Structured Methodology for Knowledge Acquisition in Kit Component Selection

by

Richard Griffin Keiser

B.S. Mechanical Engineering
Carnegie Mellon University, 1994

Submitted to the Department of Mechanical Engineering
and the Technology and Policy Program
in Partial
Fulfillment of the Requirements for the Degrees of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
and
MASTER OF SCIENCE IN TECHNOLOGY AND POLICY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1997

Signature of Author: _____
Department of Mechanical Engineering
June, 1997

Certified by: _____
Dr. Daniel E. Whitney
Thesis Supervisor, Department of Mechanical Engineering

Accepted by: _____
Richard De Neufville
Chairman, Technology and Policy Program

Accepted by: _____
Ain Sonin
Chairman, Department Graduate Committee

Structured Methodology for Knowledge Acquisition in Kit Component Selection

by

Richard Griffin Keiser

Submitted to the Department of Mechanical Engineering
and the Technology and Policy Program
in Partial
Fulfillment of the Requirements for the Degrees of

Master of Science in Mechanical Engineering
and
Master of Science in Technology and Policy
June, 1997

ABSTRACT

Knowledge-Based Systems (KBS or KB Systems), computer programs that incorporate engineering or design rules into the design process, represent the next phase of evolution in the product development process. Through the relationships of physical laws, geometric properties and assembly features, KBS can automate many design steps, thereby decreasing project development lead time, increasing worker productivity, and decreasing design errors, all contributing to significant cost savings. There are also negative aspects to KB systems: automated systems, once in place, may decrease creativity in the design process; they may be difficult to alter to reflect changes in capabilities; in addition, these systems are difficult, time-consuming, and costly to develop. Many of the benefits and drawbacks of a KB system are dependent not only on the choice of process to be automated, but also on the development methodology for the KB system. Many KB systems are not fully implemented because the automated system fails to meet the ambitious goal of capturing all of the knowledge of the process within the rules it executes. Other KB systems are unsuccessful because the rule set is not correctly identified. This is often attributed to inadequate communication between the groups whose knowledge is to be embodied in the rules and a lack of framework for developing rules. In this thesis, I propose a structure for the process by which rules are built. This structure is embodied in a knowledge acquisition tool that will (1) facilitate communication between functional and cross-functional groups to improve and accelerate the development of rules, and (2) assist in the identification of rule exceptions and the possibility of user interfaces with the system for "complexity management." Section 1 is an introduction to KB systems. Section 2 formally defines KB systems and their components. Section 3 describes the general steps in KBS development. Section 4 is an introduction to knowledge acquisition tools. Section 5 presents the design of a knowledge acquisition tool, KATMO, for rules development in selection processes. The methodology of this tool is comprised of three steps: knowledge acquisition, knowledge organization and rule identification. In Section 6, an application of this tool to the development of a prototype KBS for a drill kit assembly is described. Section 7 concludes with an exploration the policy implications of this work for manufacturing firms engaged in the development of KB systems.

Thesis Supervisor: Dr. Daniel E. Whitney
Senior Research Scientist at the Center for Technology, Policy and Industrial Development.

Acknowledgments

This thesis is dedicated to my parents whose guidance and wisdom brought me to the exciting and challenging environment of MIT; to Elena for love and support; and to Mark whose success has made me very proud.

I am also grateful to my Grandparents for their generous support of my studies and their frequent correspondance which I very much enjoyed.

I am thankful to have had the opportunity to work with Charlie Fine and Dan Whitney, whose insights and experience were a valuable asset to my development at MIT, and the completion of my thesis.

In addition, I would like to thank Bill Bullock, Brad Gale, Linda Poole and Neal McCollum at LMTAS for facilitating the project which is the subject of this thesis.

Special thanks to Mike Packer for his candor, accessibility and insight into aircraft production.

Special thanks to Steve Woods for warmly receiving me and facilitating my experience at Boeing. Thanks to Mike Kerstetter for his valuable insight into KBS at Boeing.

Finally, I would like to thank David Seidel, Brad Leech, Matt Quinn, Danny Drake, Barry Kal, Phone Wang, John Lumpkin, Roy Rajan, Steve Coffed, and Dave Kelly for their contribution to this work and for making my experience at LMTAS enjoyable, entertaining and rewarding.

Table of Contents

List of Figures

1. Introduction

Time-to-market with new products is an important issue for manufacturing companies: Fast product development enables a company to react quickly to changing market needs and can therefore be a source of competitive advantage.

The desire to improve and accelerate the product development process has led to several organizational innovations over the past few years. These include:

Company focus:	Companies now emphasize customer awareness as a key component to developing successful products
Company structure:	Companies have become less hierarchical in order to facilitate communication between managers and employees
Project Teams:	Companies now use cross-functional teams to facilitate communication across departments

One technological innovation that can facilitate productivity gains in product development is system automation. By encoding knowledge into computer programs called *Knowledge-Based Systems (KBS or KB systems)*, many tasks may be automated [Masud]. One area where KB systems present a significant advantage is in the execution of iterative tasks. Design iterations and revisions are important drivers of lead time. By automating iterations, expert systems not only accelerate the product development process, but also enable a company to quickly experiment with alternative design solutions [Boeing].

There are other advantages associated with using KB systems:

Higher Quality: Because these systems embody the knowledge of specialists or *domain experts*, quality solutions are generated consistently.

Decreased Variability: Because the encoded knowledge is not applied in a subjective manner, the variability of solutions can be reduced. In the case of KB systems for component selection, decreased variability also results in decreased inventory requirements (see 6.3).

Expertise Retention: By capturing expert knowledge and appropriating it to the organization, KB systems retain skills after the departure of the expert [Field].

KB systems are not, however, without drawbacks. The most immediate drawback is that expert systems themselves require a long development lead time [Mills] and are expensive. These factors may contribute to low level of manager support [Field]. In addition, many of the advantages derived from KB systems have associated disadvantages:

Limited Applicability: Because generating high quality solutions is dependent on the depth and scope of the knowledge embodied in the KB system, the KBS may only be applicable to a small range of problems.

Decreased Creativity: Decreased variability in solutions may also lead to decreased creativity. By relying on an KB system, an organization may not consider all potential design solutions or all potential methods of arriving at a solution.

High Maintenance: Knowledge of systems and processes changes over time. As a result, a KB system must regularly be updated and tested.

The successful development and implementation of KB systems is dependent on several factors including: management and end-user support, the choice of the process to be automated, the development methodology, and the choice of expert system tools.

Many expert systems are not implemented because the system fails to meet the ambitious goal of capturing all of the knowledge of the process within the rules it executes. This goal is often unrealistic because of the number of exceptions to the rules. Other systems are unsuccessful because the set of expert knowledge is not correctly identified. This is often attributed to inadequate communication between the system developers and domain experts, and a lack of framework for developing rules.

While hundreds of tools exist to facilitate the development of expert systems, these tools are rarely used by manufacturing companies or KBS consultants in the development of KB systems (see Section 4.5). In fact, paper is still the most common medium for recording and analyzing solicited knowledge [Concentra]. This situation reflects the need for further research in the domain of knowledge acquisition and knowledge acquisition tools, particularly in the introduction of these tools into production environments.

In this thesis, I propose a structure for the process by which rules are built. This structure is embodied in a knowledge acquisition tool that will (1) facilitate communication between functional and cross-functional groups to improve and accelerate the development of rules, and (2) assist in the identification of rule exceptions and the possibility of user interfaces with the system for "complexity management." Section 1 is an introduction to KB systems. Section 2 formally defines KB systems and their components. Section 3 describes the general steps in KBS development. Section 4 is an introduction to knowledge acquisition tools. Section 5 presents the design of a knowledge acquisition tool, KATMO, for rules development in selection processes. The methodology of this tool is comprised of three steps: knowledge acquisition, knowledge organization and rule identification. In Section 6, an application of this tool to the development of a prototype KBS for a drill kit assembly is described. Section 7 concludes with an exploration the policy implications of this work for manufacturing firms engaged in the development of KBS systems. The structure of this thesis is represented in Figure 1.1.

Section 1 Introduction

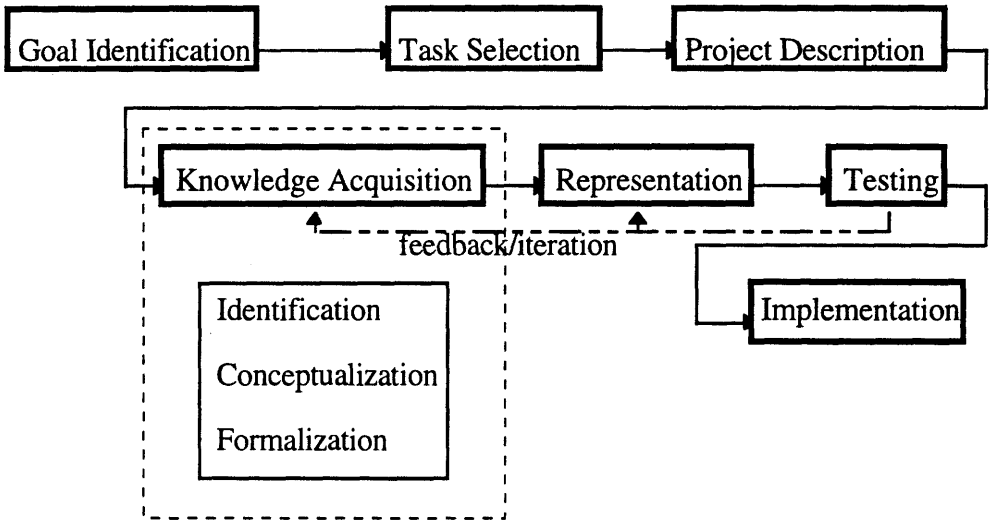
Advantages and Disadvantages of KBS

Section 2 Introduction to KBS

Definition of KBS

Components of a KBS

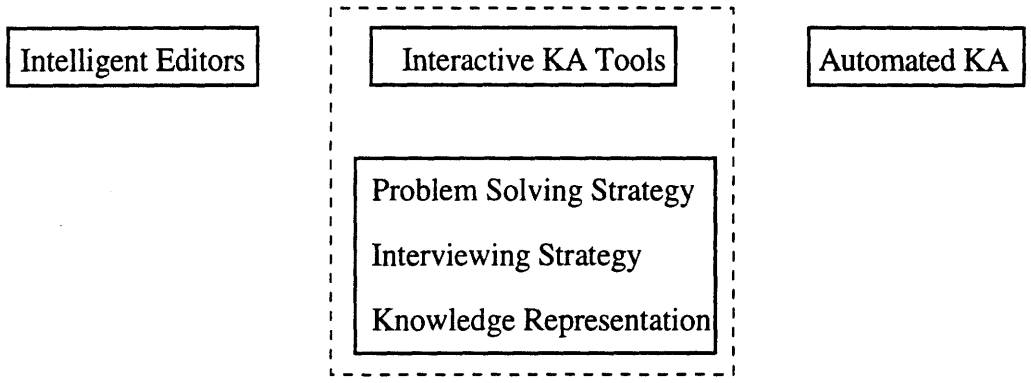
Section 3 KBS Development Process



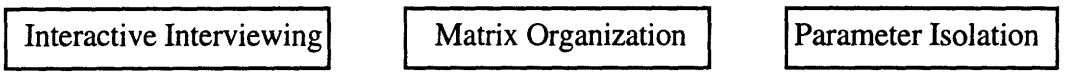
a.

Figure 1-1 Thesis Flow Diagram

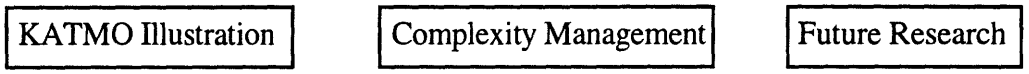
Section 4 Knowledge Acquisition Tools (KA Tools)



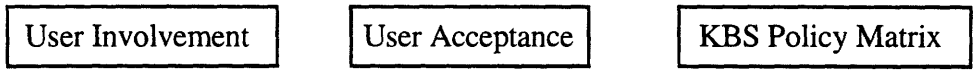
Section 5 Knowledge Acquisition Through Matrix Organization (KATMO) Tool



Section 6 An Application of KATMO: Powerfeed Drill Component Selection



Section 7 KBS Technologies and Business Policy



b.

Figure 1-1 Thesis Flow Diagram

2. Introduction to Knowledge-Based Systems

Knowledge-based systems are an outgrowth of a field of Artificial Intelligence, termed *Expert Systems*, from the 1960s. Expert Systems initially focused on the construction of high performance programs in specialized domains, a pursuit that sought to understand and encode knowledge that underlies human expertise [Hayes-Roth et al.]. Initial successes in the field of expert systems were achieved by Feigenbaum, who created "DENDRAL," an expert system that interprets data from a mass spectrometer, in 1973. Further development in expert systems has led to a set of principles, tools and techniques that are now referred to as *knowledge engineering* [Hayes-Roth, et al.]. Today, types of expert systems are employed in high technology companies throughout the world for thousands of different operations.

2.1 Defining Knowledge-Based Systems

There exist many different terms in the academic literature referring to KB systems, these include: expert systems, automated systems, knowledge-based expert systems, rule-based design systems, etc. While some authors use these phrases interchangeably, others distinguish between terms. One common distinction between KBS and expert systems is the notion that an expert system is a smaller set of knowledge-based systems including only those systems enhanced with a computer application [Masud]. Another distinction is that knowledge-based systems embody knowledge, and that this knowledge is not necessarily expert knowledge [Smith; Schmoldt and Rauscher]. In this thesis, I will conform in notation with those authors who choose to make this latter distinction. Specifically, I will define a knowledge-based system as a computer system "which embodies knowledge about a specific problem domain and can thus be used to apply this knowledge to solve problems from that problem domain [Smith]."

2.2 Components of a Knowledge-Based System

Knowledge-based systems are comprised of essentially two main parts: a knowledge base and an inference engine. In addition, there are usually subprograms to facilitate interactions between the system and its environment; such as the user interface and interfaces for communicating with external databases [Buchanan and Shortliffe]. The knowledge base is composed of formal rules, definitions, facts, and expert level heuristics for solving or diagnosing problems [Edosomwan]. The inference engine provides an organized procedure and controls for applying the knowledge base in solving problems [Edosomwan]. Figure 2-1 illustrates this structure.

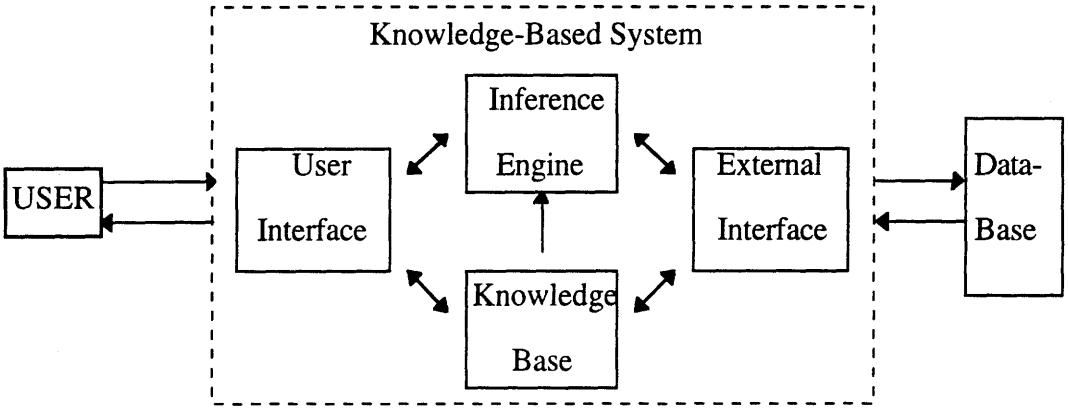


Figure 2-1 Knowledge-Based System Structure [Adapted from Buchanan and Shortliffe]

These components typically interact with the user in the following way. A user inputs information concerning a particular problem. The user-interface then translates that information into a form understood by the inference engine. Based on that information, the inference engine accesses the knowledge base and executes the relevant rules. The result of this execution is the solution generated by the KBS. This solution is returned to the user.

3. The Development of Knowledge-Based Systems

The development of an knowledge-based systems is often a complicated, expensive, and time consuming task: KBS development projects can last between several weeks and several years and can cost, including implementation, more than $1M. An awareness of the costs of developing knowledge-based systems served as an impetus for much research on the stages of knowledge-based system development. Many models of this process have been developed (see for example Buchanan et al., Schmoldt and Rauscher, Smith and Kandel, Weiss and Kulikowski, etc.). I have based the diagram in Figure 3-1 on a schematic by Bachrach because it was derived from recent contact with both companies using KBS applications and consultants who develop KBS systems. I have combined elements of this diagram with research by Buchanan et al. for the purpose of further discussing the Knowledge Acquisition phase, thus facilitating the explanation of knowledge acquisition tools later in Section 4. Figure 3-1 represents the development process of a KBS as discussed in this thesis.

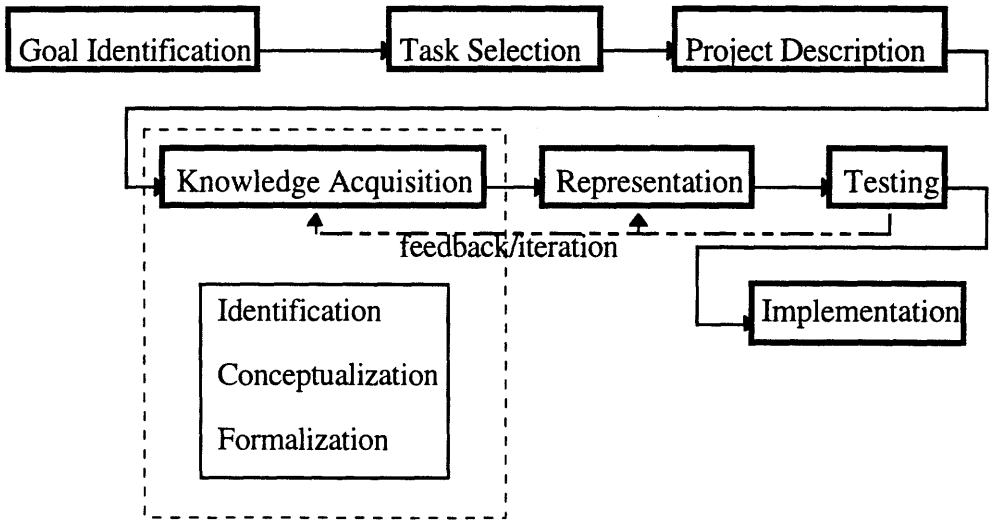


Figure 3-1 KBS Development Path [adapted from Bachrach and Buchanan et al.]

The process of developing a knowledge based system first begins with a decision to explore KBS as a potential solution to a problem. While many other possible solutions may exist, and the process by which companies decide to pursue system automation is important, these issues will not be considered here. The development process discussed will therefore begin with the assumption that the decision to develop a KBS has been made. The following sections briefly describe each phase of the KBS Development model. A thorough discussion of these topics can be found in Schmoldt and Rauscher, Buchanan et al., Payne and McArthur, and Morik et al.

3.1 Goal Identification

The process of KBS development begins with identifying the goals of the development process. Several possible goals of this effort were identified in the introduction, these included: decrease lead time, decrease variability, improve quality, etc. Understanding the purpose of the project will help in narrowing down the number of possible tasks appropriate for automation [Bachrach]. For example, if decreasing lead time for a complete design process is the project goal, the development group might choose to analyze those tasks which either contributed significantly to lead time individually or those tasks which involved many iterations. Once goals are clearly defined, and several candidates for automation have been identified, task selection is performed.

3.2 Task Selection

Task selection involves the analysis, rating and evaluation of potential candidates for automation. Because the success of a KBS application is highly dependent on the scope and complexity of the task selected, this phase is critical [Laufmann et al.]. Various methodologies have been proposed for evaluating potential tasks [Casey; Slagle et al., Laufmann et al.]. Laufmann et al. present a thorough ranking system that rates a potential task on two levels of detail and along four dimensions: clarity of goals and measures of success, appropriateness of task for a KBS, availability of resources, and non-technical considerations (for example, organizational fit of KBS). In this methodology, many criteria along each dimension are rated as either very positive

for a KBS application ("++"), positive ("+"), neutral ("•"), negative ("-"), or very negative ("--"). The number of criteria receiving each ranking are then tallied and entered into a scoring grid for evaluation. Laufmann et al. suggest several guidelines for the selection of tasks based on the ranking. A sample scoring grid is depicted below.

Dimension	Level One						Level Two					
	++	+	•	-	--	wt.	++	+	•	-	--	wt.
Goals	2	6	1	0	0		0	2	0	0	0	
Appropriateness	0	2	2	0	0		7	45	11	4	0	
Resources	0	4	0	0	0		2	7	1	0	0	
Nontechnical	2	5	1	0	0		4	11	1	0	0	

Figure 3-2 Scoring Grid for a Candidate Task [Laufmann et al.]

One advantage of the Laufmann method is that it not only indicates whether an application is well suited to a KBS approach or not, but it also identifies along what dimension(s) the task is lacking [Schmoldt and Rauscher]. It is therefore conceivable that a task initially judged to be deficient along one or more dimensions, could be redefined, i.e. scope could be decreased, and then the task could receive a favorable rating [Schmoldt and Rauscher].

3.3 Project Description and Planning

Once a particular task has been selected, the KBS development process continues by defining more specifically the goals, scope, and schedule of the development effort. Bachrach identifies several aspects of the project that should be clarified at this time. These include:

1. What the Application will do: Match project goals to application requirements.

2. What Knowledge is Needed to Develop the Application: Match application requirements to potential knowledge sources (see 3.4.1.1 Identification).
3. How the Application will be Developed: Define more precisely the development steps.
4. How will the Developed System Interface with the Existing Environment: Understand interface issues with external systems.
5. Who will Use the Application: Identify users and user interface issues and potential affects of the application on existing jobs (see Section 7).

Another important step is the identification of the KBS development team members and its leader. Forming a capable team for the KBS development process is important because of the cross-functional nature of many KBS systems. Because many knowledge-based systems require knowledge from several sources and this knowledge must be coded in a computer application, team members with appropriate skills must recruited.

3.4 Knowledge Acquisition

Knowledge acquisition, or *knowledge engineering*, is the process of transferring and translating problem-solving expertise from some knowledge source to a program [Buchanan et al.]. There are many potential sources of knowledge, these include: human experts, textbooks, databases, and past experiences. The role of the *knowledge engineer* is to interact with these sources, form a knowledge model that represents the acquired knowledge, and encode this knowledge in a computer program. This process is represented in Figure 3-3.

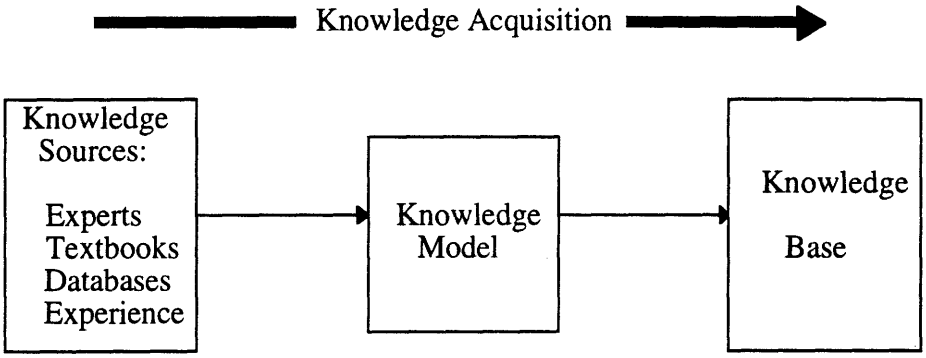


Figure 3-3 Knowledge Flow in KBS Development. [Adapted from Schmoldt and Rauscher].

The knowledge acquisition phase of KBS development is widely recognized as the bottleneck; there are several reasons for this:

1. Nature of Knowledge Storage. While an expert may fully understand his or her domain, the manner in which the human mind stores knowledge can differ greatly from the preferred structure of a computer program.
2. Semantics. Even people within the same organizations use different terminology to describe the same processes or objects. Because KBS applications often require knowledge from experts in multiple domains, establishing a common vocabulary can be problematic.
3. Many Processes are not Organized in their Current Form. The process by which a task is currently performed in a company may not be formal and may depend on many diverse criteria. Because knowledge-based systems depend on rules and heuristics for decision-making, unorganized processes may be difficult to capture with a KBS.

4. Difference in Knowledge Level between the Knowledge Engineer and the Human Expert. Often the person acquiring knowledge for a KBS application, the knowledge engineer, is not also the domain expert. In this case, the knowledge engineer must develop a certain level of understanding to converse with the expert. This education process can be time consuming.

3.4.1 Stages of Knowledge Acquisition

Literature on knowledge acquisition is abundant. Many variations and different descriptions of the process have been formulated. Buchanan et al. assert that in the process of developing a KBS, the knowledge engineer progresses through various stages. Buchanan et al. then identify five different stages of knowledge acquisition: identification, conceptualization, formalization, implementation, and testing. I have chosen to describe identification, conceptualization and formalization as stages of knowledge acquisition, while discussing implementation and testing in the context of the entire KBS development effort. In this manner, the explanation parallels the steps of the knowledge acquisition tool introduced in Section 5. The next three sections are taken heavily from Buchanan et al.

3.4.1.1 Identification

The knowledge acquisition phase of knowledge-based system development begins by identifying the participants in the process. One manner in which this is done is through a series of interviews of domain experts by the knowledge engineer. This process involves understanding the knowledge and expertise of each domain expert. Based on these interviews, appropriate experts are identified for consultation during the entire KBS development process. Another objective of the identification phase is to more precisely describe the problem the system attempts to address. This involves formal discussion of the aspects of the problem, its characteristics and its subproblems. This understanding facilitates the conceptualization of the knowledge based system.

3.4.1.2 Conceptualization

During the conceptualization phase, the knowledge engineer and domain experts make explicit the problem characteristics identified in the previous stage. Several important questions are addressed:

1. What processes are involved in the problem's solution?
2. What types of data are available?
3. Can a diagram of system hierarchy be built and can causal relationships be labeled?

Forming an accurate concept of the system is a long and iterative process, through which the knowledge engineer must repeatedly interact with domain experts. The output of this phase is the documentation of key concepts and relations.

3.4.1.3 Formalization

Formalization involves mapping the key concepts, subproblems and information flow characteristics identified during the conceptualization phase into a more formal representation based on knowledge engineering representation tools or KBS application structures. As such, this stage focuses on matching or structuring the collected knowledge into a form that can be more easily translated into a computer program. The important result of this stage is uncovering the underlying model of the process by which the desired solution is reached.

3.5 Representation

Knowledge Representation completes the Formalization process (3.4.1.3) by coding the organized knowledge into a prototype system [Smith]. There exist many possible knowledge representation schemes. These include: IF...THEN...ELSE rules, frames & scripts, objects, networks and logic [Schmoldt and Rauscher]. The representation chosen for the system should match the structure of the selected knowledge-based application. The development of a prototype is very valuable because it tests the adequacy and completeness of the formalization and the basic ideas describing

the system [Buchanan, et al.]. (Much research has been devoted to the domain of knowledge representation. For a detailed treatment of this subject see for example Payne and McArthur, Morik et at., and Marcus).

3.6 Testing

The testing stage evaluates the prototype system and the representational forms used to implement it. This is done by running the system with examples and assessing the results. If manually generated solutions to the examples exist, the KBS results can be compared to these results along some performance dimensions. Sell identifies five dimensions of KBS performance; Schmoldt and Rauscher add two final criteria:

1. Consistency: is advice free from contradiction?
2. Completeness: can the application solve any problem within its domain?
3. Soundness: can the system deliver correct answers
4. Precision: is the strength of answers in line with the data and knowledge on hand?
5. Pragmatics: can those for whom it was designed use it?
6. Representative: is the solution methodology representative of the intended design?
7. Robust: the application should degrade gracefully as the quality of input data is reduced.

Often the testing phase will identify system deficiencies and require further interaction with domain experts to improve the existing knowledge model; these measures may result in expanding the knowledge base.

3.7 Implementation

Implementation is the introduction of the KBS into the design environment. Important aspects of this process include educating the users of the system and the interfacing of the KBS with external

systems or databases. With respect to databases, it may be necessary to introduce translating programs in order to facilitate communication between the KBS application and other programs or sources. Reengineering of existing processes may be required to accommodate the KB system. Another issue in system implementation is the impact of the KBS on the user and his/her job function. These implications are explored in Section 7.

3.8 Summary

Knowledge-based system development can be viewed as a product development process that moves through several stages: goal identification, task selection, project description, knowledge acquisition, knowledge representation, testing and implementation. While this process is represented as discrete phases, KBS development is a continuous process of learning and structuring knowledge. Knowledge acquisition, the process of transferring and translating problem-solving expertise from a knowledge source into a computer program [Buchanan et al.] is critical to KBS development because this process captures and structures the knowledge upon which the KBS is built. Because of the importance of the knowledge acquisition process and the time and resources required to complete this process, many tools have been developed facilitate knowledge acquisition. These tools will be discussed in the next section.

4. Knowledge Acquisition Tools

In the previous section, the KBS development process was described as a series of tasks through which system goals are met by gathering and structuring knowledge about a system, and then coding that knowledge into a KBS application. As discussed in Section 3, this process is both time-consuming and costly. Because the Knowledge Acquisition phase of KBS development is widely recognized as the bottleneck, much research has been devoted to developing tools to facilitate this process. Knowledge acquisition tools serve as a link connecting the knowledge source to the knowledge-based system:

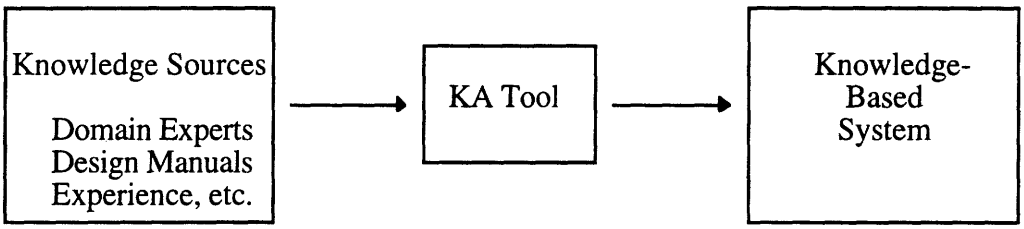


Figure 4-1 The General Function of a Knowledge Acquisition Tool

The capabilities of knowledge acquisition tools vary greatly. While some tools aim to enhance the KA process by providing computerized means of amending rules and knowledge structures, called "intelligent editors," other tools seek to fully automate the entire transfer of knowledge from its source to a computer program via machine learning. Buchanan and Wilkins categorized several well-known knowledge acquisition tools along two dimensions, complexity and level of automation (see Figure 4-2). While this representation gives insight into the intended use of the tools, Buchanan and Wilkins note that the representation does not reveal any information about the quality of the tool.

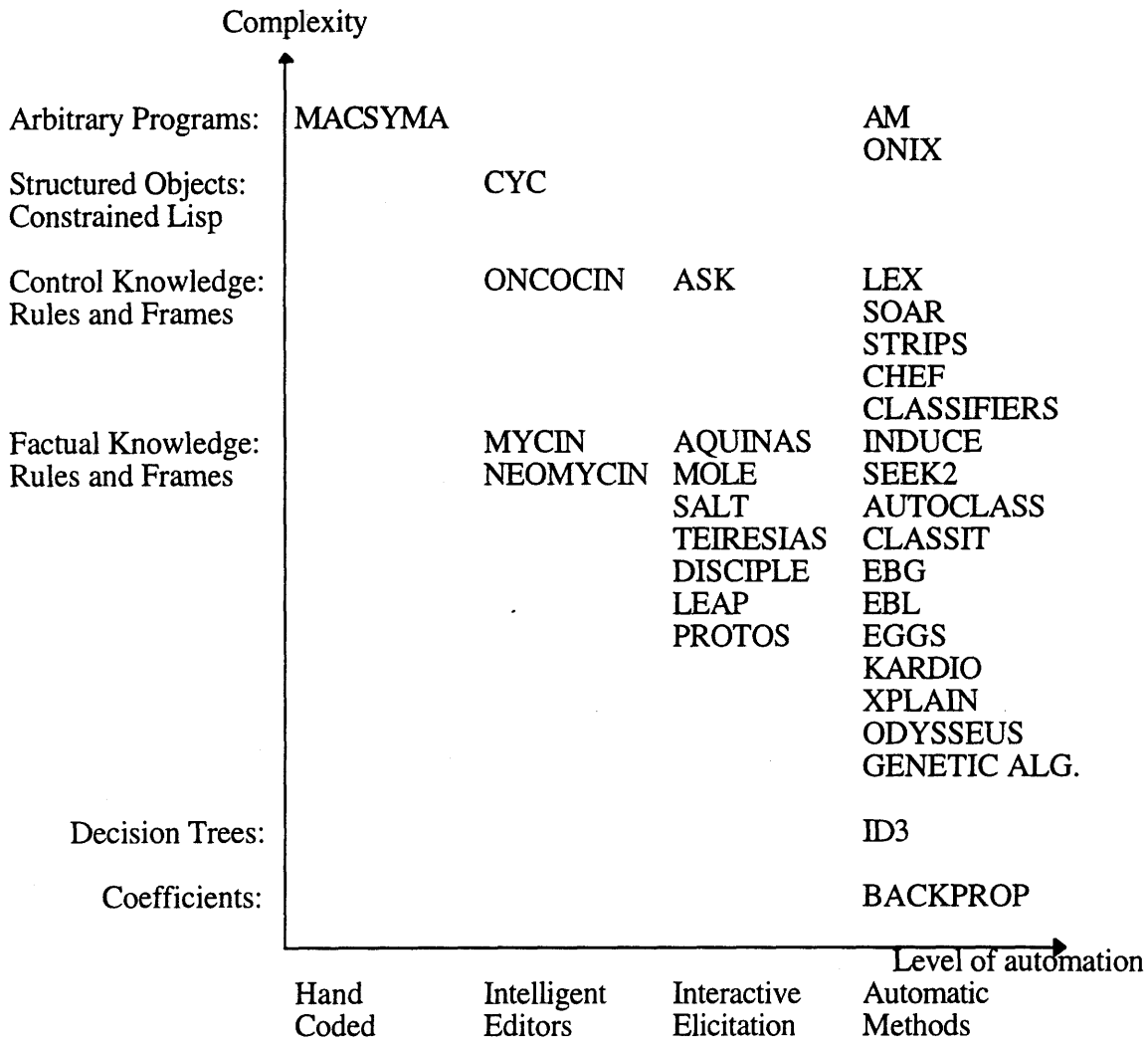


Figure 4-2 Level of automation of KA tool construction vs. complexity of the knowledge structure [Buchanan and Wilkins]

As illustrated in the above diagram, there are essentially three types of knowledge acquisition tools: Intelligent Editors, Interactive (or Semi-Automatic) Tools, and Automatic (or Machine Learning) Tools. These tools are described in the following sections.

4.1 Intelligent Editors

Intelligent editor programs facilitate the modification, expansion, and verification of information inputted to the knowledge base. An editor assists a knowledge engineer or a programmer in various ways. Most importantly, an editor acts as a translator between the user's understanding of the knowledge and the coded form of that knowledge in the KBS knowledge base. This function of an intelligent editor is represented in Figure 4-3. While the task of modifying the knowledge base may be completed by anyone in the KBS development process, I have labeled the user, "Knowledge Engineer."

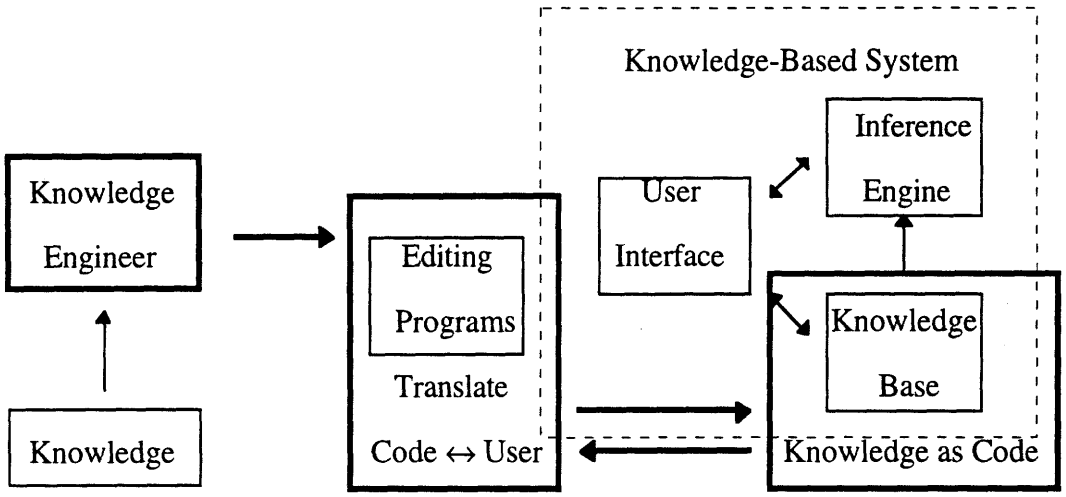


Figure 4-3 Function of a Intelligent Editor

Buchanan et al. identify three important functions of intelligent editors, these include general bookkeeping, helping the user avoid syntactical errors, and checking for semantic inconsistencies. Payne and McArthur describe a fourth important function of editors, providing graphical representations of knowledge.

1. Performing Bookkeeping Functions: General bookkeeping functions such as keeping track of unfinished editing, recognizing modifications to the knowledge base, and maintaining records of

dates, times and users altering the knowledge base are simple but important tasks in monitoring the evolution of the KBS [Buchanan et al.]. Through the automation of these tasks, knowledge engineers and programmers can focus on improving the content of the knowledge base rather than documentation.

2. Avoiding Syntactical Errors: Editors with an understanding of the knowledge-based system's building language with which they are coupled can recognize typographical and syntactical errors made by the user during knowledge-base modification [Buchanan et al.]. Several editors include such features as notifying the user of the error, and presenting the user with options for amending the error [Buchanan et al.]. These features help the user maintain a data entry format consistent with the language of the KBS code.

3. Avoiding Semantic Inconsistencies: Editors that check the semantics of the knowledge base identify inconsistencies within the knowledge base structure. Such analysis is typically performed by comparing the facts about objects in the knowledge base [Buchanan et al.]. Because hierarchically organized representations make the interrelations between facts in the knowledge base explicit, these systems are well-suited to semantic checking [Buchanan et al.].

4. Providing Graphical Representations: As the knowledge base expands, it becomes increasingly difficult to understand the elements and interrelations of elements that reside in the KBS [Payne and McArthur]. Graphical representations, such as causal loop diagrams or repository grids (see Boose, 1985), enable the user to isolate and display relationships between objects in the knowledge base. These representations can contribute to understanding of the system [Payne and McArthur].

4.2 Interactive Acquisition Tools

Interactive acquisition tools expand upon the capabilities of intelligent editors by actively querying the user for knowledge, translating that knowledge into a coded form, and then structuring the knowledge to form a knowledge base. In contrast to intelligent editing tools which are used mainly by either a knowledge engineer or a KBS system programmer, interactive acquisition tools are

typically designed for use by the domain expert [Klinker]. While this approach provides a more direct link between the knowledge source and the KBS knowledge base, Buchanan et al. note that such programs essentially replace one type of communication problem (expert to knowledge engineer) with another (expert to program).

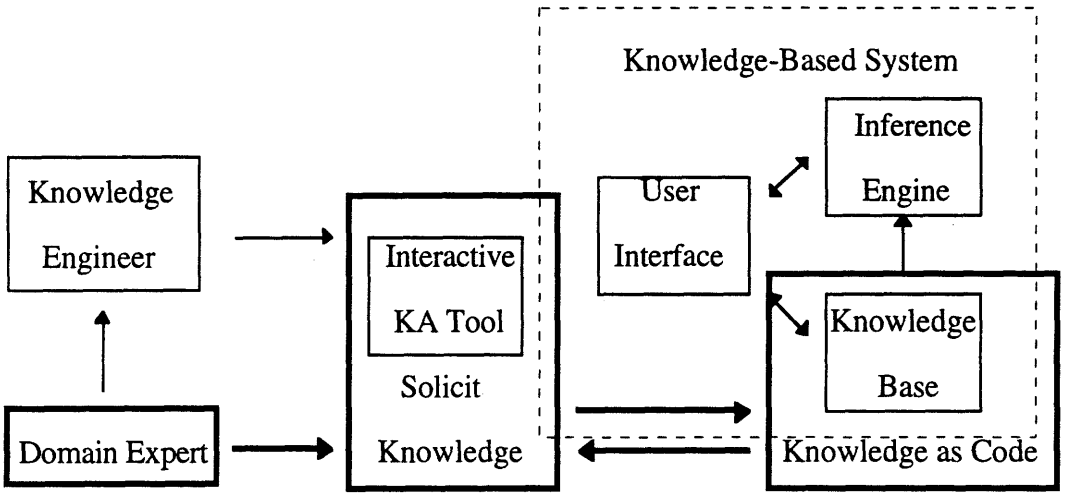


Figure 4-4 Function of a Interactive Knowledge Acquisition Tool

Despite this challenge, much progress and success has been achieved in the development of interactive KA tools. The following sections describe three important elements that differentiate interactive knowledge acquisition tools: problem-solving methods, knowledge acquisition strategies, and knowledge representation forms.

4.2.1 KA Tools and Problem-Solving Strategy

McDermott defines a KBS's problem-solving strategy as its method of performing the "identification, selection and implementation of a sequence of actions that accomplish some task within a specific domain." Klinker elaborates on this definition:

A system's problem-solving method:

1. Establishes and controls the sequence of actions necessary to complete a task
2. Defines the order in which subtasks are executed in order to complete the overall task
3. Identifies the type of domain-specific knowledge that is applicable within each step, thereby making the roles of different knowledge explicit
4. Defines the way knowledge interacts during problem-solving [Klinker].

While some automated knowledge acquisition tools do not presuppose a problem-solving method, so called "general tools" [Klinker], many applications attempt to enhance their capabilities by embodying a specific problem solving method. Such methods have been named, "Role-limiting" and are defined as problem-solving methods that "strongly guide knowledge collection and encoding [McDermott]."

In a detailed study attempting to establish a taxonomy of KBS problem solving methods, McDermott identifies the following subclassifications of Role-Limiting problem solving methods:

Subclassification 1: *Cover-and-Differentiate*

Solution by proposing solution candidates and then seeking information about the problem that will differentiate the candidates [see Eshelman].

Subclassification 2: *Propose-and-Revise*

Solution by proposing a value and then checking that value against all relevant constraints [see Marcus]. If the proposed value is not acceptable, the value is revised and re-submitted.

Subclassification 3: *Acquire-and-Present*

Solution by identifying desired pieces of information, gathering the information via search and query, and presentation of the information according to an appropriate strategy [see Klinker].

Additional discussion of problem-solving methods may be found [Boose], and [Schreiber et al.].

4.2.2 KA Tools and Knowledge Acquisition Strategy

Different knowledge acquisition strategies are employed by KA tools depending on the type of problem the KBS is designed to solve and the manner in which the solution is to be achieved. Two common techniques, interviewing and modeling are described in the following sections.

4.2.2.1 Interviews Techniques for Knowledge Acquisition Tools

KBS development entails the synthesis of knowledge from many sources. Different means of acquiring knowledge may be better suited than others depending on the domain expert's preferences and type of knowledge. In an analysis of the development of an expert diagnosis system for personal illnesses, Kahn identifies eight interviewing strategies for eliciting knowledge. While these descriptions reflect the domain in which they were developed, their mention here adds insight to different means by which a knowledge acquisition tool can elicit information.

1. Differentiation: Seeking illness symptoms that provide leverage in distinguishing between diagnosable events.
2. Frequency conditionalization: Determining if there exist background conditions under which one cause is more or less likely to occur
3. Symptom distinction: Seeking out the special characteristics of a symptom that identify it as having been cause by one rather than another event.
4. Symptom conditionalization: Eliciting conditions in which a symptom is more strongly expected, given a problem.
5. Path division: Eliciting an intermediate symptomatic event that is produced by a problem and leads to the end symptom.

6. Path differentiation: Eliciting intermediate events differentiate hypothesized events that can result in otherwise similar symptoms.
7. Test differentiation: Distinguishing the reliability of different tests.
8. Test conditionalization: Determining the conditions under which the reliability of a test may vary.

Another dimension in which interviewing tools can be differentiated is the order in which questioning is structured: Top-down or Bottom-up. Marcus asserts that a bottom-up strategy is more appropriate for soliciting knowledge from human experts who understand their individual domains much clearer than how the different domains interact. When a goal or assembly can be identified as the output of the system, a top-down approach decomposes the elements and facilitates the identification of the interactions between system elements.

4.2.2.2 Modeling as a means for Knowledge Acquisition Tools

Certain tools use models and graphics as a means for knowledge acquisition. These tools are designed to enhance the knowledge acquisition process by supplementing interviews with graphic representations. One type of modeling tool, Causal Model Based Knowledge Acquisition Tools (CMBKATs) is described by Charlet et al. CMBKAT's identify two types of knowledge upon which the KBS is built: Heuristic knowledge and causal knowledge. Heuristic knowledge describes the rules that are to be imbedded in the KBS; causal knowledge describes the relationships between elements in the KBS [Charlet et al.]. Boose classifies seven other techniques as forms of modeling.

1. Cognitive modeling: Models that captures the thought process and human problem-solving method
2. Conceptual modeling: Models that build graphical or other multi-level models
3. Consistency analysis: Analysis of knowledge for consistency or completeness

4. Decision analysis: Probabilistic inference and planning using diagrams and related techniques.
5. Domain Modeling: Models of the knowledge domain
6. Ontological and Linguistic Modeling: Language-based models
7. Simulation: Simulations to verify knowledge bases or produce rules.

As with the interviewing tools described in the previous section, modeling tools may pursue either top-down, bottom-up or mixed acquisition strategy.

Many modeling approaches to KA derive their strength from their representations. Such models present the KBS builders—either domain experts or knowledge engineers—with a graphic picture of the knowledge base [Payne and McArthur]. In previous research, these graphics have taken the form of causal loop diagrams, flow diagrams, matrices, etc. Examples of two possible representations are given in figures 4-5 and 4-6.

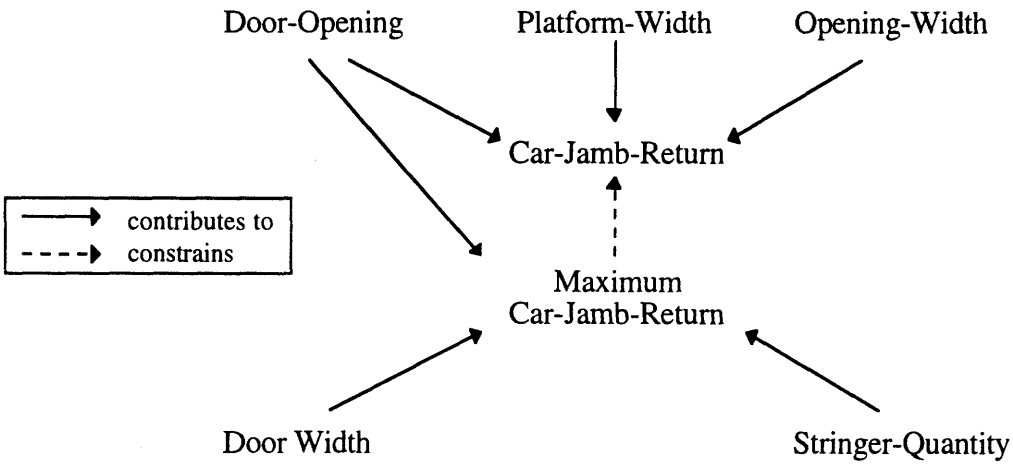


Figure 4-5 "Links among Knowledge Pieces" from SALT [Marcus]

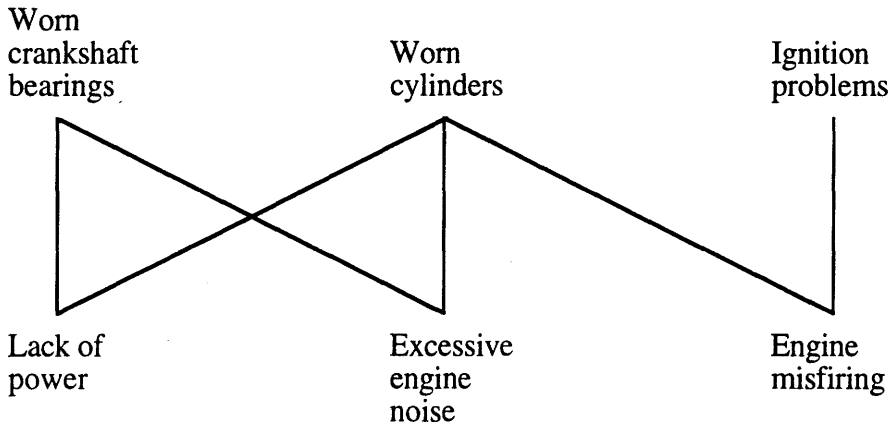


Figure 4-6 Network Representation of MOLE Knowledge Base [Eshelman]

4.2.3 KA Tools and Knowledge Representation

Knowledge representation refers to the means by which KBS elements, their characteristics and their interrelations are described within the KA tool. Different KA tools employ different representations to describe the systems they model. The most common representation is decision rules. Decision rules typically are in an IF... THEN... ELSE structure. Figure 4-7 is a sample of the knowledge representation and rule language of EMYCIN, a KA tool for developing consultation programs.

```
                <rule>  ::= (IF <antecedent> THEN <action> (ELSE <action>))
          <antecedent>  ::= (AND {<condition>}+)
           <condition>  ::= (OR {<condition>}+) | (<predicate><associative - triple>)
<associative-triple>  ::= (<attribute> <object> <value>)
```

Figure 4-7 Sample knowledge representation from EMYCIN [van Melle 1981]

Another common representation is that of a data structure. Data structures capture relevant information about each element in the system and display that information as a descriptive package.

Figure 4-8 illustrates the data structure of RLL, a complete expert system designed to build expert systems. This figure is from Barstow et al.

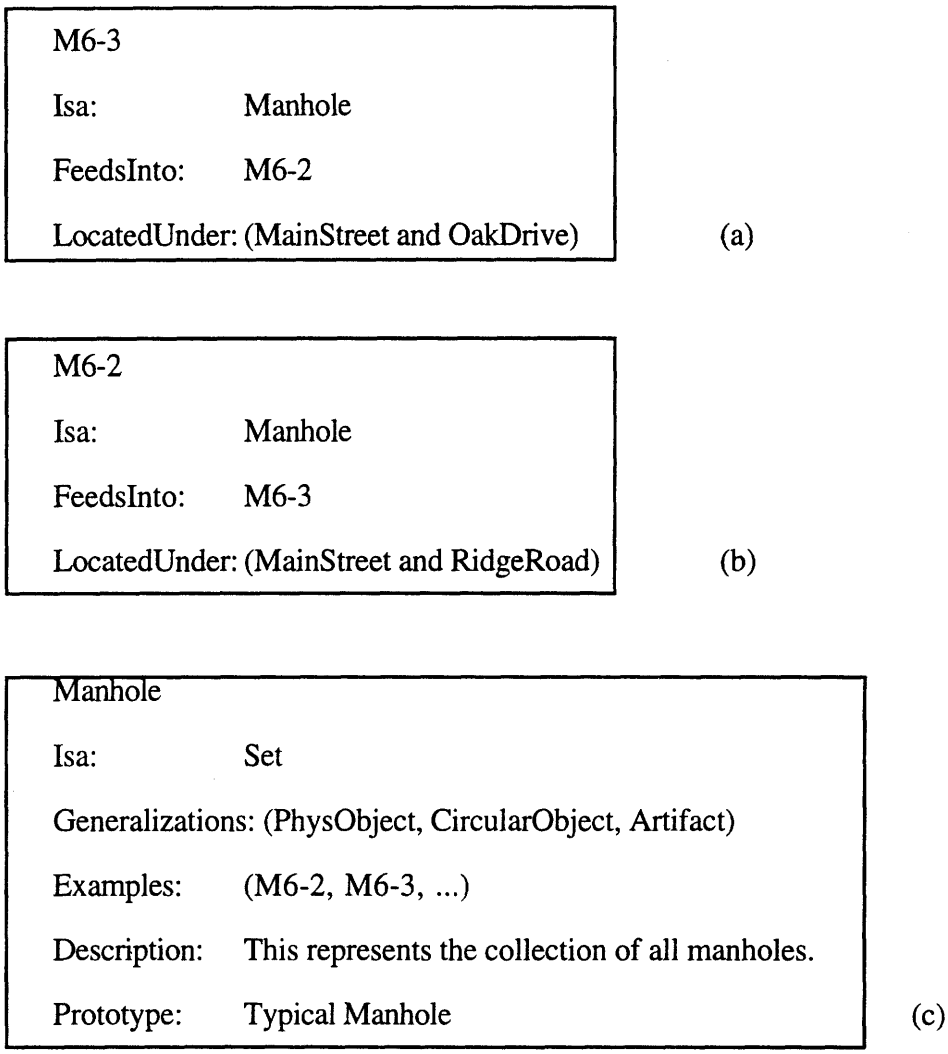


Figure 4-8 Framelike Data Structure of RLL

One characteristic of this data structure is uniformity of notation [Barstow et al.] This characteristic facilitates data organization within the tool.

Many types of knowledge representations exist. While these will not be evaluated here, several other representation schemes will be mentioned: hypotheses, findings, vectors of symbols, objects with associated vector pairs, etc.

4.3 Automated Acquisition Tools (Machine Learning)

Automated acquisition tools attempt to automate the entire KBS development process, thus directly connecting the knowledge source to the KBS, with little or no intervention from a knowledge engineer or KBS programmer (see Figure 4-9). While automated tools share many of the design issues as interactive tools such as knowledge representation, all information is processed internally.

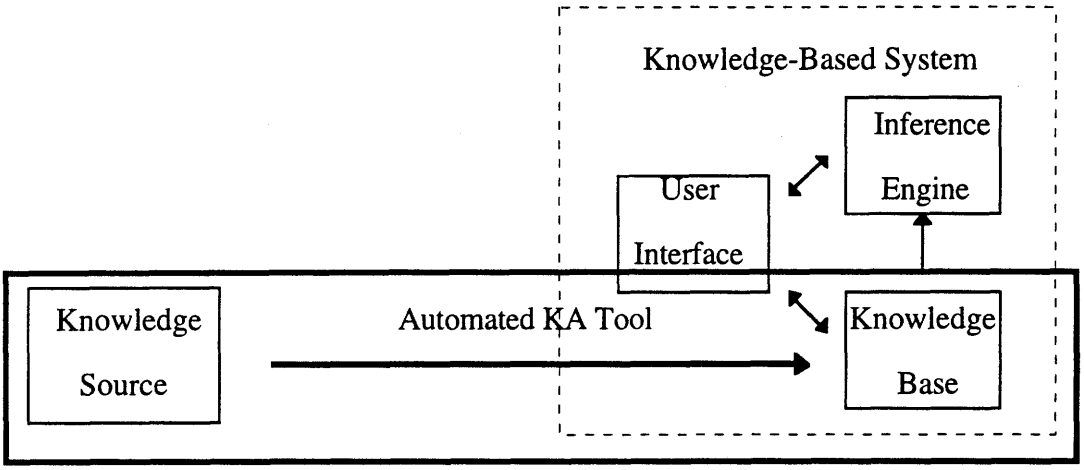


Figure 4-9 Function of a fully-automated KA Tool

Automated methods typically apply algorithms to make generalizations or induce knowledge from given examples [Boose]. Much of this work falls under the category of induction, one focus area of artificial intelligence. Inductive learning is the process of "acquiring knowledge by drawing inductive inference from teacher- or environment-provided facts [Michalski]." This process involves repeated operations of generalization, transformation, correction and refinement of knowledge representations [Michalski]. Because knowledge at the source may be in any form,

usually some common representation must be achieved before an inductive process can begin. Previous research has achieved inductive learning from many sources, these include: spreadsheets, decision trees, textbooks, statistical data, and cases. While automated tools seek to eliminate entirely the bottleneck of knowledge acquisition, most tools are still the subject of continuing research.

4.4 Survey of Historically Important KA Tools

The literature on different KA tools is abundant. While a survey of all existing tools will not be completed here, several tools are worth noting due to their historical and technological importance.

TEIRTERAS [Davis] is one of the first knowledge acquisition tools. TEIRESIAS is invoked when a system user is displeased with the performance of the expert system. TEIRTERAS determines the inadequacies in the expert system's knowledge base and then queries the user to acquire that knowledge [McDermott and Dallemagne]. TEIRTERAS is oriented toward the syntax of rule-based systems and heuristic classification [Buchanan and Wilkins].

EXPERT [Weiss et al.] is a general system for developing consultation models and has contributed to many medical applications [Barstow et al.]. EXPERT's knowledge representation is comprised of three components: hypotheses, findings and decision rules.

EMYCIN [van Melle et al.] is a general system for developing a consultation program that can request data about a case and perform an interpretation or analysis. EMYCIN's knowledge representation is comprised of production rules.

KAS [Duda et al., 1979] is a system for constructing rule-based diagnostic systems [Waterman and Hayes-Roth]. KAS represents knowledge as either probabilistic inference rules (IF

<antecedent> THEN <rule-strength>etc.) or partitioned semantic networks [Waterman and Hayes-Roth].

AQUINAS [Boose] is a collection of integrated tools that combine different strategies to build a model of the system knowledge. These tools include objects for managing hierarchical structures, uncertainty and induction. Knowledge in AQUINAS is represented in several forms including rules, hierarchical trees and rating grids.

KADS [Wielinga and Breuker, 1986] is a model-based method that uses rapid prototyping as an exploratory tool in the KA process [Smith]. KADS classifies the system knowledge to be acquired into several levels and then uses this information to shape an informal model.

SALT is a knowledge acquisition tool that generates designs based on a process of proposing a value, checking the values against relevant constraints, and then revising the values if constraints are not met [Marcus and McDermott]. SALT is one of the first tools that treats problems of construction (of a solution) rather than analysis (via decomposition for example).

EXACT [van Steenbergen, 1991] is a model-based knowledge acquisition tool for selection tasks. EXACT was designed as a tool for knowledge engineers to use in structuring knowledge through the design process. The tool proposed in the next section has many similarities to this tool.

4.5 Need for a Production Capable Tool

While the research on knowledge acquisition tool methodologies is abundant, and the number of tools and test applications that have been completed is large, knowledge acquisition tools have not become an important part of the development of KBS systems in American high technology companies. In an informal survey of eight leading U.S. manufacturing companies and three companies that specialize in creating KB systems, only one division of one manufacturing firm had

formalized KA software tools as part of the KBS development process [Bachrach and Keiser]. In addition, not one of the companies whose core business is KBS system development used KA tools. Some reasons mentioned for not using KA Tools are listed below:

1. Unfamiliarity: Several Companies were not familiar with existing KA tools and the potential applications of those tools.
2. Limited Applicability: One KBS development company felt that each KBS project was different and that no software tool would be applicable outside a very limited domain.
3. Immaturity: One company felt that KA tool technology was at present still to immature for use in a production environment.

The aim of this thesis is not to survey each KA tool and to evaluate it against dimensions such as range of applicability or product maturity. However, the author seeks simply to point out that the infrequent use of knowledge acquisition and other knowledge-based system development tools may suggest that previous tools have shortcomings.

4-6 Summary

Knowledge acquisition tools facilitate the elicitation and structuring of knowledge. The scope of the capabilities of KA tools varies widely. Interactive KA tools emphasize the user's role in generating the knowledge base, but attempt to guide the process in an efficient manner. These tools can be described by their structure (general or role-limiting), problem-solving strategy, knowledge acquisition strategy, and knowledge representation form. While many tools have been developed, few have gained acceptance or use in production environments. The next section presents the design of knowledge acquisition tool that aspires to implementation as a production tool. This tool does not write or interpret code, cases or text, but rather acts simply as a interactive knowledge structurer that decomposes assemblies into components and identifies linkages between

component parameters. While this tool has been used to develop a specific application, the selection of kit components, it is not domain specific.

5. Knowledge Acquisition through Matrix Organization (KATMO) Tool

This section describes the design of a prototype knowledge acquisition tool for selection tasks, KATMO. KATMO's design resulted from a KBS development project which executed the selection of powerfeed drill components used in aircraft assembly. While KATMO has not yet been developed into software, its methodology was particularly effective in its test environment.

KATMO structure is consistent with a *propose-and-revise* problem solving methodology in which it generates a list of required parameters for system elements and checks those parameters against a list of available selection items. If no match is found, alternative elements or parameters of elements are proposed, and the new values are checked against the list. KATMO's interviewing strategy is one of hierarchical decomposition. This process seeks to decompose assemblies into elements and then to parameters in order to capture complete knowledge about the parameter relationships in the system. While KATMO's initial design does not seek to encode knowledge, its intended representation structure is IF...THEN...ELSE... rules.

KATMO acts as a knowledge structurer to aid the KBS development team in the decomposition of kit components, the organization of element parameters and relationships, and the identification of rules relating parameters. KATMO is comprised of a three phase methodology:

1. Knowledge acquisition through interactive interviewing
2. Knowledge representation through matrix organization
3. Parameter isolation and rule identification

These phases are consistent with the knowledge acquisition steps in the KBS development model proposed in Section 3. This model is again shown in Figure 5-1.

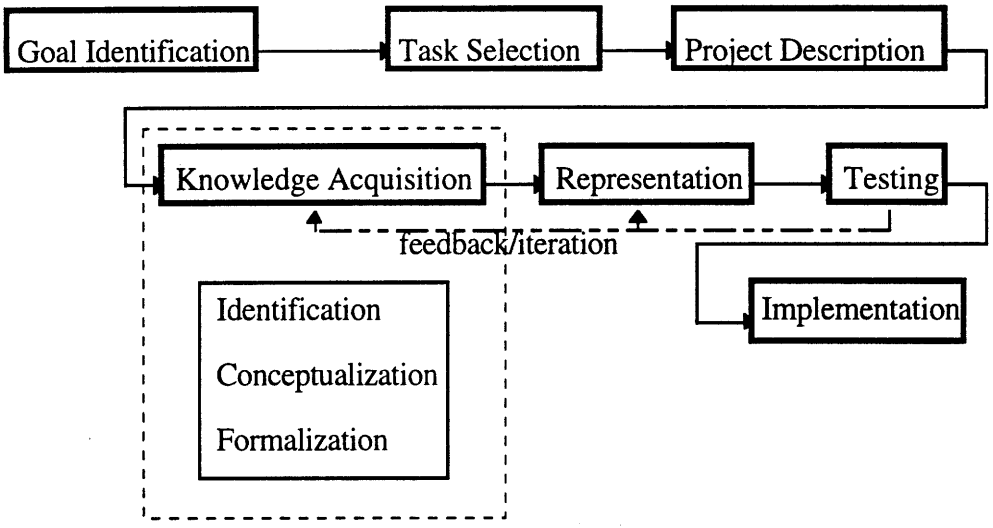


Figure 5-1 KATMO's Functionality in the Context of KBS Development

KATMO's intended use is the structuring of knowledge for selection tasks, where selection is between existing solid parts. KATMO resembles EXACT [van Steenbergen], another KA tool for selection tasks, along several dimensions:

1. Both tools are model-based—tools designed as models of a task. In this case, the task is selection.
2. Both tools employ a "mixed initiative" [van Steenbergen] dialogue protocol for eliciting knowledge—either the tool or the user can direct the interviewing process.
3. Both tools generate as an output a "knowledge document" describing the system, rather than executable code for a KBS.

The above similarities reflect the design of the tool—an aid for structuring knowledge about a system. While similar in structure, KATMO seeks to expand upon the capabilities of EXACT in certain aspects. Specifically:

1. KATMO focuses on the process of selecting kit components that comprise an assembly. This models adds the complexity of integration.
2. KATMO attempts to capture the sequential nature of design steps in the selection process. This step is organized in a matrix based on work by Steward and expanded upon by Eppinger and Eppinger et al.
3. KATMO recognizes that the existence of many rule exceptions may impede system automation. To reduce this barrier, KATMO identifies locations for user interfaces for "exception management."

The following sections describe KATMO's three phase design.

5.1 Knowledge Acquisition through Interactive Interviewing

Efficient development of KBS systems requires a complete understanding of the process for which rules are written. Because most design processes span across functional and departmental boundaries, specific process knowledge must be gathered from several sources. Assembling this information is often an arduous task as designers, engineers and manufacturing experts struggle with cultural, experience, and semantics differences in order to understand each other's concerns. In response to this challenge, KATMO's knowledge gathering approach focuses on simplicity. The knowledge gathering phase of the tool, Knowledge Acquisition through Interactive Interviewing (KATII), was designed for use in both Integrated Product Team type environments and one-on-one interviews. Under tool-directed interviewing, KATII questions the user(s) in a systematic manner designed to hierarchically decompose an assembly. Because only one level of the hierarchy is displayed at a time, users from different experience bases can begin to understand each others' inputs on a very simplistic level; because the questioning structure probes to the very bottom of the hierarchy, process complexity is fully captured. The tool also allows the user to direct questioning to facilitate knowledge model enhancement and editing. In a group setting, the

interview questions are intended to be projected before the audience (much like a Powerpoint presentation), in individual sessions, a computer screen will suffice.

The query structure of these questions is illustrated in Figure 5-2. This algorithm is more completely represented as a flow chart in Figure 5-3. The display format of these questions is shown in Figure 5-4. As noted above, the simplicity of these screens is a powerful tool for increasing cross-functional understanding and facilitating communication. In addition, the use of drawings of the components assembled in the kit will facilitate this knowledge acquisition process.

System Query	User Response
Identify a Kit Assembly:	
What Elements Comprise this Assembly?	
element1	
element2	
element3	
etc.	
What parameters define "element1?"	
parameter1	
parameter2	
etc.	
What influences "parameter1?"	
influencingFactor1/parameter	
influencingFactor2/parameter	
etc.	
Etc.	

Figure 5-2 KATMO's Interviewing Query Structure

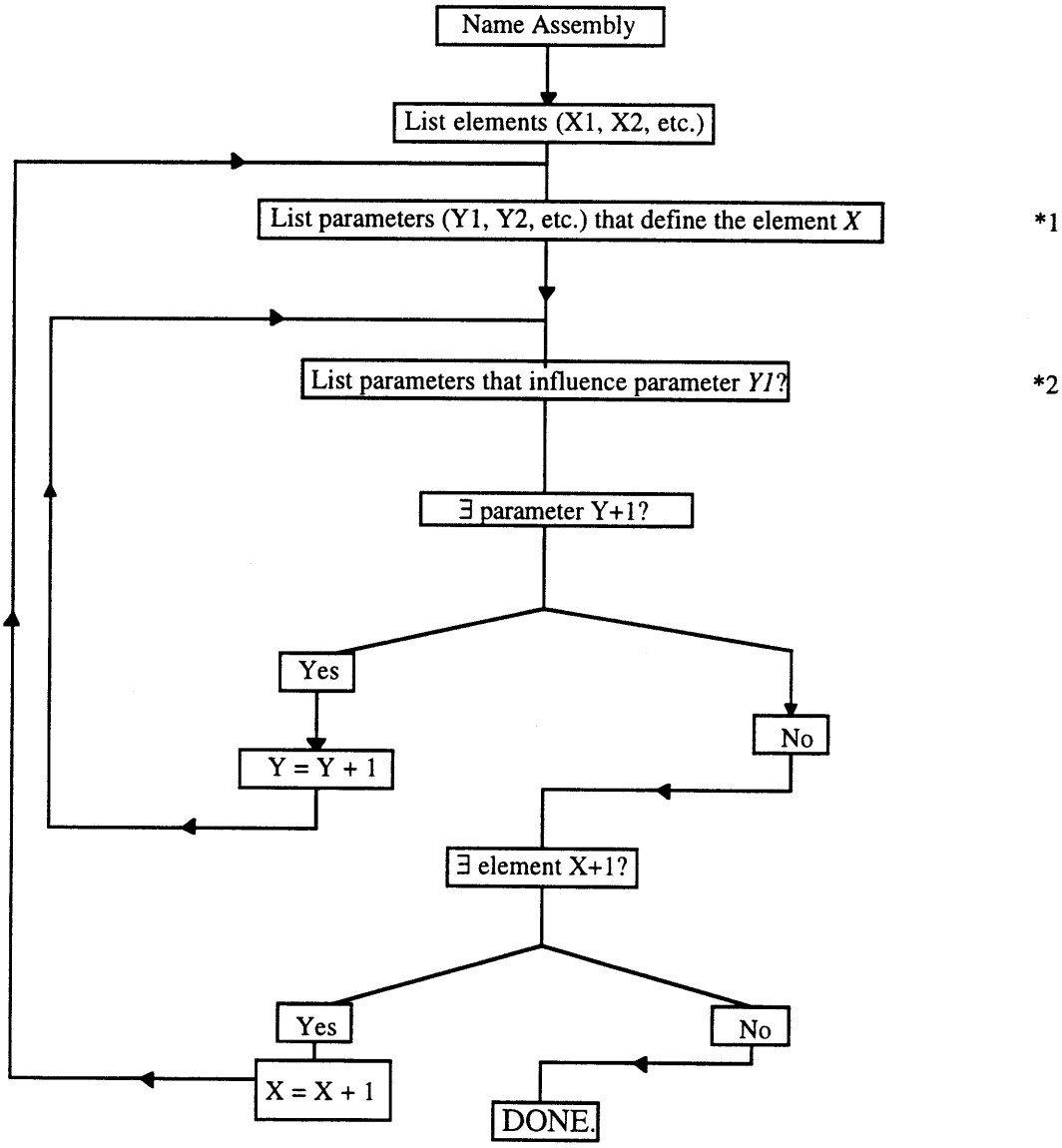
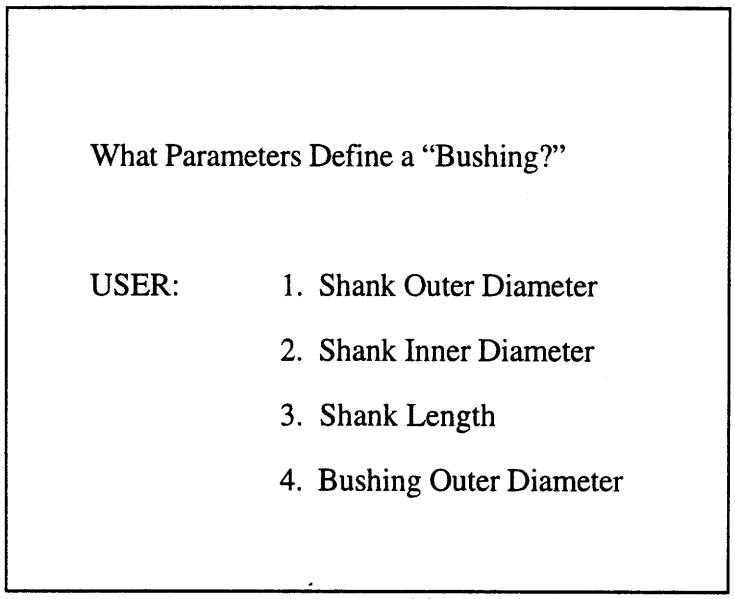
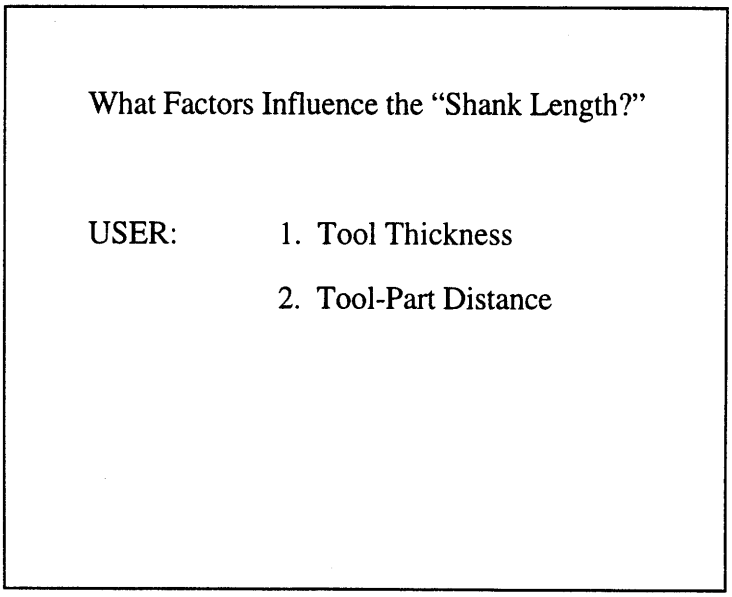


Figure 5-3 Interviewing Algorithm



(a)



(b)

Figure 5-4 Display Format of KATMO's Interview

As illustrated in Figure 5-2, in the selection process KATMO acquires three types of inputs: *elements*, *parameters* and *influencing factors*. These inputs and their organization in a matrix are discussed in the next section.

5.2 Knowledge Representation through Matrix Organization

Phase one of KATMO acquires system knowledge through interactive questioning of knowledge sources (domain experts). This assembled information represents participant's current knowledge of the system in its current state. While this information is certainly valid, as-is processes in complex organizations rarely reflect their most efficient structures [Gross]. The role of the second phase of the tool, Knowledge Structuring through Matrix Organization (KSTMO), is to organize system information in a matrix according to a sequencing algorithm. This algorithm is designed to minimize iterations by separating independent design parameters that can be evaluated separately, from dependent variables that form loops.

This organization is achieved through the application of a Design Structure Matrix (DSM). This representation offers an advantage over other formats (such as a PERT chart and SADT documents) for system analysis by explicitly displaying information loops [Eppinger and Gebala]. A blank DSM is shown in Figure 5-5.

In KATMO, the information obtained in the knowledge acquisition phase is entered in a matrix for organization. However, in order for this matrix to be correctly built, the data inputs obtained in the first phase must be classified. Classification is important because much irrelevant data may be obtained in the knowledge acquisition phase of the tool. In the interviewing process, KATMO instructs the developers to name all parameters that "define" an element. This command is structured in this manner in order to elicit complete information about an element so that users do not overlook parameters critical to the selection process (and because it is very difficult to intuitively know which parameters are important for the selection process and which are not). However, only a subset of total parameters listed are important for the selection process—users will likely list influencing factors and parameters that they believe to be important to the system but in fact are not. KATMO classifies data in order to identify the which data is relevant and to discard

unnecessary data from the system. The three types of data—elements, parameters and influencing factors—are defined below.

5.2.1 Elements

In kit component selection, the *elements* are the items which are to be selected, i.e. they are the components which are assembled to form the kit. The interviewing structure is designed to decompose elements further to the parameter level in order to establish relationships between parameters. While it is technically incorrect to relate elements directly to other elements (they are really related on the parameter level), if an element is always associated with another element, this may be the most efficient representation. This statement is clarified in the following example.

In automobile final assembly, the door of the car is assembled to the body. These elements are related most explicitly on the parameter level: the dimensions of the opening in the car frame and the diameter of the door hinge hole relate to the dimensions of the door and the diameter of the door interface. However, if door x is always assembled to car frame y, it may be most efficient to relate the elements directly.

5.2.2 Parameters

Parameters are attributes of an element that are necessary to completely describe that element. For example, the parameters of an office trashbin include its dimensions (including the angles between surfaces) its material, its surface finish, its color, etc. Without knowing the name of this object, such a description would enable its complete definition. For selection tasks, some parameters are important for selecting an element and some are not. I define three types of parameters:

Type-one parameters (t1): Those parameters necessary for selecting an element for its intended purpose. In other literature, these parameters have also been called *constraints*.

Type-two parameters (t2): Those parameters associated with a selected element, that affect the selection of another element.

Type-three parameters (t3): Those parameters that are necessary for completely describing an element, but have no consequence on the selection process. For example, in a machining environment, the color of tool may have no bearing on the selection of that tool, but it may still necessary to for a complete description.

A parameter may be simultaneously type-one and type-two.

5.2.3 Influencing Factors

Influencing factors (if's) are those that influence the selection process, but are not parameters of any element to be selected. Influencing factors are external influences on the system.

In order for KATMO's matrix to be built, the data types described above must be classified. The identification of elements is easy: these are entered explicitly by the user. The differentiation between parameter types (one, two and three) and influencing factors is more subtle. In KATMO's interviewing structure, elements are defined by parameters (see Figure 5-3, *1), and then the user is queried for factors that influence those parameters (see Figure 5-3, *2): "List parameters that influence parameter *Y*." If parameter *Y* is influenced by another parameter, parameter *Y* is necessary for the selection of the associated element and is a type-one parameter. If parameter Y is not influenced by another parameter, it is not relevant for the selection of the associated element but may still be important to the system. The remaining classification rules are described below.

1. Parameter Y is an influencing factor if:
    i. Parameter *Y* is not influenced by another parameter; and
    ii. Parameter *Y* is not associated with an element; and

iii. Parameter Y does influence another parameter in the system.

2. Parameter *Y* is a type-two parameter if:

   i. Parameter *Y* is not influenced by another parameter; and

   ii. Parameter *Y* is associated with an element; and

   iii. Parameter *Y* does influence another parameter in the system.

3. Parameter Y is a type-three parameter if:

   i. Parameter *Y* is not influenced by another parameter; and

   ii. Parameter *Y* is associated with an element; and

   iii. Parameter *Y* does not influence another parameter in the system.

In KATMO, these classifications are used for the correct mapping of element-parameter relationships in the matrix: these classifications determine the position of entries in the matrix. Type-one parameters are selection criteria for an element: the row of an element receives an input from each type-one parameter associated with that element. Type-two and type-three parameters are determined only after the selection of the element. The rows of type-two and type-three parameters receive an input from the associated element. Type-three parameters are distinguished from type-two parameters in that the columns of type-three parameters are blank. (This signifies that they influence no other data in the matrix). Type-two parameters have at least one entry in their columns (signifying that they do influence other data in the matrix). These classifications in a matrix are illustrated in 6.5.

Once all elements, parameters and influencing factors have been classified, a matrix mapping their relationships can be built. In this matrix, system data are listed in the same order along both the horizontal and vertical axis of the matrix. A solid red block within the matrix indicates that a specific parameter or influencing factor in the corresponding column is influenced by the parameter

in the corresponding row. A blank or open template matrix is shown in Figure 5-5; a sample matrix of unorganized information is shown in Figure 5-6. In this matrix, element 4 is influenced by elements 2 and 6. The diagonals are blocked out because of the implied redundancy. In the remainder of this section, the word *element* is used in matrices and discussions as a generic term to illustrate the matrix organization process. The precise classifications of matrix data is again described in the discussion of the application of KATMO in 6.5

		element 1	element 2	element 3	element 4	element 5	element 6	element 7
		e1	e2	e3	e4	e5	e6	e7
element 1	e1	■						
element 2	e2		■					
element 3	e3			■				
element 4	e4				■			
element 5	e5					■		
element 6	e6						■	
element 7	e7							■

Figure 5-5 Knowledge Matrix: Blank Template

		element 1	element 2	element 3	element 4	element 5	element 6	element 7
		e1	e2	e3	e4	e5	e6	e7
element 1	e1							
element 2	e2							
element 3	e3							
element 4	e4							
element 5	e5							
element 6	e6							
element 7	e7							

Figure 5-6 Knowledge Matrix (Unorganized)

Once the Design Structure Matrix has been built, the matrix is organized. The organization criterion defined for this application is the minimization of iteration steps within feedback loops. This process therefore separates the elements into those that can be solved sequentially, and those that cannot.

		element 1	element 3	element 7	element 2	element 5	element 6	element 4
		e1	e3	e7	e2	e5	e6	e4
element 1	e1							
element 3	e3							
element 7	e7							
element 2	e2							
element 5	e5							
element 6	e6							
element 4	e4							

Figure 5-7 Knowledge Matrix (Organized)

In Figure 5-7, the first five elements (elements1, 3, 7, 2 and 5) form a lower-triangular matrix. This indicates that all information is feed-forward without the presence of a feedback loop. I have termed such elements as *non-coupled* to signify that they do not require iterations to solve. Conversely, elements 6 and 4 are *coupled*: element 4 requires information from element 6 and visa-versa. (I.e. the loops lack causality, whereas the non-loops are causal.) In order to solve these relationships a means for breaking the loop must be defined.

5.3 Parameter Isolation and Rule Identification

Once the sequence of the selection process has been organized, design or assembly rules can be written for many of the interrelationships between process characteristics and influence factors. As defined above, the relationships between element parameters that are identified in the knowledge organization stage are primarily of two forms: coupled or non-coupled.

In order to develop these rules for both coupled and non-coupled cases, information is extracted from the DSM, which describes the interrelationships of all elements in the process, and represented as a simplified version of the Structured Analysis and Design Tool (SADT). This modified SADT format shows only immediate predecessors influencing each element and serves to isolate relationships. An example of such an extraction is given below.

Figure 5-8 Sample Matrix Extraction (Non-coupled elements from Figure 5-7)

While the process of extracting information from the matrix does not alter any of the relationships captured in the matrix, the isolation of elements serves to simplify the presentation for the construction of rules [Lockheed].

Figure 5-9 represents the coupled parameters in isolation. In the case of coupled parameters, no sequencing may be performed until the circular loop of interrelationships has been broken. I propose three means to achieve this: standard procedures, selection policies and user-determined selection. Additional information of treating feedback loops is given in Marcus.

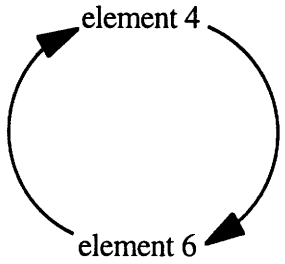


Figure 5-9 Sample Matrix Extraction (Coupled elements from Figure 5-7)

5.3.1 Standard Procedures

When a circular loop between relationships is identified, it has not been fabricated, but was a part of the existing design process. Accordingly, there currently exists a method—though the method may be ad-hoc—for managing the loop. In order to break the loop, the KBS developers may decide to incorporate existing standard procedures into the rule set. In the case of Figure 5-9, standard procedures could dictate that element 4 is always chosen first, thus breaking the loop.

5.3.2 Selection Policies

Companies often seek to regulate the processes by which activities in the organization are performed. For example, a company may have a policy of incorporating only standard fixtures sizes into its assembly operations. In the same manner, a circular loop may be broken according to

a chosen policy that may be different from the current standard procedures. For example, in Figure 5-9, management might decide that permitting element 4 to dictate element 6 has resulted in excessive variability in design. As a result, a decision could be made to stipulate that while element 4 will still be chosen first, only a subset of all element 4's, may be considered as a design choice. One plausible example of this is the decision to permit only standard fastener sizes to be used in designs.

5.3.3 User-Determined Selection

In many cases, it may be impossible to use standard operating procedures or design policies to break coupled relationships; this may be a result of the inherent complexity of the system. In this case, KATMO follows the methodology described in the KA tool SALT [Marcus].

When SALT encounters a loop, it notifies the user of the situation, identifies the parameters in the loop, and allows the user to specify the values of one of the parameters so the system can continue solving. This process is illustrated for the loop in Figure 5-9. Note that the information pieces presented to the user are those which influence the elements in the loop (see Figure 5-7 for verification), additional information could be provided if desired.

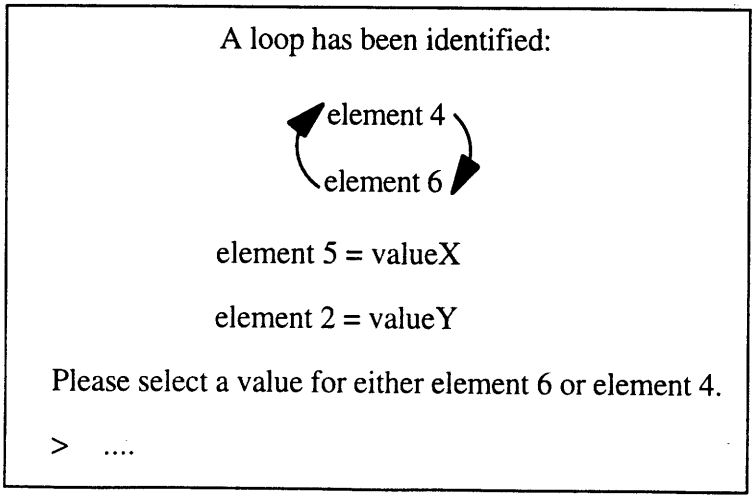


Figure 5-10 Loop Breaking via User Selection

This methodology for breaking loops discussed can be expanded to apply for deterministic relationships as well. I have termed this general concept, *complexity management.*

5.4 Complexity Management

Even if relationships are deterministic, that does not necessarily imply that rules can be written to accurately describe the process. If rules are high-order (defined below), or if there exist many exceptions to the rules, building the necessary knowledge base may be impossible or not economically viable. These two cases are described below.

5.4.1 High-Order Rules

The way in which elements are chosen in a selection process may be very case sensitive. Certain components of a system may be appropriate only if three or four (or more) circumstances are present. I propose one measure of the complexity of any given rule to be the number of statements included in that rule necessary to specify the desired outcome. I refer to this measure as the *order* of the rule. Thus, an *n*-order rule, requires *n* statements to specify the conditions of selection.

The following is a plausible example of a 4th-order rule for selecting a drill bit:

IF (a material = Titanium) AND
IF (the hole diameter < 0.500 inches) AND
IF (the hole depth > 1.25 inches) AND
IF (use of coolant? = TRUE)
THEN (drill bit = DBIT152)

Rather than attempting to construct high-order rules (I have chosen an order of 5 or greater to be considered a "high-order" rule) to specify the selection of *every* possible element in the system, KATMO permits system developers to identify points of user intervention to break loops. In this

scenario, as discussed in 5.3.3, the KBS would present the user with the relevant pieces of information and allow the user to decide. For example, in Figure 5-7, if the relationship between elements 3, 7, and 2 were very complex, rather than developing rules, the values of elements 3 and 7 (which influence element 2) could be presented to the user and the user could determine element 2.

5.4.2 Rule Exceptions

Rules in spoken languages, in engineering, and in systems in general often have exceptions. Rule exceptions pose problems for KB systems because they can usually only be incorporated by specifying high-order rules. If a rule has many exceptions, the rule-writing process may degenerate into hard-coding and/or the knowledge base may become difficult to manage. As an alternative to writing high-order rules for rule exceptions, I again propose to apply the notion of user interaction to the system. By presenting the user with the information pieces needed to make a selection, the system obviates the need for complex and tedious rule-writing while leveraging user knowledge.

5.5 Summary

KATMO is a knowledge acquisition tool that involves three phases: knowledge acquisition, knowledge organization, and parameter isolation for rule development. The power of this tool is derived from the simplicity of the user interface and the design structure matrix used to organize interrelationships. Sections 5.2.1-5.2.3 describe the data types included in the matrix. These specifications are critical to correctly identifying parameter-element relationships. Sections 5-3 and 5-4 describe rule writing and the management of rule complexity. These are important features of KATMO. Rules become cumbersome and difficult to capture if they require high-order statements. Many automated applications have failed in the past because of their attempts to capture all knowledge within their knowledge bases, without providing the user effective means of interacting with the KBS. The failure of such systems undermines the credibility of automated design

systems. The design structure matrix of KATMO facilitates the identification of potential points of user interaction. This allows KBS developers and users to efficiently manage rule exceptions and preserve a certain level of "engineering judgment" within the system. The advantage of this feature is that it enables users to selectively automate portions of a complex network. In this manner, the capabilities and applications of the KBS can be greatly expanded.

By recognizing through its design that KA tools are but one tool that may facilitate the KA process, rather than a panacea to the KA bottleneck, KATMO aspires to implementation in many user environments. The following section presents the use of KATMO's methodology in the development of rules for powerfeed drill equipment.

6. An Application of KATMO: Powerfeed Drill Component Selection

This section presents an application of the KATMO development tool to the selection of kit components for powerfeed drill equipment. This project resulted in the creation of a prototype knowledge-based system that is currently being expanded to a production system at a large aerospace company. Before exploring the use of KATMO, it is important to explain the selection process and context in which KATMO was employed.

6.1 Introduction to Powerfeed Drill Equipment: Applications and Context

While the aerospace industry inspires romantic sentiments from both enthusiasts and engineers alike, much of a plane's design integrity depends on the less glamorous field of holes and fasteners. Accurate holes are critical to good airplane design for several reasons:

1. Variations in the size and placement of holes on one part affect the placement of adjacent parts. Depending on the nature of the error, the fatigue life of the part may be adversely affected.
2. Variations render expensive tooling jigs (costing as much as $2 Million each) useless, as the assembly no longer will fit.
3. Variations result in panels that do not match closely. Outer panels that are not smooth are associated with negative aerodynamic effects and may deteriorate from advanced capabilities such as stealth.
4. Variations in parts increase the assembly's susceptibility to stress fractures or failure.

These considerations have resulted in the development of a serious "science of drilling holes," in which holes with tolerances as small as 0.0050 inches are required on dimensions such as the center-center hole alignment. To meet these demands, several companies supply complex and expensive powerfeed drills and components. Powerfeed drill equipment is "positive feed" meaning that once the feed mechanism is engaged, the drill bit will extend a predetermined amount (typically 0.001-0.002 per revolution of the drill). The feed mechanism is mechanically coupled via gear reduction to the primary motor. Powerfeed drills require additional components such as a bushing to "lock" the drill in place to support reactionary loads from the drill. This enables the powerfeed equipment to provide a consistent drilling force resulting in high quality holes. In contrast, standard drilling equipment requires that the operator push against the material being drilled; this is very tiresome, nearly impossible with some materials, and deters from the accuracy of the hole [Seidel]. Assembled powerfeed drill kits typically cost between $3,000 and $5,000.

Powerfeed drill set-ups, or kits, can comprise many elements. These typically include: a motor, a nose piece, a bushing collar, a chuck, a cutting tool, and a bushing. These components are shown in a standard assembly in Figure 6-1.

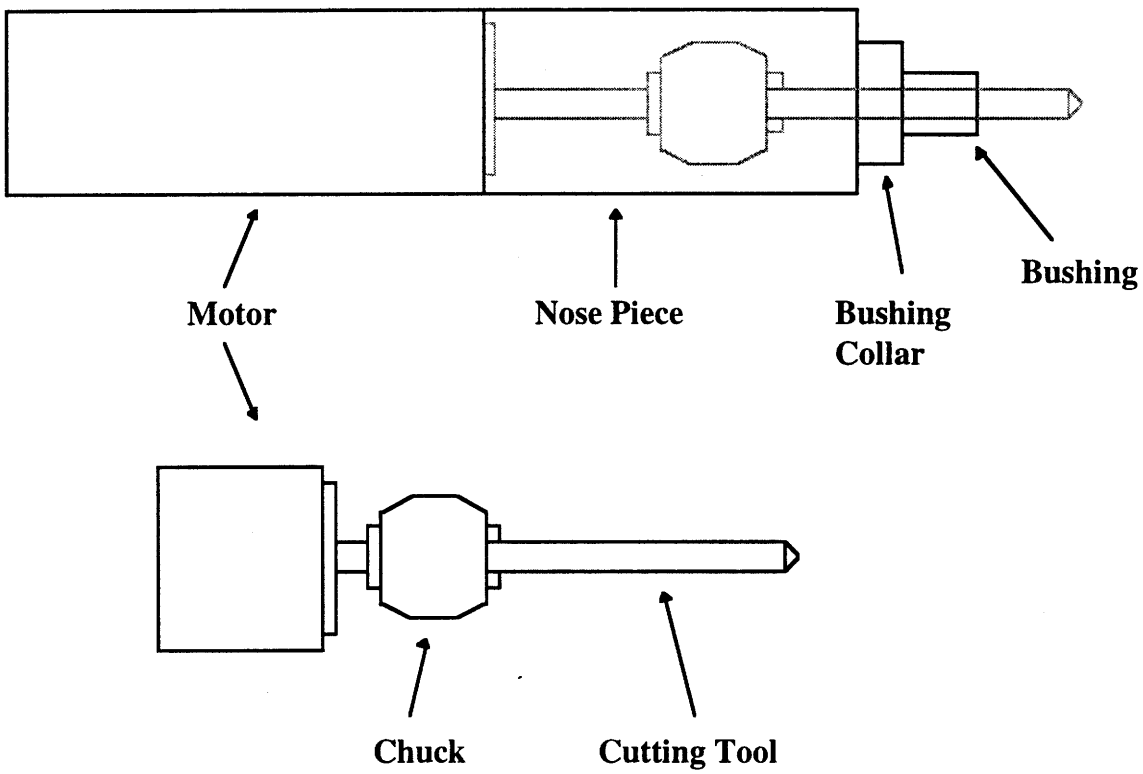


Figure 6-1 Powerfeed Drill Assembly

The powerfeed drill kit assembly is inserted into a tool fixture, or jig, which lies on top of the part to be drilled. This tool jig secures the kit to facilitate secure and accurate drilling. The operating environment of the powerfeed kit is shown in Figure 6-2.

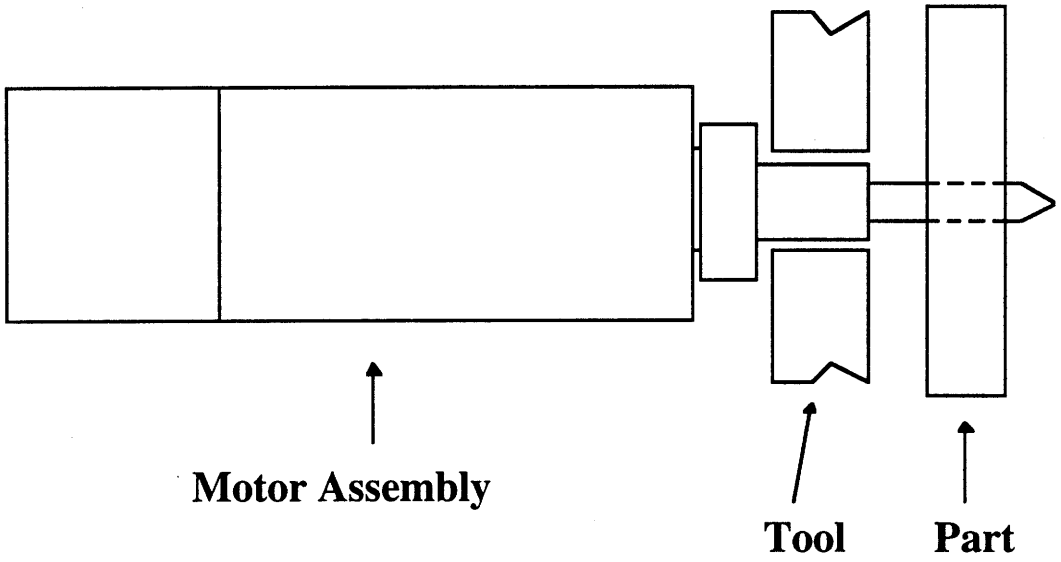


Figure 6-2 Powerfeed Drill Kit Operating Environment

6-2 Powerfeed Drill Equipment and the Product Development Process

The product development process in the aerospace industry can be thought of, at a high level, as sequential but over-lapping phases that include: identifying product requirements, conceptual design, detailed design, tool design and assembly operations design. In the context of the specification of powerfeed drill equipment, once a part is designed, a tool is built to assist in the manufacturing and assembly of the part, and then based on characteristics of both the part and tool design, a drill kit is specified (see Figure 6-3). Because of the precision at which holes must be drilled, drill equipment must be matched exactly to the drilling requirements.

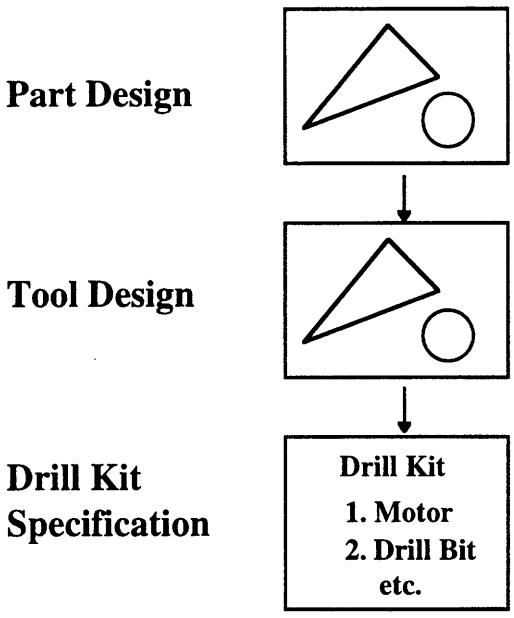


Figure 6-3 Powerfeed Drill Kit Specification Process

6.3 Case History

This project began as an effort to increase the efficiency and quality of the delivery of powerfeed drill kits. In the year prior to the development effort, incorrectly assembled kits and procurement delays associated with kit components resulted in the near shut-down of a main assembly line at an aerospace company. The specification of powerfeed drill kit components was judged well suited for enhancement through automated design because there exist heuristics for component selection. However, there were considerable technical and organizational challenges to the development of the system. In particular, the amount of data on kit components was sizable and its analysis complicated: there were thousands of unique components. The tremendous amount of kit component inventory resulted from an unregulated design process in which part designers specified any configuration of hole sizes and tool designers built more kits components to match each new design. In this process, part designs were passed to tool designers, who in turn passed their designs to shop workers who assembled the drill kits. This sequential process did not take

into account the availability of kit components or alternative combinations of kit components that could be used to fulfill the same tooling requirement.

Because the end product was a discrete assembly of known components, the process of decomposing the assembly and mapping the interrelations of components seemed a logical development path. In searches of academic literature concerning selection tasks, this approach was confirmed to be viable. Through the process of knowledge acquisition, the importance of representation tools for capturing that knowledge became clear. After the knowledge had been assembled in several forms (SADT charts, spreadsheets, input-output statements), a matrix was selected for knowledge organization because of its explicit capture of feedforward information and feedback loops. The individual pieces of knowledge about components in the system that were acquired through the development of the KBS were not new to the organization. However, the *assembly* of that knowledge in the resulting knowledge document enabled the organization to better understand the process by which powerfeed kits were developed. In addition, through the analysis of the assembled information, methods for selecting elements were determined that were previously unknown.

The prototype KBS demonstrated the potential quality and efficiency of the process. The benefits of this system are many: (1) By using "expert knowledge" to assemble kits, components are assembled correctly the first time, eliminating work-arounds; (2) Because kit component specification is automatic, kits can be quickly reconfigured to meet Engineering Change Notifications; and (3) Because kit component specification is electronic, statistical data on kits can be captured quickly and efficiently—this enables the reuse of a smaller set of kit components for many design configurations, which in turn enables inventory reduction. In a preliminary analysis of specified kit component data, powerfeed equipment inventory could be reduced by an estimated 40%.

The KBS development process with KATMO will be explained in detail in the three following sections.

6.4 KATMO in Interactive Interviewing

The first phase in applying KATMO is acquiring knowledge through interactive interviewing. As described in Section 5-1, KATMO's interviewing tool can be either algorithm-led, or user-led. The following example will be described as if the system's algorithm was directing the elicitation of knowledge.

KATMO's interactive interviewing process is top-down in structure and begins with requesting the name of the assembly. Consequently, the tool seeks to decompose the assembly interactively. This is done by first, understanding the elements that comprise the assembly, and second, by eliciting parameters of those elements. KATMO's interviewing structure proceeds until no further decomposition is desired. Then, data types are classified and fed into a design structure matrix for organization. Through this decomposition, the tool seeks to obtain "complete" information about elemental and parametric interrelations. If an unidentified parameter is entered as an influencing factor, KATMO inquires for further information about the new parameter and that parameter is entered into the system.

Eliciting all of the parameters of an element is a difficult task. Many parameters are not readily identified or are thought to be of no significance by the developers. I recommend two approaches for collecting information:

1. Define as parameters all characteristics necessary to fully describe the object without a name.
2. Analyze a database or ordering brochure for the element. Identify all the parameters necessary to distinguish one element from another.

The above techniques were employed in the development of the prototype KBS discussed in this section. The interviewing process was executed in both team and one-on-one engagements. Figure 6-4 presents the results of this interviewing. An abridged schematic is shown for presentation.

System Query	User Response
Identify a Kit Assembly:	Powerfeed Drill Kit
What Elements Comprise this Assembly?	
element1	Motor
element2	Nose Piece
element3	Bushing
etc.	
What parameters define a "Motor?"	
parameter1	Capacity
parameter2	Max Stroke
etc.	
What influences "Motor Speed?"	
influencingFactor1/parameter	Part Material
influencingFactor2/parameter	Hole Diameter
etc.	
Etc.	

Figure 6-4 Interactive Knowledge Acquisition Structure (Abridged)

6.5 KATMO in Knowledge Organization

Once information has been collected, it is classified automatically according to the algorithms described in 5.2. Then, this information is entered in a matrix for organization. While many tools address the issue of knowledge organization well, KATMO seeks to incorporate sequencing information into its rule building structure. Accordingly, KATMO employs a Design Structure Matrix to illustrate element interrelations and organize information according to sequencing algorithms. This process is similar to that performed by DeMAID [Rogers], a software program that optimizes matrix structures according to a specified algorithm.

Through KATMO's Knowledge Organization phase, information collected through interactive interviewing is displayed in matrix form with element names, parameters and influencing factors on the vertical and horizontal axes. A red square denotes a relationship. For presentation, an abridged version of the matrix developed for powerfeed drill equipment is shown. Complete matrices are given in Appendix A. Figure 6-5 shows that matrix in its "unorganized" form.

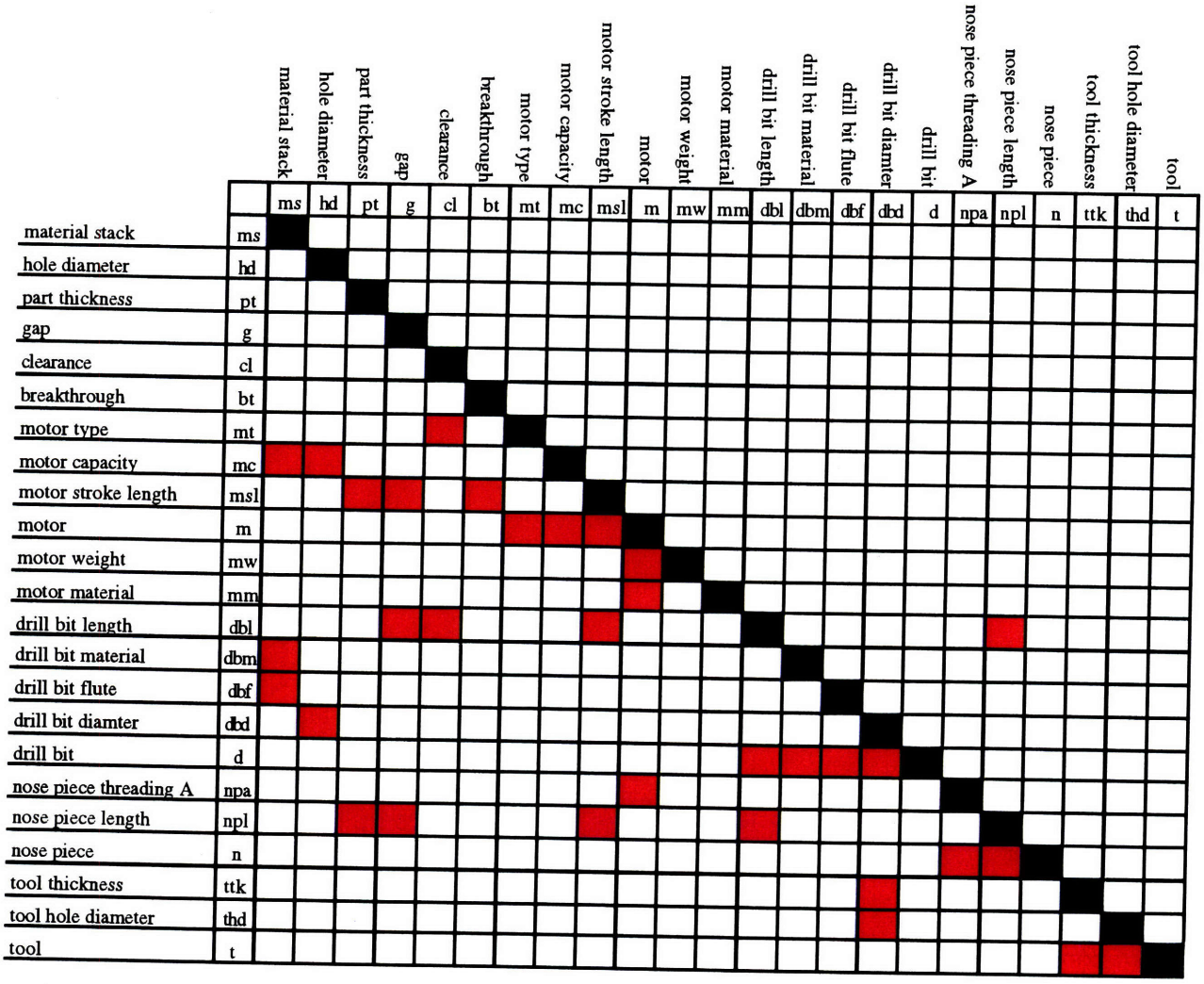


Figure 6-5 KATMO's Design Structure Matrix for Knowledge Acquired through Interactive Interviewing (Unorganized)

An analysis of the above matrix yields information about the data types in the system. *Material stack*, *hole diameter*, *part thickness*, *gap*, *clearance* and *breakthrough* are Influencing Factors: they are not a characteristic of any element in the system (their rows are blank), but they do influence the selection of elements.

In addition, all three parameter types are present. Five parameters are associated with the motor: *motor type*, *motor capacity*, *motor stroke length*, *motor material* and *motor weight*. The first three

parameters (motor type, motor capacity and motor stroke length) are the selection criteria for the motor (type-one parameters). *Motor stroke length* is both a type-one and type-two parameter: it is both necessary for selecting the motor, and it affects two other parameters in the system. *Motor weight* and *motor material* are type-three parameters: they are necessary for fully describing a motor, but are superfluous for the selection of elements—this is clear in the matrix as no red squares appear in the columns of these parameters. As type-three parameters have no bearing on the selection process, they should be excluded from the matrix. A revised matrix is presented in Figure 6-6.

In practice, it is not necessary for the user of KATMO to develop such a detailed knowledge of the data types in the system—the classification is made automatically by KATMO and can be determined from the matrix as illustrated above. However, in organizing a selection process—with or without KATMO—it is critical to determine which system data are important for selection and which are irrelevant. As previously mentioned, the classification of data reflects the relationships underlying the selection process—in KATMO these relationships are captured by the position of influence entries (red squares) in the matrix. Incorrect classification reflects incorrect information about which parameters are critical for selection of an element. As the relationships between parameters and elements are used to write rules, the quality of the knowledge represented in the developed KBS system will be negatively affected.

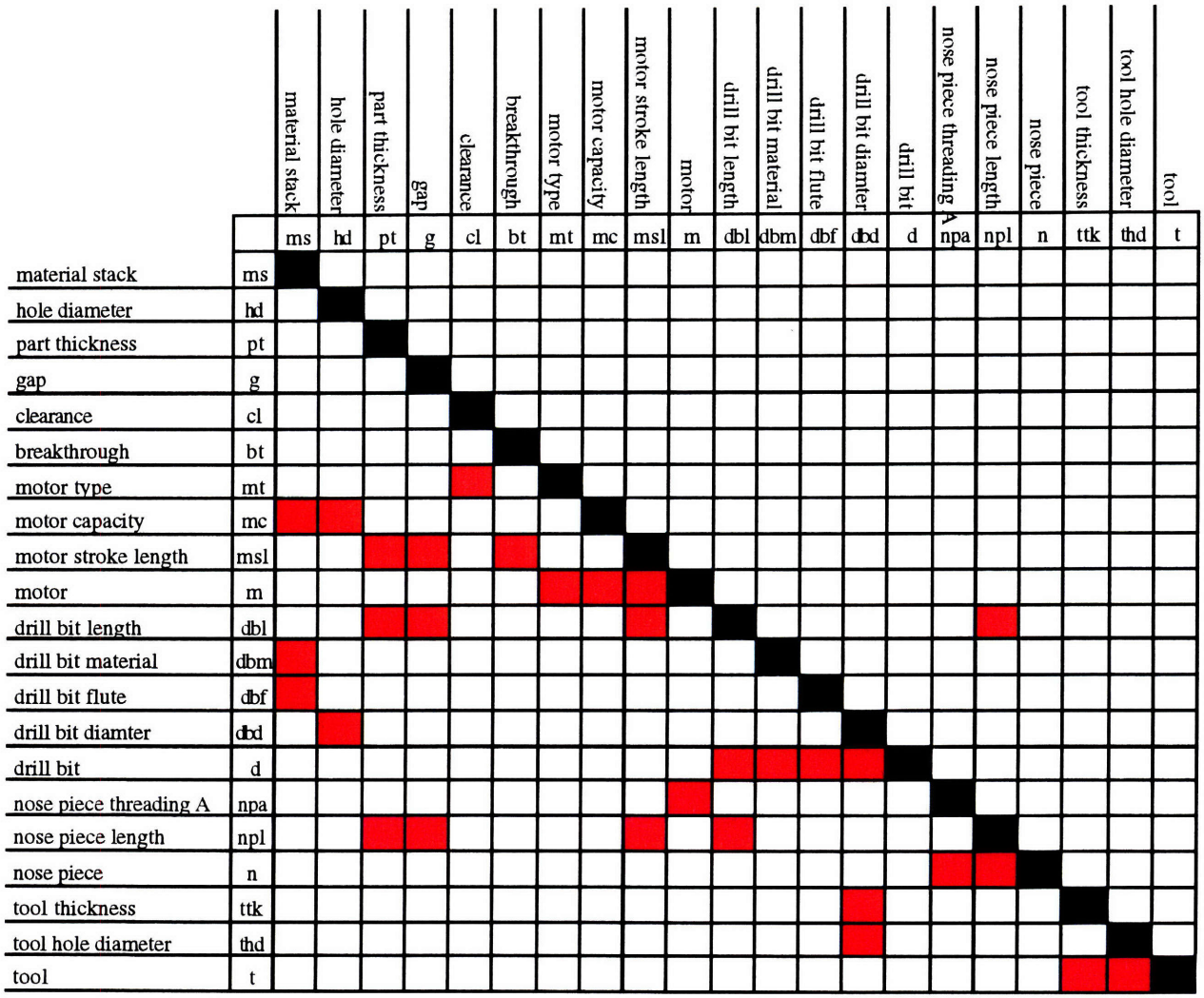


Figure 6-6 Relevant Data Pieces of KATMO's Design Structure Matrix (Unorganized)

Once knowledge has been displayed in a matrix form, it can be organized according to any number of algorithms: in KATMO the algorithm minimizes feedback loops. The organization of the matrix in Figure 6-6 is shown in Figure 6-7.

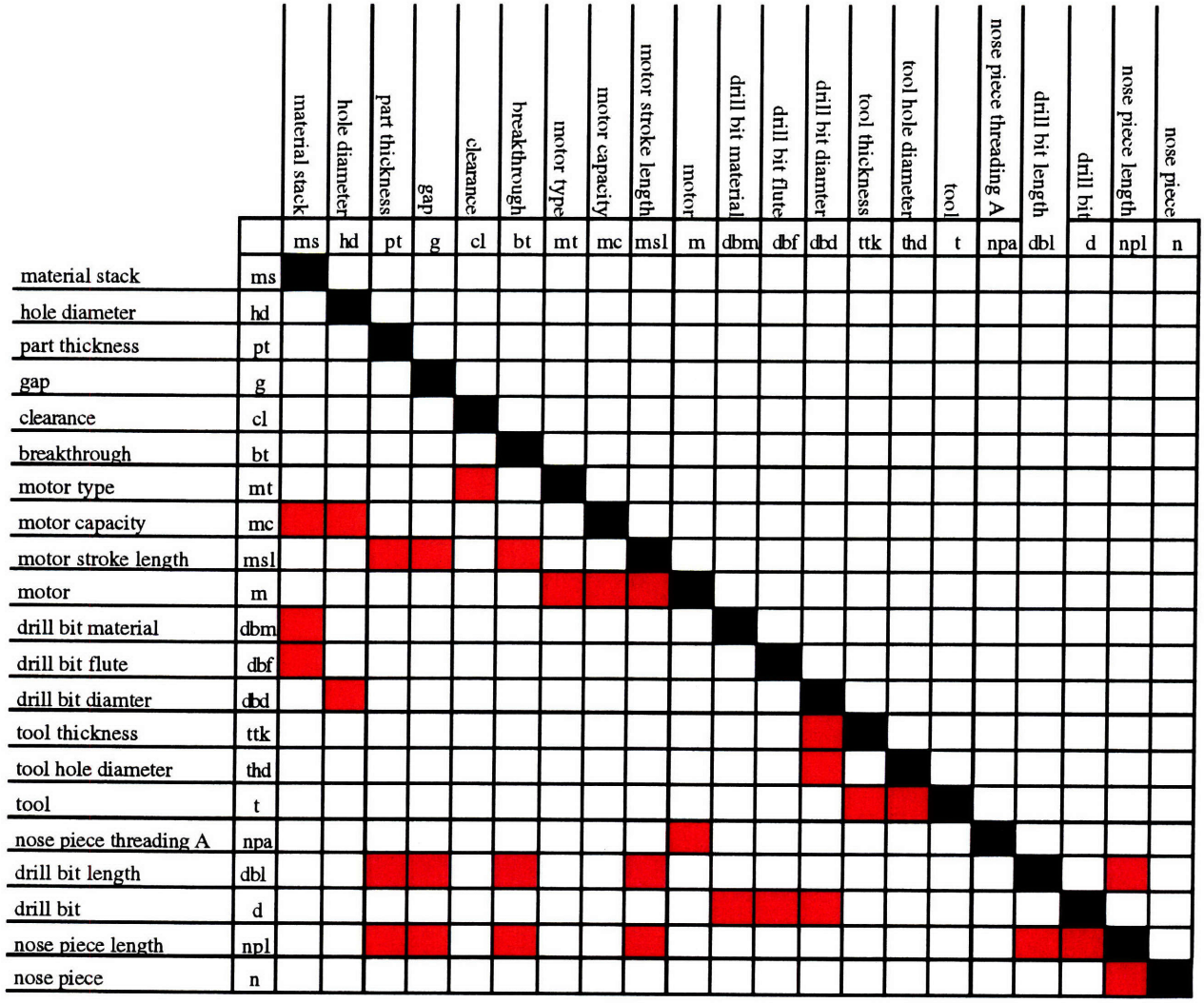


Figure 6-7 KATMO's Design Structure Matrix (Organized)

As discussed in Section 5, organized information is either coupled, indicating a circular loop, or non-coupled, indicating that given the available information, the relationship is deterministic. Figure 6-8 highlights the coupled regions in the matrix, all other relationships are non-coupled.

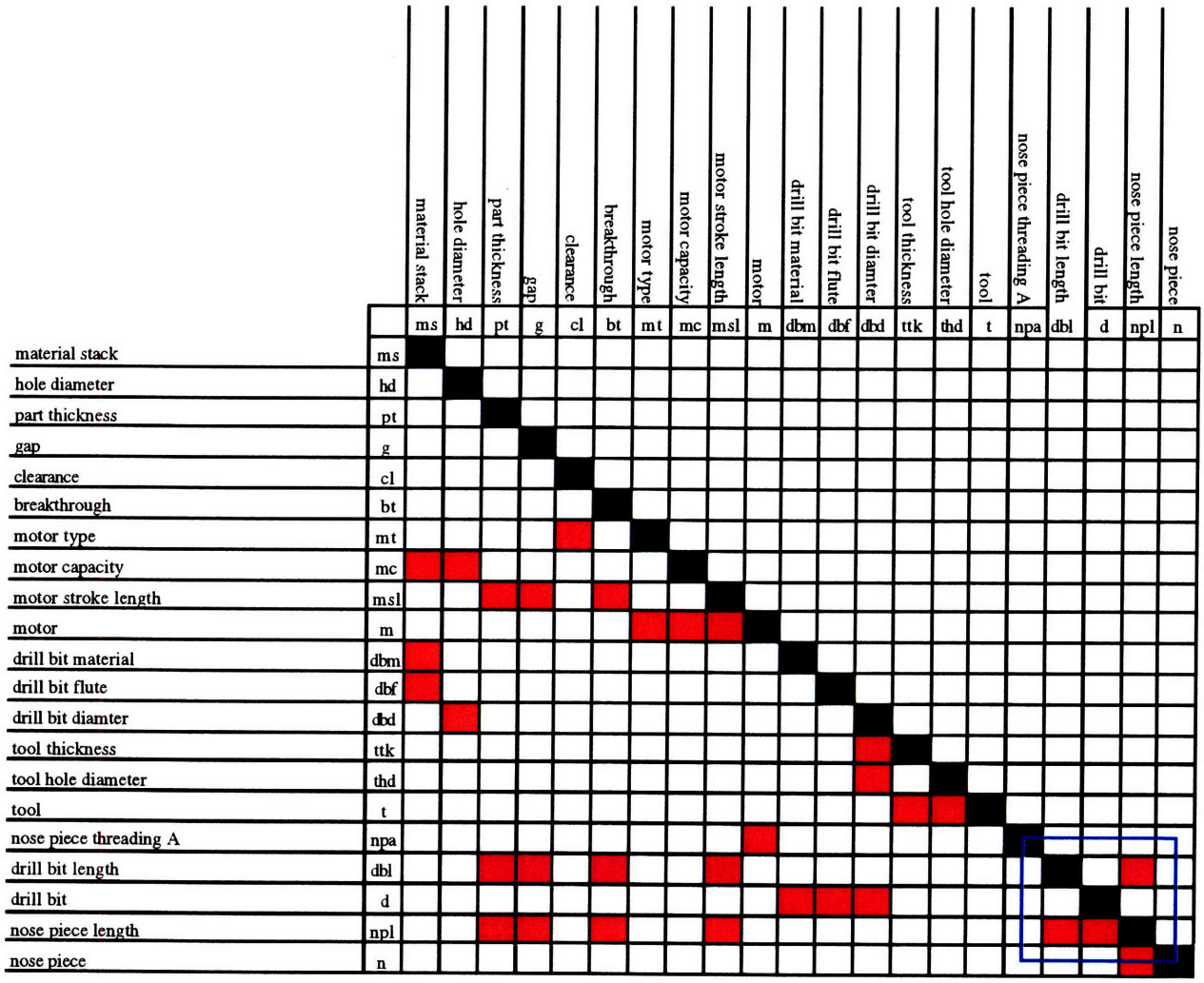


Figure 6-8 Coupled Region of KATMO's DSM

(indicated by the blue rectangle)

As illustrated above, most elements were non-coupled. This indicates that given the required information, the relationship between parameters is deterministic. In contrast, there is a coupled relationship between drill bit length and nose piece length. This indicates that both elements influence each other simultaneously.

6.6 Parameter Isolation and Rule Identification

The final step in KATMO's methodology is to isolate parameter relationships for the identification of rules. It must be stressed that interrelationships between elements do not link elements directly, but rather, link parameter(s) associated with that element. These parameter and element relationships are captured in the interviewing process.

While the process of isolating parameters does not alter any of the information presented in the KBS, its presentation as a linkage of two or three properties, as opposed to network of every system property (which is what the complete matrix reflects), served to simplify the specification of rules. Parameter isolation will be discussed in two sections, for coupled and non-coupled elements, respectively.

6.6.1 Parameter Isolation for Non-coupled Elements

Parameters that are non-coupled can be specified without iteration. Figure 6-9 illustrates the isolation of three non-coupled parameters extracted from the matrix in Figure 6-8.

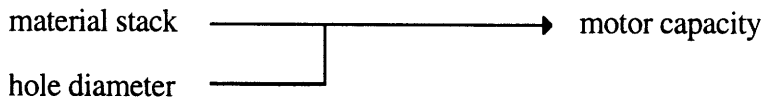


Figure 6-9 Parameter Isolation

This representation shows that both the material stack and the hole diameter influence the motor capacity.

Once parameters have been isolated, rules between elements are identified. Figure 6-10 shows a sample rule for the above relationship. (In this example, motor capacity was measured in terms of maximum hole diameter size per material).

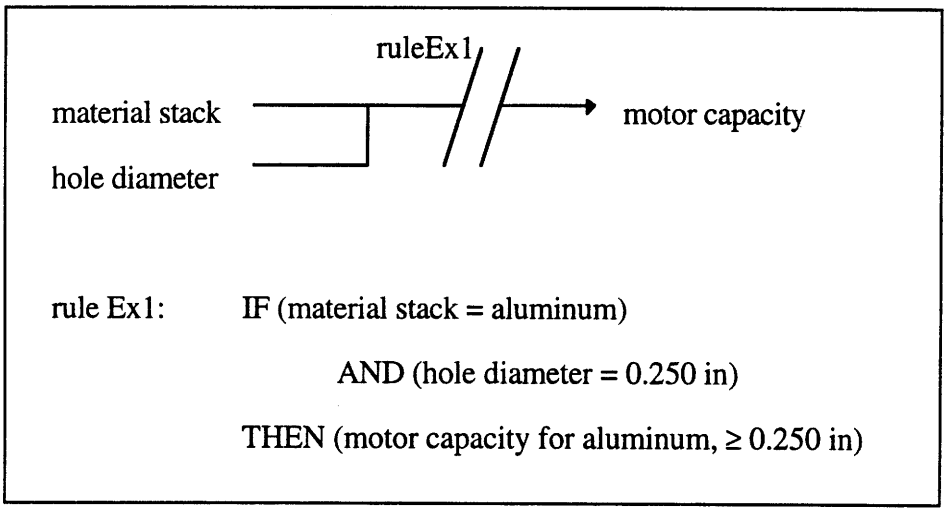
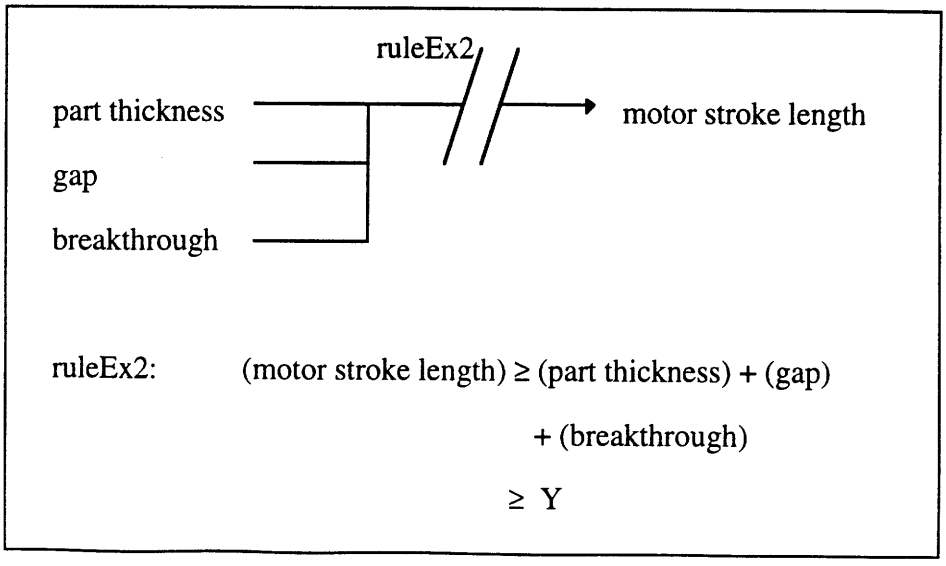
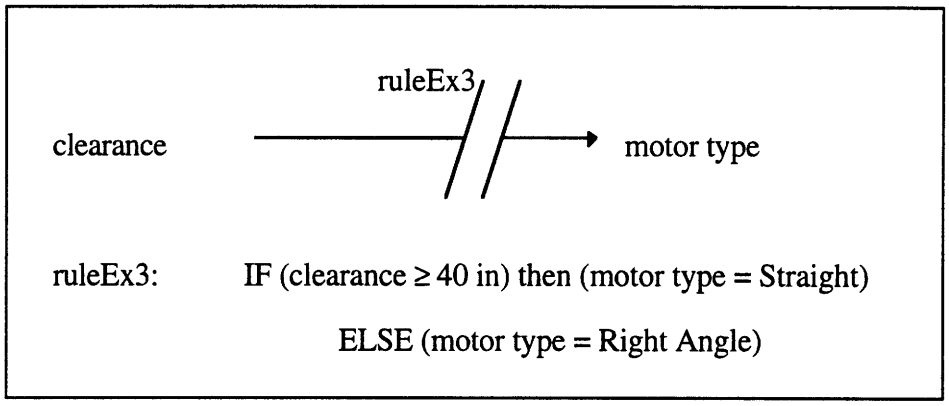


Figure 6-10 Sample System Rule for Motor Selection

Because the above rule stipulates only the conditions for determining motor capacity, and motor capacity is but one parameter that defines a physical motor, this process must continue until a complete set of rules for the motor can be determined. From the matrix in Figure 6-8, three components—clearance, motor capacity and motor stroke length—define which motor should be selected. These rules are developed below.



(a)



(b)

Figure 6-11 Sample System Rules for Motor Selection (Cont'd.)

Having defined the three parameters that determine the selection of a motor, the motor can be identified. The motor can be selected in one of two ways: either the data from the rules can be collected and then used to query a motor database, or individual rules may be built for the selection of each motor. In the case where there exist many motors, querying a database is most efficient, however, when only a few motors exist, writing individual rules for each motor is acceptable. Figure 6-12(a) integrates the previous rules for motor selection in a list that can be used to query a database of possible motors. Figure 6-12(b) gives an example of a rule for the selection of one motor from a database of motors in stock or a catalogue. (It is not necessary to assemble all relevant parameters before executing a database query. A query could be performed for each parameter independently before their assembly).

motor capacity:	(for aluminum, ≥ 0.250 in)
motor stroke lth:	(≥ Y)
motor type:	Straight

(a)

ruleEx5: IF (material stack = titanium/aluminum)
AND (hole diameter ≤ 0.313 in)
AND (hole depth ≤ 1.250 in)
AND (clearance ≥ 40 in)
THEN (motor = MOTOR150)

(b)

Figure 6-12 Final Rules for Motor Selection

In this selection application, both database querying and formal rule building were executed. With respect to database queries, the query must usually be associated with a policy in case the query returns no matching element. For example, Figure 6-12(a) outlines the three parameters against which motors in a motor database are compared. If the query is executed and no matches are found, without a policy for re-submitting the query, the KBS cannot proceed further with the selection process. A policy which modifies one or more of the parameter values and then re-submits the query, would make the system more robust. In the context of the motor parameters in Figure 6-12(a), if no matches were returned in the first query, the motor capacity requirements could be increased, and the new set of parameters could be submitted. More detailed examples of database queries with KBS rules are given in Appendix B. An awareness of the parameters of elements in the database is key to both devising rules for selection and gathering parameters to submit in a database query.

6.6.2 Parameter Isolation for Coupled Elements

As defined in section 5.2 and illustrated in Figure 5-8, coupled elements form a circular loop that must be broken in order to solve the system. Sections 5.3.1 - 5.3.3, defined three means of

uncoupling coupled systems. Figure 6-13 represents a loop of coupled elements from the matrix in Figure 6-8 and identifies the relationship linking them.

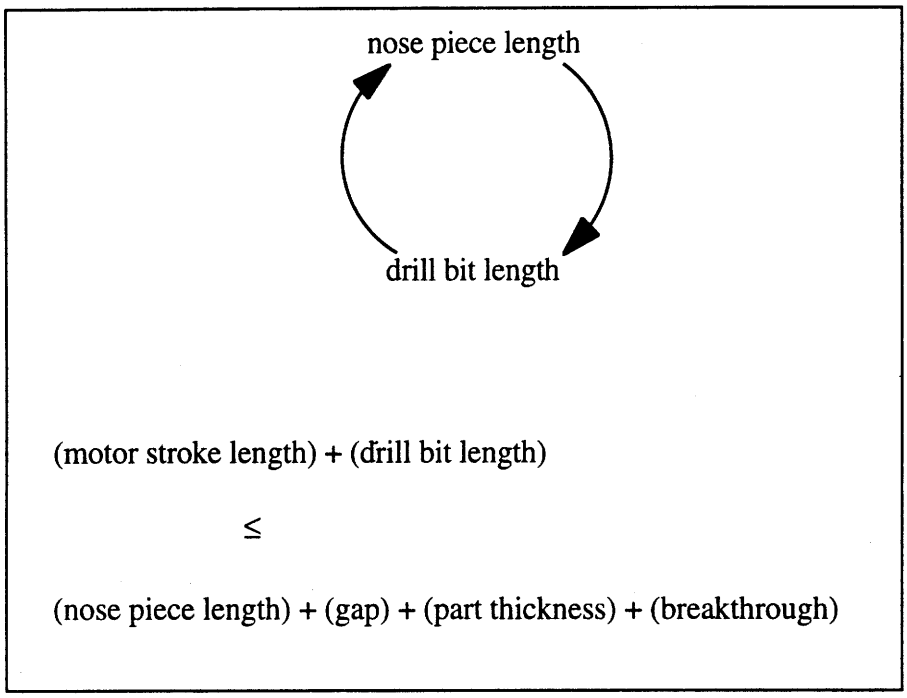


Figure 6-13 Coupled Elements in Isolation and Corresponding Relationship

In the relationship shown in Figure 6-13, all parameters have been defined except the coupled parameters, *drill bit length* and *nose piece length.* In order to solve the system, a policy must be devised for breaking the loop. In this case, there were several available nose piece lengths and fewer drill bit lengths. A decision was made to break the loop by stipulating that a standard drill size of 5" would be used.

6.7 KATMO's Output

As discussed in the introduction of section 5, KATMO's output is not code, but rather, a knowledge document. This document is comprised of two components: the design structure

matrix mapping relationships and the parameter isolations. These documents serve as a roadmap to the coding of the KB system. By not attempting to bridge the gap between knowledge acquisition and executable code, KATMO is more versatile in its application to a number of KB programs.

6.8 Discussion of KATMO

KATMO's methodology facilitates the generation of a KBS by providing an efficient way of soliciting and structuring knowledge. The tool however, has not been developed into software and thus cannot be fully evaluated. One aspect of KATMO's knowledge acquisition process that may proves challenging for users of the tool is in the decomposition of elements to the parameter level and the identification of parameter links. For example, domain expert may be very familiar with the concept that the X nose piece is used with Y motor, but not familiar with the fundamental link: between the outside threading on motor Y which interfaces with the inside threading of nose piece X. Another aspect of the tool that requires validation is the process of automatic parameter classification. This method was developed after the development of the prototype KBS and has not been tested in a production environment.

6.9 Future work with KATMO

To fully evaluate and further develop KATMO will require first, that the methodology presented here be embodied in a software tool, and second, that additional KBS applications be developed following the methodology. These applications should be pursued to production maturity as this provides both the greatest test of the methodology and the greatest opportunity for learning. Additional KBS applications will also determine if KATMO has uses other than component selection. In quick development experiments, KATMO's methodology has also proven effective in the construction of items (much like the KA Tool SALT [Marcus]). This should be further explored. Through experimentation, the quality and capabilities of the tool will be augmented.

7.0 Knowledge-Based Systems Technologies and Business Policy

The introduction of knowledge-based systems and the use of knowledge acquisition tools in a business environment must be regarded in the same manner as the introduction of any new technology: as a tool that presents potential benefits, but one that is not without consequences for the user environment. An understanding of these consequences and the ability to pursue policies that encourage the acceptance of the technology is critical to successful implementation. In the following sections, issues of business policy will be discussed with respect to KBS technologies—KB systems and KA tools. In particular, policies concerning end-user involvement with system development, user acceptance of the technology, and KBS maintenance in the context of technological innovation are addressed.

7.1 Policies of User Involvement

The success of a KBS development project cannot be evaluated in a binary manner with implementation as the only criterion. Equally important is user "buy-in"—acceptance and use of the new system by the user. In a study of thirty-four software development projects (including expert systems), Leonard-Barton and Sinha found that the degree and type of user involvement in the development process related strongly to project success [Leonard-Barton]. Leonard-Barton describes four models of user involvement. These are defined below.

1. Delivery Mode, or "Over the Wall:" No end-user involvement in the development process. The development team conceives and develops the product based on its perception of user needs.
2. Consultancy: Periodic consultation of end-users by the development team concerning the desired functionality of the product.

3. Co-development: Users are included in the development team. Strong influence of the final product by members of the user community.
4. Apprenticeship: Users assume full responsibility for the design and development of the product with software developers serving only as guides with programming capabilities.

While noting that all levels of user involvement can result in project success, certain circumstances seemed more appropriate for each method. In particular, Leonard-Barton concluded that when work in the user environment was well established and the process or system was well understood, consultancy-level user involvement seemed sufficient interaction for project success. In contrast, with novel systems or highly specialized applications, co-development was essential [Leonard-Barton].

In company interviews conducted by Keiser (see 4.5), OEM employees were not optimistic about the potential for success of KBS development efforts that included low-level or moderate-level user involvement. According to those companies interviewed, high-level user involvement—co-development or apprenticeship—was essential to the success of the project. (Because Leonard-Barton's data includes software systems other than KBS, this statement does not necessarily contradict those results).

In practice, there may be many impediments to achieving the desired level of user involvement. For example, managers may be reluctant to free up workers for participation in a KBS development project when that project is not directly funded under their budget and there are no local financial rewards associated with the project's success. Another scenario in which appropriate level of user involvement may not be possible is when the KBS project requires knowledge and skills from an adversarial labor union. In this case, delivery mode involvement may be the sole option.

Either of the above scenarios can hinder the development of a KBS, as most KBS development efforts require knowledge from numerous participants, and the use of some KA tools, such as KATMO, derive much of their benefits from use in team environments. Currently, some companies attempt to address the first issue of conflicting manager objectives by permitting employees to charge hours to projects rather than to departments. While this practice does remedy the completely contrary situation in which one manager pays an employee to work on a the project of another manager, it does little to balance time constraints and distribute appropriate rewards. A non-frequent occurrence is that a manager pulls a KBS team participant from the project whenever projects directly under his/her control need attention.

One solution to aligning conflicting managerial obligations may be to treat KBS project planning in a manner similar to large, company-wide development projects. In the case of the latter, planning and projecting resources requirements are established up front by all parties involved. In this scenario, rather than one department "owning" the project and employees from other departments being "leased" for their contribution, all managers whose employees are involved in the project will have an incentive for supporting its development.

7.2 Policies for Achieving User Acceptance of KBS Technology

Knowledge-based system are a tool for improving performance within a company; their appeal is derived from their ability to automate time consuming or difficult tasks, thus decreasing development time. Whenever a task is automated, the job function of the person formerly assigned to that task is altered. Understandably, user resistance to the technology may be encountered as a worker recognizes that his role and responsibilities are changing and possibly diminishing. This is particularly difficult when the worker has unique domain knowledge and is asked (or assigned) to participate in the development of the very technology that may obviate his position. In an effort to bolster employee commitment to automated systems, some companies have sought to reduce

employee anxiety by not threatening job security and through the choice of systems to be automated.

7.2.1 Supporting Job Security

Unlike introducing a system such as an automated assembly line which can be designed miles from its future environment and then introduced to replace an entire line of workers, KBS usually automates only a portion of an entire design or manufacturing process. In addition, KBS introduction usually requires both the knowledge and the support of those workers who will interact with the system. Galvanizing the participation of these individuals is impossible if the knowledge they share will be used to displace them or their co-workers. As a result, companies must make explicit their intentions of *enhancing* the capabilities of its workers through KBS technology rather than *replacing* them. As one KBS developer noted, KBS projects are very easy to sabotage. Even if the system is developed successfully, workers who feel threatened can refuse to use the system, or can deliberately misuse the system. Consequently, companies may have significant difficulties in trying to develop quality KBS systems without supporting the job security of their current workforce. By promoting job security, employees will be more apt to contribute to KBS development.

7.2.2 Choosing Desirable Tasks to Automate

An additional step that companies can take in garnering support for KBS is through the tasks they choose to automate. Many employees prefer some elements of their job function to others. For example, a design engineer may enjoy designing parts more than checking the manufacturability data of those parts. By automating iterative and time consuming tasks—or any task a worker would rather not do—KBS enables a worker to increase the time he spends on tasks he enjoys [Ford]. By selecting to automate tasks that workers dislike, companies can quickly generate user support of a KBS project.

7.3 KBS Maintenance and Technological Innovation

Henderson and Clark define four types of innovation that affect products and companies: incremental innovation, modular innovation, architectural innovation, and radical innovation. These categories and their definitions are given below.

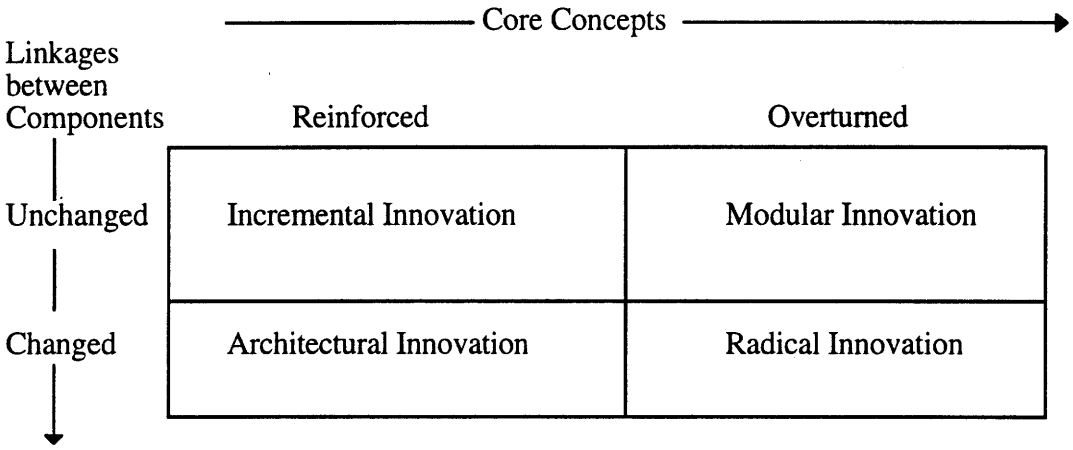


Figure 7-1 "A Framework for Defining Innovation [Henderson and Clark]"

1. Incremental innovation refines and extends an established design, while improvement occurs in individual components.
2. Modular innovation changes only the core design concepts of a technology; the linkages between components are unchanged.
3. Radical innovation establishes a new dominant design and a new set of core design concepts that result in changes to the linkages between components.
4. Architectural innovation change the linkages between the components of a product, while leaving the core design concepts unaltered [Henderson and Clark].

These concepts add insight into the role of KBS and its appropriateness for automating certain systems in a manufacturing organization. Issues of KBS maintenance become important in the presence of innovation, which implies some level of change of knowledge. KBS maintenance is discussed in the context of the different types of innovation identified in Figure 7.1, and is then linked back to policies of KBS task selection. The concept of "product architecture," defined by Ulrich as "the scheme by which the function of a product is allocated to the physical components," also adds insight to the discussion.

7.3.1 KBS Maintenance and Incremental Innovation

As defined on the previous page, incremental innovation entails the refinement of components of a product. In the context of powerfeed drill kits, such innovation might include changing the angle on a drill bit to provide a more precise hole, or improving the design of a nosepiece to allow better chip disposal. These examples reflect a change in knowledge of an individual component, or element, in the kit. While important to identify, these innovations present no great problem to an existing KBS system—the KBS can easily be edited to incorporate the new knowledge into its existing knowledge base.

7.3.2 KBS Maintenance and Modular Innovation

As did incremental innovation, modular innovation implies no direct change to the linkages between components in the system. However, modular innovation is more far-reaching in that it involves the introduction of new elements in the system. Again using the example of powerfeed drill equipment, a modular innovation might be the substitution of a drill motor with a "drill gun" (an imaginary product used here only for illustration), which produced a hole through punching rather than drilling. The net result of this innovation for the KBS is a substitution of one "module" of knowledge with another; however, the architecture of the system remains the same.

7.3.3 KBS Maintenance and Architectural Innovation

In architectural innovation, the elements of a system are reconfigured in a new way, while knowledge about the functionality of individual elements does not fundamentally change. An example of such a change might be the re-design of a motor to incorporate the nosepiece, thus combining the functionality of two components in one. In this case, the functionality of the formerly separate nosepiece—to provide a link between the motor and the bushing collar and to provide chip disposal—has not changed, but the interface between the nosepiece, motor and bushing collar has been altered. With architectural innovation, despite some components remaining unchanged, the interrelations described in the KBS system are no longer valid. (This is equivalent to reconfiguring the relationships illustrated in the matrices presented in 5. and 6.). The previous KBS is thus inadequate to address the design challenges of the new system.

7.3.4 KBS Maintenance and Radical Innovation

Radical innovation involves fundamental changes to a system's components and the relationships between components. An example might be the replacement of powerfeed equipment with equipment that dispenses a high-strength curable material which could be used to bond parts chemically, rather than fasten them mechanically. In such an example, virtually all knowledge captured within the existing KBS is obsolete; an entirely new system would need to be developed.

Potential difficulties in KBS maintenance with respect to the level of maturity of the technology to be enhanced with a KBS also have implications for initial KBS task selection. Mature technologies are presumably past stages of radical innovation. Mature systems are widely understood, can be easily decomposed and thus, may be good candidates for KBS technology. These systems may undergo incremental or modular innovations, which result in improvement, but do not change the structure of the system. In contrast, immature or developing technologies undergo periods of radical innovation, which profoundly alter the architecture of a product or system. Under such levels of change, the development of a KBS may be a futile exercise, as the acquired knowledge

quickly becomes obsolete. Therefore, in order to achieve a maximum return for the effort invested in developing KBS systems, mature systems may be better candidates for automation as their processes, and the associated knowledge, are less likely to change in a manner that would obviate the KBS.

7.4 Policy Summary and Discussion

The policy issues discussed in the preceding sections are summarized in the matrix below.

policy issue	suggested policy	impediments/risks
1. end-user involvement in KBS development	high-level user involvement	-conflicting management objectives -user resistance to technology
2. encouraging user acceptance of KBS technologies	use KBS as a means to enhance, not replace workers	company desire for productivity returns
	choose to automate tasks that workers want to automate	does the automation of these tasks provide real savings to the company?
3. task selection in the context of technological innovation	automate mature processes	does the automation of these tasks provide real savings to the company?

Figure 7-2 KBS Policy Matrix

As noted, there are impediments or risks associated with each policy. In particular, the issue of company savings is raised twice with respect to task selection. While a workers may be enthusiastic about the automation of the mundane and tedious aspects of his/her position, those tasks are not necessarily those which can provide the largest productivity gain for the company. This point also applies to the policy of automating only mature technologies: established technologies may be well functioning and not be large drivers of product development lead time or process inefficiencies. Therefore, before committing resources to any KBS project, a cost-benefit analysis must be performed to determine the potential financial impact of the KBS application.

Another issue of policy concern is that of replacing workers. The policy recommendation is to emphasize worker enhancement rather than worker replacement. While guarantying job security is the most desirable outcome from the employee point of view, in the long-run it is unrealistic. KBS technology is attractive to companies because it replaces human effort in certain areas, enabling fewer employees to accomplish the same tasks. The substitution of automated systems for human engineers or designers in turn enables cost-cutting and productivity gains. While it is unlikely that companies will be able to develop KB systems if employees believe they will be immediately replaced those systems, in the long-run, the need for productivity improvements will likely lead to such a substitution. The substitution of automated systems for employees is however, also associated with risks. If KBS is used to replace workers by automating systems that undergo architectural or radical innovation— which renders the KBS obsolete, the company may find itself in the precarious position where the technology it has used to capture valuable knowledge is now obsolete, and sources of that knowledge are no longer available.

7.5 Implications of KBS Technology for the Manufacturing Organization

Unlike upgrading a single unit of the manufacturing organization such as one machine in a production line, KB systems (and other automated manufacturing technologies) are powerful because they can serve to integrate knowledge of cross-functional tasks [Hayes and Jaikumar]. This integration brings employees together along the goal-oriented lines of the system to be automated. Unfortunately, this integration is subject to obstacles.

The process by which KATMO creates a knowledge document is first by gathering information, second, organizing that information, and third, isolating pieces of the system for automation. This methodology derives its strength from the explicit identification of element and parameter relationships and their effective organization. This methodology also highlights elements that have no relationship. A similar methodology has been applied to the organization of company tasks for

the completion of a large automotive development project [Eppinger and McCord]. This example will be discussed because it highlights parallel issues in developing KB systems:

In reorganizing a group or company to complete a task, first the process steps are identified. Then, process steps are organized according to the most efficient or otherwise desirable sequence. The organizational structure that is identified through this process organization, however, rarely reflects the current organization of the company [Gross]. For example, instead of a manufacturing team being co-located with a design team to assess a production plan, their separation may result in additional process steps and iterations such as "Design Team sends Manufacturing Plans to Manufacturing Team in Plant A for Approval," "Manufacturing Team makes Changes to Manufacturing Plans," "Send Manufacturing Plans back to Design Team," "Change Design," etc. These steps, and much time and money, could be eliminated given an alternative team organization which encouraged a closer contact between the design and manufacturing team. The process of mapping process steps, enables a company to (1) identify value added and non-value added tasks and then eliminate them in the pursuit of increased efficiency and productivity [Womack], and (2) identify alternative organizational structures that better address the objectives of the company [Eppinger, Gross].

Unfortunately, both eliminating tasks and reorganizing company structures requires flexible organizations in which managers readily relinquish and accept authority over employees. Relinquishing authority however is often seen as tantamount to demotion [Boeing]. In many manufacturing companies, as managers are promoted they increase both their salary and responsibility in terms of the number of people they supervise. A diminution of their supervision is seen as an attack on their status and is fought vigorously [Boeing].

This example is relevant for KBS because KBS development and implementation also imply changes to the work and system-user environment. Organizing a task for automation requires a

formal structuring of the process. This structuring reveals the inefficiencies of the current process. Re-organizing these processes often requires shifts of managerial control. As a result, the decision to alter company processes, like company organizational structures, often require the vision and intervention of top-level management.

7.6 Summary

The successful introduction of KBS technologies is not dependent solely on the quality of the development methodology or any tools used in the development process. Business policy issues such as involving users in the KBS development process, achieving user support of KBS technologies, and selecting tasks for automation that provide long-term benefits to the company are also important to KBS project success. In addition, KBS introduction may have far-reaching implications for the organizational structure of the company, requiring top-level management initiative. An understanding of the benefits and challenges of KBS along both technology and policy dimensions is important for the successful introduction of the technology.

8. Conclusion

Knowledge-based systems and knowledge acquisition tools are exciting technologies that present the potential to automate many design tasks, enabling companies to decrease lead-times and increase quality. In this thesis, the design of one tool for facilitating the development of KB systems, KATMO, was introduced. This tool is comprised of three phases: knowledge acquisition through interactive interviewing, knowledge structuring through matrix organization, and parameter isolation and rule identification. The intended used of KATMO is in the structuring of knowledge relating to selection tasks. A detailed explanation of the structure of this tool was given in Section 5. In Section 6, the application of KATMO to the development of a prototype KBS for the selection of components in a powerfeed drill kit was described. This application highlights features of KATMO which include a means for determining an explicit mapping of system element and parameter relationships, and complexity management—the process of identifying locations where rules are difficult to capture, and introducing policies or user interaction to manage the interrelations. This process enables *selective automation* in which the elements of a system which can be easily captured with rules are automated, and more difficult relationships are left for the user to manage. Selective automation extends the applicability of automated systems. In Section 7, business policy issues with respect to KBS technologies were addressed. In particular, the importance of involving users of KBS in the development process and the means for achieving user acceptance of the technology were discussed. Finally, the role of KBS in the context of technological innovation was analyzed. This revealed the potential inappropriateness of KBS to novel systems because these systems are likely to change frequently and profoundly, obviating the knowledge captured in the KBS knowledge base.

While KB systems are themselves an immature technology, their application in production environments has resulted in many successful systems. KA tools however, despite their large potential to accelerate the KBS development process, have not been widely employed in production

environments. This thesis hopes to further research toward the development of a production ready KA tool.

Bibliography

[Bachrach] Bachrach, H., "Formal methods for the development of Design Automation Applications," Masters Thesis, MIT, 1997

[Barstow, et.al] Barstow, D.R., Aiello, N., Duda, R.O., Erman, L.D., Forgy, CL, Gorlin, D., Greiner, R.D., Lenat, D.B., London, P.E., McDermott, J., Nii, H.P., Politakis, P., Reboh, R., Rosenschein, S., Scott, A.C., van Melle, W., Weiss, S.M., in Building Expert Systems, Hayes-Roth, F., Waterman, D.A., Lenat, D.B., Addison-Wesley Publishing Company, Reading, MA, 1983

[Boeing] Interviews at The Boeing Company

[Boose, 1991] Boose, J.H., "Knowledge Acquisition Tools, Methods, and Mediating Representations," in Knowledge Acquisition for Knowledge-Based Systems, IOS Press, Ohmsha, Ltd., Amsterdam, 1991

[Boose] Boose, J.H., "A survey of knowledge acquisition techniques and tools," in Readings in Knowledge Acquisition and Learning, Buchanan, B.G., Wilkins, D.C., Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Boose and Bradshaw] Boose, J.H., Bradshaw, J.M., "Expertise transfer and complex problems: Using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems," in Readings in Knowledge Acquisition and Learning, Buchanan, B.G., Wilkins, D.C., Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Buchanan, et.al.] Buchanan, B.G., Barstow, D., Bechtal, R., Bennett, J., Clancey, W., Casimir, K., Mitchell, T., Waterman, D.A., in Building Expert Systems, Hayes-Roth, F., Waterman, D.A., Lenat, D.B., Addison-Wesley Publishing Company, Reading, MA, 1983

[Buchanan and Shortliffe] Buchanan, B.G., Shortliffe, E.H., Rule-Based Expert Systems, Addison-Wesley Publishing Company, Reading, MA, 1983

[Buchanan and Wilkins] Buchanan, B.G., Wilkins, D.C. (editors), Readings in Knowledge Acquisition and Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Casey] Casey, T., "Picking the Right Expert System Application," AI Expert, Sept., 1989, pp. 44-7

[Charlet and Gascuel] Charlet, J., Gascuel, O., "Knolwedge Acquisition by Causal Model and Meta-Knowledge," Proceedings of EKAW-89, Third European Workshop on Knowledge Acquisition for Knowledge-Based Systems," Paris, July 1989, pp. 212-26

[Concentra] Interviews with Concentra

[Davis] Davis, R., "Interactive transfer of expertise: Acquisition of new inference rules," in Readings in Knowledge Acquisition and Learning, Buchanan, B.G., Wilkins, D.C., Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Davis] Davis, R., in Rule-Based Expert Systems, Buchanan, B.G., Shortliffe, E.H., Addison-Wesley Publishing Company, Reading, MA, 1983

[Edosomwan] Edosomwan, J.A., "Ten Design Rules for Knowledge Based Expert Systems," in Expert Systems, Botten, N.A., Raz, T., Industrial engineering and management press, Institue of Industrial engineers, Atlanta, Georgia, 1988

[Eppinger et al.] Eppinger, S.D., Whitney, D.E., Smith, R.P., Gebala, D.A., "Organizing the Tasks in Complex Design Processes," ASME Design Theory and Methodology Conference, Chicago, IL, 1990

[Eppinger and McCord] Eppinger, S.D., McCord, K.R., "Managing the Integration Problem in Concurrent Engineering," MIT Sloan School of Management Working Paper, no. 3594, August, 1993.

[Eshelman] Eshelman, "MOLE: A Knowledge-Acquisition Tool for Cover-and-Differentiate Systems," in Automating Knowledge Acquisition for Expert Systems, Marcus, Kluwer Academic Publishers, Boston, MA, 1988

[Eshelman et al.] Eshelman, L., Ehret, D., McDermott, J., Tan, M., "MOLE: A tenacious knowledge-acquisition tool," in Readings in Knowledge Acquisition and Learning, Buchanan, B.G., Wilkins, D.C., Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Field] Field, J.D., "A Rule-Based Design System for Aircraft Engine Tooling," Masters Thesis, MIT, 1992

[Ford] Interviews with Ford

[Gebala and Eppinger] Gebala, D.A., Eppinger, S.D., "Methods for Analyzing Design Procedures," Design Theory and Methodology, DE-Vol. 31, ASME, 1991

[Gross] Interviews with Dr. David Gross, The Boeing Company

[Hayes and Jaikumar] Hayes, R.A., Jaikumar, R., "Manufacgturing's Crisis: New Technologies, Obsolete Organizations," in Strategic Operations, Hayes, R.H., Pisano, G.P., Upton, D.M., The Free Press, NY, NY, 1996

[Hayes-Roth, et.al.] Hayes-Roth, F., Waterman, D.A., Lenat, D.B., Building Expert Systems,Addison-Wesley Publishing Company, Reading, MA, 1983

[Hendersen and Clark] Henderson, R., Clark, K., "Architectural Innovation: The reconfiguration of existing product technologies and the failure of established firms," ASQ, 35, pp. 9-30, 1990

[Kahn] Kahn, G., "MORE: From Observing Knowledge Engineers to Automating Knowledge Acquisition," in Automating Knowledge Acquisition for Expert Systems, Marcus, Kluwer Academic Publishers, Boston, MA, 1988

[Keiser and Bachrach] Phone interviews were conducted between January, 1997 and May, 1997 with KBS developers and employees at several companies.

[Klinker] Klinker, G. "KNACK: Sample-Driven Knowledge Acuqisition for Reporting Systems," in Automating Knowledge Acquisition for Expert Systems, Marcus, Kluwer Academic Publishers, Boston, MA, 1988

[Laufmann et al.] Laufmann, S.C., DeVaney, M., Whiting, M.A., "A methodology for evaluating potential KBS Applications," IEEE Expert, V5, Number 6, Dec., 1990, pp. 43-61

[Leonard-Barton] Leonard-Barton, Dorothy, Wellsprings of Knowledge, Havard Business School Press, Boston, MA, 1995

[Lockheed] Interviews at Lockheed Martin Corporation

[Marcus] Marcus, S., "SALT: A Knowledge-Acquisition Tool for Propose-and-Revise Systems," in Automating Knowledge Acquisition for Expert Systems, Marcus, Kluwer Academic Publishers, Boston, MA, 1988

[Marcus and McDermott] Marcus, S., McDermott, J., "SALT: A knowledge acquisition language for propose-and-revise systems,"in Readings in Knowledge Acquisition and Learning, Buchanan, B.G., Wilkins, D.C., Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Masud] Masud, A.S.M., "Knowledge-Based Systems and Industrial Engineering," in Expert Systems, Botten, N.A., Raz, T., Industrial engineering and management press, Institue of Industrial engineers, Atlanta, Georgia, 1988

[McDermott] McDermott, J., "Preliminary Steps Toward a Taxonomy of Problem-Solving Methods," in Automating Knowledge Acquisition for Expert Systems, Marcus, Kluwer Academic Publishers, Boston, MA, 1988

[McDermott, et al.] McDermott, J., Dallemagne, G., Klinker, G., Marques, D., Tung, D., "Explorations in How to Make Application Programming Easier," Knowledge Acquisition for Knowledge-Based Systems, IOS Press, Ohmsha, Ltd., Amsterdam, 1991

[Michalski] Michalski, R.S., "A theory and methodology of inductive learning," in Readings in Knowledge Acquisition and Learning, Buchanan, B.G., Wilkins, D.C., Morgan Kaufmann Publishers, San Mateo, CA, 1993

[Mills] Mills, Robert, "Engineering Software: Mechanical Design," CAU, Volume 10, Issue 12, December, 1991, pp. 14-22

[Morik et al.] Morik, K., Wrobel, S., Kietz, J-U., Emde, W., Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications, Academic Press Inc., San Diego, CA, 1993

[Motoda et al.] Motoda, H., Mizoguchi, R., Boose, J., Gaines, B. (editors), Knowledge Acquisition for Knowledge-Based Systems, IOS Press, Ohmsha, Ltd., Amsterdam, 1991

[Nielsen and Walters] Nielsen, N.R., Walters, J., Crafting Knowledge-Based Systems, John Wiley & Sons, Inc., New York, New York, 1988

[Offutt] Offutt, D., "SIZZLE: A Knowledge-Acquisition Tool Specialized for the Sizing Task," in Automating Knowledge Acquisition for Expert Systems, Marcus, Kluwer Academic Publishers, Boston, MA, 1988

[Payne and McArthur] Payne, E.C., McArthur, R.C., Developing Expert Systems, John Wiley & Sons, Inc., New York, New York, 1990

[Rogers] Rogers, J.L., "DeMAID: A Design Manager's Aide for Intelligent Decomposition User's Guide," NASA Technical Memorandum 101575, March, 1989

[Rogers et al.] Rogers, J.L., McCulley, C.M., Bloebaum, C.L., "Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes," Artificial Intelligence in Design, Kluwer Academic Publishers, Netherlands, 1996

[Schmoldt and Rauscher] Schmoldt, D.L., Rauscher, H.M., Building Knowledge-Based Systems for Natural resoure Management, Chapman & Hall, New York, New York, 1995

[Seidel] Description provided by David Seidel, LMTAS

[Sell] Sell, P.S., Expert Systems: a practical introduction, John Wiley & Sons Inc., New York, 1985

[Slagle and Wick] Slagle, J., Wick, M., "A Method for Evaluating Candidate Expert Systems Applications," AI Magazine, Vol. 9, No. 4, 1988, pp. 44-53

[Smith] Smith, P., An Introduction to Knowledge Engineering, International Thomson Computer Press, London, 1996

[Smith and Kandel] Smith, S., Kandel, A, Verification and Validation of Rule-Based Expert Systems, CRC Press, Ann Arbor, MI, 1993

[Steward] Stewar, D.V., Systems Analysis and Management: Structure, Strategy and Design, Petrocelli Books, NY, 1981

[Ulrich] Ulrich, K., "The Role of Product Architecture in the Manufacturing Firm," MIT Sloan School of Management, Oct., 1992

[van Steenbergen] van Steenbergen, M.E., "EXACT: A Model-Based Knowledge Acquisition Tool for Selection Tasks," IEEE/ACM International Conference on Developing and Managing Expert System Programs, IEEE Computer Society Press, Los Alamitos, CA, 1991

[Weiss and Kulikowski] Weiss, S.M., Kulikowski, C.A., A Practical Guide to Designing Expert Systems, Rowman & Allanheld Publishers, Totowa, New Jersey, 1984

[Womack and Jones] Womack, J., Jones, D., Lean Thinking, Simon and Schuster, NY, NY, 1996

Appendix A: Complete Matrices

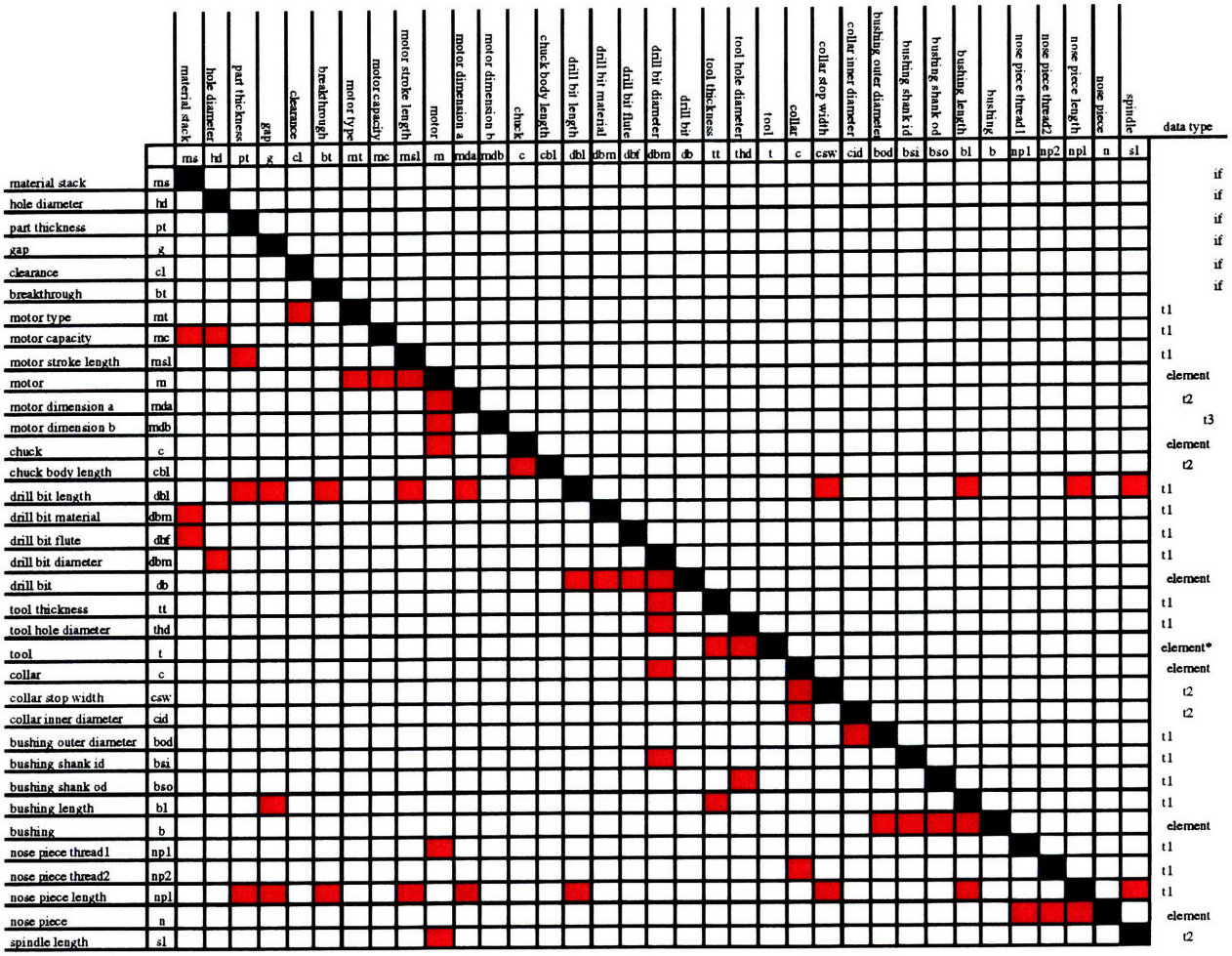


Figure A-1 "Unorganized" Powerfeed Matrix

* *Tool* is listed as an element but is not part of the selection process. Tools are designed and include many design features. The selection process above determines two parameters of a tool. For that reason, the tool element was included in the matrix.

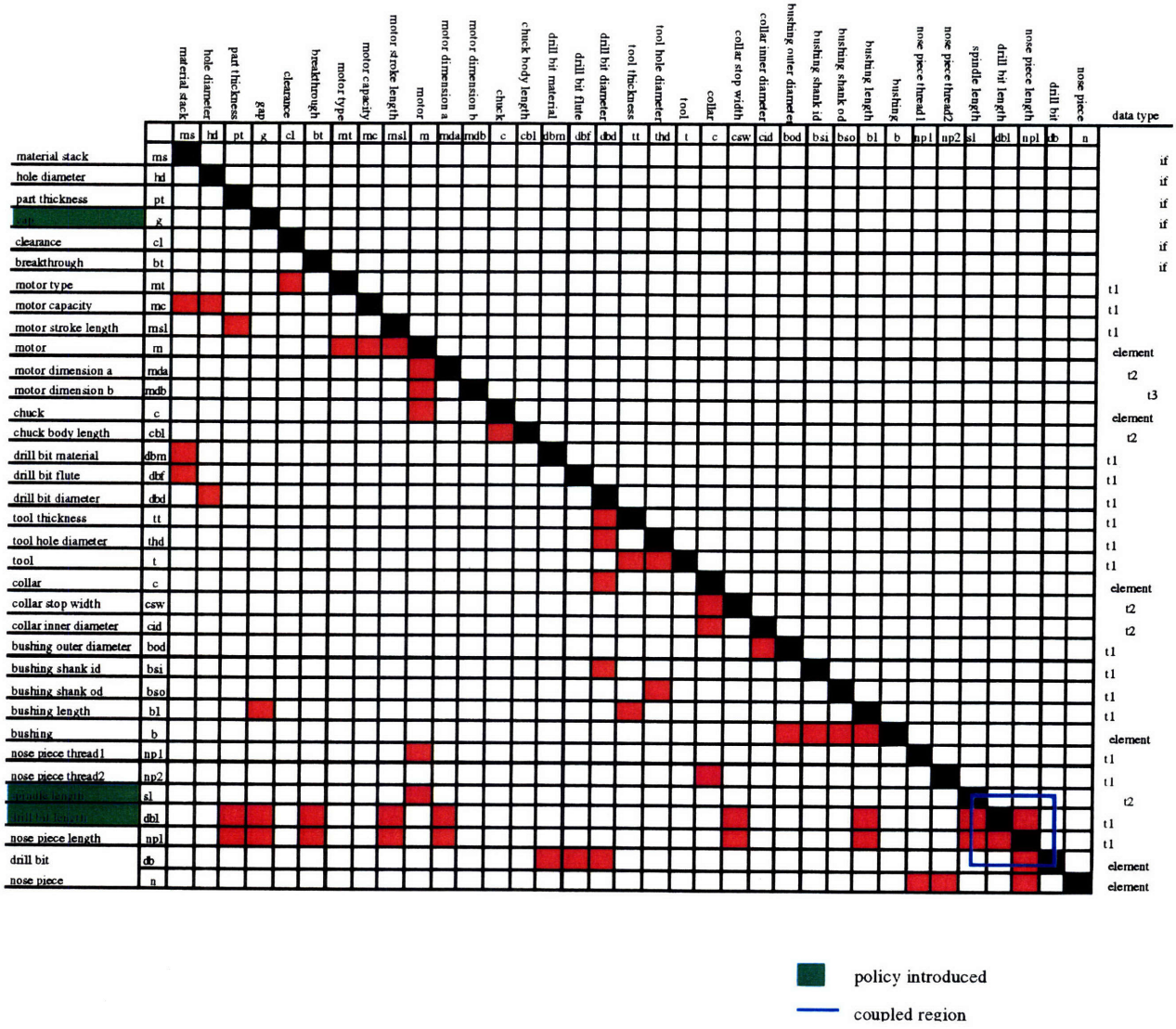


Figure A-2 "Organized" Powerfeed Matrix

policy1: gap policy type: Selection Policy
The gap between the bushing and the part depends on the design of the tool and the location of the part to be drilled. While the gap is not a constant distance for every hole, a policy was devised to assign the gap a value so other elements could be calculated. Following the computation, that user is asked if the gap value is correct. If it is not the case, a new value is entered and the selection process is re-initiated. The gap value is asked after one iteration because in most cases, the assigned value is valid.

policy2: spindle length policy type: Selection Policy
The spindle of the motor is adjustable. In order to execute the computation, a value was assigned to the spindle. In the case of the need for fine tuning, the actual spindle length can be adjusted during the kit assembly process.

policy3: drill bit length policy type: Selection Policy
The drill bit length and the nose piece length form a loop. Because these are parameters defining elements, the selection process cannot be executed without breaking this loop (Red square in drill bit row). Because more nose piece lengths existed than drill bit lengths, drill bit length was assigned a value. The procedure is to query a database for a certain drill bit length value. If no matches are found, the system modifies the value and re-executes the query.

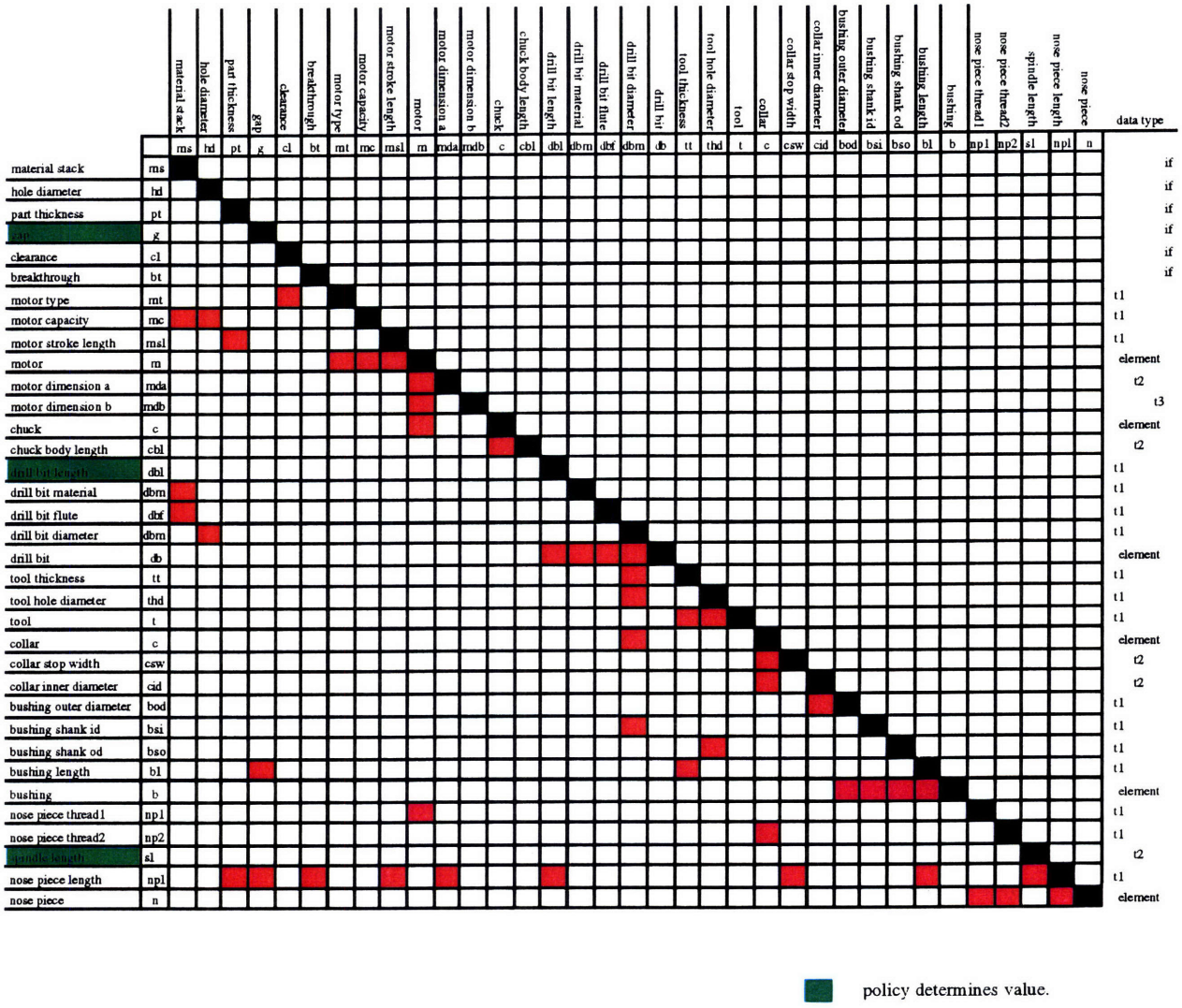


Figure A-3 Policy Managed Matrix--Solved System

Appendix B: Rule Set

Sample Data							
(disguised)							
database1							
Motor	Hole Diameter Capacity	type	Max Stroke Lngth	motor dimension A	Chuck	Chuck Body	Nose Piece Thread 1
M15D-S125	Al-0.375, Ti-0.313	Right Angle	1.250	0.375	Chk1	0.5	1.578-18NS-LH
M16D-S125	Al-0.500, Ti-0.375	Straight	3.000	0.500	Chk2	0.75	1.578-19NS-LH
M17D-S125	Al-0.375, Ti-0.313	Straight	3.000	0.500	Chk2	0.75	1.578-20NS-LH
rule1							
IF (material = Ti) AND (hole diameter < 0.375) AND (part thickness < 3.000) OR							
IF (material = Al) AND (hole diameter < 0.500) AND (part thickness < 3.000)							
THEN (Motor = M15D-S125).			**Element Defined:**	**Motor**			
				Chuck			
database2							
part material	drill bit material	drill flute					
Composite/Aluminum	Cobalt	R					
Titanium	High Speed Steel	L					
Fiberglass	Carbide	L					
Al/Ti/Al	Cobalt	R					
rule2							
hole diameter	drill bit diameter						
	drill diameter = hole diameter - 0.05"						
Query drill database for diameter, if match, accept, if no match							
	drill diameter = hole diameter - 0.075"						
repeat with increments of 0.025".							

rule3: policy							
drill bit length							
Query drill database for drill bit length = 5", if match, accept, if no match							
	drill bit length = 5.5".						
repeat with increments of 0.5".			**Element Defined:**	**Drill Bit**			
database3							
drill bit diameter	drill bit diameter						
lower bound	upper bound	tool hole diameter	tool thickness	collar	collar id	collar length	collar stop width
0	0.199	0.375	0.375	C1	0.490	1.250	0.645
0.2	0.399	0.438	0.500	C2	0.620	1.500	0.450
0.4	0.599	0.500	0.500	C3	0.950	1.500	0.450
0.6	0.799	0.500	0.750	C4	1.500	1.500	0.500
			Element Defined:	**Collar**			
database4							
Collar	Nose Piece Thread2						
C1	1.578-18NS-LH						
C2	1.578-19NS-LH						
C3	1.578-19NS-LH						
C4	1.578-20NS-LH						
database5							
collar id	bushing od						
0.490	0.500						
0.620	0.650						
0.950	0.100						
1.500	1.600						

rule4							
drill bit diameter	bushing shank id						
	bushing shank id = drill bit diameter + 0.002"						
Query bushing database for bushing shank id, if match, accept, if no match							
	bushing shank id = drill bit diameter + 0.005"						
repeat with increments of 0.003".							
rule5							
tool hole diameter	bushing shank od						
	bushing shank od = tool hole diameter - 0.100"						
Query bushing database for bushing shank od, if match, accept, if no match							
	bushing shank od = tool hole diameter - 0.200"						
repeat with increments of 0.100".							
rule6							
part thickness	bushing shank length						
	bushing shank length = part thickness + 0.02" (gap value)						
Query bushing database for bushing shank length, if match, accept, if no match							
	bushing shank length = part thickness + 0.04"						
repeat with increments of 0.020".							
Query bushing database with all assembled parameters:							
	bushing shank id						
	bushing shank od						
	bushing shank length						
	bushing od		**Element Defined:**	**Bushing**			

rule7					
Nose Piece length upper bound					
motor dimension a + 0.25" (spindle value) + Chuck Body + Drill Bit Length <=					
Nose Piece Length + Collar Stop Width + Bushing Shank Length + Breakthrough					
rule 8					
Nose Piece length lower bound					
Motor Stroke Length + Motor Dimension A + 0.25" (spindle value) + Chuck Body + Drill Bit Length >=					
Nose Piece Length + Collar Stop Width + Bushing Shank Length + Part Thickness + 0.02" (gap value) Breakthrough					
Query bushing database for nose piece length.					
Query bushing database with all assembled parameters:					
	nose piece length				
	nose piece thread1				
	nose piece thread2		**Element Defined:**	**Nose Piece**	
			Selection Process Completed.		