# THE ROLE OF COGNITIVE APPORTIONMENT IN INFORMATION SYSTEMS

Oystein D. Fjelstad
*University of Arizona*

Benn R. Konsynski
*University of Arizona*

# THE ROLE OF COGNITIVE APPORTIONMENT IN INFORMATION SYSTEMS

Oystein D. Fjelstad and Benn R. Konsynski
Management Information Systems Department
University of Arizona

## ABSTRACT

As the number of information system users increases, we are witnessing a related increase in the complexity and the diversity of their applications. The increasing functional complexity amplifies the degree of functional and technical understanding required of the user to make productive use of the application tools. Emerging technologies, increased and varied user interests and radical changes in the nature of applications give rise to the opportunity and necessity to re-examine the proper apportionment of cognitive responsibilities in human/system interaction. Examples illustrate the opportunities afforded by such an examination. A framework is presented that illustrates many of the tradeoffs that occur in a reapportionment activity. A knowledge-based architecture is proposed to facilitate both static and dynamic reapportionment decisions.

## THE APPORTIONMENT CONCEPT

As the number of information system users increases, we are witnessing a related increase in the complexity and the diversity of their applications. The increasing functional complexity amplifies the degree of functional and technical understanding required of the user to make productive use of the application tools. Aspects related to this increased understanding include human memory requirements, system command interpretation and command formulation, and problem-solving strategies. In most current system implementations the responsibility for obtaining an active understanding of the system and ensuring a correct and effective user-system interaction resides with the user. Advances in interface technology, network technology, applied automated reasoning, and radical changes in the nature of applications and system configurations give rise to the opportunity and necessity to re-examine the proper apportionment of task responsibilities in information systems.

System architectures are changing into multiprocessor architectures with servers, workstations, and networks facilitating cooperative work among users. Flexible support environments facilitating end user computing are emerging. The distinction between system development and system use becomes fuzzy in such environments. The UNIX operating system, with its flexible tool collection, is an example of a powerful but complex environment, where the mapping of system opportunities to user problems is a new cognitive task introduced to the user. Several of these cognitive responsibilities could be reapportioned among the user and the participating system processors by the application of knowledge-based reasoning techniques. One approach would be to focus on problem description vs. detailed specification of the solution. This has been the goal of fourth generation languages. However, few true fourth generation languages have appeared. The framework suggested in this paper promotes the application of techniques from artificial intelligence injected into several aspects of information systems development and use.

An information system can be viewed as a formalized theory for the execution of tasks, where the tasks and sub-tasks are distributed among available processors such as the system users, workstations, servers and networks. Traditional system architectures promote strong assumptions regarding feasible task allocations among the user and the system software and hardware configuration. Many of the cognitive responsibilities in dialogues are by default allocated exclusively to the user, the human processor. Cognitive type tasks like reasoning, learning and adaptation are assumed either to be the responsibility of the user or to be processed by separate independent systems. It is the purpose of this paper to examine the opportunity for reapportionment of the cognitive responsibilities in general, and the system dialogues in particular. Further, we will provide an architecture that facilitates both a static reorganization of responsibilities and a framework for dynamic reallocation.

The relevant processors that participate in the current system environment include: the users, personal workstations, network processors, network servers, and host application computers. Each participant represents an opportunity for allocation of responsibilities, and each carries certain unique qualities and capacities. The current state of network technology and automated reasoning provide an opportunity to reassess the assumptions regarding who can and should be responsible for particular tasks. Through an extension of the limits of reasoning support, more desirable allocations of tasks can be examined.

Organizations provide, through their formal structures and mission assignments, a structure for decomposition of tasks and allocation of responsibilities among participants (i.e., systems, personnel, and other organizational entities). With the rapid proliferation of local area network technology, opportunities for a more dynamic assignment of responsibilities arise (McKenney, 1985). The identification of organizational resources to be applied in problem solving is a meta-problem-solving activity (Kotteman and Konsynski, 1984; Konsynski, *et. al.*, 1985(b)). The resources applied in this context include: people, machines, models and knowledge bases. Of special concern in this paper is the decomposition of cognitive tasks and appropriate allocation of these tasks among the available organizational participants, both human and technological. A framework for resource identification and task allocation in information system environments, along with examples of implementation, is

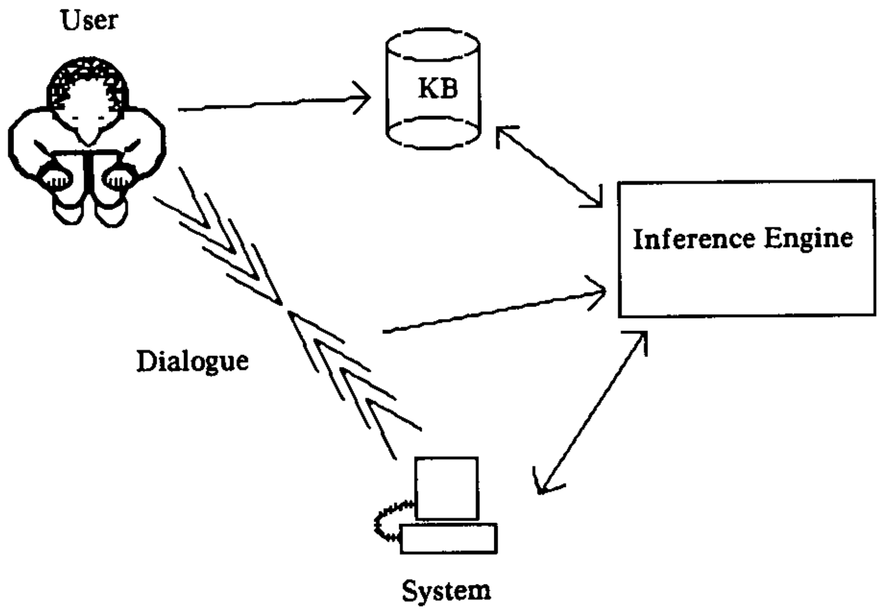


**Figure 1.** Reapportionment of Cognitive Responsiblities.

presented. Examples of cognitive tasks that fall outside traditional system architectures are:

- Maintenance of knowledge of the system's potential in the context of the problem situation. For example, what situations can the IS adequately process, and how can inappropriate situations be recognized.

- Creation of problem solving strategies; the layout of data structures and application of associated operators.

- Elicitation of problem statements and matching with available tools and resources.

- Determination of appropriate processor(s), server, workstation, network or users and user groups.

- Evaluation of implementation and system usage.

It is important to note that the cognitive tasks required in the system application process vary over time. As subordinate problems are resolved, relations, useful functions, and essential information are determined. Less useful actions and information are eliminated from the process. Useful solution mechanisms are identified and chunked into memorable patterns.

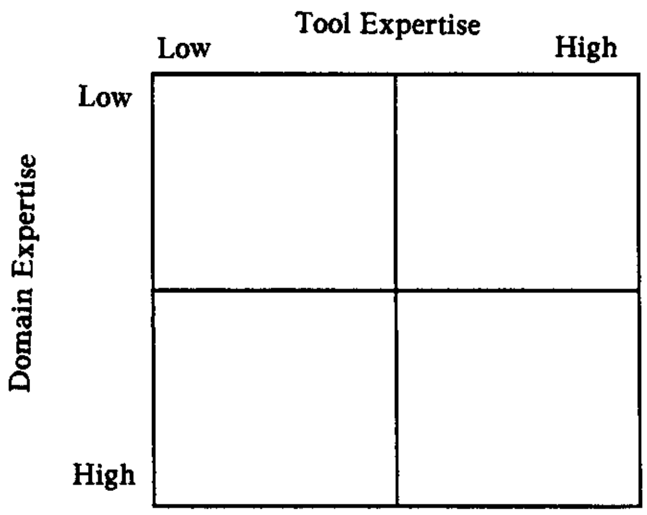


**Figure 2.** Tool/Domain Relationship.

Classical examples exist in game-playing research (Chase and Simon, 1973), where novices are found to spend much of their time learning the rules and moves, while experts recognize patterns of positional opportunity. In a system context this would relate to the learning of primitive commands in a novice context, while identifying solution patterns in the expert situation. The user-system relationship can be characterized by Figure 2.

A particular user of the system can be located within this matrix based on his association with the problem domain and the tool environment. The nature of the dialogue support that is appropriate for a particular user will depend on his/her position within this framework. The user's relationship to the tool environment and the problem domain will, necessarily, be dynamic over time. As the user participates in problem solving using the tool, a one-way familiarization takes place. The user learns effective and ineffective actions to perform in problem solving in that particular domain. Learning of tool, problem domain, and the specific problem in the context of tool usage takes place in the process.

Due to the dynamic nature of the user-problem-system relationship, a reapportionment effort must address the identification of these dimensions of experience at any point in time. Further, the identification of an appropriate functional support environment, and an effective delivery support environment are contingent upon anticipation of the initial and the elapsed experience of the user. Examples of support facilities include factors such as context sensitive help, provision of system-assisted generation of macros, dynamic and adaptive dialogue reorganization, and adjustments to system command syntax and explanation.

In order to facilitate a dynamic reallocation of these "cognitive" responsibilities, we require an effective and flexible knowledge-base of user, problem and delivery environment knowledge. A knowledge based approach to reapportionment of cognitive responsibility requires access to metalevel descriptions of the tool environment and its relationship to functional domains, profiles of the user with respect to level of expertise in system usage and functional domain, and a metalevel description of the dialogue in progress in order to develop user models and provide support delivery. In cooperative multiuser, multiprocessor environments it is also necessary to keep profiles of personnel expertise and availability, as well as profiles of applications and processors. User expertise, formalized models and knowledge fragments are among system resources for which the information system should provide support in identification, communication and allocation. Modeling of processors and applications, and modeling of task apportionment within an information system is critical to the success of a reapportionment effort. The examples presented suggest that a knowledge-based approach may facilitate an assessment of task apportionment and suggest reapportionment strategies.

## ILLUSTRATIONS OF KNOWLEDGE BASE SUPPORTED DIALOGUES

The presented research effort is directed toward the development of architectures for the facilitation of knowledge-based dialogue management systems (Kuo, 1985). Of special concern is the support of task reapportionment in DSS-dialogues. The situations described below illustrate the opportunity for reapportionment of responsibilities in user-DSS interaction, through knowledge- based dialogue management. The examples are drawn from prototype systems for model management, adaptive dialogues, and intelligent workstations. The systems are currently in a development stage in the MIS department and not all features described below are implemented.

### Spreadsheet Manager

Spreadsheets have become popular as generators (Sprague and Carlson, 1982) for relatively simple model-based DSSs. Spreadsheet models are often developed independently by individuals for specific purposes, and they are seldom well documented. However, these models often constitute the only available formal specification of assertions held in specific decision situations. As such they represent an organizational knowledge base of assertions, values, and reusable models.

The spreadsheet manager is an ES based system for management of spreadsheet models developed in Lotus 1 2 3. The following classes of functions are available: CLASSIFY, STORE, CONSULT, SEARCH and RETRIEVE. Each function is briefly reviewed below for the sake of demonstrating the functionality of the system. (For a discussion of AI-based model management see Konsynski and Elam, 1986.)

The CLASSIFY function uses a combination of simple heuristics expressed as production rules, and direct interaction with the user to classify specific instances of spreadsheet models. The heuristics are based on the type of functions applied and the variable names used (names are stored in a separate data dictionary). The initial classification is done in forward chaining mode, and forms the basis for structured query of the model builder for final classification.

The STORE function associates a description of the model with a direct reference for subsequent retrieval.

The CONSULT and SEARCH functions complement each other, and provide two alternative means of accessing a model. SEARCH facilitates relational searches for attributes such as: name of model builder, model parameters, model domain, etc. The CONSULT function guides the user through a structured query in an attempt to identify models or model templates that may be useful in the situation at hand.

RETRIEVE allows selection for retrieval among worksheet alternatives. The retrieve function also includes the ability to provide feedback to the knowledge-base with respect to the application of the selected model, i.e., suggest new areas of potential use or restrict existing definitions. The spreadsheet manager illustrates one aspect of the proposed knowledge-based dialogue architecture, the need to maintain profiles of system resources and use these to offer guidance to the users. The dialogue component contains knowledge chunks describing the resources available in the system. The knowledge base is unique in the sense that it is not based on expert knowledge, but rather on the profiling of system resources as they evolve. This is analogous to the need to dynamically model the user-system relationship in tutoring systems. The dynamic profiling of system resources becomes especially important in group DSS situations where multiple participants contribute asynchronously to the model base.

## An Adaptive Dialogue

The potential of adaptive dialogues should be explored in an effort to realize effective reapportionment of cognitive responsibilities in DSS. In adaptive dialogues, the presentation and elicitation processes change over time as the usage patterns of function evolve. This is critical in DSS where the initial states are highly unstructured. As structure emerges in the process and solution, the patterns adopted by the user become structural entities that may play an important role in an effective user-system dialogue. User presentation and action expression preferences develop as the dialogue proceeds. A dynamic dialogue capability should exploit implicitly expressed preferences to insure an expedient resolution of the decision- making situation, and improve the decision-making process through recognition of the user processes.

Digital's ALL-IN-ONE shell was used as a laboratory for the study of dialogue requirement specification. The usage of the ALL-IN-ONE system was examined via the construction of Markovian models of keystroke patterns. The potential for automatically adapting, or restructuring the dialogue to adapt to an individual's style and functional requirements was examined. In the version studied facilities existed for the examination and deletion of individual mail items; however, several keystrokes were required for each selection and deletion. A "learned" task was dynamically developed, in which the user could peruse and delete a file using familiar editing functions. The experiment demonstrated that a performance improving dialogue model could be adaptively developed. The performance of the user was improved through the use of more efficient access methods (fewer keystrokes and less display time) and reduced learning time through common access methods. The common keystroke patterns observed were treated as a script. A macro facility emerged to support a minimization of keystrokes and an economy of the presentation space. Adaptive dialogue support may be an effective means of facilitating the reapportionment of cognitive responsibilities in the user-DSS interaction. Dynamic dialogues help by allowing the migration of dialogue decisions from the user to the flexible dialogue manager and, possibly, certain decisions from the system to the user.

## Distributed Electronic Mail Processing

Evolving office computing environments are typical examples of multi-processor configurations. Some offices will have a complement of PC's, file and printer servers, and mainframe machines. Each class has unique capabilities that justifies its inclusion in the configuration. There is, however, very little true distributed processing where a task is decomposed into subtasks that can be allocated according to the availability of functionally well-suited processors.

An example of functional decomposition of a task can be found in a simple electronic mail system. The task of sending mail can be decomposed into mail creation, mail submission, and mail delivery. Among the alternative processors for these tasks in the above office environment example are: the user, a personal computer workstation, and a mail server. What processes to execute where should be determined by the available configuration, and the nature of the sub-tasks that are needed to accomplish the goal of communicating an electronic message. As demonstrated by Zisman [1977], in some situations the message may be generated automatically based on the occurrence of external events. Other situations call for user participation. The determination of human vs. PC as message origination candidates can be based on multiple factors, such as message recipient, message content, and processors available. A mail system called MASH (Mail Access Supporting Heterogeneity), currently being developed at the University of Arizona, provides a decomposition of mail origination and receipt tasks into subtasks that are distributed based on the availability of processors and environmental knowledge. The workstation will recognize and script communication with heterogeneous network and host environments.

In a workstation-based environment a major part of the dialogue will be carried out in the local workstation to provide the user with a responsive dedicated dialogue environment and to minimize network connections. On portable lap top computers with limited dialoging capabilities the dialogue is more constrained, and in a terminal mode the dialogue responsibility is allocated to the connecting server machine. The server will recognize the functional capabilities of its connecting processors and adapt its protocol to the situation at hand. An example is the detailed specification of the

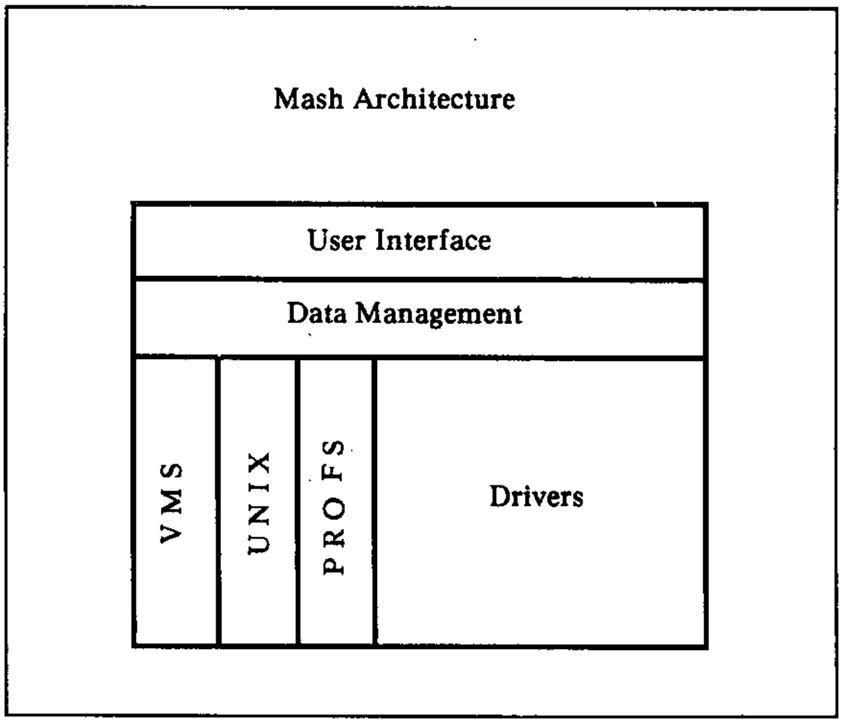


**Figure 3.** Mash Architecture.

mail routing information. In MASH, a callable inference engine is used to assist in the determination of message receivers and message handling responsibility, based on information contained in the message, job and authorization responsibility and other information about the individuals and project assignments.

## KNOWLEDGE REPRESENTATION

The interaction between a user and a system is itself a problem solving activity. Anderson (1985) defines problem solving as a goal-directed sequence of cognitive operations. The definition does not contain any reference to the assigned processor(s) for these cognitive operations. Cognitive operations can be carried out either by a human or by a machine. In an interactive environment, the problem-solving task is characterized by a sharing of cognitive tasks between the user and the system.

The tool portfolio provided in most current interactive problem- solving environments is specifically directed toward the execution of functions that relate to the general problem space. The general problem space refers in this context to device independent functions such as: calculation of net present value, or average of a time series. Expert system technology has been suggested as a facilitator of intelligent user interfaces. Suggested applications include explanation capabilities, provisioning of familiar terms, and tutoring of the user (Turban and Watkins, 1986). Expert systems are generally defined to be systems that can exhibit performance comparable or above expert performance in a problem domain (Hayes-Roth, *et al.*, 1983). As illustrated by the preceding examples, the reapportionment of cognitive responsibilities may call for expert system elements in the architecture. However, there are several additional problems related to areas such as user modeling, task profiling, dialogue and technology representation (Gaines and Shaw, 1986) that require a broader definition of knowledge and performance than what is provided by the ES paradigm. The purpose of the research described in this paper is the identification of elements in a knowledge-based architecture for reapportionment of cognitive responsibilities in information system dialogues. Kuo (1985) identifies three dimensions of knowledge for dialogue modeling: task, user and technology.

### User Modeling

User modeling can be viewed as the process of developing a model of the user that can guide the interaction. The type of knowledge that should ideally be available to the dialogue management system depends on the purposes for which the knowledge is required to draw inferences. Rissland (1984) identifies seven types of knowledge that must be maintained in an effort to achieve an intelligent user interface: user, user's tasks, tools, domain, modalities, how to interact, and evaluation. The diversity of the knowledge that must be maintained indicates that a single knowledge representation scheme will not be sufficient. Rather the aim would be toward pragmatic integration of knowledge from multiple representations.

**Intelligent Tutoring Systems** - Intelligent tutoring system researchers have been concerned with the development of user models that reflect the student's knowledge of a domain in such a form that it can be applied in the identification of skill or knowledge deficiencies, and provide a basis for the development of a tutoring strategy, and the execution and monitoring of tutoring plans (Sleeman, 1985). This research provides a basis for the representation of technological environment knowledge and potential knowledge about the problem-solving domain. A frequently encountered structure for user models in ITS's is a hierarchy of concepts where subconcepts are asserted as known or unknown by the student (Peachey and McCalla, 1986)

**Cognitive Style** - Models of users' cognitive style have been suggested as a basis for design of individual user interfaces (Mason and Mitroff, 1970; Alavi and Henderson, 1981). Huber (1983) argues that there is currently little support for cognitive style oriented design as important in DSS usability.

**User Classification** - Rich (1983) with the perspective of eliciting user characteristics and anticipating user preferences, discusses user modeling and provides a taxonomy of alternative user models. The user models are classified into: a) canonical models vs. individual models, b) long-term models vs. short-term models, c) explicit models vs. implicit models. The selection of appropriate user modeling strategy for an information system will depend on the relationship between the user and the system, and the user's and the system's purpose.

**Representations** - Zissos and Witten (1985) review common forms of user model representation: a) parametric form where a small set of values characterize the user for a particular task, b) discrete event forms where keystrokes or sequences of keystrokes are massaged into a finite state pattern, c) a framelike form in which domain knowledge is used to identify explicitly the user's performance with each concept.

**Performance Evaluation** - Card, Moran and Newell (1980) present a model of user task representation that can be useful in the assessment of user performance. The model shown at the bottom of the page was developed to predict performance for expert users in the application of word processing systems.

Carlsson and Stabell (1986) have modified the prediction model for spreadsheet usage. Their findings indicate that although the model may be appropriate for prediction of performance in mechanistic routine tasks, it is difficult to apply it for modeling of problem solving activities. Several aspects of the problem solving activity are not reflected in the interaction time with the device. Elkerton and Williges (1985) suggest a performance profile methodology to construct a model of expertise and describe a user's performance as a subset of the experts skills.

**Environment Perception** - Stabell provides a framework for assessment of the decision maker's perception of the decision environment. Stabell (1978) points out that a task model would help define which conceptual system should be considered as relevant to the individual's information processing behavior in a concrete decision making situation. He presents a framework for measurement of integrative complexity for the evaluation of managers' perceptions of their information environment that can be used in profiling the users' relationship to the tools and resources in a specific DSS, and serve as an index of the users' conceptual systems development. Stabell measured integrative complexity as:

$$IC = \frac{1}{[k(k-1)]} \times \sum_{j=1}^{k-1} \sum_{i=j+1}^{k} \left| c_{ij} - d_{ij} \right| \quad \text{where}$$

$$T_{execute} = T_k + T_p + T_h + T_d + T_m + T_r$$

Where T=Time for task, K=Keystroking, P=Pointing, H=Homing, D=Drawing, M=Mental operator, R=Response from system.

IC= integrative complexity; K=number of constructs; $c_{ij}$ = value of the least node in common construct i and construct J in the hierarchical cluster obtained using the connectedness rule; and $d_{ij}$ = value of the lowest node in common between constructs i and j in the hierarchical cluster obtained using the diameter rule.

## Task Representation

Greenstein and Revesman (1986) make a distinction between function and task where function is defined relative to the means of transformation from one state to another, while task is described as related to activities executed to achieve a certain goal. The GOMS model (Card, *et al.*, 1983) provides a mechanistic task description composed of goals, operators' methods and selections. Croft (1984) develops a task representation where a task is described in terms its name, a description, references to sub-tasks, conditions, associated information elements, and a completion criterion. Tasks and sub-tasks represent transformations that yield results and side-effects during execution. Insight into a problem domain is often a side-effect of a specific problem-solving activity that can successfully be applied in new situations. A script (Schank and Abelson, 1977) represents a control structure for the execution of sub-tasks or functions. The script describes the preconditions for the execution of functions, and determines the scheduling of task execution.

In multiprocessor/participant environments canonical task representations are important to ensure successful implementation regardless of processor selection. Greenstein and Lam (1985) gives an example of dynamic task allocation in an experimental air traffic control study, where the responsibility for directing an aircraft landing can be carried out either by the person or by the decision support system. The allocation of responsibility is controlled through the system dialogue.

## DIALOGUE AND TECHNOLOGY MODELING

A need for descriptive and normative models of the user-system relationship arises in the assessment of task apportionment among the participants. The descriptive dialogue models have origins in computer science research, and provide formal frameworks for dialogue specification, or architectural descriptions of dialogues.

### Dialogue Descriptive Models

Moran (1981) presents an extensive model for the user-system interaction in the Command Language Grammar model. The model distinguishes between a task-level, a semantic level, a syntactic level, and a physical interaction level. Benbasat and Wand (1984) suggest a model based on interaction events, directed at the development of a dialogue generator that can provide customized dialogues based on a set of dialogue descriptive tables. Models using Finite State Machines (Jacob, 1982) and BNF notation to describe dialogues also fall in the category of dialogue descriptive models.

Sprague and Carlson (1982) give examples of alternative forms of interaction: Question-Answer, Command Language, Menu Interface, Input Form - Output Form, and input in context of output. They conclude that there are tradeoffs in the selection of interaction styles, and that alternative forms may be appropriate under alternative circumstances. They also provide the ROMC, representations, operations, memory aids, and control model, as an approach to identify the necessary capabilities of a DSS.

Kieraas (1985) introduces a comprehensive device representation model referred to as a Generalized Transition Network (GTN), describing the dialogue structure of a device. The model is directed at the analysis of user complexity of a device, but the precise description of system states may prove useful for tracking the users through the system, and give reference to descriptions of system features, functionality and pitfalls.

### Technology Models

Gaines and Shaw (1986) emphasize the need for models of computer systems to be developed that are well-suited to the analysis of human computer interaction. Models reflecting the technology with which the user interacts are needed in order to assess the allocation of tasks between user and system. Mathematical models and Monte Carlo simulation models have been suggested for technology modeling (Greenstein and Lam, 1985). The research presented here is focused around a knowledge-based approach to the development of IS technology models. The knowledge base should facilitate inferences about a component or an object's applicability, and the side-effects of applying it (Rissland, 1984). Zissos and Witten (1985) implement advice objects that describe concepts in the EMACS editor. The advice objects contain descriptions directed at the identification of missing concepts in the user's representation of the editor, and explanation strategies for different classes of user's are referenced. Chalfan (1986) describes a knowledge system that integrates heterogeneous software for design applications. The software modeling emphasis is placed on description of prerequisites for software invocation to enable dynamic determination of overall system control and data flow.

A major part of user complexity of a device can be attributed to differences between the user's mental task representation and the task representation provided in the device (Kieraas, 1985). Multipurpose software such as financial planning packages, personal database systems, etc., are popular because of their applicability in the execution of a wide variety of tasks. The consequence for the device-user task mapping is that with the exception of low level editing, and navigational tasks, it is not possible to have a perfect task structure mapping between user and technology. This necessitates a translation between a goal oriented high level task structure, and a detailed execution oriented task structure. The models must be able to support description of a) attributes of technological components such as input devices, software modules, etc., b) planning with respect to what components should be applied in achieving a goal, and the relationship among those components, and c) the composition of specific scripts of invocation sequences. Three model dimensions have been identified in order to support the above requirements:

1. **Object attributes:** Kuo (1985) presents a list of attributes including input and output requirements, primary purpose, side-effects, cost associated with invocation and application, etc. An object oriented representation with class characteristic inheritance (Goldberg and Robson, 1983) is well suited for this type of information.

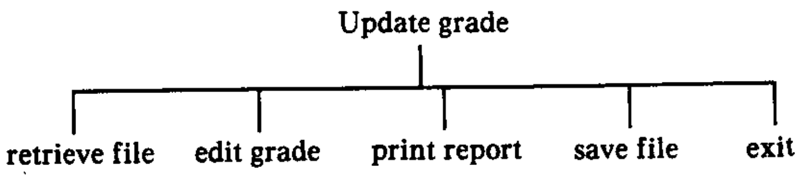
2. **Functionally oriented plan decompositions:** These can be predefined either by domain experts, or sufficient information for dynamic generation of plans can be provided (Sacerdoti, 1974). A single plan hierarchy is similar to a functional structure chart (Orr, 1977) The combined hierarchies will form a network in which atomic components and higher level virtual components participate in the achievement of multiple goals.

3. **State transition representations:** Functional task hierarchies may not correspond to the frequently found hierarchical access structure of the device. The support of specific script composition requires access to a precise description of invocation sequences for the technological component such that a plan can be translated into an executable command sequence.

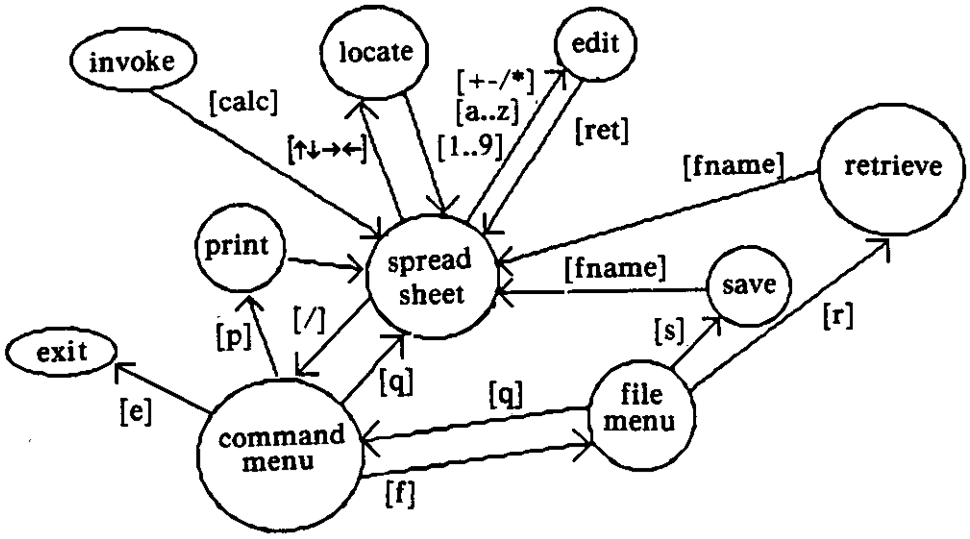
Figure 4 provides a simplified representation of a plan for updating the grade record of a student. The records are kept in a spreadsheet that must be accessed through a spreadsheet manager in order to perform the update. A transition network for the implementation of an associated executable script is also illustrated.

## "Black Box" Problems

Classes of problems exist where the user may benefit from exercising a higher degree of control over system processes than current architectures allow. Many system operations such as program compilation, model optimization, and reasoning in an expert system take place in a "black box" controlled by the machine based on an *a priori* user specification. Much of the



i) Simple plan for updating grade



ii) Subset of state transition diagram for spreadsheet where update program is implemented (commands are enclosed in brackets).

[/] invoke main menu
[f] invoke file menu
[r] "gradesheet" retrieve file
[ ] locate cell
[a..9] input data
[/] invoke main menu

[p] print worksheet
[/] invoke main menu
[f] invoke file menu
[s] "grade sheet" save worksheet
[/] invoke main menu
[e] exit

iii) Command sequence implementing the plan.

**Figure 4.** Script Generation.

reason for this can be found in historically inadequate representation mechanisms for user interaction with system processes. Some of the responsibilities embedded in black box activities could be shared with users, ensuring user control over the outcome of the processing, if well suited representational schemes that can represent and explain the system task execution are devised and interaction mechanisms provided. As an example illustrating reapportionment from the system to the user, we have prototyped a microcomputer-based expert system shell. The system's reasoning processes are modeled in a worksheet, allowing the user to assess the validity of intermediate results and intercept the reasoning to modify or perform sensitivity analysis on the facts in the database and the knowledge base itself. The worksheet interface provides a dynamic explanation facility for the inference mechanism and allows the user to share the responsibility in a co-cognitive environment between user and machine. The same interactive strategy on critical system decisions would be applicable to other "black box" processes if the system is equipped with sufficient explanatory mechanisms.

## ARCHITECTURES FOR IMPLEMENTATION OF COGNITIVE REAPPORTIONMENT

The facilitation of a reapportionment depends on the identification of cognitive tasks in the user-system interaction, the identification of available processors with associated implementations of activities, and selection among the feasible alternatives. The satisfaction of these tasks must be accommodated in the support system architecture. Our research and experience with the prototype implementations suggests that a general, knowledge-based approach can facilitate some aspects of the desired reapportionment. The architecture suggested includes knowledge bases with user information, problem oriented task representations, and delivery environment knowledge. The knowledge bases contain metalevel descriptions directed at external profiling of the user, the system, and the user-system relationship, to accommodate application of background inferencing minimizing the constraints imposed on the application environment.

The descriptions provided below are based on sample implementations of the tool environment portfolio. The suggested architecture represents the interaction between the user and the system as instances of scripts (Schanck, 1977) for execution of one or more problem-solving tasks. The composition of scripts represents a meta-problem-solving activity. Sub-tasks can be recognized and assessments can be made with respect to their implementation. Sub-tasks represent functions that yield results and side-effects during execution. Insight into a problem domain is often a side-effect of a specific-problem solving activity that can successfully be applied in new situations. A script represents a control structure for the execution of sub-tasks or functions. The script describes the preconditions for the execution of functions, and determines the scheduling of task execution. The support for a transfer of user scripts to the system in the user-machine interaction is currently very limited, allowing mainly for primitive and static scripts, providing automa-

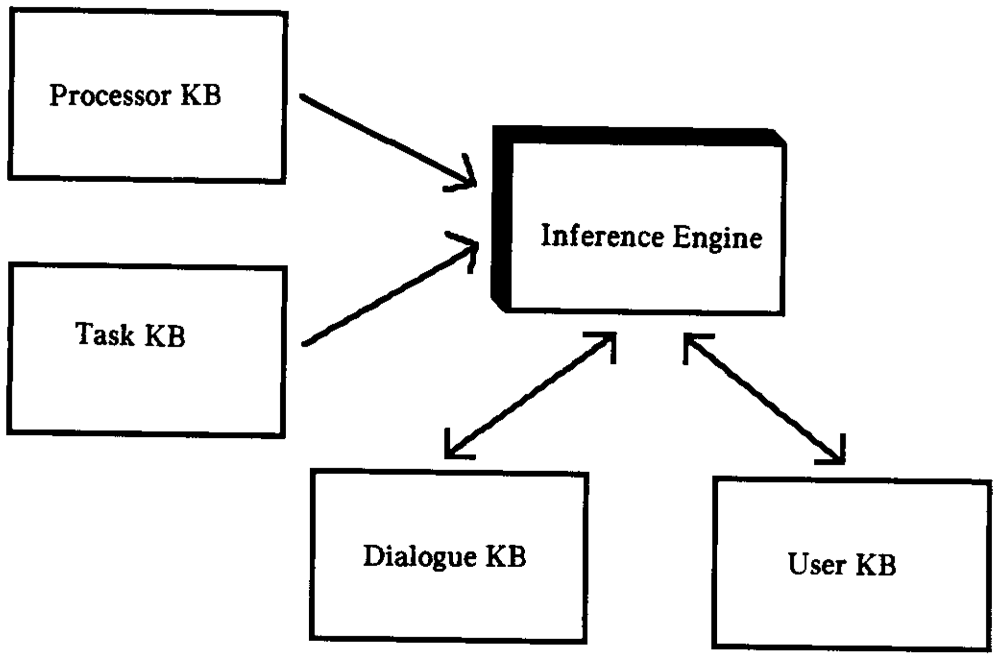


**Figure 5.** Knowledge Base Structure

tion of fixed execution sequences for system functions. An example of such fixed scripts are the batch command scripts found in most operating systems at the shell level. Scripts are abstractions of a task. Scripts synthesize the effects and side-effects of executed sub-tasks into a single cognitive element. They shift the responsibility for detailed sub-task scheduling during task execution from the user to the system. However, among the disadvantages of static scripts is the lack of dynamic adaptability to changing contexts. A situation where a corporate budget is consolidated from of a series of spreadsheet models from divisions and departments can be used as an example.

A simple script for budget consolidation would go through the lowest level of budget models and create aggregated results, until the desired level was reached. A problem with such a simple procedure surfaces if a lower level model changes. The script must either be modified to incorporate only the affected aggregates, or the user must personally direct the new consolidation effort if major time-consuming reprocessing is to be avoided. In either case the user is forced to carry out a set of problem-solving tasks, including change identification, and task rescheduling for a problem that a more dynamic script approach could have handled. An example of dynamic scripting is scripts that use inference mechanisms for task identification and task scheduling. The UNIX "make" utility represents a dynamic script. The "make" utility is directed at compilation and linking of large systems with multiple separately compiled object files. "Make" uses a simple inference mechanism to derive which files need to be manipulated and recompiled before final linking, based on file modification history and a description of file interdependencies. The function effectively implements a "do as I mean" command for the compile and link process, where the user is relieved of specifying exact control for each new situation.

The change from static scripts to adaptable inference-based script formulation has a strong impact on the information that needs to be exchanged in the user-system dialogue. In the currently dominating static control environments the user provides specific control over the actions to be executed. In an inference-based, goal-oriented interaction mode the system needs to elicit the relationships between tasks and the rules for execution. Appropriate sequencing can be determined by the scripting mechanism. A dynamic scripting mechanism requires formal representations of system objects, tasks and their alternative implementations in alternative environments, and available processors to be effective. The elicitation of task-object relationships can take place through direct specification or through abstraction mechanisms implemented as side-effects of task execution. It has been recognized that documentation should be a by-product of an effective systems design process (Teichrow, 1974). In the same way knowledge elicitation related to the provisioning of intelligent dialogues should be a side-effect of the problem solving interaction between the user and the system. Model and data characteristics, such as explicit and implicit relationships between model components, (Blanning, 1984) parameters, data structures, and information sources, are abstracted into an intelligent dictionary system that is used in the identification and description of tools and resources.

The knowledge base task representation contains descriptions of task outputs and side-effects, implementation independent sub-tasks that can be identified in the task, and implementation of activities with associated data structures in alternative processor environments. The detailed implementations are in the form of callable library utilities for system processors, and formalized dialogue specifications, where the user will participate in the implementation. The task representation is adopted from the framework suggested by Croft (1984). The user profiles are implemented as prototypes (Rich, 1983) containing descriptions of the alternative system users, and their present position in the presented tool-functional domain expertise framework. The user profiles are based on initial structured queries of the user, and later dynamically updated in the course of user system interaction.

## Matching of Available Tools to Problem Description

A major part of learning how to use a system or an application is the abstraction of application functionality into concepts that can be matched against a problem statement. The process is recursive in the sense that the problem statement is used to evaluate the tools, and the tool functionality is applied in the exploration of the problem statement. The situation is similar to that found in an info-center, where a user's functional needs, and proficiency must be evaluated against the available tools portfolio. A production-rule-based system that queries the user for profiling information, has been developed to guide a user in the selection of appropriate resources in the info-center. The same matching problem arises in the use of any nontrivial problem solving application. The user must select problem structures and operators that can adequately describe and solve his problem. The applicability of alternative strategies depends on the problem at hand, the users' familiarity with the tool, and the users level of expertise. The challenge is to represent an application's functionality, and to profile a users problem situation and level of sophistication sufficiently for the generation of plans (Schmidt, 1978) and suggestion of problem representations and operator selection. Using the above knowledge representations an application can be described such that its potential use in alternative problem situations can be assessed. The profile of the users and the user's problem situation is developed over the life-cycle of the users interaction with the system. Certain elements of the user profile, and the problem profile can be extracted directly from the users' application of the tool. Other elements can be elicited in a backward chaining mode of reasoning. Queries are made as scenarios within the context of the tool, and suggestions are made through generation of sample problem statements that serve as templates that can be adapted to the specific problem. The following example is drawn from financial modeling.

Assume a novice user who is unfamiliar with the package wants to perform profitability calculation. The user needs to learn the package at hand. Combinations of the following occurrences would be indications of a novice user: The time elapsed between execution of commands is relatively long; the amount of backtracking or undoing of operations is considerable; there is a lack of coherent references in the operations performed in the application. The system could at this point assume an inexperienced user, confirmation could be obtained through a direct query. After focusing in on the users application domain, some simple templates for ROI and NPV calculation could be presented in context of the tool, with a replay of the key sequences required to execute the commands. When the user indicates completion of a specific instance of the calculation the system can query concerning the extension or generalizations of the model.

The tool profiles and user profiles are implemented as semantic nets. The meta level descriptions of dialogues, and task implementations in alternative environments are implemented using a script-oriented framework, while the matching of processors, tools and tasks is done using production rules.

## CONCLUSIONS

The recent increased focus in dialogue management, network technology and application of artificial intelligence gives rise to the opportunity and necessity to re-examine the proper apportionment of cognitive tasks in information systems. We are now able to question the effective assignment of functional tasks across the multiple processors involved in the emerging IS environments. Research in cognitive psychology and artificial intelligence provide insight into the process of identification of cognitive tasks, and knowledge representation and reference required in a reapportionment effort. It has been the purpose of this paper to bring to attention the issue of apportionment of cognitive tasks among multiple processors in information systems in general, and information system dialogues in particular. The presented examples illustrate the opportunities afforded by such an examination. The examples were gathered from areas of intelligent workstations, dynamic dialogues and model management. A framework was presented that illustrates many of the tradeoffs that occur in a reapportionment activity. A knowledge-based architecture was proposed to facilitate both static and dynamic reapportionment decisions. Issues that arise in the assessment of the reapportionment problem include:

- Desirability and applicability of reapportionment
- Feasibility and form of reapportionment
- Necessary and sufficient conditions for reapportionment
- Scheduling or migration control of functional tasks
- Mechanisms for reapportionment

The current research efforts relate to each of these issues. A key aspect is the development of technology and user profiles. It is appropriate to recognize that distributed processing concepts are emerging in both the hardware/software environment as well as in task management in IS environments. The partnership role of the system in the cognitive activities will not be realized until we are able to accommodate a full assessment of the proper apportionment of cognitive tasks among all participating intelligent processors --both human and system.

## REFERENCES

Alavi, M. and Henderson, J. "An Evolutionary Strategy for Implementing a Decision Support System," *Management Science*, Volume 27, Number 11, 1981.

Benbasat, I. and Wand, Y. "A Structured Approach to Designing Human-Computer Dialogues," *International Journal of Man-Machine Studies,* Volume 21, 1984

Blanning, R. W. "A Relational Framework for Join Implementation in Model Management Systems," *Decision Support Systems,* Number 1, 1984.

Card, S. K., Moran, T. P. and Newell, A. "The Keystroke-Level Model for User Performance Time with Interactive Systems," *Communications of the ACM,* Volume 12, 1980.

Card, S. K., Moran, T. P. and Newell, A. *The Psychology of Human-Computer Interaction,* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

Carlson, E. D. "Developing the User Interface for Decision Support Systems," in *Building Decision Support Systems,* J. Bennet (ed.), Addison-Wesley, Reading, Massachusetts, 1983.

Carlsson, S., Stabell, C. B. "Spreadsheet Programs and Decision Support: A Keystroke-Level Model of System Use," *Proceedings of the IFIP-86, DSS A Decade in Perspective.*

Chalfan, K. M. "A Knowledge System that Integrates Heterogeneous Software for a Design Application," *The AI Magazine,* Summer 1986, pp. 80-84.

Chase, W. G. and Simon, H. A. "The Mind's Eye," in *Visual Information Processing,* Academic Press, New York, New York, 1973.

Croft, B. W. "The Role of Context and Adaptation in User Interfaces," *International Journal of Man-Machine Studies,* Volume 21, 1984.

Elkerton, J. and Williges, R. C. "A Performance Profile Methodology for Implementing Assistance and Instruction in Computer-Based Tasks," *Journal of Man-Machine Studies,* Volume 23, 1985, pp. 135-151.

Gaines, B. R. and Shaw, M. L. G. "Foundations of Dialogue Engineering: The Development of Human-Computer Interaction. Part II." *International Journal of Man-Machine Studies* Volume 24, 1986 pp. 101-123.

Goldberg, A. and Robson, D. *Smalltalk-80: The Language and its Implementation,* Addison-Wesley, Reading, Massachusetts, 1983.

Greenstein, J. S. and Revesman, M. E. "Application of a Mathematical Model of Human Decisionmaking for Human-Computer Communication," *IEEE Transactions on Systems, Man, and Cybernetics,* Volume 16, Number 1, 1986.

Greenstein, J. S. and Lam, S. T. "An Experimental Study of Dialogue-based Communication For Dynamic Human-Computer Task Allocation." *International Journal of Man-Machine Studies,* Volume 12, 1985, pp. 605-621.

Hayes-Roth, F., Waterman, D. A. and Lenat, D. B. (eds.) *Building Expert Systems,* Addison-Wesley, Reading, Massachusetts 1983.

Huber, G. "Cognitive Styles as a Basis for MIS and DSS Designs: Much Ado About Nothing," *Management Science,* Volume 29, Number 5, 1983.

Jacob, R. "Using Formal Specifications in the Design of a Human-Computer Interface," *Human Factors in Computer Systems Conference,* 1982.

Keen, P. G. and Bronsema, G. S. "Cognitive Style Research: A Perspective of Integration," *ICIS Proceedings,* 1981.

Kieraas, D. and Boviar S. "The Role of Mental Models in Learning to Operate a Device," *Cognitive Science,* Volume 8, 1984.

Kieraas, D. "An Approach to the Formal Analysis of User Complexity," *International Journal of Man-Machine Studies,* Volume 22, 1985.

Konsynski, B. R., Kottemann, J., Nunamaker, J. and Stott, J. "PLEXSYS-84: An Integrated Development Environment for Information Systems," *Journal of Management Information Systems,* Volume 1, Number 3, 1985.

Konsynski, B. R., Greenfield, A. and Bracker, W. E. "A View on Windows: Current Approaches and Neglected Opportunities," *NCC Conference Proceedings,* 1985.

Kottemann, J., Konsynski B. R., "Information Systems Planning and Development: Strategic Postures and Methodologies," *Journal of Management Information Systems,* Volume 1. Number 2, 1984.

Kuo, F. "An Architecture for Dialogue Management Support in Information Systems." University of Arizona, unpublished dissertation, 1985.

Mason, R. O. and Mitroff, I. J. "A Program for Research on Management Information Systems," *Management Science,* Volume 19, Number 1, 1970.

McKenney, J. *The Influence of Computer Based Communication on the Organization.* Harvard Business School, Working Paper 9-785-053, 1985.

Moran, T. P. "The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems," *International Journal of Man-Machine Studies,* Volume 15, 1981.

Newell, A. and Simon, H. *Human Problem Solving,* Prentice Hall, Englewood Cliffs, New Jersey, 1972.

Orr, K. T. *Structured Systems Development,* Yourdon Press, New York, New York, 1977.

Peachey, D. R. and McCalla, G. I. "Using Planning Techniques in Intelligent Tutoring Systems," *International Journal of Man-Machine Studies,* Volume 24, 1986, pp. 77-98.

Rich, E. "Users are Individuals: Individualizing User Models," *International Journal of Man-Machine Studies,* Volume 18, 1983, p. 199-214.

Rissland, E. L. "Ingredients of Intelligent User Interfaces," *International Journal of Man-Machine Studies,* Volume 21, 1984, pp. 377-388.

Sachredoti, E. D. "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence,* Volume 5, 1974, pp. 115-135.

Schanck, R. C. and Abelson, R. *Scripts, Plans, Goals, and Understanding,* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

Schmidt, C. F., Sridharan, N. S. and Goodson, J. L. "The Plan Recognition Problem: An Intersection of Cognitive Psychology and Artificial Intelligence," *Artificial Intelligence,* Volume 11, 1978.

Simon, H. A. "A Behavioural Model of Rationale Choice," in *Models of Man,* Wiley, New York, New York, 1957.

Sleeman D. "UMFE: A User Modelling Front-End Subsystem," *International Journal of Man-Machine Studies,* Volume 23, 1985, pp. 71-88.

Sprague, R. H., Jr. and Carlson E. D. *Building Effective Decision Support Systems,* Prentice Hall, Englewood Cliffs, New Jersey, 1982.

Stabell C.B. "Integrative Complexity and Information Environment Perception and Information Use, An Empirical Investigation," *Organizational Behaviour and Human Performance,* Number 22, 1978.

Stabell, C. B. "A Decision-Oriented Approach to Building DSS," in *Building Decision Support Systems," J. Bennet (ed.,* Addison-Wesley, Reading, Massachusetts, 1983.

Teichrow, D. "Problem Statement Analysis: Requirements for the Problem Statement Analyzer (PSA)," in *System Analysis Techniques,* J. D. Cougar, and R. W. Knapp (eds.), John Wiley and Sons, New York, New York, 1974.

Turban, E. and Watkins P.R. "Integrating Expert Systems and Decision Support Systems," *MIS Quarterly,* Volume 10, Number 2, June 1986, pp. 121-136.

Zissos, A. Y. and Witten, I. H. "User Modelling For a Computer Coach: A Case Study," *International Journal of Man-Machine Studies,* Volume 23, 1985, pp. 729-750.

Zisman, M. D. *Representation, Specification and Automation of Office Procedures,* The Wharton School of Business, Working Paper 77-09-04, 1977.