on each separate occasion, with each occasion being treated as independent. Barsalou's assumption that conceptual knowledge is not stored in long term memory arises from his observation that category typicality and representativeness are of an unstable nature, varying with different contextual information both within and between individuals from one occasion to another. Johnson (1983) has observed a similar phenomenon, and has argued that the context effects arise due to multiple representation of some entities in memory, further exacerbated by different effects arising due to different purposes of carrying out categorisation. This latter point has been eloquently discussed by Medin and Wattenmaker, (1987), Lakoff (1987) and McCauley (1987) who from different perspectives all argue for the use of idealized cognitive models and theories of the world as a basis for classification. The main thrust of their arguments is that depending on the models or theory of the world we are following at the time, this will affect how we classify. McCauley in particular is outspoken in his condemnation of views of categorisation, as in Barsalou (1987) and Brooks (1987) which espouse no centralised, or a decentralised view of categorisation. A centralized view of categorisation assumes that there are higher order structures which might guide categorisation (e.g. idealized cognitive models or theories of the world) and thus explain the empirical findings. On the other hand the decentralized view attempts to explain the finding that entities might be judged typical on one occasion and yet not on others, as indications of at best, the instability of conceptual structures and at worst, the possibility of no conceptual structures in long term memory. In terms of task knowledge structures we believe that overriding goals and the tasks that accomplish them will provide one aspect of the user's current model of the world especially since most behaviour is goal driven. We explicitly assume that TKSs are represented in long term memory. TKSs represent the different aspects of knowledge that are possessed by a task performer and as such are a convenient summary representation, constructed by the task analyst, to represent the task knowledge held by one or more task performers and which once constructed will provide a basis for task-centred system design.

What are the benefits of assuming that tasks can be represented as concepts? The main benefits are in the extrapolation of the utility of conceptual organisation to task knowledge, i.e. benefits associated with assuming tasks have an internal structure and that task entities differ in how representative they are of the task. These related assumptions are brought into play because we need to be able to describe and predict how the user will carry out tasks and the relationships between the various aspects of task knowledge. The structure in tasks should give us some indication of the task entities which go together (are carried in parallel), which prime, precede or follow on from one another. Tasks would be unstructured if the task world formed a set of tasks in which all possible elements and attributes of tasks could co-occur with equal probability combined with all other possible elements or attributes of tasks. This is obviously not the case; task elements or behaviours do not occur independently of one another. Some pairs or even n-tuples of task elements are quite probable whereas others are improbable; some groupings of attributes while being logically possible may never occur in reality. Elements of tasks are generally carried out according to some feasible temporal ordering, designated by a plan. An example of feasible temporal ordering might be the case of a builder who is building a house who cannot begin to build until the bricks have arrived. An architect designing the layout of a house cannot design the upstairs layout until s/he knows how many bedrooms are required. The same architect might have to simultaneously consider certain other related task elements. For instance, in designing bathroom layout, the respective position of the bath is considered at the same time as the positions of the wash-hand basin and toilet. Empirical evidence for the representation of plans in long term memory can be found in the work on programming tasks (Green et al, 1987). Further evidence for structuring in tasks comes from the work of Byrne (1977) on tasks in domains such as cookery and meal planning. Having identified the plan and any alternative plans to be activated depending on the satisfaction of each plan's pre and post-conditions, we can then acquaint the software designer with these task details in order that entities which precede or prime one another can be provided for in an appropriate manner, for instance spatially close on a single screen, or as automatic activation of

one after the other, and so on. (For fuller implications of task structure for system design see Johnson et al, 1988b).

Our view of task structure is similar to that of Garner (1974) in describing how people represent physical objects based on the structure of those objects in the real world. However, the structure of objects in the real world, in terms of their correlated features and the psychological representation of those features has been a matter for debate recently, see Neisser, (1987) for a direct perception view of perception of features. In considering the structure of tasks in terms of correlated features, i.e. in terms of things that go together because they might share some particular features, we must address the wider problem of why some things go together and why some do not. Medin and Wattenmaker (1987) argue that what makes a concept cohesive has received a variety of answers mainly to do with similarity of features. However, they argue that similarity based approaches to conceptual coherence are insufficient to explain the richness of categorisation. They argue for a theory based approach which emphasizes that coherence derives from both internal structure of a conceptual domain and the position of the concept in the complete knowledge base. According to Medin and Wattenmaker concepts are viewed as embedded in theories and are coherent to the extent that they fit people's background knowledge or naive theories of the world. If we view goal driven behaviour, executed in tasks as one aspect of people's naive theories of the world, then we need to look no further for reasons that task entities go together or are related, than the fact that together they comprise a coherent view of how the task is carried out. More specifically, entities are related through a common goal, irrespective of whether or not they share common perceptual or functional features.

The third assumption which we have made, and which also is a direct implication of treating task knowledge as conceptual knowledge is that some task entities are more representative of the task than are others. The psychological basis for this argument arises from research into categorisation in cognitive psychology in the last fifteen years. In

this space of time the traditional view of categorisation has become systematically challenged by both the probabilistic and exemplar views of categorisation. The traditional or classical view dates back to Aristotle and treats categories as having necessary and sufficient criteria which determine whether or not a particular instance is a category member or not. Classification viewed in this light is taken to be all-or-none. "All" meaning that the instances have all the necessary and sufficient criteria, and "none" for those instances not having the necessary and sufficient criteria. The probabilistic and exemplar views challenged the traditional view on the grounds that some category members appeared to be more representative or typical of a category than did others, (Rips, Shoben and Smith, 1973; Rosch 1973). The empirical evidence for the claim that exemplars differ in typicality comes from Rosch et al (1976), McCloskey and Glucksberg (1979) and Smith (1978) who argued that the grading examples from good to poor examples, is central to predicting how long it takes to classify something as a category member with typical exemplars being identified faster than atypical exemplars. Graded structure is also central to the observation that typical exemplars are generated more often than atypical exemplars (Rosch et al, 1976). Typicality also plays a role in learning where typical exemplars are learnt more quickly than atypical instances. For a full review of the three different approaches to categorisation and their theoretical implications, see Smith and Medin (1981) and Johnson (1983).

In terms of TKS we assume that knowledge about objects and their associated actions differ in how representative they are of the TKS of which they are a part. We argue that in the same way that a "robin" might be a typical instance of the "bird" category so might "house plan" be typical of the task "design house room layout". In addition the typical task entities are just those which might constitute central task procedures which are important for successful task performance. Empirical psychological evidence for procedural centrality and action/object representativeness in task behaviour has been obtained by Leddo and Abelson (1986) who found that for tasks such as borrowing a book from a library there were

particular task segments which were more central to, and more representative of, going to the library than other segments.

Knowledge in TKSs

A TKS is a summary representation of the different types of knowledge that are recruited and used in task behaviour. A TKS is related to other TKSs by a number of possible relations. For a full discussion of the knowledge in TKSs, see Johnson et al, 1988b.

One form of relation between TKSs (within role relations) is in terms of their association with a given role. A role is assumed to be defined by the particular set of tasks that an individual is responsible for performing as part of their duty in a particular social context (a "job" is one form of social context). A person may take on a number of roles, for example "author", "referee", "teacher" etc. There are tasks associated with each of these roles. There will be a TKS for each task that a role may be required to perform. Those tasks that are related because they are performed by the same role will have the role relation property associating their respective TKSs.

A second form of relation between TKSs (between role relations) is in terms of the similarity between tasks across roles. This occurs when a person is performing a similar task under different roles. For example, a person may carry out the task of "arranging meeting(s)" while assuming the role of "chairman", "client", "husband". Each task may be performed differently in one or other respect. However, a single person assuming all these roles would have a knowledge structure for each task and also knowledge (not necessarily explicit) of the relations between these tasks. Thus the TKSs for each of the "meeting arranging" tasks would have relational links to other "meeting arranging" tasks performed by that person when assuming a different role.

Within each TKS there are other knowledge representations. The goal structure identifies the goals and subgoals contained within the TKS. A goal is assumed to be the desired state of

affairs that a particular task can produce and forms part of a person's conceptualisation of their task world while carrying out a given role. A subgoal is a lower level goal. The goal structure also includes the enabling and conditional states that must prevail if a goal or subgoal is to be achieved. In this way the goal structure represents the plan for carrying out the complete task. A plan is the particular formulation and possible ordering of subtasks that are undertaken to achieve a particular goal. The plan defined in the goal structure is carried out through a procedural structure which contains alternative procedures for achieving a particular subgoal. A procedure is a particular element of behaviour that forms part of a subtask. As there may be alternative sets of procedures for achieving a particular subgoal there are also conditional and contextual groupings of these procedures. In this way strategies are represented. Strategies are particular sequences of procedures. The procedures rely on other knowledge of the objects and actions which when combined constitute a given procedure statement. Actions and objects are the lowest level of elements of tasks.

Action and object knowledge is represented in a taxonomic substructure. Within the taxonomic substructure information about the properties of the action(s) and object(s) are represented. This includes, the representativeness of the object, the class membership and other attributes such as the procedures in which it is commonly used, its relation to other objects and actions, its features (such as what the object can do or possess) and its typicality rating. These are described in more detail in Johnson et al (1988a) along with a number of techniques for identifying and modelling TKS components. Johnson et al (1988b) provide a method for applying these task models to design using examples from real tasks that have been analysed and real designs in complex domains that have been proposed.

Superficially, TKS might appear to have certain similarities to GOMS in particular in respect to the goal structure. However, there are fundamental theoretical and practical differences. To reiterate an earlier point, TKS is distinct from GOMS in that GOMS can be construed as an evaluative method, not pertinent for design generation. TKS is appropriate for design generation because it encompasses techniques for collecting, analysing and modelling the task

knowledge that people possess. This information will then provide a set of requirements which can be taken into account in devising a design solution by indicating which parts of tasks need support. GOMS does not, by contrast, have a methodology for identifying task knowledge, rather it models the knowledge required to use a particular design solution. The theoretical underpinning to TKS is also very different from that of GOMS. We make explicit claims about treating tasks as conceptual structures in memory. Additionally, GOMS makes no assumptions about the structure of tasks, nor does the GOMS approach have any views about object representativeness or procedural centrality. Following the GOMS approach, there is no reason to assume that any particular part of a task is any more central or important to the successful execution of the task than any other, whereas we have provided previously in this paper empirical evidence that such notions do have validity.

Finally, with GOMS usability is related to how long it takes for an expert to complete a task, where elements of that task performance are considered to be additive and there is a component for "thinking time". GOMS, as extended by Kieras and Polson (1985) argues that complexity can be measured by counting production rules. There is no scope in their extensions to GOMS for different classes of rule. Usability in TKS relates to system support for representative and central task elements, where usability is deemed to decrease depending on the level of support for task procedures. Empirical evidence for the improvement in usability afforded by modelling TKSs is provided by Davis (1988) in a small-scale pilot study of graph and table drawing. The first part of the study identified representative objects and actions, central procedures and sequencing of task procedures for the above two tasks across a population of 12 subjects. An experiment was then carried out in which three further groups of subjects were required to undertake graph and table drawing tasks using one of three different interfaces with the same underlying functions.

One interface was structured so that it positively supported representative and central task elements, and task sequencing identified by the TKS modelling stage. The other interface was unstructured; representative and central task elements were supported but

representative objects were not identified with their associated actions and the sequencing of task procedures was not supported. A further control group had an interface which contained neither central nor representative task features, and which had no explicit task structure.

The results of the pilot study showed that subjects found the structured interface easier to use, and also this interface had a higher preference value from subjects. Additionally, the structured interface resulted in quicker task execution, and the resulting graph and table drawings were better quality in that they were more complete. Also the unstructured but central and representative interface design produced better performance than the control group interface, but less than the structured interface group. The findings support our theoretical view that TKSs provide important information about users that can be used to design improved user interfaces.

II. Task analysis and system design.

In the first half of the paper we outlined the psychological basis of TKS, contrasted TKS with other approaches to task analysis and discussed the knowledge which was assumed to comprise TKSs. The theoretical assumptions made in TKS have implications for the composition of the data collection and analysis techniques we have devised, (Johnson and Johnson, 1987a,b). These techniques therefore support system designers in identifying task structure and representative and central task elements, and also in constructing task models (TKSs). For a full discussion of the implications of the psychological basis of TKS for system design we would refer readers to Johnson et al (1988).

Having outlined an approach to task analysis, in this section of the paper we consider how task analysis and TKS in particular might contribute to system design. In order to facilitate the introduction and use of task analysis in system design we need to know what software design practice entails, and also how to provide tools to support designers in carrying out task

analyses. A current view of design processes and practices in system design is provided by Johnson and Johnson (1989) who carried out in-depth interviews with three system designers who were representative of their respective companies. Two purposes of the interviews are of relevance here; first, it gave us a view of what these designers considered to be the potential contribution of task analysis to various stages of system design and thereby, an idealized view of how task analyses might eventually contribute to system design. This idealized view of the potential contribution of task analysis to system design is being taken account of in our extended version of TKS (Johnson and Johnson, 1991a). The second purpose of the interviews, as an initial stage in our intention to produce on-line task analysis tools, was to identify designers' tool requirements. Below we summarise some of the survey findings. (Full details of the survey can be obtained from Johnson and Johnson, 1989).

The designers who participated in the survey thought that the contribution of TA might be:

At the feasibility/initial planning stage:

- *Identifying and documenting any new functions/new tasks the computer may support.
- *Identify potential functionality of the system from user perspective.
- *Identify user population and characteristics of that population.
- *Identify characteristics of interface to be developed.
- *Allocation of function between user and system.
- *Assess scope/degree of larger-scale TA to be undertaken later in development lifecycle.

At the requirement/analysis stage:

- *Identify and document user/UI requirements comprising details about;
- *hierarchical structure of tasks (goals and subgoals)
- *how users achieve goals and subgoals
- *listing and ordering of undertaking task procedures
- *frequency with which particular procedures were carried out by users
- *reasons why and circumstances under which one procedure was used in preference to another
- *inputs and outputs from each procedure
- *events, data used, actions, objects
- *standard set of properties relating to objects and actions, e.g. frequency, time taken, etc
- *expectations the user entertains about the system after user has carried out an operation
- *division between user and system.

At the Design stage/User interface Development/Dialogue design:

- *Provide initial input to guide dialogue and screen design, comprising;
- *details of what users expect to have available to them at any one time;
- *the structure and sequence of their usage of system facilities;
- *the names and form of representation to be given to screen-presented objects and events;
- *information that should be available in given contexts,(i.e. design of screens);
- *structure between contexts, (i.e. mapping between screens)
- *how much to put on the screen at once with reference to number of commands
- * what information should go on screens and the grouping of that information
- *what commands are needed to support user operations and what those commands will be
- *User testing.

At the prototyping stage:

- *Guide initial format and presentation of prototype by indicating how the screens should look
- *Identify data that has to be displayed.
- *Identify operations and sequencing of procedures
- *Ensure dialogue specification is represented in a format that can be understood and verified with end users and to carry this out.

At the validation stage:

- *User testing

At the update and maintenance stage:

- *Identifying, documenting and cataloguing user problems.

This is the idealized view of the potential contribution of task analysis in the future to system design, and this view could be taken as a specific set of requirements of an approach to task analysis, whether on- or off-line. The designers were also asked about the desired characteristics and criteria they thought that the *proposed* TA tools must satisfy; the results are summarised below:

- *easy to use
- *on-line
- *can be used on a small-scale at first then used more extensively later
- *must have a facility for registering and defining objects and actions
- *facility to list characteristics and attributes of objects
- *facility to help in constructing procedures and task hierarchy
- *facility to show task hierarchy and procedures graphically - and allow procedures to be expanded into detailed operations by some diagrammatic convention
- *assist in process of transforming task model into representations used by designers.

Developing a task analysis toolset.

In the final subsection we present our initial thoughts about how a toolset to support designers in developing TKS models could be designed and implemented. Knowledge Analysis of Tasks (KAT) is the name we have given to our framework for task analysis and the resulting toolset. The basis for the development of the KAT tools are the data collection and analysis techniques, and the designers' tool requirements. The designers' tool requirements however are very general and it is not easy for the designers to say exactly what they want. In addition, a particular problem for the designers was in stating how the task data could be represented in an optimal manner such that integration to system design is possible, (see Johnson and Johnson, 1989). Because the designers' tool requirements tend to be very general, we have in the design of the

tools, mainly taken account of what aspects of a TKS it is important for designers to gather data about and which aspects are repetitive and time consuming, such as generating generic (common across occurrences) task elements. However, we feel that it was of benefit to identify what if any tool requirements designers might have, especially as this is seldom done, and thus provide for an informed and principled approach to the development of the tools. We will consider later whether any of these general requirements are met.

It is important to stress that the tools are not yet implemented. Before this takes place we will carry out extensive consultations with designers using paper versions of the proposed tools as mock-ups and story-boards. There is a wide range of techniques that can be employed (Muller, 1991) to ensure that the tools once implemented will be useful and usable. Not only must the tools be easy to use, but their deployment should also result in better designed systems, therefore empirical studies to this end are also planned. (A brief outline of the tool design specifications is given later; for a fuller account, see Johnson and Johnson, 1990).

With respect to satisfying the designers' general tool requirements, the specifications we have put together are for on-line tools thus satisfying one of the designers' requirements. Another requirement was that the tools must be easy for designers to use. As previously stated, the resulting tools would have to be tested extensively in order to assess how easy they were to use. The TA tool specifications incorporate introductory sections to task analysis to aid the system designers. The implemented tools will support the construction of a task model but this can be undertaken to a lesser or greater extent thereby making the tool appropriate for small scale use at first and more extensive use later. An object/action database or dictionary will allow TA tool users to register, define and characterize attributes of objects and actions and their interrelationships. A goal-oriented structure to be completed by designers should, with the addition of appropriate help facilities, aid in the construction and graphical representation of task hierarchies and procedures.

Further requirements for the task analysis tools, resulting from the survey was that the proposed tools must be versatile and flexible, capable of integration with different approaches to, and stages in design, should not impede design stages and should be capable of integration into both structured design methods and other design lifecycles. These requirements stem from our finding that design practices vary considerably from designer to designer, even within the same design team, with some stages being eliminated, grouped together and so on. The content and function of the task analysis tools should provide end-user/task information and users' interface requirements, relate identified requirements back to end-users in a non-technical fashion, and lessen the need for designers to have to rely only on their intuition. The TA tools should also provide the basis for getting user feedback, decrease the number of end-user problems associated with using the system in undertaking tasks and provide a basis for including end-users in quality assurance procedures.

Additionally, the authors required that the tools must support designers in carrying out task analyses and constructing TKSs. A further requirement, in order to aid integration into system design methods is that an independent subset of the proposed tools should be specifically tailored to one of several potential structured design methods. SSADM has been chosen, for a number of reasons, largely beyond our control, as the structured design method for our tool subset. The reason why SSADM was chosen is that it is a UK government standard, and our collaborators at the time, ICL, were in the process of training designers in SSADM. Therefore, there was an opportunity to introduce designers to our tools at the same time. Several approaches could be taken to relating task analysis to system design. It might, for instance, be advantageous to represent task knowledge in one or more of the data representation formats of SSADM and other structured design methodologies, i.e. data flow diagrams, entity life histories and so on. However, when we asked designers if this was appropriate, each of them indicated that the task data should be kept separate. The approach which was taken by us was to identify for all phases of SSADM whether task data/user requirements was indeed appropriate, what that data would consist of, how it would be collected and modelled and what *specific* stage of SSADM it would be address. Therefore, the authors identified for all

stages of SSADM the potential contribution of task data, and we are presently extending SSADM, i.e. adding in more stages to take account of this potential contribution. In the present version of our tool specifications, we have already identified for all the stages in the feasibility, system analysis and system design phases where task analysis would be appropriate and what that contribution would be.

The KAT toolset.

In this paper we are only able to briefly outline some of the features which characterize the proposed tools. In the full report (Johnson and Johnson 1990) the authors indicate how the tool screens might look, the menu labels and content, and also describe the functions that will be available to the users, and/or will be carried out automatically by the implemented TA tools. Here we only have space to provide brief details of the screen contents and functions.

The format of the TA tool screens will be as follows: First, screen titles, centred at the top of the screens will identify and refer to each of the individual screens. Shaded menu headers situated below the screen titles, will be visible at all times when the screen has been selected. Each of the menu headers will comprise pop-up menus with menu items which will be activated when the menu header is selected. Therefore all the menu items will be available to the user at any time that the user chooses, on selection of the appropriate menu header. Some of the menus will also have within them cascading menus. On the user selecting a menu item the pop-up menu will disappear and the appropriate text as a result of the menu selection will appear on the screen. Text on the screen will be presented as on-line scrollable text. Below we give a general indication of the contents of each of the screens.

A total of four screens and a number of associated windows will complete the KAT toolset. **Screen 1 Knowledge Analysis of Tasks (KAT)** will always be present when the application is started. This first screen will be mainly an information screen giving general details about KAT and the KAT tool. This screen will provide an introduction to KAT activities. When the user actually wants to undertake these activities, other screens will be activated automatically as

a result of the user selecting either “Colecting data”, “Analysing data” or “Modelling tasks”. In addition Screen 1 provides the user with a short-cut to the TKS screen (see later). Screen 1 will comprise six menu headers with pop-up menus within these headings. The menu headers will be “Information”, which will provide details about reports and papers which give details about the theoretical underpinning to KAT and other sources of information. The second header, “Task analysis” will have five items on a pop-up menu providing a) a general introduction to task analysis, b) general guidelines to follow when carrying out task analyses, c) a glossary of task analysis terms, d) details about the scope of a task analysis and finally, e) a list of the knowledge components which comprise TKS.

The third header of Screen 1 will be “Data Collection”, which will comprise two items on a pop-up menu, a) an introduction to data collection and b) a menu item “Collecting data” which when selected will advise the user that the “Data collection and analysis screen” will be opened. The fourth header of the KAT screen will be “Data Analysis” again with two menu items, a) an introduction to data analysis and b) a menu item “Analysing data” which again will activate the “Data collection and analysis screen”. The fifth menu header “Task modelling” comprises three menu items, a) an introduction to task modelling, b) an automatic activation of the TKS screen, labelled “Modelling tasks” and c) automatic activation of the system integration screen, labelled “System integration”. The final header of Screen 1 will be “TKS” which will have four menu items; a) “Create” b) “Edit” c) “Delete” and d) “TKS”. Choosing any of these menu items will provide automatic activation of the TKS screen. On choosing “Create” the computer once on the TKS screen will ask for the title and also about the goal and taxonomic substructures, which are substructures of the TKS (see Johnson and Johnson, 1987b for a discussion of what these relate to). Choosing “Edit” or “Delete”, results in the user being asked for the title of the TKS to be edited/deleted which is then displayed on the screen. Choosing “TKS” will activate the screen without any further activity until the user pulls down and selects an item from the menu on that screen.

The Data Collection and Analysis Screen comprises four menu headers, "Data collection", "Knowledge sources", "Collection techniques" and "Data Analysis". The pop-up menu for the "Data collection" header has four items, a) an introduction to data collection, b) an introduction to knowledge components which make up TKS, c) a menu item "Actions and Objects" which also has a cascading menu to items giving information about how to identify actions and objects and their representativeness and typicality, and to accessing the "Object database" and the "Action database". On the user selecting the "Object database" or the "Action database" either the Object Database window or the Action database window respectively, will appear on the screen. For detailed descriptions of the functions of these windows we would refer readers to Johnson and Johnson, (1990). Briefly the windows provide a register of the actions and objects and their characteristics which can be added to, amended, queried and so on. The final item on this menu is d) containing details about procedures and plans such as their identification and their relationship to goals and subgoals.

The second menu header "Knowledge sources", has two menu items, a) details of background literature on how to find out about people's tasks and b) a menu item "People" which has a cascading menu to details about choosing people to participate in a task analysis in terms of their number, expertise, characteristics and also about eliminating subject bias. The third menu header "Collection techniques" has eight menu items covering a) introduction to collection techniques, b) Interviews with cascading menu to information about interviews, their advantages, disadvantages, appropriateness and also a proforma questionnaire. The same information is able under separate menu items for c) observation d) concurrent protocol analysis e) retrospective protocols, f) the use of concurrent protocols versus retrospective protocols, g) the problems with protocol analysis and finally, h) problems with collection techniques generally.

The final menu header of the "**Data collection and analysis screen**" is "Data analysis" which has two menu items a) an introduction to data analysis b) details about generification with a cascading menu to introducing generification and the identification of generic actions and

objects. The goal structure item on this cascading menu will activate the goal structure window will support the inputting of plans, procedures, goals and subgoals, which it will then present to the user in both graphical and textual (including rule-based) form. A final item on this menu is computer supported generification. Computer supported generification procedures will be supported by a generification window which when activated will identify generic actions and objects from those entered by the user, dependent on a user-defined threshold. As a result the window will display a comprehensive list of generic objects and actions, listed alphabetically and with the corresponding frequency also quoted. This list can then be transferred to the taxonomic substructure.

The Task Knowledge Structures (TKS) screen will display the empty slots of the TKS which have to be completed by the designer. Extensive help facilities will be available at this screen to indicate how the slots can be filled and referring to each of the other screens and windows for techniques and methods by which the TKS might be completed. There are five menu headers, "TKS", "Roles", "Tasks", "Taxonomic substructure" and "Goal substructure". Under the "TKS" menu header are five items, a) an introduction to TKS b) a dictionary consisting of a scrollable list of already constructed TKSs c) a menu item Object database which on selection will activate the "Object database window", d) a menu item Action database which on selection will activate the "Action database window", e) a menu item System integration which will on selection automatically activate the System integration screen. Under the "Roles" menu header there are three menu items allowing the user to a) create, b) edit and c) delete role relationships. The "Tasks" menu header similarly has three menu items allowing the user to a) create, b) edit and c) delete links between TKSs. The "Taxonomic substructure" and "Goal substructure" menu headers have menu items which introduce the user to the substructures and allow them to be created, edited and deleted and for the automatic transference of items within the two databases to the goal and taxonomic substructures. Two windows are associated with each of these menu headers, (for full details please see the report referred to earlier).

Finally, once the designer is satisfied with the partially complete or fully completed TKS, the task model can then be transformed into one of the various representational formats which will make up the System integration screen. The **System Integration Toolset Screen** has three menu headers, "Information", "Task modelling representational formats" and "SSADM". Under the "Information", menu header there are two menu items, a) which discusses the potential integration of task analysis and b) KAT, to the feasibility/initial planning, requirement/analysis, design, prototyping, validation and update and maintenance stages of a design lifecycle. The "Task modelling representational formats" menu header will have four menu items, a) an introduction to the different formats TKS could be represented in, b) a view of the use of English as a medium for TKS representation, c) the use of the diagrammatic/graphical representation of TKS given by the goal substructure window and finally d) the use of production rules which can be constructed directly from the TKS via the goal substructure window. The last menu header of the **System integration screen** "SSADM", has four menu items which cover a) introduction to SSADM, the contribution of KAT at the feasibility stage (b), at the systems analysis stage (c) and systems design stage (d).

This completes the brief details of the proposed KAT screens and windows. We have taken into account the aspects of TKS which need to be identified, analysed and modelled according to the theoretical underpinnings of TKS. We have also attempted to meet the very general requirements requested by the designers in our survey data. In the next stage of the research the KAT tool designs will be tested by using a variety of techniques to see how well they satisfy designers' requirements. Following satisfactory testing the tools will then be implemented.

Summary.

At the beginning of this paper we argued that one way of assisting designers to build usable and useful computer systems was to provide them with information about how users plan and carry out goal directed behaviour. We then discussed the uses of task analysis in system design, followed by a description of the theoretical background to TKS. The results of a survey concerned with identifying what designers wanted, needed and expected in the way of tools to

support task analyses and their integration into system design, were presented. However the requirements were very general and the survey small-scale so it will be necessary to extensively test the paper based specifications before implementation. Finally, we provided a very brief description of a set of design specifications for proposed KAT tools to support designers in carrying out task analyses and their integration into system design. Other research being conducted on TKS involves the development of a formal notation to express TKS, incorporation into an advanced prototyping environment and also the use of formal methods including algebraic expressions, first and second order logic to express TKS elements. In addition to this research in a new project concerning interactive dialogues for explanation and learning, the framework of TKS is being extended to provide users of information systems with satisfactory explanations which allow them to successfully use those systems and learn from the interaction.

Acknowledgements.

We are grateful to ICL URC for funding the project "The development of task analysis as a design tool", and also to ESPRIT basic research for funding the further development of the psychological basis of TKS.

References.

- Anderson, J. R. 1976. Skill acquisition. Harvard University Press.
- Annett, J. and K.D. Duncan. 1967. Task Analysis and training design. *Journal of Occupational Psychology* 41, 211-221.
- Barsalou, L. W. 1982. Context-independent and context-dependent information in concepts. *Memory and Cognition*, 10, 82-93.
- Barsalou, L. W. 1983 Ad hoc categories. *Memory and Cognition*, 11, 211-227.
- Barsalou, L. W. 1987. The instability of graded structure. In U. Neisser (ed)., *Concepts and conceptual development: Ecological and intellectual factors in categorization*. Cambridge University Press, Cambridge.
- Brooks, L. E. 1987. Decentralized control of categorization; the role of prior processing episodes. In U. Neisser (ed)., *Concepts and conceptual development: Ecological and intellectual factors in categorization*. Cambridge University Press, Cambridge.
- Byrne, R. 1977. Planning meals: Problem solving on a real data-base. *Cognition* 5, 287-332.

- Card, S.K., Moran, T.P. and A. Newell. 1983. *The psychology of human-computer interaction*. Hillsdale, N.J: Lawrence Erlbaum Associates.
- Cromer, R. 1986. Hierarchical planning disability in the drawings and constructions of a special group of severely aphasic children. Personal communication.
- Davis, S. 1988. Knowledge structures in the human computer interface. Unpublished manuscript. Queen Mary College, University of London.
- Fleishman, F. and G. Quaintance. 1984. *Taxonomies of Human Performance*, New York: Academic Press.
- Galambos, J.A. 1986. Knowledge structures for common activities. In J.A. Galambos, R.P. Abelson, and J.B. Black (eds) *Knowledge structures*. Hillsdale, New Jersey: LEA.
- Garner, W.R. 1974. *The processing of information and structure*. New York: Wiley.
- Graesser, A.C. and L.F. Clark. 1985. *Structures and procedures of implicit knowledge*. Norwood, New Jersey. Ablex Publishing Corp.
- Green, T.R.G., Bellamy, R.K.E. and J.M. Parker. 1987. Parsing and gnirap: a model of device use. In H.J. Bullinger and B. Shackel, (eds) *INTERACT 87. Proceedings of the Second IFIP Conference on Human-Computer Interaction*, 1-4 September. Elsevier: North Holland.
- Johnson, H. 1983. *Categorisation in children and adults*. Unpublished PhD thesis. Birmingham University.
- Johnson, H and Johnson, P. 1987a. The development of task analysis as a design tool: Collecting and analysing task data. Report to ICL.
- Johnson, H and Johnson, P. 1987b. The development of task analysis as a design tool: A representational framework for task modelling. Report to ICL.
- Johnson, H and P. Johnson. 1989. Integrating task analysis into system design: Surveying designers' needs. *Ergonomics*, 32, 1451-1467.
- Johnson, H and P. Johnson, 1990. The development of task analysis as a design tool. Design specifications for tools to support task analyses. Report to ICL, January.
- Johnson, H and P. Johnson. 1991a. Extending TKS for system design. Manuscript in preparation.
- Johnson, P. 1982. Functional equivalence of images and movements. *Quarterly Journal of Experimental Psychology*, 34A, 349-365.
- Johnson, P. Diaper, D. and Long, J. 1984. Tasks, skill and knowledge; Task analysis for knowledge based descriptions. In B. Shackel (ed) *Interact84*. London North-Holland.
- Johnson, P. 1985. Towards a task model of messaging. In Johnson, P. and Cook S. (eds.) *People and Computers; Designing the user interface*. Cambridge, Cambridge University Press.
- Johnson, P. Johnson, H and F. Russell. 1988a. Collecting and generalising knowledge descriptions from task analysis data. *ICL Technical Journal*, 6, 137-155.
- Johnson, P., Johnson, H., Waddington, R. and A. Shouls. 1988b. Task related knowledge structures: Analysis, modelling and application. In D.M. Jones and R. Winder (eds), *People and Computers: From research to implementation*. Cambridge: Cambridge University Press, 35-62.

- Johnson, P and H. Johnson. 1991b. Knowledge Analysis of Tasks: Task analysis and specification for human-computer systems. In A. Downton (ed) Engineering the human-computer interface. London: McGraw Hill.
- Karmiloff-Smith, A. 1986a. Some recent issues in the study of language acquisition. In J.Lyons, R.Coates, M.Deuchar and G. Gazdar. New horizons in linguistics, 2. Harmondsworth:Penguin Books.
- Karmiloff-Smith, A. 1986b. Children's problem solving. In A. Brown and M.Lamb (eds) Advances in developmental psychology, Vol III. New Jersey:Erlbaum.
- Kieras, D and P.G. Polson. 1985. An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, 22, 365-394.
- Lakoff, G. 1987. Cognitive models and prototype theory. In U. Neisser (ed)., Concepts and conceptual development: Ecological and intellectual factors in categorization. Cambridge University Press, Cambridge.
- Leddo J. and R.P. Abelson. 1986. The Nature of explanations. In J.A. Galambos, R.P. Abelson, and J.B. Black (eds) Knowledge structures. Hillsdale, New Jersey: LEA.
- McCauley, R. 1987. The role of theories in a role of concepts. In U. Neisser (ed)., Concepts and conceptual development: Ecological and intellectual factors in categorization. Cambridge University Press, Cambridge.
- McCloskey, M and Glucksberg, S. 1978. Natural categories: Well-defined or fuzzy sets? Memory and Cognition, 6, 462-472.
- Medin, D. and Wattenmaker, W.D. 1987. Category cohesiveness, theories and cognitive archeology. In U. Neisser (ed)., Concepts and conceptual development: Ecological and intellectual factors in categorization. Cambridge University Press, Cambridge.
- Minsky, M. 1975. A framework for representing knowledge. In P.H. Winston (ed) The psychology of computer vision. New York: Mc Graw Hill.
- Muller, M. 1991. PICTIVE: an exploration in participatory design. Human Factors in Computing Systems (CHI'91), 225-231.
- Neisser, U. 1987. From direct perception to conceptual structure. In U. Neisser (ed)., Concepts and conceptual development: Ecological and intellectual factors in categorization. Cambridge University Press, Cambridge.
- Payne, S. J. and Green, T. R. G. 1986. Task-Action Grammars: a model of the mental representation of task languages. Human Computer Interaction , 2, 93-133.
- Rips, L. J., Shoben, E.J. and Smith, E. E. 1973. Semantic distance and the verification of semantic relations. Journal of Verbal Learning and Verbal Behavior, 14, 665-681.
- Rosch, E. 1973. On the internal structure of perceptual and semantic categories. In T. E. Moore (ed) Cognitive development and the acquisition of language. New York: Academic Press.
- Rosch, E. 1978. Principles of categorisation. In Rosch, E. and LLOYD, B. (eds.) Cognition and Categorisation. New Jersey, Lawrence Erlbaum Associates.
- Rosch, E. 1985. Prototype classification and logical classification: The two systems. In E.K. Scholnick (ed) New trends in conceptual representation: Challenges to Piaget's theory? New Jersey: Lawrence Erlbaum.

Rosch, E., Mervis, C., Gray, W., Johnson, D., and P. Boyes-Braem. 1976. Basic objects in natural categories. *Cognitive Psychology*, 8, 382-439.

Rosson, M.B., Maass, S., and W.A. Kellogg. 1988. The designer as user: Building requirements for design tools from design practice. *Communications of the ACM*, 31, 1288-1298.

Schank, R.C. 1982. *Dynamic Memory: A theory of reminding and learning in computers and people*. New York: Cambridge University press.

Smith, E. E. 1978. Theories of semantic memory. In W.K. Estes (ed) *Handbook of learning and cognitive processes*, Vol 6, Potomac, MD: Erlbaum.

Smith, E. E. and Medin, D. 1981. *Categories and concepts*. Cambridge, MA; Harvard University Press.