CRESUS: An Integrated Expert System for Cash Management

Pete Shell, Carnegie Mellon University, Gonzalo Quiroga, Juan A. Hernandez-Rubio, Eduardo Encinas, Union Frnosa S.A., Jose Garcia, and Javier Berbiela, Norsistemas Consultores S.A.

CRESUS is a unique application of state-of-the-art expert system technology to the real-world financial problem of cash management. By automating the work of company treasurers, it saves substantial amounts of both money and time every day. Real-world test cases show that CRESUS performs better and much faster than the human expert: In minutes, it generates a combination of operations that efficiently balances all banking accounts in a 15-day period. It uses user-friendly window technology to control the human-machine dialogue and has been integrated into the work environment of a major electric company in Spain. Written in Common Lisp using UNIX workstations, it was jointly developed by Union Fenosa, Carnegie Mellon University, and Norsistemas Consultores.

The Cash-Management Problem

This chapter describes our success with developing CRESUS, a cash-man-

agement expert system. In the CRESUS project, we explored the uncharted territory of automating cash-management decision making. First, we explain the motivation for turning to an expert system solution and then describe the technology itself. Finally, we explain the many benefits that CRESUS has brought us.

The cash-management decision-making process is a complex task that requires not only a highly skilled treasurer but also the availability of appropriate software tools. Each day, the treasurer has to make the following logistical decisions: how to coordinate the company's forecasted collections and payments through banking accounts; whether to borrow money or invest surplus money and how to do each; how many fund movements to make and between which accounts and with what amounts; and for each of these decisions, which financial instruments to use.

The goal is to minimize the combined cost of these operations: The credit lines used should be as inexpensive as possible; the balances of the banking accounts are preferably zero (to avoid overdrawn accounts and idle funds); and the commissions of instruments used in payments, collections, and fund movements should be the minimum.

The treasurer's job is made more complex by many factors:

First is the nature of the financial market, where the diversity, complexity, and continual change of its instruments make it difficult to choose among the many bank offers.

Second is the large number of banking accounts, collections, and payments.

Third is the interaction among the operations that causes each decision to affect the others. For example, the financial status on one day depends on the previous day's actions.

Fourth is the inherent uncertainty of collections and payments that causes continuous changes in the situation.

Fifth is the necessity of immediate response time that makes it difficult to plan the decisions.

Because of the complexities of this environment and the limitations of human memory, the treasurer must greatly simplify the problem by using several heuristics that ignore much of the interaction among the operations. When the cash flows are large, the cost to the company can be significant.

Union Fenosa attempted to solve this problem through linear programming techniques but failed. They solved the task of structuring and organizing the information but not the searching and evaluation that is inherent to optimization processes. The problem was that the search space is so large that it was impossible to simulate all the alternatives within a reasonable response time.

System Objectives

CRESUS meets all these needs through the use of AI techniques, automatically finding the set of decisions that result in the minimum cost after simulating and evaluating those possible decision combinations with the highest likelihood of obtaining this minimum cost.

CRESUS is useful not only for the daily management of the treasury problems but also as a powerful tool that allows treasurers to enhance their knowledge about the problem and infer the ideal conditions of their financial instruments. It is the only tool that we know of that automates the cash-management process.

The main objectives of this tool are the following:

First is to optimize the global cash flow by automatically generating optimal solutions. To achieve this goal, the system simulates a set of possible solutions to the current period, evaluating the cost of each and choosing the best. This approach offers an objective criterion on which the treasurers base their decisions and allows them to analyze the elements that most affect the cost.

Second is to plan future periods, taking into account the influence of each day's decision on the remaining days of the period. This approach allows the user to avoid undesired situations in the short term.

CRESUS also provides these features:

First, it presents the user with a simultaneous vision of all relevant information, globally and detailed, through X-11 WINDOW technology.

Second, it provides continuous checking for user errors and inefficiencies. When inefficiencies are found, corrective actions are suggested to the user and, if desired, automatically performed by the system.

Third, report generation allows the user to make effective management decisions and analyze their quality in terms of cost.

Fourth, it provides flexibility in managing company cash flows, with no limits on volume of information.

Fifth, it provides multilingual interaction with the user. We currently provide Spanish and English modes, and it is easy to add new languages through message files.

With all these functions, the treasurers have a powerful tool able to satisfy their needs and assure a high degree of quality in decisions that have to be made to solve the cash-management problem.

System Components

The expert system combines expert knowledge, such as constraints on how much money to borrow, pay back, or invest, cost evaluation, and

operator generation, with a hybrid, multilevel *K*-best search strategy. This expert system is tightly integrated with the manual window-based cash-management component. The main modules are as follows:

Global improvements: Automatic search and execution of the best solution of the period

Flagger: Rule-based detection of user inefficiencies and suggestions for improvements

Simulator: Simulation of the user's transactions' user and global cost evaluation at any point of the period

User interface and data manager: Window-based screening of the account balances, individually and globally

Each of these functions is detailed in this section, with an emphasis on the expert system modules.

The Data Manager

The *data manager* maintains the frame hierarchy that represents the problem domain (figure 1). It provides an intelligent database (Pylyshyn 1985) by integrating a traditional database with AI techniques. This module uses a frame-based language called PARMENIDES (Shell and Carbonell 1988) to define data types and instances and allow intelligent manipulation of these instances. The data manager also guarantees data validity and consistency each time data are modified and provides file input-output (I-O) operations.

PARMENIDES also provides inheritance through a class hierarchy, demons, and user-defined relations. The data manager uses these if-added demons to interface with the other modules. When a frame is modified, it can call the simulator to infer new frame values if other frames depend on the changed value. It also automatically calls the user interface to update relevant windows to redisplay the changed data. Finally, the data manager provides these frames to the rule-based flagger module and the global improver module (see discussion later).

PARMENIDES was chosen because it not only provides powerful AI tools such as inheritance and procedural attachment, it also allows fast access to slots in frame instances, a feature that many frame languages don't have. Furthermore, it is well integrated with the FRULEKIT (Shell and Carbonell 1986) production system used in the FLAGGER module.

The Simulator

The *simulator* simulates the effects of user financial operations so that the treasurer can more efficiently plan the company's transactions. Each time that a transaction is added, modified, or deleted, the simulator computes the effects of this change and tells the data manager to

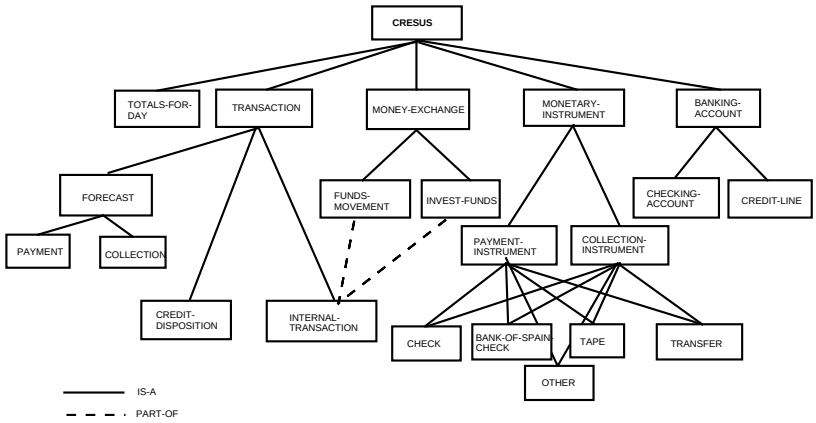

Figure 1. CRESUS Class Hierarchy (simplified).

update the relevant frames. The simulator is also responsible for evaluating the cost of the cash-management decisions.

The User Interface

The *user interface* makes it easy for the user to interact with each part of CRESUS. It is based on the X WINDOWS technology (Jones 1989), which allows the user to both make decisions and see their effect. It also facilitates a high level of integration between all the other modules.

Through this interface, the user can control and view the expert system's performance, perform data maintenance, make all types of transactions, request reports, move between different days of the simulated period, and record or play back a work session.

In figure 2, sample user interface windows are shown. These managing windows show information about abnormal banking accounts, credit-line dispositions, payment and collection forecasts, and fund movements.

The Flagger

The FLAGGER locally detects and corrects inefficiencies in the user's decisions, thereby finding local optima. By *local*, we mean that it restricts its attention to only one or two user operations at a time and only considers the current day. This approach is in contrast to the global searcher (see next subsection). The flagger is based on production rules (Forgy and McDermott 1977) that encode expert knowledge in the form of data-driven if-then pairs. The rules allow the system to be used by users with a lower degree of expertise or for training purposes.

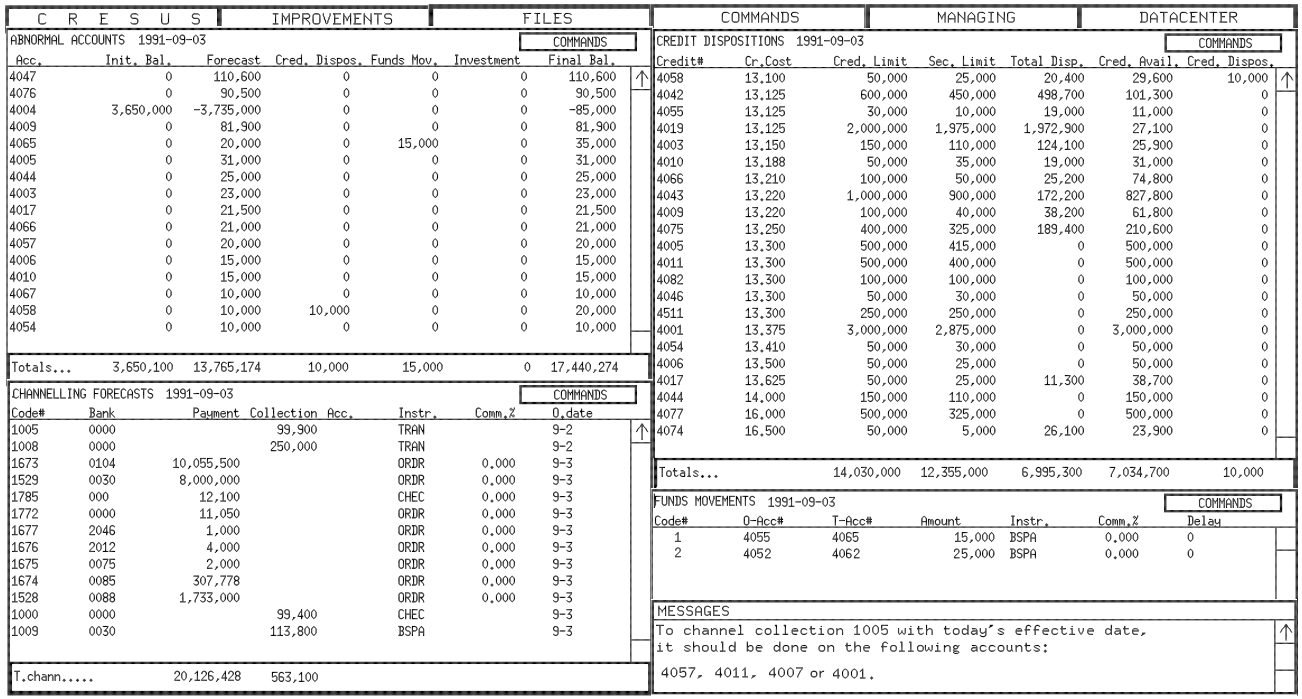

Figure 2. CRESUS User Interface Managing Windows.

Each time the user makes a decision that could be optimized, one or more rules flag the user with a message that explains the inefficiency and suggests an alternate operation. If this suggestion is accepted by the user, it automatically replaces the original operation, and all windows and data structures are updated.

The action part of the production rules encodes two things: a method to explain the inefficiency to the user and a method to correct the inefficiency if the user decides to correct it.

The flagger uses the FRULEKIT production system language, a modern Common Lisp implementation of OPS5. FRULEKIT was chosen for the following reasons:

First, it is tightly integrated with the PARMENIDES frames that are used by the rest of the system. Unlike some other production systems that copy frames into their rule system, FRULEKIT matches directly on the frames themselves, and the right-hand actions apply to frames.

Second is the fact that FRULEKIT has an open architecture. It is possible to call Lisp functions on both the left-hand and right-hand sides. This structure allowed us to write our own notation for specifying what messages to present to the user and encoding knowledge about how to correct the inefficiencies.

Third is the reason maintenance facility. FRULEKIT allows the programmer to designate right-hand-side items as either beliefs or side-effects. *Side-effects* are actions that can't be taken back, whereas *beliefs* are actions that should be taken back whenever the left-hand side of the rule no longer matches. This shortcoming exists in many production systems. For example, when there is an inefficiency (for example, the user made a fund movement from account A to account B and then from B to C), a message is posted that the funds could have been moved directly from A to C. However, if this action is corrected by the user, then FRULEKIT automatically takes away the message.

The Global Searcher

This module represents the most sophisticated part of CRESUS. Instead of making only incremental improvements like the flagger, the *global searcher* constructs entirely new solutions from scratch. Thus, it attempts to find a more global optimum than the flagger. It is also global in the sense that it takes the entire period into account instead of focusing on only one day without regard to the others.

The searcher's solution comprises a set of decisions, or operators, that require the lowest global cost to balance the bank accounts. An operator can do any of the following actions: move money between two accounts, borrow or pay back money from a credit line, channel a col-

lection or payment from or to an account, and invest money.

Before the search, the user can establish certain constraints, such as disposing of a certain amount of money from a given credit line or making a concrete collection to a given account. These constraints are respected by the system even if they are not considered efficient (that is, the system understands that they might obey some other criteria of the treasurer's).

The search space is extremely large. Solutions typically consist of a sequence of about 50 operators, and at each step of the search, on the order of 100 operators can apply. For this reason, a heuristic search is necessary because it is possible to search only a small fraction of this space. Furthermore, by partitioning the search space into subspaces with relatively little interaction, the problem becomes more tractable (this observation was first put forth by Minsky [1963]). The interaction is usually handled by encoding expert knowledge. In CRESUS, these subspaces are as follows:

First is the daily search space versus the global search space. When CRESUS searches for a daily optimal solution, it doesn't consider the global implications of each operator during this search. The global implications are considered between daily searches (see later discussion).

Second is the space of operators for a specific operator type. Inside the daily search, CRESUS only considers operators of one type at a time (the specific operator types are described at the beginning of this section). Although there is interaction between certain types of operators, this interaction is handled by putting more expert knowledge into the evaluation and generation functions. For example, the evaluation function would prefer a fund movement that balances an expensive credit line over one that balances a less expensive credit line because it would avoid having to borrow from the expensive credit line.

The searcher uses a multilevel heuristic search based on the I-BEAM search algorithm originally developed for the HARPY speech-recognition system (Lowerre 1976; Newell 1978), as we describe later. The expert's knowledge about cost evaluation and operator generation is incorporated into the searcher. Parameters affecting the search were tuned based on the performance of the searcher on typical test cases.

The global searcher is composed of the following submodules:

Searcher: This top-level module performs the searching, using all the other modules.

Generator: Its function is to generate new operators that improve a given search state.

Evaluator: It evaluates the local and global cost of a set of operators. The *global cost* is defined as the cost associated with every operator that has been applied plus the estimation of the cost needed to complete the search.

Global data manager: This module is responsible for the initial data storage and the intermediate and final results.

The global searcher performs its work in two different stages: setup and search.

In setup, the search goals are set for each day of the period: whether to borrow, pay back, or invest money and how much. These goals depend on the total balances for each day. At the end of the search, the following must be accomplished: (1) all the accounts are balanced, (2) the global cost is minimum given the conditions preceding the search, and (3) the number of operators that define the global solution is minimized.

After the setup phase, the search begins. It is divided into two different levels: within-day search and between-days search. Both the within-day and between-days searching modules are based on the I-beam search algorithm. The *beam searcher* is a breadth-first searcher that considers a number of different partial solutions simultaneously. This approach yields higher-quality results than only considering the locally best partial solution at each step because local optima are not necessarily part of the optimum global solution. The greater the number of partial solutions that are considered, the better the chance that a more optimal solution will be found. However, it is infeasible to consider every possible solution because there are so many. Thus, there is a trade-off between a high-quality solution and a large search space and between a fast solution and a smaller search space.

During a within-day search, the searcher stores the *K*-best partial solutions that it has found so far on every level of the search (*K* is known as the beam width). It always chooses to further explore the *K*-best solutions that look the most efficient based on an evaluation of the state of this solution.

For example, in figure 3, *K* is set to 5. At the first level of search, the best states are the ones labeled with evaluations of 50, 45, 40, and 40. At level 2, the best states are those labeled 80, 75, 78, 72, and 73. Note that the states with evaluations of 65, 62, 55, and 63 are not retained. This process would continue until complete solutions are found. Note that the total number of nodes on each level does not grow after level 3 because the number of states to explore is limited to *K*.

During the between-days search, the searcher performs a beam search by combining the daily searches of each day. In other words, during the between-days–search phase, the best global solution is found based on the comparison of the cost associated with each daily solution.

Because the uncertainty grows as the searcher looks forward in time, the value of *K* decreases as the day being searched increases. With this

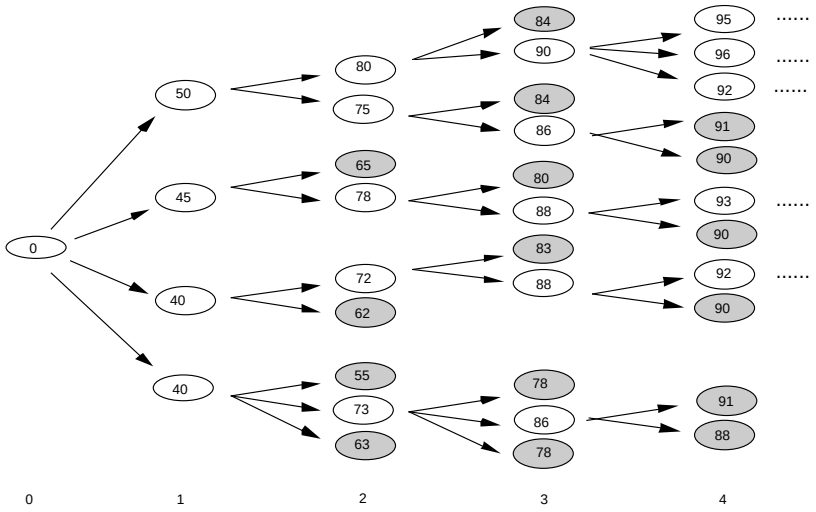

*Figure 3. Sample Within-Day Beam Search (*K *= 5). Shaded nodes are not explored any further.*

algorithm, even though a particular solution might look like the best one on day *d*, the searcher evaluates the strength of this solution globally. By combining solutions over more than one day, we are globally searching for the most efficient solutions.

CRESUS Benefits

CRESUS has benefited Union Fenosa in a number of ways:

First is the time savings in the daily management of the treasury. We estimate this savings to be 70 percent of the decision-making time and 30 percent of the treasurer's total work time.

Second is the improvement in the quality of the treasurer's work. By using a tool that simulates and optimizes a priori the cash-management decisions, the treasurer is free to spend more time analyzing and searching for the best banking conditions.

Third is the augmentation of knowledge about the problem and detection of points for improvement. Because of the comparison of optimal suggested solutions with those conditioned by company policies or the conditions of certain banks, those points where the costs are concentrated arise and imply an analysis process for its improvement.

Fourth is the economic savings. The quantitative savings realized by the CRESUS searcher is significant. We compared the cost of the expert's solution using CRESUS manually with the global searcher's solution over a two-week period in January 1992. The comparison was made by com-

puting the total cost of each solution: the cost of borrowing money, the fees associated with fund movements, and the fees for having overdrawn accounts minus the money earned by investing. The results show that Union Fenosa will save over a half million dollars a year:

	Week 1	Week 2	Projected annually
Manual use	$113,060	$34,360	$3,832,920
CRESUS searcher	$91,180	$28,740	$3,117,920
Difference (abs.)	$21,880	$5,620	$715,000
Difference (prc.)	19.35%	16.35%	18.65%

(The annual projections are computed by taking the average of the 2 weeks and extrapolating to a 52-week period.) The savings changes from week to week because a number of real-world conditions are always changing, such as the volume of the operations, the current conditions of the financial market, and the point at which the savings are measured. However, with these results, we estimate that CRESUS will pay for itself in one year.

Another benefit expected over time is the solution of the expert-substitution problem. By making it easier to extend and share the expert's knowledge with other people, CRESUS should allow us to smoothly handle times when the expert is away from the company.

Development

The application was developed starting in January 1989 and was finished by December 1990. One of Union Fenosa's main goals was to ensure that the technology would be transferred effectively to its personnel. Thus, a mixed team was formed, including personnel from both Union Fenosa and Carnegie Mellon University. The team initially included a project leader, a senior knowledge engineer, and two part-time programmers from Carnegie Mellon and a junior knowledge engineer and a domain expert from Union Fenosa.

Development Process

The development process was split into three phases:

Phase 1—Initial Prototype: During this phase, we defined the project plan, configured the development team, selected the hardware and software tools, and validated the conceptual design by implementing a prototype of the non-expert modules of CRESUS (the user interface, the data manager, and the simulator).

Phase 2—Advanced Prototype: In this phase, we scaled up the non-expert modules, designed the expert modules (flagger and global searcher), and implemented the flagger. Finally, we installed the prototype at the customer site so that the end user could validate it.

Phase 3—Final System: In the final phase, we implemented the global searcher, completed the integration of all modules, and extensively verified and validated the application.

Verification and Validation

At the end of each phase, we verified and validated the application. The initial prototype was validated by the domain expert as well as the senior manager, who was responsible for project support. At this time, we didn't pay much attention to verifying the prototype because we were mostly interested in proving the concept. After having finished the other phases, we spent about a month testing the entire application. We used several approaches. To test the simulator, we developed a record-playback option that allowed the domain expert to check for errors. To test the global searcher, we gave small test cases to test for correctness, ran it on several real-world data sets to test for quality, and wrote an auto-test program to run the searcher through several combinations of parameters.

Deployment

The application is currently being used by Union Fenosa, an electric company that during 1991 moved nearly $50 million (both collections and payments), with the number of bank operations approaching 125,000.

Deployment began in January 1991 and took 8 months of work for a team of 3 people, each dedicating 30 percent of his time to the project. The work covered everything from the study of the appropriate hardware platform to user training. Since September 1991, CRESUS has been used daily for treasury decision making in the company. Even though several people from the department are involved in the data-entry and report-making processes, the cash-management decision making is centralized by one person (the treasurer).

The first step in deploying the system was to completely study the data-collection process to feed the company data to and from the system. Because of the volume and the complexity of the study, we chose a decentralized data approach, where we built interfaces with other applications and enhanced the CRESUS data manager module to support

these interfaces.

The main CRESUS software can run on any UNIX workstation with X-11 WINDOWS and is currently integrated into the environment of Union Fenosa. It runs on a SUN SPARCSTATION that is connected to a MACINTOSH II CX, a DOS Compaq 386 PC, other personal computers (PCs), and a Laser Writer printer. The PCs are used for data entry and report generation. This hardware is connected through the TOPS local area network (LAN) developed by Sun Microsystems, which offers a user-friendly environment.

This structure lets us obtain automatically, through LAN, the ASCII files generated in other environments and built with different software and combine it with data the system needs for it to function. In addition, we took advantage of the MACINTOSH to design automatic reports that directly read the simulation and optimization results of CRESUS.

With respect to user training, we can distinguish two phases: learning how to use CRESUS and gaining user confidence in the system. The first part did not present any problems because the CRESUS user interface was designed to agree with the treasurer's philosophy and work mode. A treasurer without knowledge of computers could learn to manage CRESUS in one day.

The second part required running both CRESUS and the old system in parallel for about three months. Here, the principle problems were convincing the treasurer to take advantage of the full capabilities of CRESUS and accepting its particular proposals when they conflicted with the work habits derived from the limitations of the old system.

Maintenance

CRESUS is working now, but we are continuing to enhance it. Because of feedback from the users, during the deployment phase, we discovered improvements that were needed in the user interface (such as sorting and searching functions in the data windows), the knowledge base, and the global improver module that we didn't take into account in the development phase.

The improvements needed in the global improver module warrant further discussion. The domain of the CRESUS system is variable, so that while it is being used, we need to change not only the banking account conditions (usual in the daily performance) but also the different types of banking instruments in use. For this reason, CRESUS must facilitate the creation of new instruments with different behavior. In addition, we wanted a greater sophistication of instrument use and desired to extrapolate the Union Fenosa solution to the cash-management problem

to other companies. Thus, we were motivated to generalize our system further. This work is developing now, and we can already say that the knowledge base is general enough that it can be adapted to different situations.

The user interface is also general enough that it allows the user to interactively change all relevant conditions. The global searcher has been made more powerful, but some of its knowledge needs to be made more declarative so that it will be easier to modify it in the future. Finally, the maintenance and modification team, experts, and knowledge engineers are, for the most part, the same staff that developed CRESUS, which makes it easier to make the needed system changes.

Conclusions

Because this work represents the first time that knowledge engineering technology was applied to the complex problem of cash management, we had no way of knowing whether our efforts would be successful. Our experience shows, however, that it is possible to effectively automate the treasurer's decision making to save significant time and money. Other cash-management tools that we looked at claim to automate the cash-management task but in reality don't perform decision making. Instead, they typically use spreadsheet programs to simulate the effect of the treasurer's operations—equivalent to the manual part of CRESUS. We found that the bulk of the benefit is when the global searcher is utilized to automatically find less costly solutions.

CRESUS was developed for a specific utility company in Spain, but we are currently attempting to market the system to other companies in Spain. With some more work, we also hope to generalize it enough to make it useful in the rest of Europe and the United States.

Acknowledgments

We would like to thank Michael Mauldin for his useful comments on a draft of this chapter and the following people for their important contributions to the project: Daniel Borrajo, Jaime Carbonell, Mike Kanaley, Todd Kaufmann, Michael Mauldin, Chris Nuuja, and Jose Prieto.

References

Forgy, C. L., and McDermott, J. 1977. OPS, a Domain-Independent Production System Language. In Proceedings of the Fourth International Joint Conference on Artificial Intelligence, 933—939. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Jones, O. 1989. *Introduction to the X WINDOW System*. Englewood Cliffs, N.J.: Prentice Hall.

Lowerre, B. 1976. The HARPY Speech-Recognition System. Ph.D. thesis, School of Computer Science, Carnegie Mellon Univ.

Minsky, M. 1963. *Steps toward Artificial Intelligence*. New York: McGraw-Hill.

Newell, A. 1978. HARPY, Production Systems, and Human Cognition. Pittsburgh, Pa.: Carnegie Mellon University.

Pylyshyn, Z. 1985. Intelligent Database Interfaces: A Survey of Some Artificial Intelligence Applications. London, Ont., Canada: University of Western Ontario.

Shell, P., and Carbonell, J. G. 1988. The PARMENIDES Reference Manual, Internal Paper, School of Computer Science, Carnegie Mellon Univ.

Shell, P., and Carbonell, J. G. 1986. The FRULEKIT Reference Manual, Internal Paper, School of Computer Science, Carnegie Mellon Univ.