

Knowledge Management Using an Expert System Written in SAS®

Anthony M. Dymond, Dymond and Associates, LLC, Concord, CA

ABSTRACT

Expert systems are a branch of artificial intelligence that encode the knowledge and performance of a subject matter expert. An expert system provides a powerful approach to managing and applying knowledge to the business. The paradigm here is to encapsulate knowledge in an automated "expert" who can utilize this knowledge for the company's benefit.

An expert system can shield a company from employees leaving with critical information. It can help make knowledge available throughout the enterprise and help solve the problem of "knowing what we know." It can implement business rules to bridge the gap between written policy and practical application. It also allows importing automated subject matter experts when needed.

This paper describes an expert system written in SAS and reviews some of the major design considerations. Several knowledge bases are demonstrated to show how an expert's knowledge is encapsulated in a knowledge base and applied at run-time.

INTRODUCTION

Knowledge as an Asset

Companies have spent time and money discovering important information about themselves, their customers, and their competition. They have invested in what SAS calls "The Power to Know." Once this knowledge becomes available, it has to be leveraged to affect the corporate bottom line. In a sense, data warehousing and data mining are "mid-game" activities. They help to make knowledge available. Knowledge management, where knowledge is applied to the company's business activities to produce tangible results, is the "end-game." This knowledge is more than an accumulation of facts. It is also the rules and procedures needed to make decisions.

One example is Customer Relationship Management (CRM), which continuously acquires information about customers and their behaviors. Knowledge is analyzed and used to sell products and services, often by tailoring offerings to individual customers. Knowledge can also be embedded in

intelligent interfaces, allowing customers to be guided through personalized product exploration and customization.

It is also important to make some of this information available to stakeholders such as suppliers. This can improve supplier relationships by controlling prices, inventories, and deliveries. In fact, there may be many localized areas both within and outside the company where information needs to be packaged and presented.

There are many ways companies can utilize knowledge. Knowledge can be summarized in reports or procedures. It can be stored and utilized through the expertise of employees. It can also be captured in a computer program (Figure 1).

Each of these approaches presents problems. Reports are filed and forgotten. Employees leave and take the company's knowledge with them. Procedures can be changed without adequate documentation. Computer programs are often incomprehensible and inherently lack flexibility.

It would be advantageous if this knowledge could be stored in an automated system in a more flexible and understandable form than conventional programming. Some of the attempts to do this are referred to as "business process management" or "business rules." These rules are interpretable by computer applications and directly drive computation. They are also understandable by nonprogrammers. In the ideal case, business units themselves can mostly maintain these rules without IT support. The business units become empowered and are more responsive and adaptable. Efficiency improves since there is less overhead spent on complex code maintenance and documentation.

The next step after automating business rules might be to capture the knowledge and behaviors of a subject matter expert and place these into an automated system. This is referred to as an "expert system." Interacting with the expert system provides an experience similar to interacting with the human expert. This approximates the human expert being continuously available. This intelligent agent then reflects the best expertise the company has in a given subject area. This can offer a significant competitive advantage when dealing with customers, suppliers, or other business issues. Automated experts can be made available

throughout the company as well as to suppliers and customers. Additional expertise can be brought into the company by acquiring expert systems from external sources.

Knowledge today is viewed as one of a company's most valued assets (O'Dell and Grayson, 1998; Stewart, 1997). It must be tended and enhanced like any other asset. Knowledge management is more than storing and exploring data. It requires finding, deploying, and applying both facts and rules to obtain competitive advantage. Embedding it in intelligent systems can significantly leverage knowledge.

Current Enterprise Computing

The current enterprise computing environment is not particularly supportive of knowledge management. Applications are comprised of focused commercial packages, locally developed programs, and large application suites covering entire areas such as finance or manufacturing. Many of these applications, whether in-house or commercial, are basically procedural and provide a set of predetermined interfaces to an underlying database. They tend to be oriented to batch processes and are often inflexible.

Isolated legacy systems continue to hide both data and procedures behind a wall of incompatibility. Corporate mergers have complicated the situation with a mix of software and hardware that is difficult for IT to manage and for the business units to utilize. Many of these applications do not integrate well at the corporate level.

At the same time, the size and complexity of projects are increasing. Development teams charged with these projects can have multiple subject matter experts and may include members from foreign offices who struggle with language and cultural barriers. Traditional programming tools such as system development life cycle design and procedural code are proving to be inadequate.

Emerging Enterprise Computing

Corporations are beginning to look at their applications from an enterprise-wide perspective, rather than depending on a piecemeal approach. They seek integrated systems that support data interchange and common standards. They recognize that software needs to be flexible to meet changing business requirements. Software should also provide both event-driven and batch-oriented features.

There is a shift to object-oriented programming for large or complex projects, especially projects that need to be addressed by multiple task groups. Object-oriented techniques effectively decompose a subject or process into modular components. These object modules can be designed to be compatible with applications across the enterprise. They improve reliability and maintainability. They support reuse at both the object and method level.

Separate programs within the objects are contained in labeled blocks of code called methods. Object-oriented systems have other valuable features such as abstraction and encapsulation that isolate object internals from the rest of the system. Of particular importance is method inheritance, which provides a set of rules for sharing methods throughout a system or the entire enterprise.

When built to be compatible with industry standards such as CORBA, the object-oriented model provides the means to execute methods in diverse applications throughout the enterprise and to support information interchange regardless of the underlying data structures.

Use of object-oriented systems sets the stage for intelligent systems. Intelligent applications drive visual and conversational interfaces with embedded intelligence, allowing the application to function as an assistant. Intelligent agents provide a semiautonomous application that acts on behalf of the user.

INTELLIGENT SYSTEMS

Procedural programming separates program code from data. Particular attention is placed on developing a complex control structure. The control structure code is intertwined with the knowledge and rules contained in the program, making programs difficult to maintain or modify. Procedural programming focuses on how things should be done, rather than what should be done.

Object-oriented systems decompose the problem to modules called objects that combine both programs and data. But object-oriented systems have little to say about control structures and provide a somewhat static view of a problem.

Intelligent systems today start with the object-oriented model but add a separate control structure. The advantages of the object-oriented model are kept, but knowledge and expertise are now separated from application control. The systems become easier to understand and faster to construct (Jackson, 1999).

A separate control structure encourages the use of a declarative approach that focuses on what should be done, rather than emphasizing the details of program control. Stating rules and other knowledge to the system accomplishes the programming. The dynamic run-time interaction between objects is determined by rules written into methods and by the separate control structure. Rules are a natural way to describe a decision process. We can say:

"IF the applicant has a good credit history AND the applicant has the resources to repay a loan THEN it is OK to give the applicant a loan."

When the control structure can be represented graphically, then the system becomes exposed and comprehensible to staff who are not programmers. By using templates and graphical tools, staff in the business units can maintain these systems in the field with minimal IT support. Intelligent systems now begin to deliver integration at the corporate level with adaptability and responsiveness in the business units.

Building an Intelligent System in SAS

An intelligent system using an object design model with a separate control system has been developed in Base SAS and SAS Component Language (SCL) (Dymond, 2000, 2001). This can be used as an independent agent or embedded into applications where it is transparent to an end user.

It uses a conventional expert system design with an inference engine, multiple knowledge bases, and a database (Figure 2). The inference engine provides both development and run-time tools. It supports the run-time environment, including the separate control structure.

The knowledge bases serve a role similar to separate applications and are executed under the control of the inference engine. A subject matter expert's knowledge is captured in the knowledge base as objects in a graph. The graph provides a mechanism to navigate between the objects, and it is implemented in this intelligent system as a search tree explored by well-known tree search algorithms such as depth-first search.

Declaring objects into the tree and requesting one of the tree search algorithms provides the designer with program control. The control system contained in the inference engine then has the information necessary to automatically provide control services. Some control fine tuning can be done through code in the methods.

The tree is a dynamic system object and system methods are available that allow detailed programmer-level control should this be necessary. Methods are available that allow one tree to load and execute another tree, providing the capability of chaining between multiple subject matter experts. The tree is also extensible at run-time, forming the infrastructure for an intelligent agent that can learn and remember.

The memory-resident database contains the data from a current session. This is a "blackboard database" that is read-write accessible by all objects in the intelligent system. Data is stored in an object-attribute-value triplet, plus an index column. The database is also a system object that can be accessed by its own system methods.

The tree, objects, methods, and database can all be saved to disk and reloaded at any time during a run. This captures the complete state of the system at any time and supports "what if" studies.

MakeLoan Knowledge Base

The MakeLoan knowledge base demonstrates the intelligent system used to implement business rules. Rules describe how a company processes a loan application. The system encapsulates the company's procedures for making loans and also embeds the best practices of a company expert.

The object decomposition and placement into a tree is shown in Figure 3. The decomposition is based on functions, and the tree is arranged based on functional dependencies and on the order objects should be encountered. The tree should be processed with a depth-first search algorithm that transverses the tree from top to bottom, and from left to right. With this in mind, it is possible to "read" the tree as follows:

Begin by gathering some information about the customer, such as demographics and the nature of the loan (StartConsultation). Decide if any loan will be made (MakeLoan). This decision will depend on the customer having satisfactory credit (CreditOK) and adequate resources to repay the loan (ResourcesOK). The resource requirement can be met by the customer having adequate income to repay the loan (IncomeOK) or by pledging sufficient collateral against the loan (CollateralOK). Once a decision has been made to make a loan, the amount to be loaned can be determined (LoanAmount). Finally, a report is prepared summarizing this loan application (Report).

The object ResourcesOK could be coded using Base SAS with system methods to read and write the blackboard database, or it could be coded by using one of the method proof templates available in the system. A code fragment from a method proof template would be:

```
*PROOF=ResourcesOKProof;
*;
*TARGET={ResourcesOK,resourcesScore};
*;
*ORCLS={IncomeOK,incomeScore}      eq 'true' ;
*      {CollateralOK,collateralScore} eq 'true' ;
*ENDCLS;
*;
*ENDPROOF;
```

which states that there is a method proof template named “ResourcesOKProof” that will attempt to find a value (either “true,” “false,” or “unknown”) for the attribute resourcesScore in the object ResourcesOK. It will accomplish this using a logical OR test on the attribute incomeScore from the object IncomeOK, and the attribute collateralScore from the object CollateralOK.

One further programming issue should be mentioned. If the applicant’s credit is not OK, then MakeLoan,makeLoanScore = “false” and the object MakeLoan and its descendents do not need to be explored further. This is communicated to the search engine by setting a search control flag attribute in the object MakeLoan. This flag would be set when the makeLoanScore attribute was set to “false,” and the system would know not to further examine the object MakeLoan or its descendents.

This system can be rapidly constructed, easily understood, and maintained at least in part by nonprogrammers in the business units. The objects and their control are defined by declaring them into the tree. Within the objects, much of the programming can be replaced by filling out method proof templates. Other system methods make it easy to open pop-up windows to gather information from users.

Mammal Knowledge Base

The Mammal knowledge base contains the knowledge and behaviors of a subject matter expert who knows how to classify mammals. It demonstrates an intelligent system built by successively subclassing from a general root node (Mammal) to specific examples of animals in the leaf nodes. The objects and tree shown in Figure 4 essentially reflect a hierarchical decomposition of mammalian animals. The sense of the tree is as follows:

Suppose that one encounters a medium-sized brown animal with a marsupial pouch, hopping on large hind legs and carrying the Australian flag. To identify the animal, a search begins at the root node of the Mammal knowledge base. The search proceeds through successively lower levels in the tree and seeks increasingly detailed information. As information is gathered, some branches of the tree are found to be unproductive. Control flags set from the methods code block further unnecessary search in these branches. The focus of the search would soon be guided to the object Kangaroo.

Several points can be made about the tree and search engine:

- These searches are very efficient. Only the most relevant questions are asked in the order needed to solve the problem.
- The process is scalable and works well for trees with thousands of nodes.
- The tree can be entered at any node. For example, if it is already known that the animal is a marsupial, then a focused search can begin at the node MarsupialMammal.
- It is easy to steer the search to prove a goal node in the body of the tree or to allow it to work down to individual leaf nodes. For example, the goal might have been just to determine if the animal is a mammal without trying to identify the specific type.
- “Goal driven” or “backward chaining” search proceeds from the top toward the bottom of the tree. It is also possible to have “data driven” or “forward chaining” where data is placed into objects at the bottom of the tree and the consequences allowed to flow upward. For example, it could have been initially stated that the animal is a Kangaroo and this fact then allowed to prove MarsupialMammal and Mammal.

MonitorOK Knowledge Base

The MonitorOK knowledge base encapsulates an expert’s knowledge from a service call center supporting monitors. The main point of presenting this knowledge base is to show that its design and construction is essentially the same as the Mammal knowledge base. Both progress by subclassifying from a general root node to specific instances at the leaf nodes.

In fact, MonitorOK and Mammal are essentially the same as MakeLoan in that they all demonstrate search within a bound knowledge space. In all cases, the answer is known to exist somewhere within the knowledge base, and the problem is to find it expeditiously.

The underlying commonality of seemingly diverse problems is one of the most interesting and important facts uncovered when remapping these problems to the knowledge representation shown here. They all belong to a family of similar problems, and a generic tool, the intelligent system, can address them all.

CONCLUSION

Gathering, retaining, transferring, and applying knowledge are of great importance for a company seeking competitive advantage. Knowledge management is significantly leveraged by intelligent systems, which can capture and implement business rules and the expertise of subject matter experts.

The intelligent system presented here utilizes an object-oriented programming model and adds a separate control structure. This allows problems to be entered in a declarative manner by using object decomposition and then placing the objects as nodes in a search tree. Methods in the objects can easily communicate with the search engine to provide a robust and simple control paradigm. System methods supporting template proofs and user query windows round out a system that is understandable and supportable by nonprogrammers in the business units.

A number of problems that seem to be very diverse are in fact remappable as search within a bound knowledge space. The intelligent system then becomes a generic tool that can be used to address these problems.

Building the system around object-oriented techniques allows it to be integratable and to share information with the emerging enterprise computing infrastructure.

REFERENCES

Dymond, A.M. (2000), "An Object-Oriented Expert System for the SAS Environment," *Proceedings of the Eighth Annual Western Users of SAS Software Regional Users Group Conference*, 8, 454-458.

Dymond, A.M. (2001), "Telling Aardvarks from Zebras with an Expert System Written in SAS," *Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference*, Long Beach, California. (CDROM paper 135-26).

Jackson, P. (1999), *Introduction to Expert Systems*, Harlow, England: Addison Wesley Longman Limited.

O'Dell, C.S., and C.J. Grayson (1998), *If Only We Knew What We Know: The Transfer of Internal Knowledge and Best Practice*, Free Press.

Stewart, T.A. (1997), *Intellectual Capital: The New Wealth of Organizations*, Doubleday.

ACKNOWLEDGEMENTS

Automated Consultant is a trademark of Dymond and Associates, LLC.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Anthony M. Dymond, Ph.D.
Dymond and Associates, LLC
4417 Catalpa Ct.
Concord, CA 94521

(925) 798-0129
(925) 680-1312 (FAX)
amdymond@dymondassoc.com

A SAS Institute Inc. Quality Partner

Mapping Tasks and Knowledge in a Company

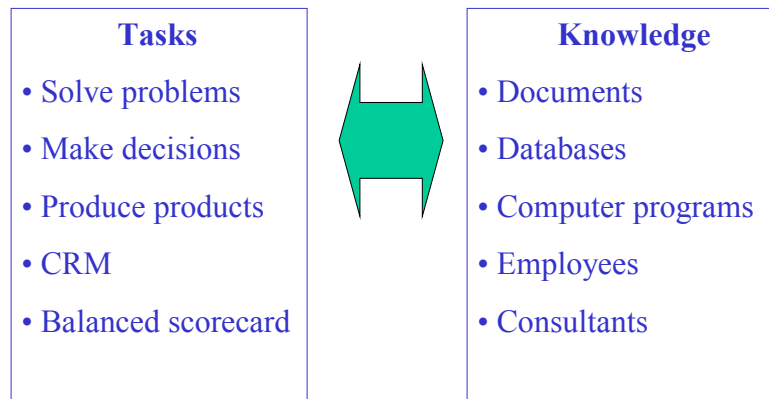


Figure 1: Knowledge used to solve enterprise tasks is stored in many forms.



Figure 2: The intelligent system is composed of an inference engine, multiple knowledge bases, and a database

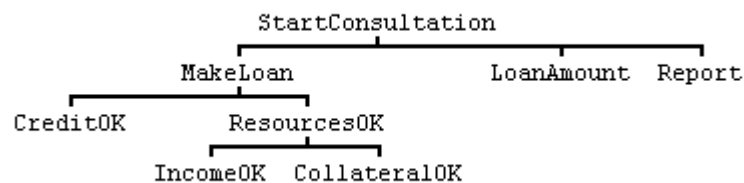


Figure 3: Tree for the MakeLoan knowledge base.

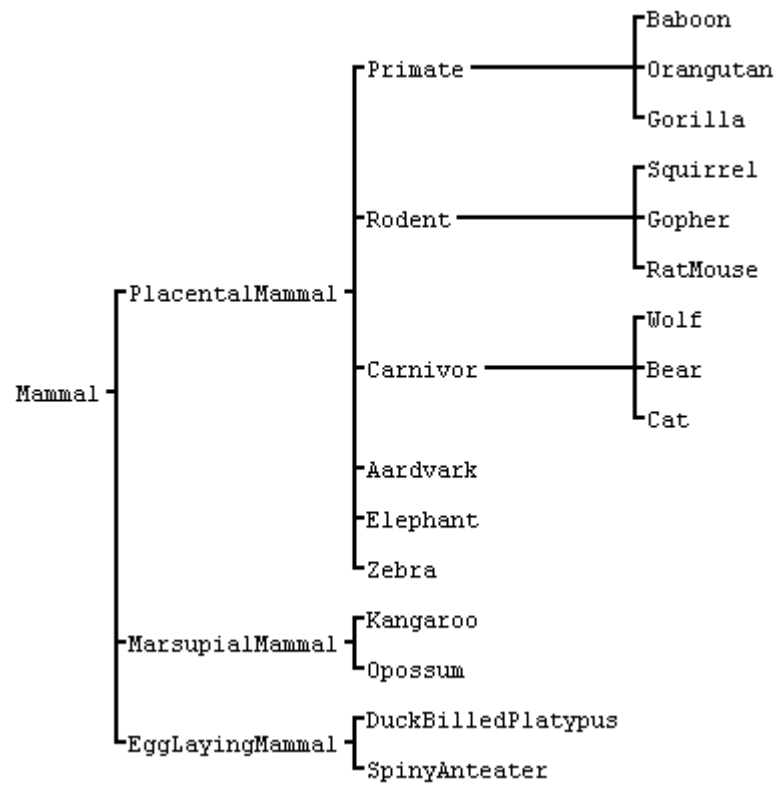


Figure 4: Tree for the Mammal knowledge base. (The tree is shown sideways with the root at the left.)

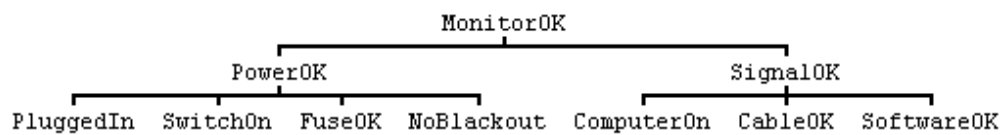


Figure 5: Tree for a service call center supporting monitors.