A PROTOTYPE KNOWLEDGE-BASED ASSISTANT FOR ACQUISITION STRATEGY DEVELOPMENT

Robert J. Hammell II, CPT, USA
Air Force Institute of Technology
Wright-Patterson Air Force Base, OH

Stephen E. Cross, Major, USAF
Air Force Wright Aeronautical Laboratories (AFWAL/AAI)
Wright-Patterson Air Force Base, OH

ABSTRACT

This paper presents an application of knowledge-based system techniques to the problem of developing acquisition strategies for government procurements. The need for automated assistance in this area is examined, and reasons why conventional programming techniques are inadequate are offered. The problem assessment is discussed, and the design and implementation of a prototype knowledge-based system is outlined. Finally, an evaluation of the system is presented.

INTRODUCTION

The acquisition strategy is one of the most important program management documents. An acquisition strategy is required for all materiel acquisition programs, regardless of dollar amount, and its primary purpose is to document to general concepts that serve as direction for the program throughout the entire acquisition cycle.

The information required in the document is extremely diverse, and input from various individuals with specific areas of expertise is necessary. The large amounts of coordination, coupled with the frequent changes in acquisition policy and the normal problems with loss of organizational expertise in a military environment, pose significant problems for the acquisition strategy developer. Additionally, the involvement of many individuals fosters inconsistencies between acquisition strategies even if they are for similar procurements within the same organization.

The purpose of this research was to demonstrate the effectiveness of applying knowledge-based system techniques to help alleviate some of these problems. This goal was accomplished by the development of a prototype "intelligent" assistant to aid the acquisition strategy developer. Additional information relating to all aspects of this research can be obtained from [1].

This paper is organized as follows: First, background information leading to the research is presented. Then, the approach to the problem is mentioned, followed by a discussion of the problem assessment, the conceptual design, the knowledge acquisition phase, and the detailed design. The tool selection process is examined, along with the prototype implementation and an illustrative example. Finally, an evaluation of the prototype is offered.

BACKGROUND

Part of the mission of the Defense Systems Management College (DSMC), located at Fort Belvoir, Virginia, is to conduct research in applied management science. The motivation for this research was the planned addition of a Procurement Package Generator by the DSMC to the Program Manager's Support System (PMSS). The Procurement Package Generator will provide assistance in the development of various program management documents, such as Statements of Work, Specifications, Commerce Business Daily announcements, Acquisition Strategies, and the like [2].

Of the documents to be included in the Procurement Package Generator module, the acquisition strategy is one of the most important. An acquisition strategy is required for all materiel acquisition programs regardless of dollar amount. The primary purpose of the acquisition strategy is to document the general concepts that serve as direction for the program throughout the entire acquisition cycle. [3:7.11 Thus, it is the program's guide for the development, support, production, and deployment of equipment. The acquisition strategy is structured in a prescribed format and must not exceed 15 pages. Thirteen specific elements must be addressed in the acquisition strategy and if any of the elements do not apply to a particular program, the rationale must be provided [4:11-22]. It is a "living document;" that is, the acquisition strategy is updated throughout the acquisition life cycle. The information contained in the acquisition strategy is typically general in the early phases of a program with the level of detail increasing as the program matures. [3:7.1]

Unlike many of the other program management documents to be included in the Procurement Package Generator, no software existed within the DoD to assist in the development of an acquisition strategy. The main reason for this is that conventional programming techniques do not allow for the efficient representation and manipulation of the vast amount of knowledge required to develop an acquisition strategy. Additionally, conventional software programs do not provide flexibility for the frequent changes that occur in acquisition guidelines and regulations.

1349

U.S. Government work not protected by U.S. copyright.
The emergence of knowledge-based systems, however, has provided new techniques that have proved beneficial in solving problems analogous to those in the acquisition strategy development area.

**APPROACH**

The approach used to develop the prototype was the standard knowledge-based system approach as outlined in [5]. The Belvoir Research, Development, and Engineering Center (BELVOIR) at Fort Belvoir, Virginia was used to form the basis for the design, capability, and evaluation of the prototype system. BELVOIR is responsible for many acquisition programs, up to 200 at any one time, and had expressed an interest to the DSMC for automated assistance with the task of developing program management documents.

**PROBLEM ASSESSMENT**

The problem assessment phase revealed that the major problem in developing an acquisition strategy is the diversity of information required in the document. The acquisition strategy is comprised of thirteen separate paragraphs. Each paragraph addresses a specialized area, such as logistics, cost, contracting strategy, and so on. Experts in these areas do exist, but the acquisition strategy developer (the project engineer, within BELVOIR) has to draw on the expertise from many different people to write the acquisition strategy. This massive coordination effort makes it hard to get the input required to develop the document.

Thus, the project engineer typically prepares a rough draft of the acquisition strategy, staffs it through the offices that should have helped develop it, incorporates the comments received, and then staffs it again. For various reasons, this review cycle may be repeated several times before approval is granted to send the acquisition strategy forward to the next level of review.

Another factor that contributes to the development and review problem is the competition for the expert's time. The personnel that must be involved with the development of an acquisition strategy for a new program also have the responsibility to support all ongoing programs within the organization. Consequently, it is hard to get the expert's time, and when the document is staffed it may get only a cursory review in some areas due to time constraints (and because they know it will be back again!). Thus, a bottleneck exists at the inboxes of the various experts.

An additional problem with developing an acquisition strategy is that the project engineer typically has little experience with writing the document. Experienced project engineers do exist, and they tend to do much better than novices at developing an acquisition strategy. This implies that the task is understood and the problem is simply one of knowledge.

Contributing to the complexity of developing an acquisition strategy are the frequent changes in acquisition regulations, policy, and guidance. Not only does the inexperienced project engineer have to locate all this written material, but he must ensure that the latest guidance is being used. This increases the slope of the learning curve for the novice.

Because each program typically has a different project engineer, and because of the frequent changes in written guidance, inconsistencies can (and do) exist in acquisition strategies even for similar procurements.

The information gained in the problem assessment indicated that the problem meets Waterman's criteria [5] for knowledge-based system development:

- it is POSSIBLE (task is understood; experts exist);
- it is APPROPRIATE (expert performance better than novice performance); and
- it is JUSTIFIED (expertise is scarce).

Thus, the problem assessment concluded that a knowledge-based system could help solve the problem of developing acquisition strategies. This was generally true throughout the thirteen paragraphs, but the problem assessment also indicated that the most difficulty was typically encountered in developing paragraph 1, Program Structure, and paragraph 3, Tailoring the Acquisition Process. These paragraphs were considered the most important by the expert, and the most challenging for which to develop a knowledge base. Consequently, paragraphs 1 and 3 were used as the basis for building the prototype.

**CONCEPTUAL DESIGN**

From information gained during the problem assessment phase, it was possible to develop a conceptual model for the prototype system. The problem assessment indicated that the major problem encountered during the development of an acquisition strategy (especially by a novice) was the diverse nature of the knowledge needed by the project engineer to write the document. This knowledge is contained in manuals, written guidance, and in the heads of experts in the various matrix support areas (logistics, contracts, etc.) as well as in the heads of experienced project engineers and the acquisition strategy reviewers and approvers. The challenge was to make all this knowledge available to the project engineer in a manner that would provide him assistance during the development of the document. This analysis suggested that an "intelligent assistant" (in the form of a knowledge-based system) with access to all the required expertise could provide the desired assistance to the project engineer. Figure 1 below graphically represents this concept.
misconceptions were corrected and missing bits of the reasoning process checking other criteria as knowledge acquisition. This phase consisted of providing this feedback to the experts, several were used for the program, and this in turn was maintained. This idea was extended somewhat by deciding that any particular area of expertise that might be shared between paragraphs (like deciding what NDI category applied) would also be represented in a separate rule base.

The primary consideration in the design of the prototype system turned out to be the preservation of consistency throughout the document. Although this aspect was barely surfaced during the knowledge acquisition phase by alluding to the fact that some paragraphs use the same information, it became a primary driver of the system design. The need for consistency throughout the acquisition strategy had several implications. First, it implied that either an elaborate form of consistency checking with a mechanism for resolution of inconsistencies would have to be developed, or a method for enforcing consistency (i.e., not allowing any inconsistencies to occur in the first place) was needed. The latter was considered the best solution.

Given that inconsistencies would not be allowed to occur, the other implications followed. For example, the user could not be asked for the answer to a particular question more than once for each acquisition strategy. While this was definitely a desired characteristic anyway, it was required to ensure that a user did not answer the question differently for different paragraphs. Given that inconsistencies would not be allowed to occur, the other implications followed. For example, the user could not be asked for the answer to a particular question more than once for each acquisition strategy. While this was definitely a desired characteristic anyway, it was required to ensure that a user did not answer the question differently for different paragraphs. Additionally, if the user was ever allowed to change any of his answers to "edit" the acquisition strategy, these new answers would have to be propagated through all the paragraphs developed to that point to ensure that consistency was maintained.

Since an acquisition strategy is a fairly lengthy document, consideration had to be given to the idea that the user might not desire to read the entire document at one sitting. This fact reinforced the need for a capability to store previous answers, deductions, and decisions so that development of an acquisition strategy could
processor interface, a built-in database, and a development tool Guru 171 was chosen. Based on well defined mechanism for interface with the user, the knowledge-based system were given the highest consideration. Tool selection is one of the most critical decisions to be made during the knowledge-based system development. The prototype needed to function logically like an expert to facilitate understanding and maintenance by personnel not intimately involved in the development of the system. The system also had to have built-in flexibility to accommodate changes in acquisition policy and guidance. This implied that textual output should be stored separately, when possible, to facilitate maintenance and that the knowledge bases should be designed so that rules could be easily added, changed, or deleted without rewriting the entire knowledge base. (The flexibility consideration is appropriate for any knowledge base and, in fact, flexibility is a major benefit provided by knowledge-based techniques over conventional programming techniques.)

Although the problem assessment phase indicated that a project engineer was usually involved with only one acquisition program at a time, consideration was still given to the fact that more than one acquisition strategy might be developed (and, hence, reside) on a single computer. The implication here was that acquisition strategy files would have to be stored, and that the system would have to be able to "recognize" which files applied to which document. A mechanism to save the files to a floppy and to load them to another computer system was also required (if the user was to be shielded from having to know himself which files went with which document).

**Tool Selection**

Tool selection is one of the most critical decisions to be made during the knowledge-based system development. The selection of an appropriate knowledge-based system development tool can greatly simplify the implementation process, just as "trying to fit the problem to the tool" can complicate implementation.

Several factors influenced which tools were considered for use. The requirements for a PC-based system, multiple rule bases, a built-in word processor interface, a built-in database, and a well defined mechanism for interface with the user were given the highest consideration. Based on these needs, the knowledge-based system development tool Guru [7] was chosen.

**IMPLEMENTATION**

The prototype uses three rule bases; a rule base for developing paragraph 1, a rule for developing paragraph 3, and a rule base for determining the appropriate NDI category. The system also utilizes over 50 procedural files, and text files, and over 90 screen forms are defined within the system to facilitate interfacing with the user. A database is used to force consistency within the document, to eliminate the need for asking the same question more than once, and to keep the user from having to develop all paragraphs in a single session. Priorities were assigned to each rule to ensure that, within each paragraph, text was written in the correct order.

The design and use of the data base within the prototype system was one of the most critical aspects of the prototype implementation. As mentioned previously, a method had to be developed to enforce consistency throughout an acquisition strategy. In the prototype system, a Guru data base is used for this purpose.

Each record holds information on a variable used in the decision process; the information consists of the name of the variable, its current value, the source of the value (user or system deduced), the reason for the value (user input or "facts" that led the system to deduce it), the when paragraphs in the acquisition strategy use the variable.

This record allows the system to "remember" that a value for a variable has already been determined. It also allows the system to remember what paragraphs are affected by the value of the variable and that if the variable is changed during the edit of any of these paragraphs, the new value must be propagated through the reasoning process of the other.

Each variable in the data base represents a piece of information that is required for the development of paragraph 1 or paragraph 3, or both. Some of the values are determined directly by user answers to questions, some are deduced from user answers, and some can take on default values if the user defers to the system. A total of 47 different variables are used by the prototype to generate paragraphs 1 and 3. The data base records are available to all three rule bases and to all of the procedural files.

In addition to providing a mechanism for consistency enforcement, the data base had an additional benefit. By using the data base to store all facts needed in the reasoning process, the information used to "solve" the problem is explicitly defined within the system. Having all the necessary information explicitly defined (and thus clearly represented for the prototype developer) made debugging, refining, and enhancing the system during the implementation process much easier.

**Illustrative Example**

The overall design of the prototype system, shown in Figure 2, can best be illustrated by an example. Suppose the Army decided a commercially available truck would best meet some vehicle acquisition strategy requirement. That being the case, the procurement should be handled as a NDI procurement. Further suppose the person tasked with developing the acquisition strategy had the prototype system available for use.
After choosing the DEVELOP AN ACQUISITION STRATEGY option, the user would be presented with a menu of the 13 paragraphs. Suppose the user picked paragraph 1 as the paragraph to develop first; at this point a consultation with the paragraph 1 rule base would be invoked. One of the first things the rule base would need to know while providing "assistance" to the user is the category of NDI procurement to be used for the particular item. Since the assistant has no knowledge of the NDI category, the paragraph 1 rule base would invoke a consultation with the NDI rule base to determine the NDI category. For this example, suppose either the user told the system or the system decided the procurement was category A (strictly off-the-shelf). Control would then go back to the paragraph 1 rule base and the "knowledge" in this rule base would be used to ask the user various questions and produce the appropriate text for paragraph 1 based on the answers.

As information is gained, either by the user answers or by deduction by the assistant (based on user answers), the information is stored in a database that is accessible by all parts of the prototype system.

After paragraph 1 is done, suppose that the user then chooses to develop paragraph 3. At this point, a consultation with the paragraph 3 rule base is invoked. It just so happens that one of the first things that this rule base needs to know is the NDI category for the procurement. But now the assistant already "knows" this information because it is in the database (even if the computer has been powered down since paragraph 1 was developed). Therefore, the user is not asked any questions about the NDI category, nor is a consultation with the NDI rule base necessary. The same thing applies to all the other information gained during the development of paragraph 1; the system already knows those things and does not have to ask (or deduce) them again. Only deductions and questions are necessary for "new" information.

The database has done two things: first, it eliminated the need for redundant questioning and deduction. More importantly, it eliminated the possibility for inconsistencies within the document (which could happen if the user answered NDI category A in paragraph 1 and then NDI category B or C when asked again in paragraph 3).

The database also comes into play. As soon as paragraph 1 is re-developed with the new NDI category (with no new questions asked since the system already "knows" all the information), all paragraphs that also use the NDI category to make decisions (in this case, paragraph 3) are automatically re-developed utilizing the new NDI category. In this manner, the database again eliminates the possibility of introducing inconsistencies within the document.

Other options provided by the assistant include the ability to continue work on a partially developed document (allowing a document to be developed at several sittings instead of one), to print, view, or delete an acquisition strategy, and to save an acquisition strategy to (or load one from) a diskette.

**EVALUATION**

The prototype was evaluated in two stages. The pre-evaluation was done when 85% of the prototype was completed (all the knowledge was implemented but not all of the prototype features were finished). The various aspects of the system were reviewed with the experts and appropriate changes were made based on their input. The changes fell into two categories: 1) changes in how some information was gained from the user were made, and 2) capabilities were added to the knowledge bases to make the text for the paragraphs more complete.

The formal evaluation was done by having one of the experts take two previously developed acquisition strategies and re-develop paragraphs 1 and 3 with the prototype. The "system" developed paragraphs were then compared to the "human" prepared paragraphs and evaluated for completeness, consistency, and correctness. The formal evaluation was very positive; in fact, no changes to any aspects of the prototype system were suggested.

The expandability of the prototype was given a positive evaluation based on the design of the system and the experience with expanding the system to make the changes and additions suggested during the pre-evaluation. The maintainability of the system was also judged to be positive based on the system design; that is, the multiple rule bases divided along logical boundaries, the procedural code split into logical modules, and
CONCLUSIONS

The prototype knowledge-based system developed in this research demonstrated the effectiveness of applying knowledge-based system techniques to the problem of developing acquisition strategies. By utilizing knowledge-based system development techniques and a knowledge-based system development tool, a useful prototype was developed in a relatively short period of time to help solve a problem that defies easy definition.

The prototype system also provided a foundation for the contracting of a complete system to assist in developing the entire acquisition strategy document. The DSMC has awarded a contract for the development of such a system, and work under the contract has begun. The prototype not only served to provide the contractor with a foundation of knowledge and a framework of features, but was also used as a valuable demonstration aid within BELVOIR to educate potential users and experts.

The concept of using a data base to provide simple but effective enforcement of consistency throughout a document was also illustrated by the prototype system. As previously discussed, consistency among the paragraphs of an acquisition strategy is absolutely essential. The use of a data base, to which the knowledge-based system has access, proved to be an excellent method of providing a mechanism to enforce the required consistency. The use of the data base also served to explicitly define within the system all facts needed throughout the reasoning process. Additionally, the data base design process forced the system developer to specifically address the facts that were being used, thus providing a more structured understanding of the information required, reasoning used, and knowledge represented within the system.

Continued efforts in the area of knowledge-based system "assistants" planned by the Defense Systems Management College has the potential to provide enormous benefits to individuals and organizations required to develop program management documents. As knowledge-based systems continue to be applied more and more to management applications, the additional potential exists to provide the same benefits to many other areas both within, and outside of, program management applications.

References