FRAMEWORK FOR PROTOTYPING EXPERT SYSTEMS FOR FINANCIAL APPLICATIONS

Jacob Y. Friedman and Atul Jain

Decision Support Group, Management Consulting Services

Coopers & Lybrand, New York, NY 10020

ABSTRACT

Analysis of difficulties in transferring expert systems technology into the financial industry applications suggests that speed-up of the prototyping phase can significantly reduce the cost and length of the entire development process. We suggest a prototype concept that is generic for certain types of financial applications and can serve as both a catalyst for the knowledge engineering process and a laboratory for knowledge gathering, validation and maintenance. We developed software tools to provide a framework for rapid prototyping by financial professionals with basic computer training.

I INTRODUCTION

High costs, a long development cycle and the lack of experienced knowledge engineers are the main obstacles to the broad commercialization of expert system technology, particularly in the "bottom line" oriented financial industry. To understand the difficulties in building an expert system (ES), one has to consider all stages of ES development.

1. Knowledge Engineering: analysis of problem domain; selection of knowledge representation, computational approach, and user interface; building of the first prototype.

2. Knowledge Acquisition: selection and training of experts; knowledge gathering and validation using the prototype; refinement of the first prototype or building a second one; prototype field testing.

3. Delivery System Development: preparation of specification and design documentation; software coding; system integration; testing.

4. System Deployment: preparation of user documentation; user training; installation and support; knowledge base maintenance.

The first two stages are referred to as ES prototyping and the last two as implementation. ES implementation is a process similar to conventional DP system development. Therefore, well formulated and accepted principles and methods of software engineering can be adopted. The prototyping phase, on the other hand, often becomes a major research project, consumes more than 70% of the time and resources allocated to ES development and requires highly skilled knowledge engineers (KE).

Transformation of the ES prototyping process from a research effort to an engineering task is an important factor in reducing the length of ES development and associated costs.

II PURPOSE OF PROTOTYPE

A. Knowledge Engineering Catalyst

The difficulties in ES development start from the very first interaction between the KE and the domain expert (DE). Typical scenario: the KE is a computer scientist with knowledge of AI principles and techniques but a limited understanding of the problem domain; the DE is very knowledgeable in the application domain but has difficulty comprehending ES technology. Knowledge engineering sessions become unstructured crash courses in both AI and application. They are complicated by differences in the individuals' backgrounds, ways of thinking and communicating, and inability to visualize abstract rules, notions, and ideas.

The Knowledge Engineering process functions much smoother when an ES for a similar problem is available and can be used as a means for exchanging knowledge, demonstrating principles and testing ideas, and sometimes as a first prototype. Even though an existing ES is not a perfect solution for the new problem and can not be used for complete knowledge acquisition (KA), it helps to keep the knowledge engineering process focused and significantly speeds up ES development.

If a similar ES is not available, the building of the first prototype in the very early stages of ES development is extremely important. Such a prototype, even with a limited useful life, can serve as a catalyst for the Knowledge Engineering process by provinding the DE with structure and concrete objects to describe his reasoning process to the KE.

B. Knowledge Base Laboratory

Some developers believe that it is important to have a prototype as close to the final system as possible at very early stages of a project. Such a prototype is supposed to speed up development of the delivery system.

In our opinion, at this stage of development it is difficult to select the optimal design of the final system. Furthermore, we do not believe that the final system intended for production is the most suitable for KA.

The system used during the KA process needs certain capabilities that the final system will not, and these features are essential tools for the most effective development of the Knowledge Base (KB). For example, the prototype should provide the DE easy access to intermediate results to permit the necessary refinements in the problem-solving process. It also needs a greater degree of user control over steps of inference to allow testing of the KB by the DE.

Testing the KB is often complicated by the fact that processing a test case requires the ES to contain knowledge about all aspects of the problem. When in the prototype version the problem is broken down into a series of relatively independent steps (at least as a first aproximation), and human intervention is provided for steps missing in the KB, the entire KA process becomes better structured, earlier tests of completed sections of the KB can be performed, and several DEs can work on different sections simultaneously.

Some advanced ES software tools provide facilities to examine and modify the KB during the test run of a final system. However, these are typically designed more for the use of the KE in debugging the system code. The DE needs an interface that is specifically tailored for a particular application and does not significantly slow him down when routinely processing real cases.

C. Knowledge Validation Device

A good prototype can be carried to the phase of KB validation by independent experts who have otherwise not participated in the process of KA.

In prototype field testing, the problem of acceptance by new users becomes very important. Users have a problem accepting a system whose operations are not completely transparent. Because of the inherent limitations of the initial KB, the user does not entrust real cases to the system and performs parallel testing only with selected cases. This slows the KB validation and limits the variety of test cases. Another limitation of on-line testing of the prototype is that it rarely handles multiple cases. In financial applications, not all input information is readily available at one time, and the user often switches back and forth between cases.

To allow on-line processing, the system used for the KB field test should be absolutely transparent, imitate the problem-solving process used by human experts and provide the user with full control of this process. These features are not important and often are undesirable for the production system, but are essential for the prototype used in KA.

D. Knowledge Base Maintenance System

The process of KB maintenance is analogous to the process of KA and requires thorough testing of new pieces of knowledge. A good prototype can be used for testing of new knowledge even if it does not exactly correspond to the delivery system. The speedup of KB testing using debugging facilities of the prototype easily compensate for possible delays caused by the conversion from the format used in the prototype to the format of the delivery system.

E. Methodology Carrier

The major problem in ES development is the lack of an established methodology and the absence of examples for different types of problems in various applications.

By continuing to use a prototype through the entire process of ES development, and later KB maintenance, we extend its role of knowledge engineering catalyst to one of a carrier of ES development methodology. A good prototype built on the base of similar problems can be a valuable guide during ES development, especially for KEs who are not very experienced.

III PROTOTYPE FEATURES

For a prototype to be extended to cover all aspects of knowledge engineering (acquisition, testing, validation and maintenance of the KB), it must incorporate a series of features not usually associated with a prototype.

The following is a description of these features embedded in the example of a generic ES prototype used by Coopers & Lybrand to develop ES for its financial clients. The prototype is intended for decision making problems that comprise a large segment of the potential ES applications in finance.

A. Domain Tailored Data Representation

From our experience, the process of Knowledge Engineering proceeds faster if the knowledge representation in the first version of the ES closely follows the representation used by human experts in solving the problem. This makes it easier to extract rules from experts and test their validity on real cases.

Decision-making processes in financial applications are typically accompanied by and organized around massive paperwork: input information collected in forms, summary data, results of intermediate analyses, underlying assumptions and the final report. Every step of the problem solution is documented on paper. If several people participate in the process, they use paper media (folders with forms and reports) as a main channel of communication.

In the prototype, data is organized on the screen in a way that imitates the paper forms used by the human experts. The on-screen form is composed of several fields that are placeholders for data (Fig. 1). Each field has a label and a value, number or string, and is mouse-sensitive. Values of "display only" fields or space allocated for them are underlined. Values of user modifiable fields are boxed. Users can enter or modify field values by clicking on the box and then, depending on the field type, enter values from the keyboard, cycling through a pre-specified list of values or selecting values from a pop-up menu.

A form can have a bar diagram to illustrate some of the numeric field values. Bars corresponding to modifiable fields are mouse-sensitive and provide another way of entering or overwriting field values by the user. Each form can have several "schedules" - instances of the same form for several instances of an object. For example, the form in Fig. 1 is scheduled by vessels and there is an instance of this form for each instance of a vessel. The schedule menu is used to select specific schedules or to create new ones.

B. Problem Decomposition

Decomposition of the problem is implemented both by grouping contextually similar data and by breaking decision process into a series of steps.

Forms in an application are organized in folders. Mouse clicking on a folder icon in the application window (Fig. 2) is

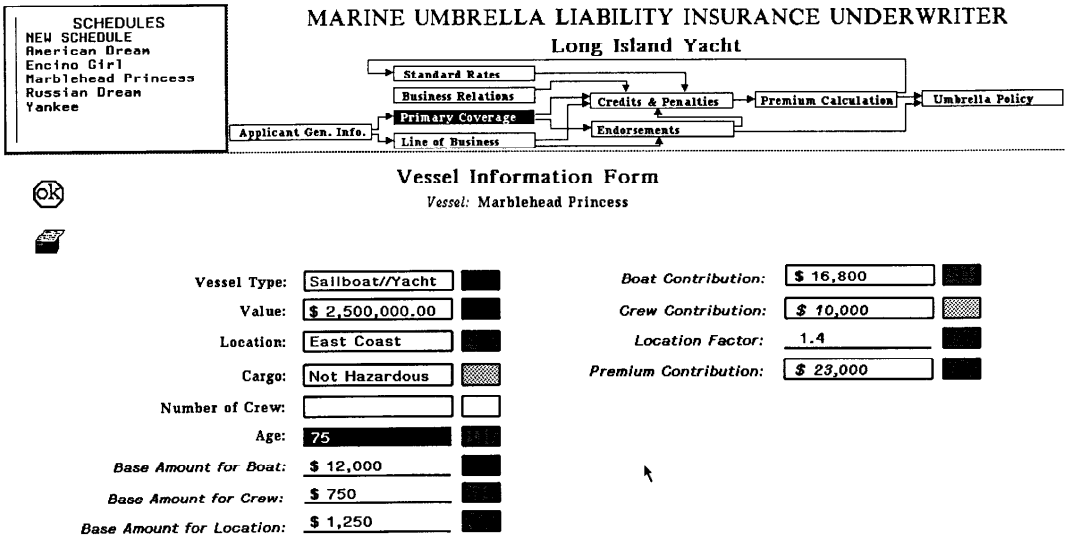

Figure 1. Interface Screen with Form Window, Context Window, and Schedule Menu

used to select a folder. Clicking on the form icon in the folder window (Fig. 3) invokes display of the corresponding form. Any folder can also be selected from the context window on the form or folder screen (Fig. 1).

The way forms are grouped in folders is intended to accomplish two objectives: to have forms with context-related fields in the same folder and, more important, to have forms containing the results of a major step of the problem solution in the same folder. Thus, the solution process can be represented as a step-by-step propagation of information (field values) from one group of folders to another. Visually, it is represented in the application window (Fig. 2) and context window (Fig. 1 as lines with arrowheads (connectors) connecting two folders, the origination folder and the termination folder.

C. KB Debugging Facilities

Such features as explanation facilities and indicators of data availability, source and quality are important in production ES, but they are even more important in prototype during the KA and KB validation.

An explanation related to specific data can be obtained by clicking on a field. The user can invoke an explanation of why its value is needed to solve the problem, or a justification of the field value was inferred.

In addition to a label and a value, each field has a value source and a value quality associated with it. The shading intensity of a box displayed on the form next to the value (Fig. 1) is used to indicate one of the three levels of value quality: "guessed", "probable" or "certain". The quality of data represented by a bar diagram is indicated by the shading of the bar.

The quality of the field value is internally represented by a number from 0 to 1. It is calculated using one of the methods described in [1,2,3] depending on the rule specification. The entire range from 0 to 1 is broken in three sub-ranges corresponding to "guessed", "probable" and "certain" for external presentation.

Each form and folder icon has a data meter (Fig. 2 and Fig. 3) that indicates how many fields are filled with values and the quality composition of these values. Total height of the meter's shaded area indicates weighted number of filled fields, while the nonshaded area represents unfilled fields. A weight factor assigned to each field is used to determine the field contribution in the data meter read-out. The shaded areas are broken into three areas, with different shading indicating how many filled fields are "guessed", "probable", and "certain".

The validation mechanism is used to allow the user to easily trace changes in field values resulting from inference.

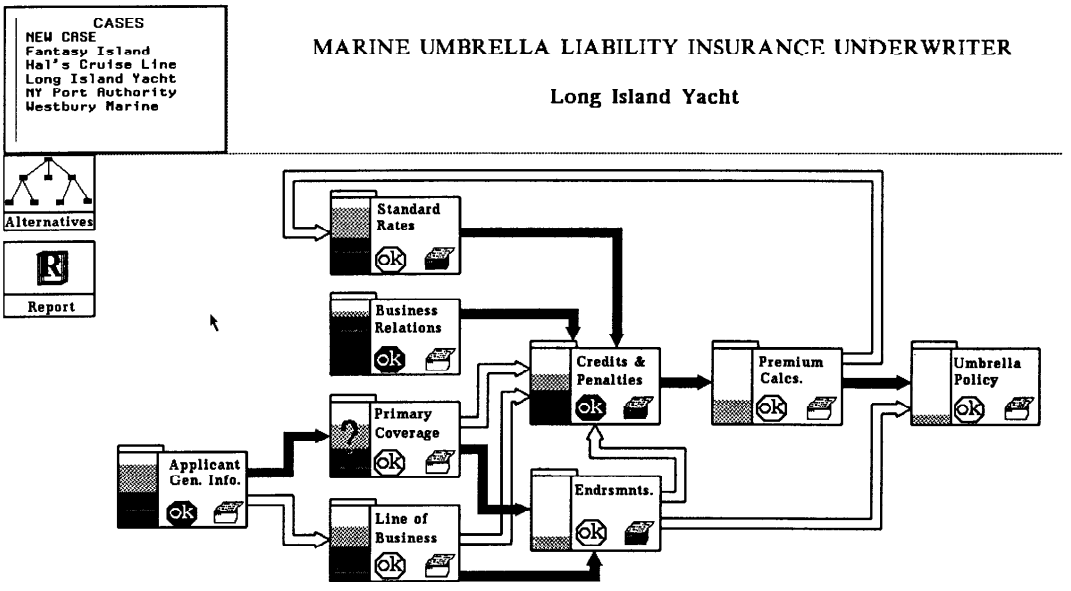

Figure 2. Interface Screen with Application Window and Case Menu

When a field value is asserted by a rule, it is not verified and is displayed in reverse video. The user can verify all fields of a schedule, a form or an entire folder by clicking on the corresponding verify icon. If as a result of later inference the field value is changed, the new value is displayed in reverse video. By clicking on it, the user can examine the old value and even restore it. The user can detect changes at any level of field hierarchy: schedule, form or folder. The corresponding verify icon is not filled if the folder, form or schedule contains nonverified values. The icon is filled if all field values are verified.

Hierarchy of folders, forms and schedules provides the user with easy access to all input data and intermediate and final results. The user has the option to enter data that is supposed to be inferred or to overwrite already inferred data. If the value of a field that is intended to be inferred was actually entered by the user, it is displayed in italic.

D. Control and Status of Solution Process

Two components are important for step-by-step execution of a decision process:

1. Clear visual representation of the problem structure, steps in the solution and their interaction.

2. User control of inference, with an indication of the solution steps that provide enough information to infer new data.

A flow chart composed of folder icons and connectors on the application window (Fig. 2) combined with folder data meters provides the visual representation mentioned above.

Each connector has a group of production rules associated with it. These interfolder rules, in their premises, refer to field values from the origination folder and assert the field values of the termination folder. Each rule can be associated with several connectors terminating at the same folder. A connector is shaded in the application window if at least one rule associated with it is ready to fire. By clicking on the connector, the user can view a list of actions that will occur if these rules are fired.

The user has to click on the data meter of a folder icon to allow firing of the inter-folder rules. A tree with the activated folder as a root, connectors as branches and other folders as nodes illustrates the scheme used to control interfolder inference. The generation of connectors most removed from the root is activated first, then the next generation, and so forth, ending with a generation of connectors terminating at the activated folder. Generations with no rules ready to fire are skipped. When there are no more rules to fire at the root level, inter-folder inference is completed. Connectors that are activated at each step through the tree are highlighted in the application window.

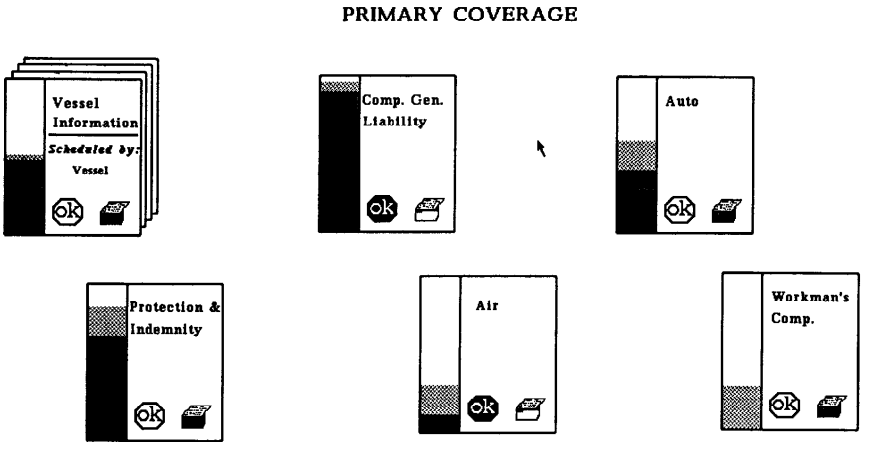

Figure 3. Folder Window

The data meters of folders are also updated. They give the user a visual indication of the information propagation through the system. The user can choose to walk through the solution process by sequentially invoking one generation of connectors at a time and reviewing intermediate results after each step. On the other hand, by clicking on the folder with the final results, the user can accomplish the entire process in one shot.

In addition to user-controlled inter-folder rules, the system has intra-folder rules that fire automatically as soon as their premises are satisfied. Intra-folder rules are local to a particular folder and usually represent a "lower level" of knowledge. These rules are invoked as result of user entry of field values or inter-folder inference.

E. Analysis of Alternative

The ability to analyze a variety of alternatives in the process of selecting an optimal solution is one of the most important features of an ES. To help develop and test rules for selection of optional alternatives, the prototype should allow the user to generate and analyze arbitrary alternatives. The alternative window (Fig. 4) is used to generate new alternatives and is used to switch between alternatives. Alternatives are shown as nodes of the alternative tree; the node corresponding to the current alternative is highlighted.

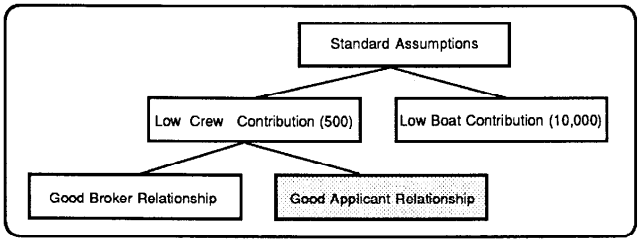

Figure 4. Alternative Window

A new alternative is generated by clicking on the node that becomes a parent alternative. The parent alternative becomes frozen and cannot be modified. The user can change any data in the new alternatives and other alternatives without children.

An alternative can be selected by clicking on the corresponding leaf of the alternative tree. The user can prune the alternative tree using the mouse. In this way he can temporarily or permanently disregard (poison) certain branches or select a subtree as the only promising set of alternatives (believe). The alternative comparison window is used to compare alternative values of any field under different assumptions. It is invoked by clicking on the field value and has the value of the field corresponding to each alternative displayed in each node. Alternative selection and pruning is also possible in the alternative comparison window.

F. Case Management System

To be able to use the prototype for on-line validation of KB, it should handle multiple cases. The case menu in the application window (Fig. 2) is used to switch between cases and to initiate new ones. When a different case is selected, the current case, including the field values, the statuses, and justifications, is stored on disk and can be restored later for continuation.

IV PROTOTYPE DEVELOPMENT

A. Difficulties of Prototype Development

Building a prototype, especially one with the battery of features described above, entails a serious programming effort, even with the help of commercially available ES software tools. These tools provide knowledge representation language, inference engine, and graphic and development utilities, but they lack a very important feature successfully used in PC spreadsheets and data base packages: a complete application structure with built-in user interface and data handling facilities. As a result, ES developers spend a lot of time programming specific features and designing the prototype structure in addition to developing the KB.

A majority of ES software tools (at least those with enough representational and computational power to solve financial applications) require extensive training and a strong programming background from a KE. That limits selection of KEs to programmers and precludes utilization of the large group of financial specialists who are successfully using PC software package.

Even experienced KEs have a problem properly incorporating the wide variety of existing ES techniques. They would benefit from a software-development environment with a library of modules tailored to specific problems, enabling them to select those most relevant to the particular application.

B. Framework for Financial Applications Prototyping

The following is a description of a software development environment (framework) for rapid building of prototypes for financial applications that have features presented in the previous section.

The framework was implemented on a Symbolics 3640 computer, by Symbolics, Inc. using both LISP and Automatic Reasoning tool (ART) from Inference, Inc. The Symbolics computer provides the necessary computational power, high resolution graphics, rich software library and development environment. ART provides the knowledge representation language, rule compiler, inference engine, and important features such as logical dependencies and a viewpoint mechanism for exploration of hypothetical alternatives.

The framework consists of three modules: interactive application editor, preprocessor and run module. The interactive application editor is an icon editor that provides the KE with an easy way (using the mouse) to specify the entire application structure: fields, forms, schedules, folders and connectors. As a result this module produces a file with the application specification.

The preprocessor module uses the file with the application specification and the file with rules specification to generate application-specific ART schemata, facts and rules, and to store them as an application file.

The run module is a set of ART rules and LISP functions generic for all applications. When loaded together with an application file, it generates the prototype for a given application that supports all generic features described in the previous section.

V. RESULTS

This framework has been used for prototyping three financial expert systems: Risk Manager, Marine Umbrella Liability Insurance Underwriter, and Merger and Acquisition Analyst Assistant. In all three cases, the framework structure was efficient and flexible enough to produce a working prototype in less than four weeks. All interactions with domain experts were performed by KEs with a background in the problem domain after short training in use of the framework. The main tasks of the software engineers were to check rule syntax and consistency and to select the method for data quality propagation.

The experts' acceptance of the "forms" concept was very encouraging. As a result, knowledge engineering sessions were very focused from the beginning. In a matter of days, the experts learned to use the prototype and started processing real cases to generate, refine and test rules. Several experts were used as sources of rules for each application. The clear and visible structure of the prototype made it easy to achieve consistency in rules derived from different experts. Use of multiple experts was also aided by the fact that the inference process was broken into several steps.

VI CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

Successful use of the framework in prototyping several ESs for financial ES applications confirmed our statement that the proposed financial applications structure and tools for prototype development with built-in features facilitating gathering, validation and testing of KB can significantly reduce the cost and time involved in ES development.

Even in the somewhat more specialized field of financial applications there are different types of problems requiring different approaches: questionnaire-driven systems, modeling in time, etc. To accommodate these problems, different frameworks should be created or new features should be added to the existing one.

ACKNOWLEDGMENTS

We would like to thank Dr. David Shpilberg from Coopers & Lybrand for his support and advice on the project and help in shaping this paper.

REFERENCES

(1) Shortfliffe, E.H. and B.G. Buchanan. "A Model of Inexact Reasoning in Medicine", Mathematical Biosciences, Vol. 23, 1975, 355-356.

(2) Duda, R.O., P.E. Hart and Nils Nilsson "Subjective Bayesian Methods for Rule-Based Inference Systems" AFIPS Conf. Proc., National Computer Conf., Vol 45, New York, 1976, 1075-1082.

(3) L.A. Zadeh, "Syllogistic Reasoning as a Basis for Combination of Evidence in Expert Systems", Proc. IJCAI-85, Los Angeles, 1985, 417-419.