ReValuator—An Expert System Approach to Actuarial Valuations

S. Meltzer and D. Sriram

Expert system technology has now matured so that task-oriented business programs can be rapidly prototyped, developed, coded, and deployed on desktop and laptop personal computers. This rapid development and deployment is especially true when the task is well defined, and the target user has little knowledge in the specified domain. This chapter sketches the successful implementation of an actuarial program designed to assist a nonactuary in detailed actuarial analysis.

During the past few years, sophisticated microprocessors have been placed in desktop and laptop computers. As a result, these computer platforms can now accommodate expert systems that once ran only on specialized computers or mainframes. The expertise captured in the knowledge-based system can be distributed on these relatively inexpensive computer platforms to locations where no expert is available for consultation.

This chapter describes the development of an expert system called the ReValuator and includes the following sections: (1) Problem Definition, (2) Description of the AI Technology Used, (3) Functional Overview, (4) Architectural Overview, (5) Example of Model in Operation, (6) Innovations, (7) Criteria for Successful Deployment, (8) Nature and Estimate of Payoff, and (9) System Evolution.

Problem Definition

The U.S. Treasury/Internal Revenue Service (UST/IRS) and the Department of Labor monitor compliance with the Employee Retirement Income Security Act (Erisa) enacted by the U.S. Congress to protect the pension rights of the rank-and-file employee. Every corporation or company that is in business for profit—as opposed to an institution that is nonprofit, such as a school or church—must comply with Erisa if it wishes to maintain its tax-deductible pension eligibility. Erisa requires that an actuarial certification, called a Schedule B, be submitted annually to UST/IRS for each pension plan. Millions of these actuarial certifications are submitted, and each one of these certifications is signed by one of 5000 federally certified actuaries.

The few certified actuaries who work for UST/IRS do not routinely review Schedule B submissions. This task is the responsibility of UST/IRS field staff members who have little if any formal actuarial training.

Like every discipline, the actuarial domain has mathematical formulas and nomenclature that are unique to the trade. The average actuarial student spends five to seven years after college graduation before achieving certification in an actuarial specialty. Therefore, field staff members are at a tremendous disadvantage when conducting an actuarial audit of a pension plan. The need to keep staff members up to date about additional legislative amendments to Erisa compounds this difficulty. Needless to say, many private actuaries are aware of this inability to monitor compliance and use their actuarial training to boost pension tax deductions to more than is actuarially needed.

Because there are not enough federal pension actuaries to validate the Schedule B submissions, the IRS deputy commissioner and the assistant commissioner of employee plans decided a stand-alone system that captures the knowledge of an expert pension actuary in an inexpensive computer would aid Erisa compliance.

Description of the AI Technology Used

If successful, the project had to prototype, develop, test, and deploy an actuarial system within six months. The system developed (1) consults with a nonactuary in choosing reasonable actuarial assumptions, (2) assists a nonactuary in understanding the implication of the assumptions chosen, and (3) produces an actuarial evaluation based on the assumptions chosen.

The rapid prototype and development was accomplished by integrating expert system technology with C language interface calls and a database. The expert system captures and reproduces the knowledge of a pension actuary. The C language calls speed procedural processing, specifically, the calculation of actuarial factors. The database stores case data.

An expert system shell was used to rapidly prototype an application and convince management that a system could be developed and tested within a specified time frame. Rapid prototyping helped to sell management on what the system could do and to establish continuing management support for the project—a key element in the successful development of any computer program. The ability to convince management cannot be understated. The decision to use an expert system shell (Nexpert by Neuron Data) was not made by IRS management; their concern was, understandably, rapid deployment. The choice was made by the programmer (Seth Meltzer). (This point is made because of the time required to learn a shell, time that is not used writing source code in some native language.)

Some managers might not be concerned with how a project is accomplished or the time savings that can be earned in later projects as a result of the time invested learning new tools; witness the relatively slow acceptance of structured techniques and object-oriented programming in the computer programming community. Therefore, the decision to use an underlying development shell can undermine the success of a project if the startup time on a shell leaves little time to construct a demonstrational system. An expert system shell was chosen for two reasons:

First, in the initial development of the IRS prototype, the actuarial experts were not certain what input-output was required. The shell enabled the programmer to rapidly code and demonstrate to the expert any new idea proposed. For example, in the initial development, the expert wanted the system to include questions concerning retirement age. Although retirement age is not usually challenged by IRS, in some occupations, the normal retirement age is significantly less then 65 (for example, professional athletes). A low retirement age can significantly increase the amount of taxable deduction. Therefore, a pension auditor might need guidelines to decide when and how a low retirement age should be challenged. Working with the shell, the programmer was able to rapidly include a normal retirement consultation in the prototype system. This inclusion would take significantly more time and effort using a native language than an expert system environment. In addition, because the expert decided to exclude the retirement age guidelines from the final system, needless programming was avoided.

Second, all available actuarial computer source code was in a native language (for example, Fortran, Basic, Cobol) written without the use of structured techniques. The code was judged not usable.

Functional Overview

The user requests an actuarial valuation consultation by inputting a pension plan name. ReValuator makes an external call that loads the pension plan records from a database. After the database file is read, the records concerning the pension plan are loaded into ReValuator. Objects are made by instantiating appropriate classes.

For example, an object is made for each type of pension plan trust fund asset read in from the database. A type of asset is, for example, cash, bank account, bond, or stock. The asset objects are instantiated from the class "asset," which has the following slots: amount, investment income, percentage return, and confidence factor of percentage return.

The system consults with the user about the reasonableness of each actuarial assumption (Fikes 1985). The user chooses to accept the private actuary's assumptions (that have been read in from the database) or decides to suggest a new actuarial assumption or assumptions. Actuarial factors based on the new actuarial assumptions are computed by external C language routines, and the actuarial valuation is computed. The program allows the user to make what-if changes to the assumptions. If the assumptions are changed, the system recalculates the actuarial valuation and again allows the user to change assumptions. If there are no changes, the session is done.

Architectural Overview

The current model, depicted in figure 1, underwent several major structural reorganizations. The knowledge base in the current model is divided into three layers (Nii and Feigenbaum 1982). The first two layers, strategy control and activator control, manage the flow of the consultation. The last layer, specialist control, supervises the resolution of the consultation subgoals, for example, the determination of an expected fund earnings rate.

The strategy control module (SCM) is a single rule set that calls the activator control module (ACM). SCM is always resident in memory. ACMs call the specialist control modules (SpCMs). ACM determines

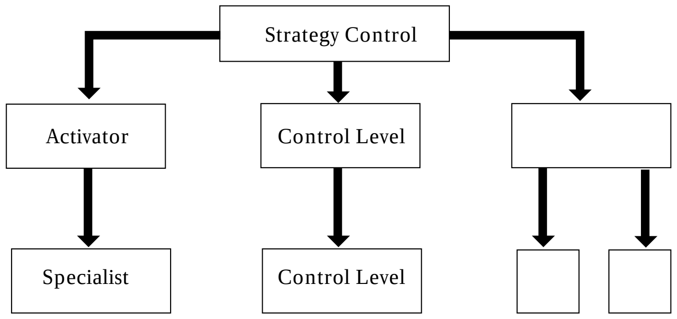

Figure 1. Architectural Overview.

SpCMs to load and the priority in which SpCMs are executed. Only one ACM is resident in memory at any instant in time. SCM unloads the current ACM before loading another ACM. SpCMs bind user variables, such as expected fund earnings rate or life expectations. More than one SpCM can be in memory at any instant in time. ACM unloads all current memory-resident SpCMs before returning control to SCM.

For example, suppose a consultation is requested to determine an acceptable expected fund earnings rate for the pension fund. SCM loads the interest rate ACM. The interest rate ACM chooses from asset split, employee salary scale, number of years till last current employee retires, and so on. The expected fund earnings rate is computed as a result of binding each applicable subgoal.

Adding an additional permanent or testing subgoal is now accomplished by adding a separate SpCM. The additional SpCM rule set can, if necessary, hide any effects from other rule sets. If a new major structure is needed, the programmer can add another ACM.

Example of Model in Operation

The following abridged example illustrates how flow control is transferred from rule set to rule set and within rule sets. The example assumes (1) the user is consulting with the system about the applicability of the assumed retirement age submitted by the private actuary, (2) the slot value ST.current (strategy level) is set to App_Ara (approval assumed retirement age), and (3) there are no current goal-driven searches.

Because there are no more goals to resolve, the system looks for any changes in slot values located in rule antecedents. It finds new slot-value changes in ST.current, and the forward chainer is triggered. The current activator level rule set is unloaded, and extraneous rules are removed from memory. The next activator rule set is loaded. Note the following example:

```
If ST.current =
ApprovalAssumedRetAge
Then ACT.H_app_ara Is TRUE
and Unload knowledge base
ActAssumptionAsset.kb Load knowledge base
ActApprovalAssumRetAge.kb
```

A context-switching mechanism is used to create a goal-driven search for the slot ACT.H_app_ara_done in the activator rule set.

To resolve ACT.H_app_ara_done, the user is asked to select one of three choices: (1) accept the actuarial assumed retirement age (ara) read in from the database, (2) input a new ara and override the assumption read in from the database, or (3) consult the system for a decision before selecting choices 1 or 2.

Choices 1 and 2 immediately resolve the goal (ACT.H_app_ara_done) without loading a specialist level rule set.

Choice 3 loads a specialist level rule set and establishes an additional goal (HAra.ara_done). To resolve HAra.ara_done, the specialist level rule set asks additional questions about the validity of ara.

Two goals are now in the search space: ACT.H_app_ara_done and HAra.ara_done. The former goal drives the activator level rule set; no rules in the specialist level rule set mention it. The latter goal drives the specialist rule set. The last rule to fire in the specialist rule set resolves the goal and passes control back to the activator rule set. The activator rule set unloads the specialist rule set from memory and passes control back to the strategy rule set. The strategy level rule set unloads the current activator rule set and continues the inference process.

Innovations

The system made it possible for UST/IRS to enforce tax laws that were previously unenforceable (hopefully, none of the readers were affected). Additionally, the enforcement is uniform across the nation; agents in Washington use same program as agents in Florida. Because of rapid prototyping, the programmer could quickly respond to comments and rapidly enhance the additional program-installing features requested by the actuarial expert or the end users. The payoff from this effort has been enormous; the specifics are discussed in the following sections.

Criteria for Successful Deployment

The first version of the system was deployed in January 1989. To train as many field agents as quickly as possible, UST/IRS national office actuaries held classes around the country several times a month from January through March 1989. By their use or nonuse of the program, the UST/IRS agents determined if the application was successfully deployed.

The typical agent has computer phobia; so the first thing the programmer advised the instructors to say is, "It's the programmer's fault if the program crashes the machine or if the program is not fully understandable." The goal was for a tightly controlled, user-friendly system. To date, the program has been deployed in every UST/IRS key district and is being used by almost 1000 UST/IRS pension agents. Instruction to other UST/IRS personnel is ongoing.

Once trained, the agents put their new tool to the test, and they have been pleased with the results. They feel it is their program because their suggestions for improving the user interface and the program in general are quickly integrated into revised versions. Since January 1989, the program has undergone several extensive revisions, including one complete rewrite with another rewrite currently being planned. Senior UST/IRS managers have given their support to continuing development and have gone into the field to encourage program use.

Nature and Estimate of Payoff

The program is expected to generate at least $100,000,000 in previously uncollectible revenue. Some optimistic estimates are much greater than $1,000,000,000. The estimate is based on a statistical sample of pension tax forms filed for the years 1985 to 1988 in which the tax-deductible disallowance on a single-person pension plan ranged from tens of thousands of dollars to hundreds of thousands of dollars. There are thousands of these pension plans. An additional payoff of the program is that the public will become aware that these tax laws are now enforceable and will be encouraged to voluntarily comply with existing and new pension plan tax code.

System Evolution

As previously mentioned, the first prototype was developed using an expert system shell. The prototype was refined over three months using the standard expert system methodology (Minsky 1975): (1) consult with an actuarial expert, (2) extract the expert knowledge, (3) engineer the knowledge, and (4) create a prototype reflecting the expert's knowledge and repeat steps 1–4. This process continued until the actuarial expert was satisfied that the prototype captured the procedures used in valuing a pension plan.

The expert system shell operates under Microsoft (MS) Windows. MS Windows is a time-slicing, multiple-process operating environment treating each window process as a virtual machine. This approach permits multiple windows to be simultaneously displayed on the screen and is useful in prototyping and development. For example, during program execution, a rule-firing audit transcript file and other useful debugging windows can simultaneously be on the screen.

Because the system was deployed on a laptop computer without a hard disk and limited random-access memory, the executable program, compiled with the expert system shell run under MS Windows (version 2) was judged too large and too slow for a final deployed system. The final deployed program was rewritten in C and assembly language. This approach cut the executable code size in half and improved the response time. For example, the development environment used multiple data segments that required each data variable to specify its address as a segment and offset. The rewritten program contained the data in a single data segment, which also improved response time because there are fewer address bus machine clocks. Some shells (including Nexpert) offer an MS-DOS run-time module, but time constraints did not permit this module to be tested.

Acknowledgments

The initial research for this project was done for a Master's thesis under the brutal but always exactly appropriate guidance of Sri Ram at the Massachusetts Institute of Technology (MIT) during the summer of 1988.

The IRS management team consisted of Deputy Commissioner John Wedick, Assistant Commissioner Robert Brauer, and AI Lab Director Ted Rogers; the IRS actuarial expert is Ken Black. Paul Brown and Shaoul Ezekiel of the Center of Advanced Engineering Studies at MIT deserve the author's gratitude and thanks.

References

Fikes, R., and Kehler, T. 1985. The Role of Frame-Based Representation in Reasoning. *Communications of the ACM* 28(9): 904–920.

Minsky, M. 1975. A Framework for Representing Knowledge. In *The Psychology of Computer Vision*, ed. P. Winston, 211–277. New York: McGraw-Hill.

Nii, H. P., and Feigenbaum, E. A. 1982. Signal-to-Signal Transformation: HASP / SIAP Case Study. *AI Magazine* 3(1): 23–35.