3rd AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop
Banff, Canada, November, 1988

Knowledge Acquisition at the Front End: Defining the Domain

Brian Woodward
Knowledge Sciences Laboratory
Department of Computer Science
The University of Calgary
Calgary, Alberta, Canada

Abstract

The Knowledge Engineering process may be partitioned into three general phases. The first phase is characterized by domain definition, application selection and resource commitment. The second reflects the numerous iterations of knowledge acquisition and knowledge-base development. The final phase is the installation, maintenance and use of the system. Unlike phase two, which is characterized by powerful methodologies, few structured techniques for domain definition exist for the first phase. Knowledge Acquisition principles, but not current knowledge acquisition tools, are applicable to phase one activity. Cognosys is a semi-automated system developed to build general content models of a domain before the use of specific knowledge acquisition tools. Cognosys is described, as are future developments and research using the tool.

Introduction

The rapid growth, development and use of knowledge-bases is largely due to powerful Artificial Intelligence methodologies for knowledge representation and knowledge processing. The "engineering" of knowledge into useful knowledge-based systems requires more than these methodologies: it requires procedures for acquiring information and knowledge in such a manner as to both accurately reflect the original human knowledge and to maintain its integrity when replicated in a machine environment. The activity of knowledge acquisition has grown rapidly to accommodate and facilitate the process of knowledge transfer from human knowledge sources, such as a domain expert, to a machine usable representation (Gaines, 1987).

The process of knowledge-base development (Figure 1) begins with the initial selection of the problem, application or domain definition; moves through iterations of knowledge acquisition activity, knowledge-base development and verification, and ends with delivery and a plan for maintaining the knowledge-based system in its application environment. The steps in this three phase process reflect decisions about the context of the knowledge engineering activity as well as decisions concerning the content of the domain and resulting knowledge-base.

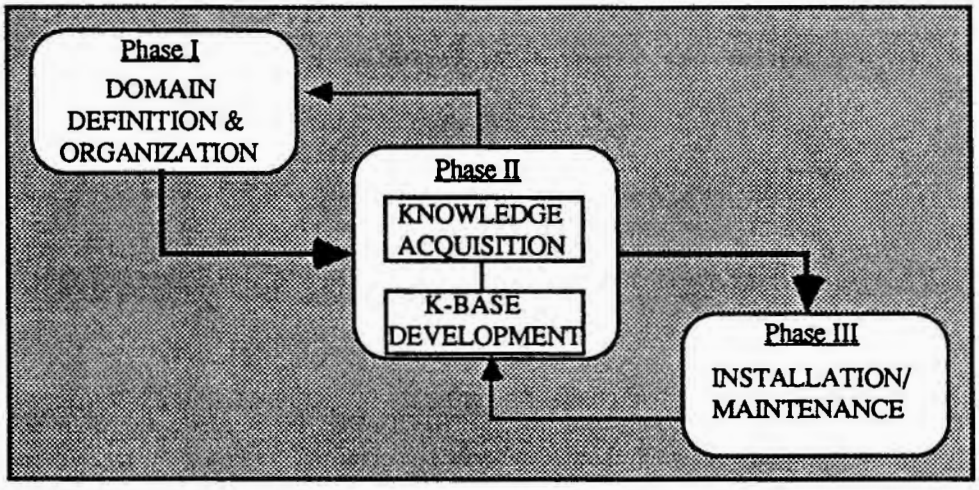


Figure 1. Phases of Knowledge-Base Development

Pragmatically, knowledge acquisition work concentrates on the middle phase. Knowledge acquisition techniques have focused primarily on the direct elicitation of knowledge from a human source in a manner which allows relatively easy representation of this knowledge in a machine environment. The intermediate representation formats used in these knowledge acquisition systems result in knowledge forms which are close-to-machine representable. Little processing is subsequently required to produce a knowledge-base from the output of these knowledge acquisition systems. (Shaw and Gaines, 1987; Abrett and Burstein, 1987; Boose and Bradshaw, 1987; Musen et al, 1987; Gruber and Cohen, 1987; Marcus, 1987). Also, within this phase, work on verification and validation is beginning (Shaw and Woodward, 1988; Stachowitz & Chang, 1988; Hofmeister, 1986; Scambos, 1986).

The predominant activity of this middle phase reflects decisions about knowledge content, organization and structure. Phases one and three, however, reflect issues of context: what application is best suited for a knowledge-based system? Who has responsibility for maintenance? What resources are needed? Domain problems which are cognitive in nature and which demonstrate a logical complexity (rather than common-sense, intuition or hard to define knowledge) are considered appropriate for knowledge-based systems (Martin & Oxman, 1988). Also appropriate are those problems which have a narrow focus and those which have a clearly identifiable pattern or procedure of reaching solutions.

Specifically in phase one, there are important issues about the content of the knowledge and its organization which need attention before phase two activity. Many authors give advice on what kinds of problems are best represented in knowledge-base form.

Knowledge-base designers (eg. Parsaye & Chignell, 1988) suggest a feasibility analysis where issues of domain problem breadth, availability of experts, and stability of procedures in the domain are considered. Others state the need for recognizing the extent of resource commitment in time and funds compared to the eventual return or savings, and the extent to which people would use the system (Martin & Oxman, 1988). Harmon et al (1988) require that the purpose of the knowledge-based system be critically specified before development.

These criteria or guidelines act only as general constraints for domain selection and definition. They do not provide a content analysis of the domain useful for guiding subsequent knowledge engineering activity. Siegel (1986) comes closest to offering a procedure for domain definition. He presents a four-step approach:

1. compartmentalize the domain into sub-problems;
2. define the inference flow;
3. compare client-system dialogues which reflect the conversational format of the proposed system; and
4. define the standards of performance.

However, the procedures for these steps are generally described and do not offer a clear set of testable instructions.

Specific techniques for domain definition and organization are lacking. Techniques which address knowledge content, organization and structure are required to help make critical decisions before concerted and expensive effort in the middle phase of knowledge engineering. Clearly defined techniques based on knowledge acquisition principles aid decision makers in phase one and affect critically the success of latter phase activity. This paper addresses the issue of the development of structured techniques for domain definition and organization. The objectives of this paper are to:

1. describe the content and contextual issues of phase one knowledge-base development activity;
2. discuss briefly how present knowledge acquisition methods are inadequate for domain organization and definition tasks;
3. present a semi-automated tool (Cognosys) which provides a structured methodology for domain definition and organization;
4. outline how Cognosys is used by knowledge-base developers;
5. discuss future developments for Cognosys; and
6. research using the tool.

Domain Organization Issues

Two main issues require consideration: the issue of context and of content in knowledge-base development. The context refers to those decisions concerning the infrastructure or development environment within which the knowledge-base is developed. These include the application parameters, potential users and uses of the knowledge-base, the availability of knowledge sources, the resource constraints of time, funds, and trained knowledge workers, and the political or management climate of deadlines and organizational expectations. These issues define the contextual setting. These decisions may be complex, requiring extensive planning and organization, and can involve many people within the organization. In research settings, these contextual issues are less onerous.

The content refers to those decisions about the knowledge-based system or the domain of interest or application. All knowledge engineering activity includes initial decisions about the domain of application. Partially influenced by the needs of the eventual users and partially by political sensitivities, knowledge-base domain definition is largely and generally defined by the availability of appropriate and interested human knowledge sources and specifically defined by the range of expertise of the knowledge sources. At some point, this domain is defined before the application of specific knowledge acquisition techniques. The definition is usually accomplished by generally focused face-to-face interviews between the knowledge engineer, if available, and the human knowledge source. On this point, descriptions of clearly defined techniques for domain definition are unavailable.

Domain definition and organization is defined as knowledge acquisition activity which incorporates procedures of knowledge elicitation and analysis to produce a domain knowledge model which is used as a basis for further knowledge refinement and structuring. This domain model represents the content and procedural knowledge of a domain at a general level. The finally agreed upon domain model sets the boundaries of the domain. The knowledge models are generated from the problem or application context and the problem demands. They are related to task demands within the domain. These models are used to define the domain and to organize subsequent knowledge engineering activity. Domain organization is a process which is used to determine the scope of the final knowledge-base and the choice of subsequent knowledge acquisition activity. Domain organization is viewed as an essential step in knowledge engineering which structures the domain to isolate and organize key areas of content, then presents knowledge in a manner that "feeds" more specific knowledge acquisition tools.

Knowledge Acquisition Tools and Domain Organization

Currently, all knowledge acquisition procedures assume problem domain definition in a manner which is conducive to starting knowledge engineering activity. The application of knowledge acquisition techniques further assumes that initial decisions have been made about the boundaries of the domain: what is included and what is not. These assumptions are demonstrated by the very few detailed descriptions of this phase one activity. In descriptions of knowledge acquisition techniques, initial domain definition is implied by the fact that the processes and techniques begin with a clearly defined domain and at a level of detail which suggests initial organization. It is assumed that an interview or a series of interviews has been completed, and that the guidelines for domain selection were followed (Waterman, 1986).

Automated knowledge acquisition systems such as KSSO (Gaines, 1987) and AQUINAS (Boose and Bradshaw, 1986), require as input a series of conceptual primitives called elements or entities which are then organized along dimensions or constructs for further processing and analyses. These tools are extremely powerful but they assume a certain level of specificity or granularity of knowledge. Once the purpose of the elicitation has been defined, the generation of cognitive primitives is easily forthcoming. The purpose itself results from a content analysis of the domain.

Other automated systems require the input of object-attribute pairs which are then organized into roles to determine domain knowledge (KNACK: Klinker et al, 1987), or ask the expert for lists of events such as hypotheses and evidence which are then associated (MOLE: Eshelman et al, 1987), or require the expert to slot-in a variety of information types in the form of events, objects, names, formulae (SALT: Marcus, 1987).

Common to these knowledge acquisition tools is the identification of specific attributes of cognitive primitives used as input. They were not built with phase one activity in mind, and their level of granularity is not suited to phase one content analysis. Each tool requires that domain definition and organization be completed prior to use of the tool or procedure. Also, these tools attempt to provide close-to-codeable output so they are extremely rigorous in their knowledge representation schemes. They have proven very effective in uncovering, organizing and representing patterns of knowledge. These qualities are essential for phase two activity but not for phase one.

Principles of Domain Definition and Organization

The present knowledge acquisition tools, though not suitable for phase one domain definition, offer clear guidelines and rigorous standards for the development of domain organization techniques. Domain organization methodology falls into the category of Knowledge Support, in that it aids knowledge engineering activity and supports and facilitates human knowledge processes. Shaw and Gaines (1987) have outlined the general design requirements for knowledge support systems. Domain organization methodology has followed these guidelines. Systems and procedures which incorporate domain organization methodology need to:

1. be domain independent;
2. access a diversity of knowledge sources including text, interviews, and observations;
3. encompass a diversity of perspectives including partial or contradictory input from human knowledge sources;
4. be directly applicable by a human knowledge source without intermediaries;
5. present knowledge from a diversity of sources with clarity as to its derivation, consequences and structural relations;
6. encompass a diversity of forms of knowledge and relationships between these forms;
7. allow users to apply knowledge in a variety of domains and to freely experiment with the implications;
8. make provision for validation and verification; and
9. be grounded on a well-developed explicit theory.

Although not speaking specifically about knowledge support methodology, Garg-Janarden and Salvendy (1987) support many of the above design principles. They say that any methodology must have a theoretical framework which helps to determine the types of knowledge for consideration and representation. These authors also suggest that any methodology should be capable of representing underlying knowledge structures which are not obvious in the surface analysis, but which are present upon a deeper analysis in the form of deep, causal and strategic knowledge. Quantitative methods should also be used wherever possible as a basis of analytic procedures. Finally, the methodology should provide procedures for examining and modifying the resultant knowledge structures. Gruber et al (1986) summarize another constraint when they suggest that knowledge acquisition techniques allow human knowledge sources to set their own level of discourse. Such systems do not demand that users organize their knowledge to fit the input requirements, but rather the reverse.

More recently, work in recognizing and defining the network of task demands within a domain has suggested another set of guidelines for domain organization methodology. McDermott (1988) states that a clear understanding of task requirements is essential for knowledge acquisition activity - that task demands structure, order and define subsequent knowledge representation and knowledge processing decisions. Chandrasekaran and Bylander (1988) have identified generic tasks which are defined by their own characteristic requirements for knowledge representation and knowledge processing, and which can act as modules for building knowledge-based systems. It is essential that domain organization methodology incorporate an understanding of task demands.

COGNOSYS

In its present form, Cognosys is best described as a knowledge engineer's tool for structuring initial discussions with a human knowledge source, and for constructing domain models. It operates in a semi-automated fashion. Some procedures of Cognosys require informed human judgment and manipulation of the knowledge. Cognosys offers a convenient format for organizing knowledge. For other procedures, Cognosys performs analyses and manipulations for the knowledge engineer. The prototype of Cognosys is in Hypertalk and runs on the Mac II or SE. The purpose of Cognosys is to aid the knowledge engineer in initial domain organization by helping to organize knowledge, in a general domain model, from the human knowledge source and to generate a useful representation of this model.

Input to Cognosys

The beginning point in the use of Cognosys is a set of written descriptions of the domain at a level which gives a complete overview of the domain. The human knowledge source is asked to generate and/or provide:

1. a descriptive statement of the overall domain;
2. a list of general objectives which must be met in order to successfully solve problems or make decisions within the domain; and
3. a collection of short paragraphs which are descriptive of the tasks required to meet each of the objectives.

The last may be from the individual's own work or from articles referred to by the individual and specifically chosen for this purpose. These three types of description act as the raw input for the knowledge engineer and Cognosys. Once this information is provided, Cognosys and the knowledge engineer cycle through the process outlined in Figure 2 . Initial discussions result in the generation of material which is processed by the

knowledge engineer and Cognosys, resulting in a model which is used as a basis for discussion between the knowledge engineer and the human knowledge source. These discussions lead to further refinement of the domain model until the human knowledge source is satisfied that it represents the emphasized domain components. The final model represents the content of the domain and the various objectives and task demands required.

Overview of Cognosys

Figure 2 outlines the three main sub-components of the Cognosys program: Analysis, Synthesis, and Support. The first two are discussed in detail, the third only generally. The knowledge engineer follows the procedural steps outlined by Cognosys to organize knowledge from initial discussions with the human knowledge source. The Analysis component on the left results in the partitioning and sub-partitioning of the knowledge and works top-down. The Synthesis path in the middle of Figure 2 starts at the bottom with knowledge units or K-units and moves upward to build a domain model. The Support component houses help information, a map of the system, and note records and storage.

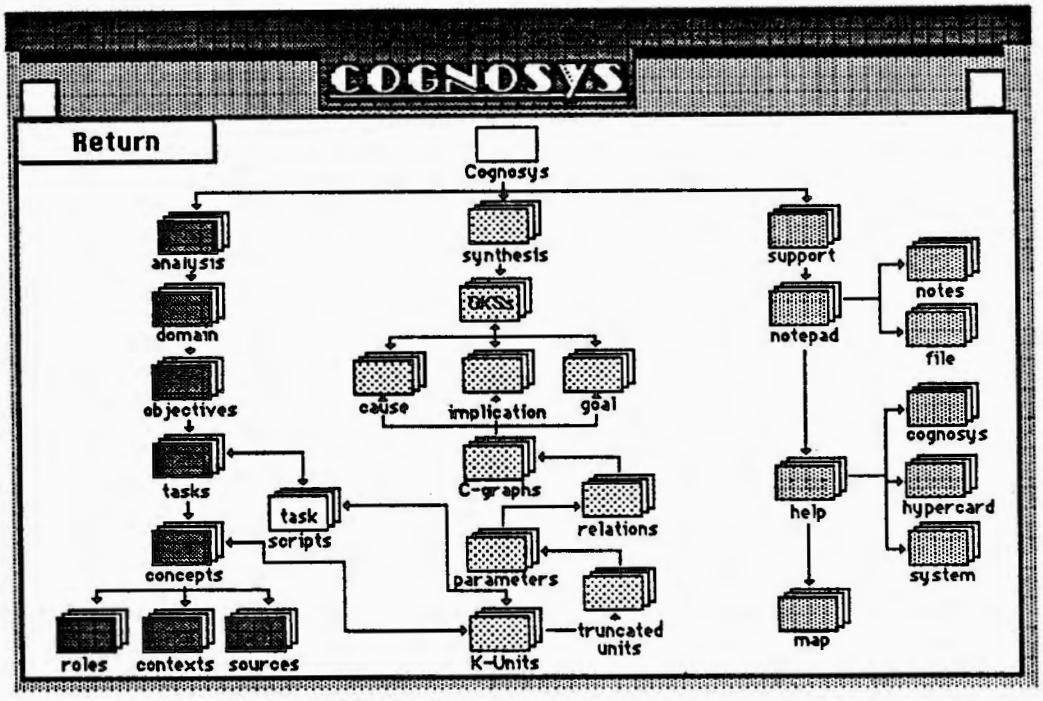


Figure 2. Map of the COGNOSYS Program

Cognosys screens correspond to the 3 sub-components of the program in that they offer different choices. The most pervasive and predominant feature is the "navigational cube" found always in the left hand corner of the screen (Figure 3). This device is used to

navigate the Cognosys program. When required, another, smaller cube appears. The cubes act as indicators of the current procedure as well as active instruments of movement to other procedures.

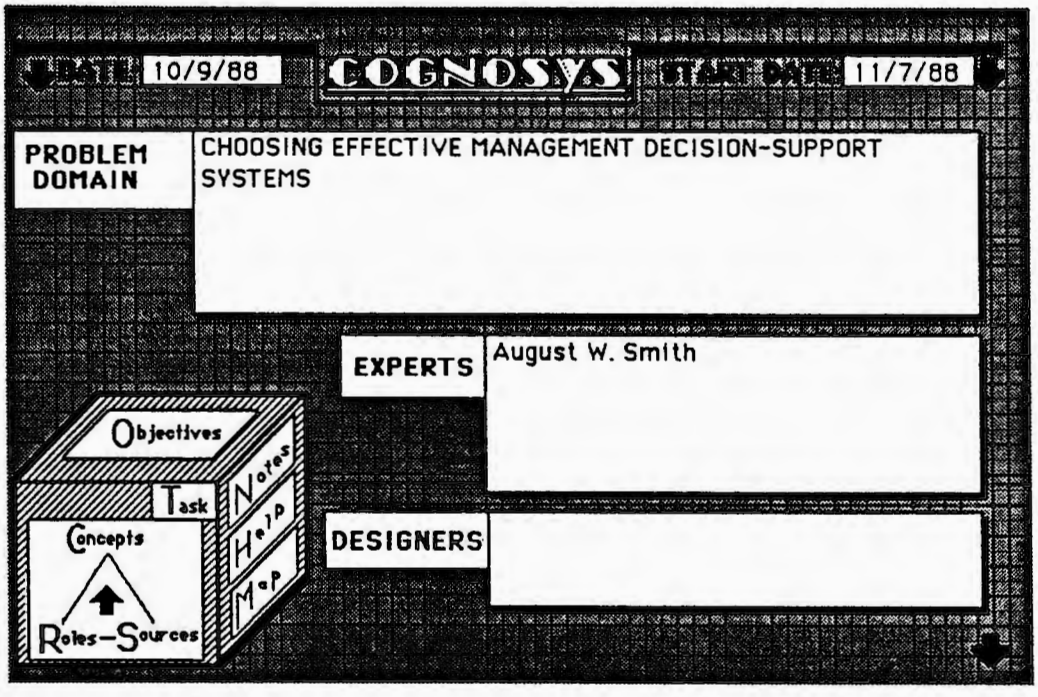


Figure 3. Navigational Cube and Domain Title

Analysis

The purpose of this component is to:

1. organize the domain into objectives (general goals);
2. analyze the objectives into tasks;
3. classify the task types;
4. analyze the tasks into functional and structural concepts;
5. classify the role of the concepts;
6. organize the source of the concepts; and
7. record any attributes of the concept.

Although not all of the information generated in steps one to seven is used in the Synthesis component, this information is necessary for two reasons. The first is to aid in search and selection routines for the cataloging of information: the second to act as potential "feed-stock" for more specific knowledge acquisition tools used later in the knowledge engineering process.

1. The Objectives Level

Figure 4 shows the screen for the first objective which has been entered manually or from file. A short title for reference is also entered. Other objectives are added until all are entered. The task level is then chosen for each objective in turn.

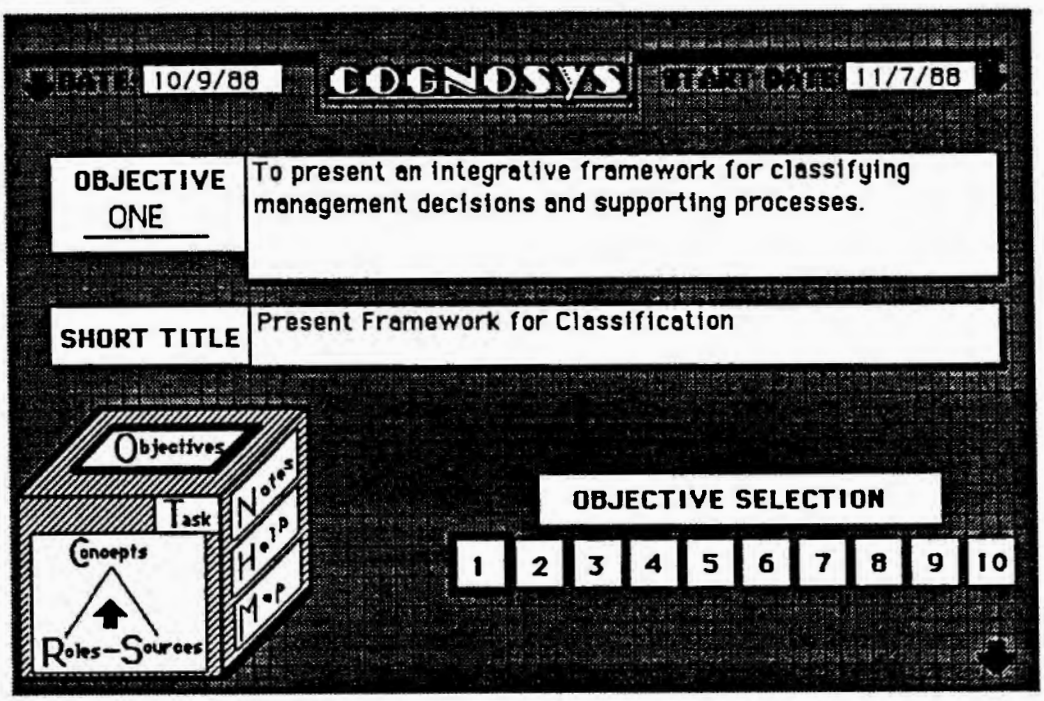


Figure 4. The "Objectives" Level

2. The Task Level

Figure 5 shows a task description for a given objective. This is a short paragraph containing 5 to 6 sentences. These descriptions are entered manually or from file. Now the task is typed using Greeno's taxonomy of task types. The knowledge engineer reviews each task description and identifies the various types: pattern finding, transformation and arrangement, or any combination of the three. Pattern-finding tasks require the decision-maker to identify a known pattern of objects, data or ideas. Transformation tasks are characterized by a given initial pattern and a desirable end pattern. The problem-solver is required to use legitimate steps to transform the initial pattern to coincide with the desired pattern. Arrangement tasks refer to those tasks for which only pieces of information are given and a final arrangement or pattern is required to meet certain design or functional criteria. Often, tasks can have more than one 'type' assignment. If this occurs, all types are recorded and the task may be further analyzed to a single type by the human knowledge

source in the next iteration of Cognosys.

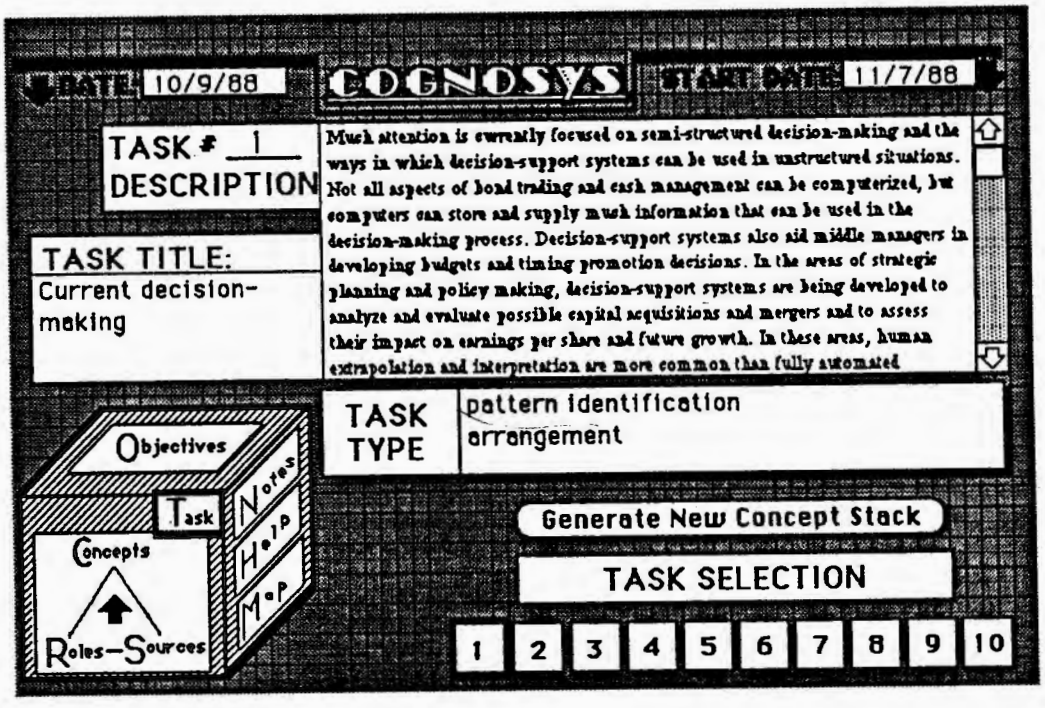


Figure 5. The "Task" Level

At this level of analysis, the Greeno categories provide an appropriate format for task typing. Chandrasekaran (1988) and McDermott (1988) have developed task-level categories for describing task demands, but these systems were developed for analysis at a more specific level of knowledge acquisition. They are extremely rigorous and useful frameworks but are not required at the domain organization level of analysis.

3. The Concept Level

This level of analysis reflects the linguistic assumption that a word (noun, adjective, verb or adverb) is a single concept (Graesser, 1985; Fauconnier, 1987). The concepts are selected from the task description paragraph by the knowledge engineer and classified as either structural (noun or adjective) or functional (verb or adverb) (Figure 6). These single word concepts act as cognitive primitives for the analysis sub-component of Cognosys. The single concepts are stored together with their task context and any attributes which may have been denoted in the task paragraphs (Figure 7).

The source of information concerning the concept is also recorded. The source could be the human knowledge source, calculated information or a data source. Finally, the role of

the concept is declared. This declaration may be a descriptor of a structural or functional concept, but more often, the best way to describe the role is to place the concept in the phrase within which it appears in the paragraph.

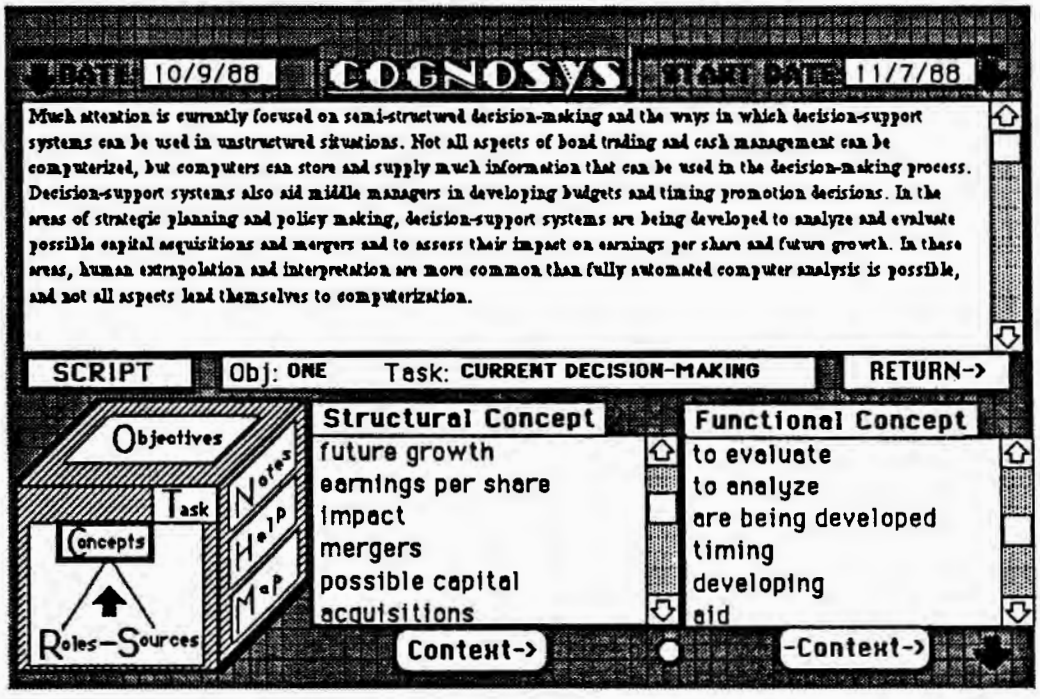


Figure 6.Classification of Concepts

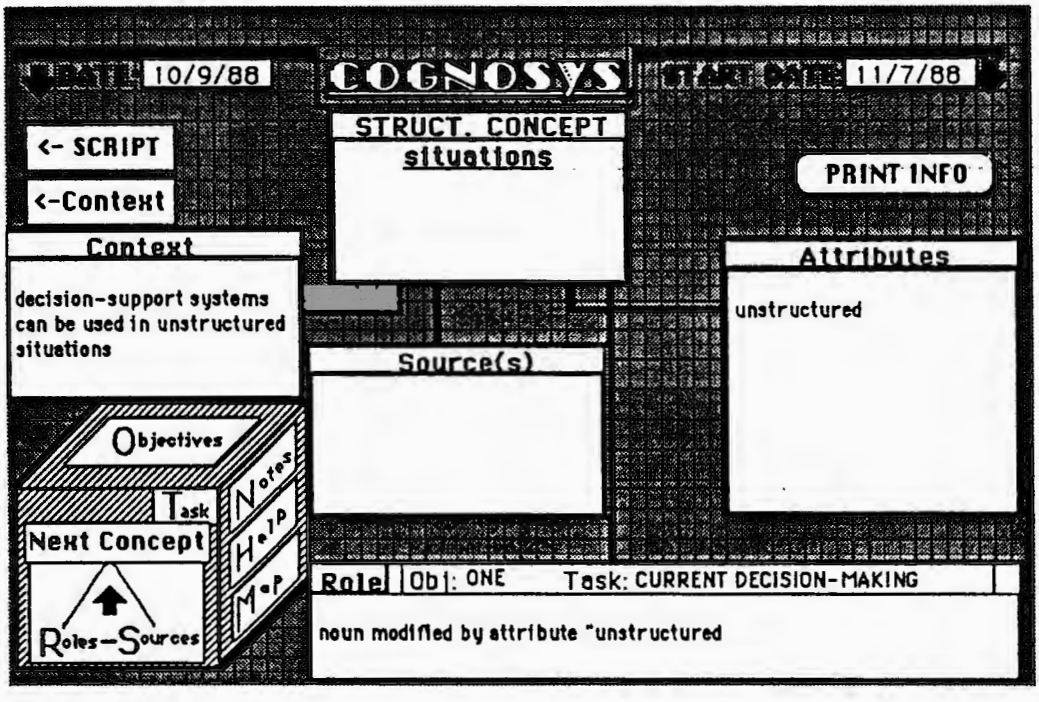


Figure 7. The "Concept" Level

Often, the first run through the Cognosys program does not provide enough information for the completion of each category for the concept description (Figure 7). This information is usually forthcoming in subsequent discussions.

4. Summary

This portion of the Cognosys program provides the knowledge engineer with a methodology for analyzing the problem domain into primary objectives and each subsequent set of tasks. The knowledge engineer requests of the human knowledge source a breakdown of the domain into required operational or procedural objectives required in order to arrive at a satisfactory solution to classes of domain problems. These objectives are then decomposed into tasks with short descriptive paragraphs. The tasks are typed and the concepts isolated, typed, and stored with their sources and attributes.

Synthesis

Most of the organization of the domain model occurs as a result of synthesis activity. The purpose of this Cognosys component is to:

1. isolate and type discrete knowledge primitives (K-Units);
2. identify relationships among K-Units;
3. display K-Unit relations for each task in conceptual graphs to form a Specific Knowledge Structure (SKS);
4. combine task-level conceptual graphs into General Knowledge Structures (GKS) for each objective; and
5. combine each GKS for objectives into a domain model.

1. Methodology

The method used to develop the GKSs is based on work by Graesser (1981) and Graesser and Clark (1985). This method of determining structures and procedures of implicit knowledge has been successful in understanding how readers comprehend written text. The primary component is that of a GKS which is described as "a rich configuration of information which contains a combination of sub-structures..[which are considered].. declarative knowledge structures" (p.4). Other examples of general knowledge formats or packages are frames (Minsky, 1975), scripts and/or plans (Schank and Abelson, 1977), productions (Anderson, 1983), memory organization packages (Schank, 1982), and thematic abstract units (Winston, 1984). These systems each provide a framework for understanding based on syntactical analysis. Graesser's GKSs concentrate rather on

mapping the content and structure used in understanding specific text. This approach was selected because it was more content-oriented than the others and because it provided a framework for modelling " the actual content and organization of knowledge in the cognitive system" (p. 47) of the human knowledge source. Also, it was chosen because of its representational quality. The representational system for GKSs is the conceptual graph. With variations, this type of system has been used successfully by many researchers; most notably Sowa (1984). The conceptual graphs used in Cognosys are, at times, more restricted than those of Sowa in that they represent a set of global relationships between and among concepts and therefore cover fewer examples or types of precise relationships. At other times, the conceptual graphs are more flexible and encompassing because they do not require the rigor of Sowa's conceptual graphs since his goal is to describe his conceptual graphs in close-to-codeable form. It is not unlikely, however, that many of Sowa's inference techniques can be used with Graesser-type conceptual graphs (C-graphs).

The C-graphs represent knowledge at two levels: the specific and the general (or the task and the objective). Specific knowledge structures (SKSs) represent task level knowledge. These are combined to form a GKS for each objective in the domain. Each consists "of a set of categorized statement nodes which are interrelated by a network of directed, relational arcs" (Graesser and Clark, 1985, p. 48). The statement node is similar to other knowledge representations formats such as Sowa's conceptual graphs and Anderson's productions. These nodes are categorized as:

1. a *state* which represents a situation remaining unchanged throughout a period of time;
2. an *event* which represents a situation which changes within a period of time;
3. a *goal* which describes a state or event which is desired; or
4. a *style* which modifies states or events by specifying the manner in which an event occurs or further distinguishes a state.

Relational arcs act to connect the nodes within a restricted framework, but they capture relationships which frequently exist between nodes in a knowledge structure. The types are:

1. *Consequent* arcs which demonstrate how one node can causally lead to another node, enable another node to happen, or show how one node results in the occurrence of another node (temporally bound);
2. *Implies* arcs which are weaker than consequent arcs but are also used to relate event or state nodes when both node types occur at the same time (temporally free);
3. *Reason* arcs which are used to denote sequences of goal state nodes such that one node is "a reason for another node";

4. *Outcome* arcs which specify whether goal nodes are achieved (temporally bound);
5. *Initiates* arcs which represent relations between state/event nodes which invoke goal nodes (temporally bound);
6. *Manner* arcs which specify how an event node is completed;
7. *Property* arcs which further define state nodes by style and/or context (temporally free); and
8. *Set Membership* arcs which reflect the set inclusion or class of inclusion of nodes.

The combinations of nodes and arcs are represented in C-graph form which can then undergo a variety of analyses and can be combined for greater complexity.

2. K-Units and their Relationships

The knowledge engineer uses the same task paragraphs as in the Analysis component of Cognosys. In this procedure, however, complete sentences are extracted from the paragraph and then truncated into noun-verb-noun units and typed (state, event, goal, style) (Figure 8). The result is a list of K-Units and their respective types.

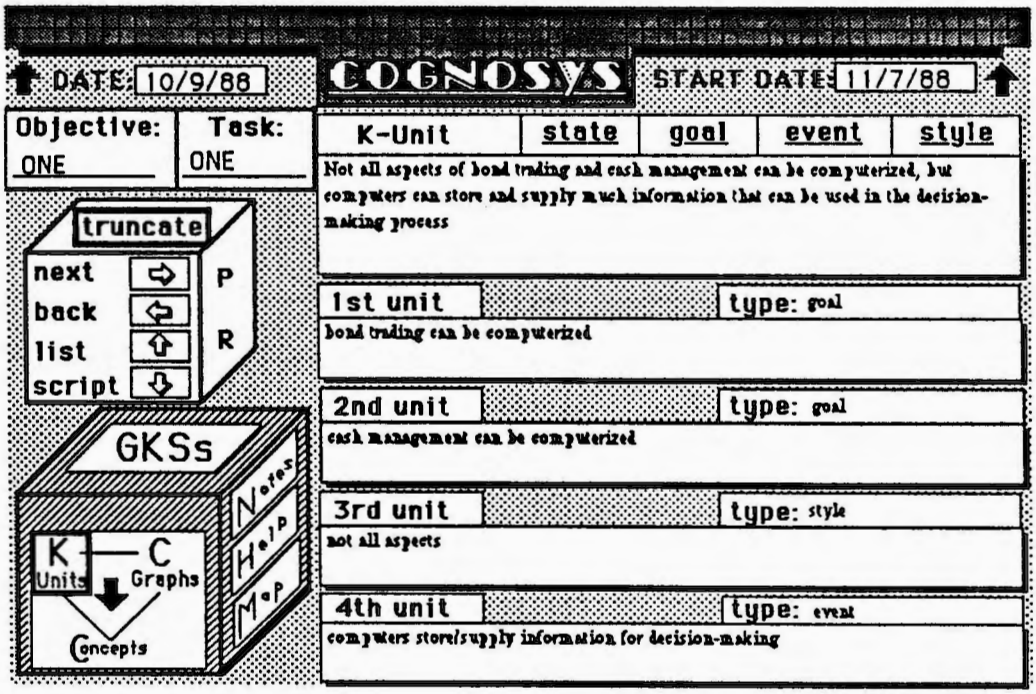


Figure 8. Truncating K-Units from Sentences

Cognosys then displays each combination of two K-Units so that the knowledge engineer can assign a relationship arc (Figure 9). This is done with reference to the task

paragraph (see Figure 7). The result of this activity is a display of K-Units and their inter-relationships.

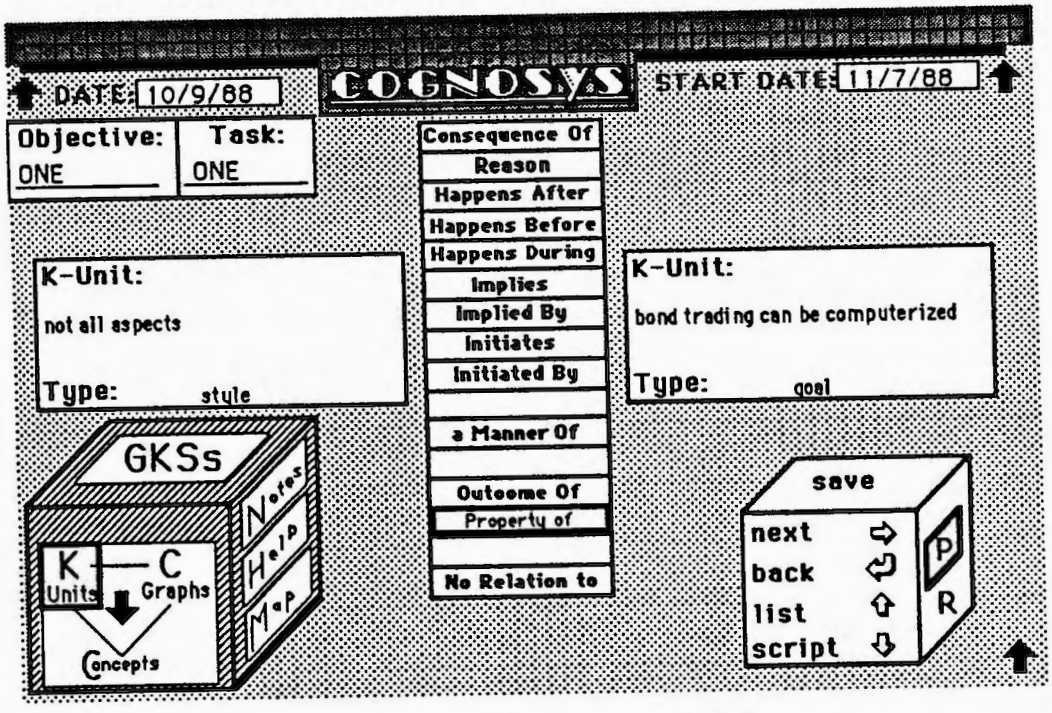


Figure 9. Assigning Relations Between K-Units

3. C-graphs

The representation of nodes and arcs in C-graph form requires analysis by Cognosys and direct arrangement by the knowledge engineer. Structural Centrality is defined as " the number of arcs that are directly connected to a given node" (Graesser and Clark, 1985, p.102) and is used to guide arrangement. Cognosys uses the list of K-Unit relations to calculate the structural centrality score for each node. This score is displayed with each K-Unit and is used by the knowledge engineer in C-graph construction. Those with higher scores are placed closer to the centre of the graph. Each C-graph represents one task. Figure 10 shows a C-graph for the example task as arranged by a knowledge engineer. This C-graph represents a specific knowledge structure (SKS) for the task.

The knowledge engineer completes this procedure for each task description for each objective. When the procedure is complete, the knowledge engineer has a series of C-graphs which can be combined to represent the objective or general knowledge structure (GKS). Because of its present hypercard environment, Cognosys cannot display a complete GKS in C-graph form. These can be constructed from the print-outs of each SKS C-graphs. Figure 11 shows a completed GKS. The nodes and arcs for the tasks are

shown. In this example, some task SKSs overlap indicating that similar or the same concepts are used for different tasks demands.

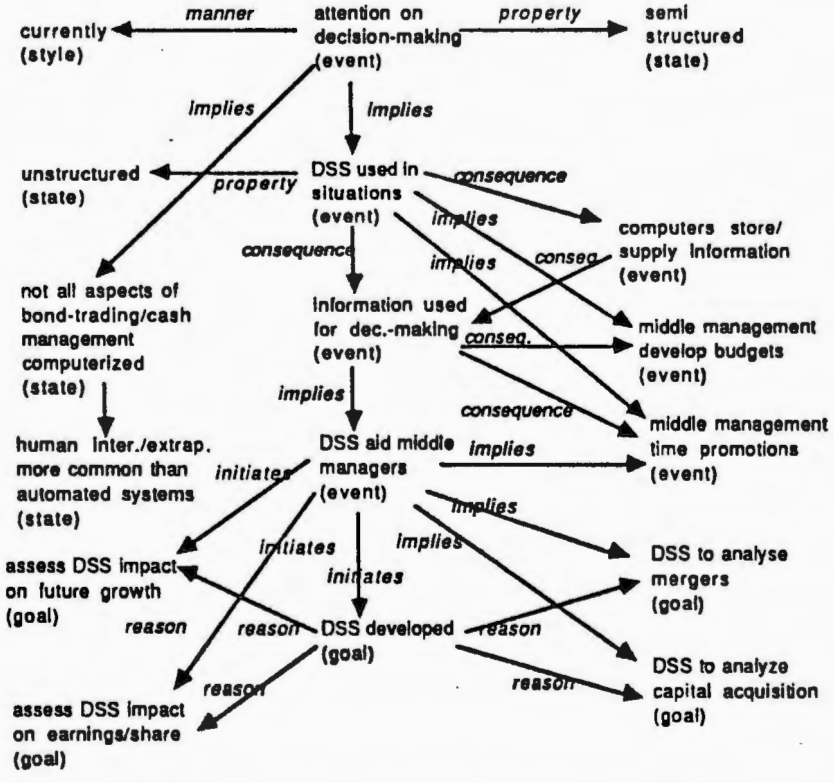


Figure 10. Sample C-graph for Task Example

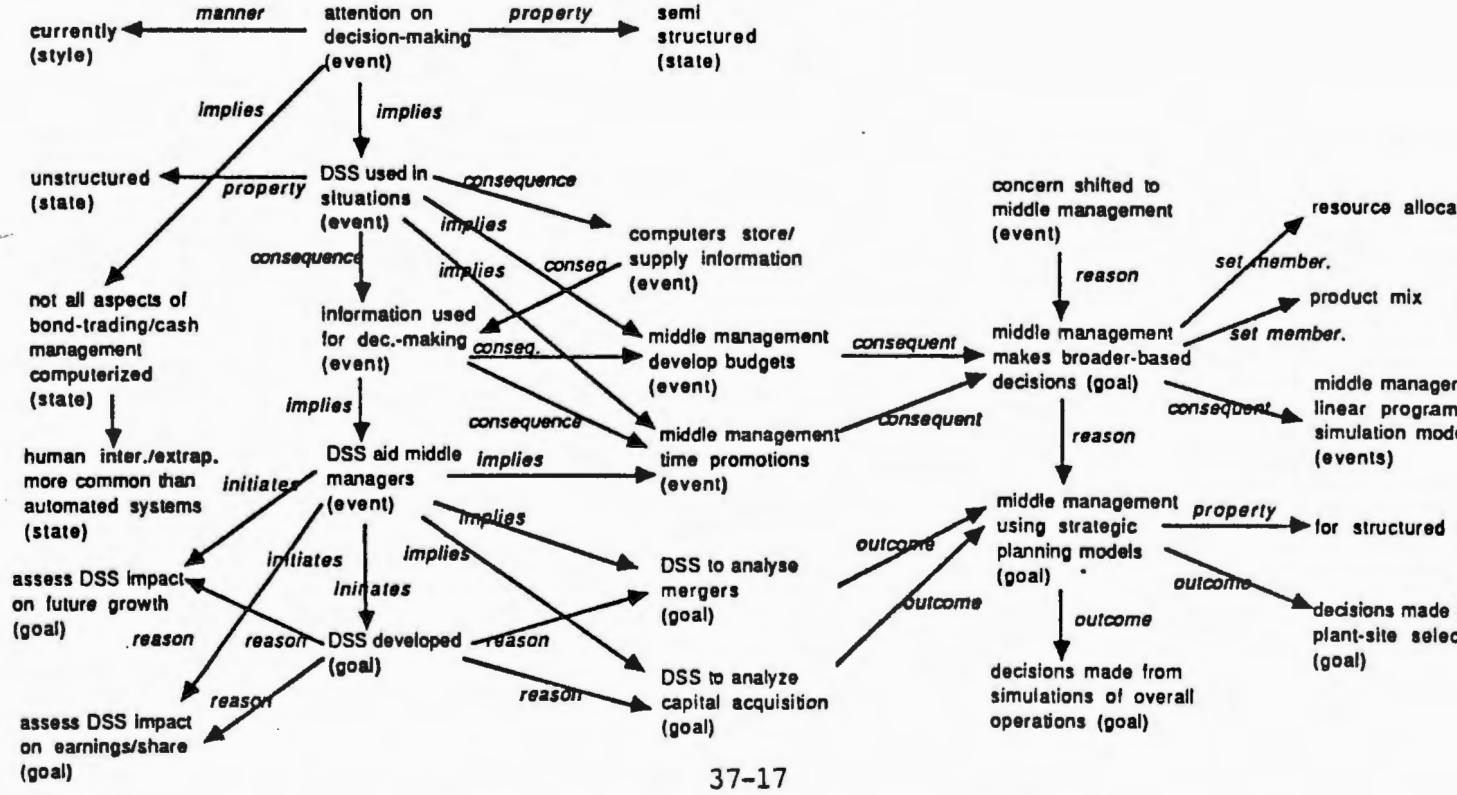


Figure 11. General Knowledge Structure (GKS) in C-graph Representation

4. Operations on GKSs

Cognosys is able to use the GKS to isolate useful sub-structures. Cause-effect, implicational, goal-oriented, and taxonomic sub-structures are useful for understanding the human knowledge source's knowledge structures for the domain. Goal-oriented sub-structures represent organized plans and activities which animate agents use to reach a set of goals. These goal structures are generated by selecting all reason, outcome, initiates and manner arcs. Sequences of events and states are graphed to display the path to goals in a temporally bound manner. These sub-structures (see Figure 12) are also useful in further discussion with the human knowledge source because they allow clarification and elaboration of temporal relationships, proper event and state sequences and goal specification.

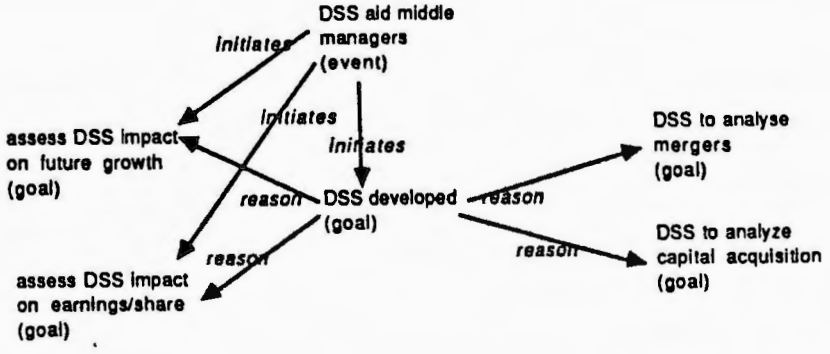


Figure 12. Goal-Oriented Sub-structure

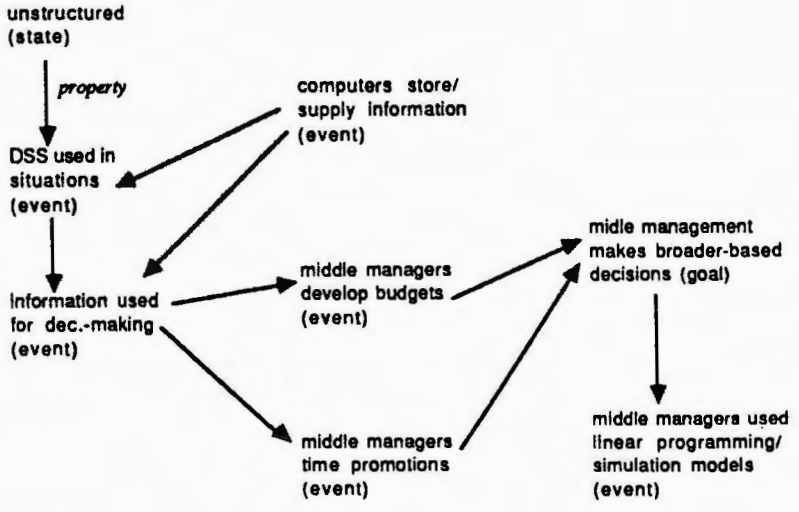


Figure 13. Cause-Effect Sub-Structure (all Consequence Arcs)

Cause-effect sub-structures capture what are specifically considered as event-effect relationships. These structures are not governed by the goal structures but represent action-initiated activity and are thus refinements of goal structures. They are generated by selecting all consequent arcs. They result in event-event node sequences which can be embellished to include related goals, states and manner nodes for further explanation (Figure 13). These sub-structures of GKSs are useful in understanding the human knowledge source's attribution of cause in the domain. This can be further elaborated in subsequent discussions.

Implicational sub-structures reflect the existence of deductive or inductive reasoning portrayed in the GKS. They are generated by selecting all implies arcs (Figure 14). These sub-structures are useful in further discussions because they allow elaboration and explanation of these inferences.

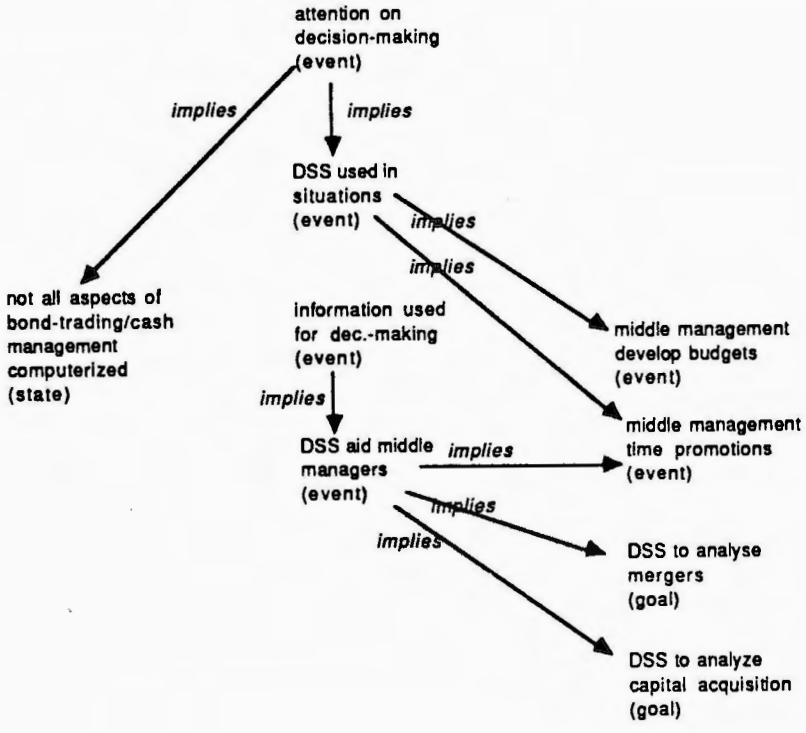


Figure 14. Implicational Sub-Structure

Finally, taxonomic sub-structures represent those relationships which suggest set and sub-set membership. These sub-structures are generated by selecting the property and set-membership arcs (see Figure 15), and are used to identify hierarchical structures within the domain.

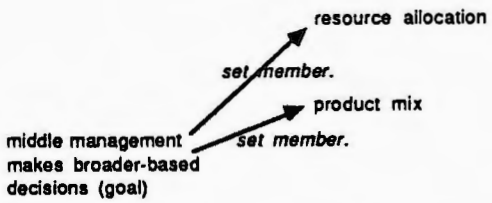


Figure 15. Taxonomic Sub-Structure

5. Summary

The knowledge engineer applies set of procedures to the task paragraphs which result in the formation of K-Units. These act as the primitives which are then typed and combined through relational arcs to form specific knowledge structures: one for each task. These SKSs are represented in C-graph form. The SKSs are then combined to form a GKS for the set of objectives. The combination of GKSs defines the domain.

Domain Definition and Organization

The problem domain is too complex for the human knowledge source, let alone the knowledge engineer, to elucidate without aid. Both individuals can attend to only parts of the domain at any one time, so they require a full model of the domain to aid in discussions about the domain and its components. Cognosys helps the knowledge engineer construct this domain model with very little time required from the human knowledge source. Through a series of clarification interviews, the knowledge engineer re-designs and tests domain models prior to expensive time in specific knowledge elicitation. Apart from clarifying nodes and arcs, portions of the model are culled or expanded as required. The boundaries of the domain are clearly identified, the central domain content and structures defined and the edges of expertise recognized. Cognosys acts as a facilitation tool in these initial design stages.

Now, decisions are made about the next steps in the knowledge-base system development. Several knowledge-based systems may be identified. The human knowledge source identifies a specific problem within the domain to focus on, or prefers to deal with one area only within the defined domain model. Decisions for resource commitment are made based on the domain model. For the knowledge engineer, general task types are identified so decisions about fine-grained knowledge acquisition tools, knowledge representation formats, and inference processes are made. Sources of domain knowledge are identified and organized.

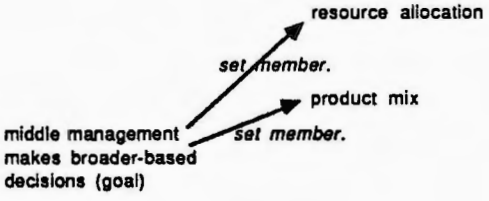


Figure 15. Taxonomic Sub-Structure

5. Summary

The knowledge engineer applies set of procedures to the task paragraphs which result in the formation of K-Units. These act as the primitives which are then typed and combined through relational arcs to form specific knowledge structures: one for each task. These SKSs are represented in C-graph form. The SKSs are then combined to form a GKS for the set of objectives. The combination of GKSs defines the domain.

Domain Definition and Organization

The problem domain is too complex for the human knowledge source, let alone the knowledge engineer, to elucidate without aid. Both individuals can attend to only parts of the domain at any one time, so they require a full model of the domain to aid in discussions about the domain and its components. Cognosys helps the knowledge engineer construct this domain model with very little time required from the human knowledge source. Through a series of clarification interviews, the knowledge engineer re-designs and tests domain models prior to expensive time in specific knowledge elicitation. Apart from clarifying nodes and arcs, portions of the model are culled or expanded as required. The boundaries of the domain are clearly identified, the central domain content and structures defined and the edges of expertise recognized. Cognosys acts as a facilitation tool in these initial design stages.

Now, decisions are made about the next steps in the knowledge-base system development. Several knowledge-based systems may be identified. The human knowledge source identifies a specific problem within the domain to focus on, or prefers to deal with one area only within the defined domain model. Decisions for resource commitment are made based on the domain model. For the knowledge engineer, general task types are identified so decisions about fine-grained knowledge acquisition tools, knowledge representation formats, and inference processes are made. Sources of domain knowledge are identified and organized.

Cognosys concepts provide a feed-stock for specific knowledge acquisition tools. For example, KSS0 requires "entities" which act as the basis of elicitations and can then be used in knowledge-base development. These elicitations are most effective and accurate when the "entities" (or concepts in Cognosys) are at the same level of generality or granularity, are ordered heterarchically, rather than hierarchically, and share general commonality or set relationship. Further refinement of concepts into entities is accomplished through a series of node ratings on three dimensions. The first is typicality which requires the human knowledge source to rate all nodes on how typical they are of the GKS. Given the GKS, how frequently would the node occur. The second is necessity, which refers to the question of how necessary a node is for the GKS to occur. In other words, would the GKS exist/occur if the node were absent? The last scale is distinctiveness, which refers to the degree to which the node is particular to a given GKS or if it appears in many GKSs with in the defined domain. Low distinctiveness scores indicate a common activity, goal or state within the domain.

Future Developments for Cognosys

As a semi-automated knowledge acquisition tool, Cognosys requires informed judgment from the knowledge engineer. Whether the knowledge engineer role is filled by a domain expert or not, using Cognosys requires knowledge about task analysis and GKS development. Successive versions of Cognosys will contain increasingly more automated procedures. The present hypercard environment has been useful for rapid development of the prototype but a more powerful object-oriented environment such as smalltalk is better suited to more automation. Specifically, automatic generation of C-graphs for SKSs and GKSs is necessary. Also, multiple windows would make some processes and decisions easier.

A wide variety of analyses are possible using GKSs: using node and arc types and combinations to identify clusters and sub-clusters of node/arc occurrence, procedures for GKS comparisons, and for identifying implicit knowledge structures. These are possible within the hypercard environment but are much more easily implemented in smalltalk.

Also, additional support processes would make Cognosys a more effective and general tool (Figure 16). For example, a knowledge-base of GKSs which allowed filing, sorting, matching, and combination retrieval would benefit the user. A library containing references of knowledge engineering tools and procedures, their purpose, and their application is useful. A knowledge-base for advising Cognosys users on the system itself and for selecting knowledge acquisition tools is possible. Finally, output formats for a wide variety of specific knowledge acquisition tools is necessary.

Cognosys as a Research Tool

Cognosys provides a useful framework for research into human knowledge structures and the usefulness of these in knowledge-base design and development. Research areas include:

1. identifying biases in GKSs for different task demands and human knowledge sources;
2. comparing GKSs over time to track knowledge development;
3. identifying the influence of human knowledge source cognitive style variables on knowledge acquisition;
4. applying Cognosys procedures to finer-grained knowledge acquisition activity; and
5. identifying GKS concomitants to generic level tasks (Chandrasekaran, 1988) and task methods (McDermott, 1988).

References

Abrett, G. & Burstein, M.H. The KREME knowledge editing environment. **International Journal of Man-Machine Studies**, Vol 27(2), 103-126.

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

Boose, J.H. and Bradshaw, J.M. (1987) Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems. **International Journal of Man-Machine Studies**,Vol. 26 (1), 3-28.

Bylander, T. and Chandrasekaran, B. (1987). Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. **International Journal of Man-Machine Studies**,Vol. 26 (2), 231-244.

Chandrasekaran, B. & Bylander, T.(1988) Task-specific problem-solving architectures. **Tutorial-AAAI**, St. Paul. Morgan-Kaufmann.

Eshelman, L., Ehret, D., McDermott, J. and Tan, M. (1987) MOLE: a tenacious knowledge acquisition tool. **International Journal of Man-Machine Studies**,Vol. 26 (1), 41-54.

Fauconnier, G. (1985) **Mental Spaces: Aspects of Meaning Construction in Natural Language**.Cambridge, Mass.: MIT Press.

Gaines, B.R. (1987) Advanced expert system support environments. **Proceedings of the Second Knowledge Acquisition for Knowledge-Based Systems Workshop**. Banff, Canada.

Garg-Janardan, & Salvendy,G. (1987) A conceptual framework for knowledge-elicitation. **International Journal of Man-Machine Studies**, Vol 26(4), 521-532.

Graesser, A.C. & Clark, L.F. (1985) **Structures and Procedures of Implicit Knowledge**. Norwood, NJ.: Ablex Publishing.

Graesser, A.C. **Prose Comprehension Beyond the Word**. New York: Springer-Verlag.

Greeno, J.G. & Simon, H.A.(1986) Problem solving and reasoning, In (Eds.), R.C. Atkinson, R.J. Hernstein, G. Lindsey & R.D. Luce, **Steven's Handbook of Experimental Psychology**. New York, NY: Wiley.

Gruber, T.R. & Cohen, P.R. (1987). Design for acquisition: principles of knowledge system design to facilitate knowledge acquisition. **International Journal of Man-Machine Studies**, Vol 26(2), 143-160.

Harmon, P, Maus, R. & Morrissey, W. (1988) **Expert Systems: Tools and Applications**. Toronto: Wiley.

Hofmeister, A.M. (1986) Formative evaluation in the development and validation of expert systems in education. **Computational Intelligence**, 2, 65-67.

Marcus, S. (1987) Taking backtracking with a grain of SALT. **International Journal of Man-Machine Studies**, Vol. 26 (4),383-398.

Martin, J. & Oxman, S. (1988) **Building Expert Systems: A Tutorial**. Englewood Cliffs, NJ.: Prentice Hall:

McDermott, J. (1988) Exploiting task structure to automate knowledge acquisition. **Invited Address, AAAI**, St. Paul.

Minsky, M. (1975) A framework for representing knowledge. In P.H. Winston (Ed.), **The Psychology of Computer Vision**. New York: McGraw Hill.

Musen, M.A., Fagan, L.M.,Combs, D.M. and Shortliffe, E.H. (1987) Use of a domain model to drive an interactive knowledge-editing tool. **International Journal of Man-Machine Studies**, Vol 26(1), 105-121.

Parsaye, K. & Chignell, M. (1988) **Expert Systems for Experts.** Wiley: Toronto.

Scambos, E.T. (1986) A scenario-based test tool for examining expert systems. **IEEE**, 131-135.

Schank, R.C. & Abelson, R. (1977) **Scripts, Plans, Goals, and Understanding.** Hillsdale, NJ: Erlbaum.

Schank, R.C. (1982) Reminding and memory organization: an introduction to MOPs. In W.G. Lehnert & M.H. Ringle (Eds), **Strategies of Natural Language Comprehension.** Hillsdale, NJ: Erlbaum.

Shaw, M.L.G, and Gaines, B.R. (1987) KITTEN: Knowledge initiation and transfer tools for experts and novices. **International Journal of Man-Machine Studies**. Vol 27(3), 251-280.

Shaw, M.L.G. & Woodward, J.B. (in press). Validation in a knowledge support system: replication and consistency with multiple experts. **International Journal of Man-Machine Studies.**

Siegel, P. (1986) **Expert Systems: A Non-programmers Guide to Development and Applications.** Blue-Ridge Summit, PA: TAB.

Sowa, J.F. (1984) **Conceptual Structures: Information Processing in Mind and Machine.** Menlo Park, California: Addison-Wesley.

Stachowitz, R.A. and Chang, C.L. (1988) Verification and Validation of expert systems. **Tutorial-AAAI,** St. Paul: Morgan-Kaufmann: San Mateo, California.

Waterman, D.A. (1986) **A Guide to Expert Systems.** Don Mills, Ontario: Addison-Wesley.

Winston, P.H. (1984) **Artificial Intelligence (2nd ed).** Reading. Prentice-Hall.