An experiment in constructing an open expert system using a knowledge substrate

by C. V. Apté
R. A. Dionne
J. H. Griesmer
M. Karnaugh
J. K. Kastner
M. M. Laker
E. K. Mays

This paper discusses an experiment in the use of an object-centered knowledge representation service to provide a common conceptual model for the construction of a large knowledge-intensive decision support tool. A core knowledge substrate forms a common resource for a variety of problem-solving activities and a basis for the rapid construction of new capabilities. FAME, a substantial expert system to aid in the financial marketing of IBM mainframes, has been built and extensively tested in the field to validate our tools and techniques.

Introduction

Early expert systems demonstrated considerable success at providing in-depth problem-solving capability on narrowly defined problems. While this class of expert system will continue to be of interest, there is also a requirement for providing knowledge-based systems for problems of ever-increasing size and complexity. Our research program for the past five years has been focused on developing techniques for large-scale knowledge-based systems. As a method for determining the effectiveness of these techniques, we chose a challenging domain in which to build a real system which made use of them. Furthermore, the domain we chose, financial marketing support for IBM marketing representatives, is a dynamic, evolving area, for which expertise is rare and not easily characterized. Thus, we faced two primary challenges—developing technology which would scale to ever-larger problems, and rapidly modifying prototypes to keep up with the dynamic marketing environment and the evolving expression of expertise.

The resulting system, called FAME (FinAncial Marketing Expertise), assists IBM marketing representatives in the area of financial marketing for large mainframe computing systems. The term *financial marketing* characterizes the financial decision processes used in the marketing of products and services of such large scale that they can significantly affect a company's financial status. The problem is that of generating a financing proposal, coupled with an acquisition plan, which

meets (or is compatible with) some financial objective. This contrasts with the applications of many early expert systems, which usually tended to be classificatory in nature. Another important distinction between this problem and many others addressed by AI researchers is that a typical financial marketing problem frequently has no one solution. There may be no definitive answer to a problem. This is not a question of merely computing financial optimality. The importance lies not only in the answer, but also in the explanation and justification that can be derived to back the answer. For this purpose, it is important to allow the user to efficiently explore the space of possibilities, and then help generate a convincing financial argument that will strengthen the selling of the resulting answer.

Since it was difficult to identify well-defined principles and expertise in the financial marketing domain, we chose an approach in which multiple problem solvers could be organized around a common conceptual model. As concepts were uncovered in the domain, they could be added to the conceptual model and easily be made accessible to all problem solvers. Since this sort of activity was ongoing, it was crucial that it be done in a disciplined way. Thus, we took the approach that the knowledge-representation language used to build the conceptual model was required to have a well-defined and enforceable semantics. This conceptual model forms a knowledge substrate on which problem solvers can be built. By employing an enforced semantics, this style of knowledge representation strikes a balance between simplifying the programming task and limiting the creation of unintentional contradictions. In contrast, the rule-based approach to building expert systems fails to explicitly represent the conceptual structure of the domain, thus under-constraining the construction of large, complex systems.

The architecture and technology we have adopted points the way to the construction of large, reusable knowledge bases which will serve as a valuable corporate resource. Such large knowledge bases would allow rapid development of new knowledge-based applications and provide a platform for sharing and evolving common corporate knowledge. We foresee that this approach to building knowledge-based systems could lead to knowledge-based system support for all marketing representative activities, including sizing, configuration, order entry, installation, and maintenance.

The FAME experiment

• *Representation language*

In designing K-Rep, the knowledge representation service [1, 2] which would serve as the common conceptual model for FAME, we chose to provide a simple, well-defined core representation and include integrated support for the interface to problem solvers. The core of K-Rep is based on KL-One [3] and includes an enforced semantics and classification-based inference.

The primary object for representation in KL-One-based languages is called a *concept.* Concepts are organized into a generalization hierarchy; a concept may be defined to be a specialization of other concepts, in which case the more specific concepts inherit from the more general. Binary *role relations* between two concepts are used to specify attributive information about concepts. Thus one can form fairly complex descriptions, since the roles of a concept may refer in turn to other (complex) concepts. The concept referred to by the role relation restricts the set of permissible concepts for that role in all further specializations. That is, inheritance may not be overridden. A subsumption relationship may be defined between concepts. A concept A is said to subsume another concept B when B possesses all of A's roles, and in all such roles the restriction of the role in A subsumes the restriction of the role in B. On the basis of this subsumption relationship, new concept definitions may be classified into the existing generalization graph such that the most specific subsumers and most general subsumees of a new concept definition are discovered. In K-Rep, concept definitions (from role relations) are restricted to be noncircular, and concept definitions may be modified any time after their initial definition.

Attached to each role is information constraining the possible set of values (*value restriction*), restrictions between roles (*role value map*), number of values (*cardinality*), and such things as the units, presentation name, documentation, presentation restrictions, and default value (*facets*). Some example value restrictions are "number greater than 5," "member of the set of known processors," and "string." Role value maps further restrict values on sets of roles. For instance, the sum of the workloads of the applications on a nonoverloaded system cannot use more than 100% of the available processor power. In K-Rep, role value maps may be any arbitrary Lisp predicate, in addition to a supplied set of inequalities. Constraints are enforced by applying the predicate to the designated role values. To facilitate concept instantiation in the presence of constraints, a constraint propagation function may be supplied which interacts with the role value restriction. The result of this propagation may be queried under program control. Using the Common Lisp condition system, constraint violations are signaled and proceed handlers are defined, thus providing a protocol for concept instantiation in the presence of constraints.

Figure 1 shows a portion of a K-Rep structure that evolved during the course of a FAME session. In the session a user is comparing the financial implications of acquiring an IBM 3090™ 200J system using an outright

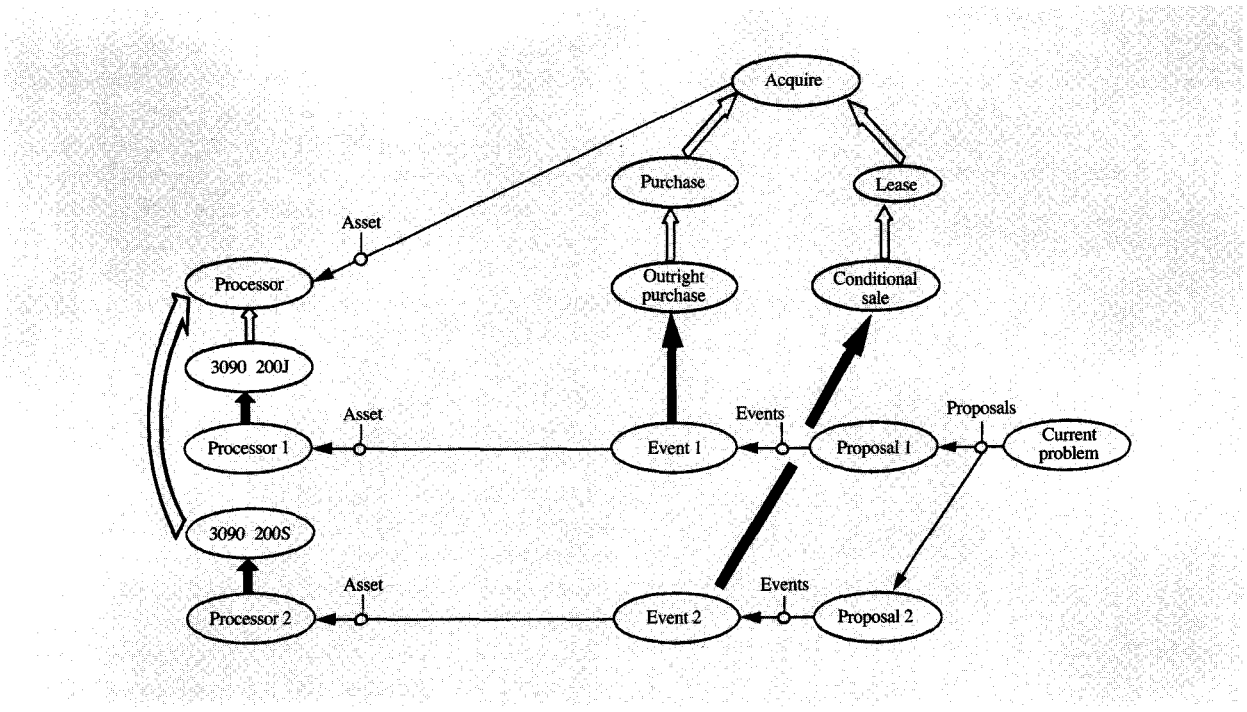

Figure 1

FAME knowledge representation in K-Rep.

purchase method of financing, with the acquisition of an IBM 3090 200S system financed through a leasing method known as a conditional sale. In this figure, an ellipse represents a concept. A concept can either be *generic* (representing a class of objects) or *individual* (representing a single object). A wide white arrow from concept B to concept A denotes that generic concept A subsumes generic concept B. In the figure the generic concept "Processor" is shown as subsuming the generic (but more specific) concept "3090 200J." A wide black arrow from concept B to concept A denotes that generic concept A subsumes individual concept B. An example in the figure is the subsumption of the individual concept "Processor 1" by the generic concept "3090 200J." A role is represented by a narrow link leaving a concept and pointing to the value of that role. "Processor 1" is the role value of the role "Asset" belonging to the concept "Event 1."

Portions of the definitions of the generic concepts "Acquire," "Purchase," "Outright Purchase," "Processor," and "3090 200J" as defined in the K-Rep language by system developers are shown in **Figure 2.** The "Asset" role of "Acquire" is defined to be the generic concept "Processor," and inherited by its subconcepts "Purchase" and "Outright Purchase." A role defined for the concept "Processor" will be inherited by its subconcept "3090 200J," and only appear in the definition for "3090 200J" if a more specific value is defined. For example, the "List Purchase Price" role for "Processor" is defined to be ≥ 0, while the "List Purchase Price" role for "3090 200J" is defined to be the more specific value obtained by interpolation over a time-valued set. Individual concepts, such as "Event-1" and "Processor-1," and the values of their roles, which represent the definition of the specific problem scenario, are defined as the user interacts with a set of menus, some of which are shown in subsequent figures.

Around the KL-One-based core of K-Rep, several extensions are necessary in order to provide a usable representation service for building large expert systems. These extensions fall into two major categories—extensions to assist in the ongoing development of knowledge bases (KBs), and extensions to the representational power of the language. To facilitate development, we have a presentation-based browser facility, a version-based control system for KB source file definitions [4], and a sub-KB mechanism for name

```
(defconcept Processor :generic ( )
  ((amount-of-central-storage (≥ 0) :units megabyte)
   (list-purchase-price (≥ 0) :units dollar)
   (mips (≥ 0))))

(defconcept 3090-200J :generic (processor)
 ((amount-of-central-storage (:set 64 128))
  (list-purchase-price
   (:time-valued-function linear-interpolate
    ((:day "11/01/89") 5112400)
    ((:day "4/01/90") 5256975)
    ((:day "10/01/90") 5257370)
    ((:day "3/01/91") 5261693)))
 (mips 45.65)))

(defconcept Processor-1 :individual (3090-200J)
 ((amount-of-central-storage 64)))

(defconcept Acquire :generic ( )
  ((asset processor)
   (acquisition-date (:day))))

(defconcept Purchase :generic (acquire)
  ((purchase-price (≥ 0) :units dollar)
   (sales-tax-treatment (:set "expensed" "capitalized"))))

(defconcept Outright-Purchase :generic (purchase)
  ( . . . ))

(defconcept Event-1 :individual (outright-purchase)
 ((asset processor-1)
  (acquisition-date (:day "3/01/91"))
  (purchase-price 5261693)
  (sales-tax-treatment "capitalized")))
```

Figure 2

Example of the FAME knowledge-representation language: definitions of some of the K-Rep concepts shown in Figure 1.

resolution and efficient KB redefinition. Figure 2 shows how the K-Rep concepts of Figure 1 are defined in the representation language.

It is our perception that demanding knowledge-based applications are never fully supported by the representation service being applied, either for reasons of expressiveness or efficiency. Rather than continually revise the core service, we chose to provide well-designed points of interface to the core service, thus allowing the knowledge engineer to incorporate techniques quickly and make them mostly transparent. In this way, a facility for computing defaults has been provided by using facets and *functional attachments.*

A unique feature of the functional attachments in K-Rep is that the returned values may be cached. As the FAME system developed, the run-time computation involved with some of the functional attachments became a considerable bottleneck (e.g., generating financial cash streams for a 60-month analysis period). However, many of the functional attachments depend only on the state of the KB alone and do not make reference to the state within the Lisp environment. In the case for which the knowledge engineer makes a declaration that the functional attachment is solely dependent on the state of the KB, K-Rep is able to cache the returned value. Appropriate cache dependencies are maintained in this process, so that as the state of the KB changes, the relevant cached values are invalidated.

The succeeding sections discuss the use of K-Rep as a knowledge-representation service in support of the FAME problem solvers.

• *FAME problem solvers*

FAME makes available to the user a variety of problem-solving specialists that would typically be used by an expert for preparing competitive proposals. Problem-solving systems in FAME include capacity planning, customer cash flow analysis, impact analysis, financial planning, marketing incentives support, and high-volume information browsing. Much of the problem solving is interactive, with the user assuming a high degree of control over the choice and sequencing of actions to be performed.

Each of the problem solvers may be thought of as an expert system in a specialized sub-area of financial marketing. They employ a variety of different techniques, ranging from heuristic search and pattern-directed rules to dynamically generated hypertext interfaces. Unlike relatively stable domains such as medicine or geology, where problem-solving paradigms remain more or less the same, the marketing domain is substantially dynamic. As market conditions change, experts devise new means to operate effectively in the new environment, and systems like FAME must support this dynamism. That is, as new problem-solving approaches are devised by experts in the field, they should be able to be rapidly incorporated within the FAME environment.

With this goal in mind, the FAME knowledge-base architecture attempts to make a clean separation between financial marketing expertise and the financial marketing consensus reality [5], i.e., those things about financial marketing that everyone in the field ought to know. Financial marketing expertise is modeled within appropriate problem solvers as they are introduced and implemented in the FAME environment. The consensus reality is, however, modeled separately and explicitly in a terminological representation (i.e., definitional representation in K-Rep of domain structures in the form of concepts related by subsumption and attribution). This substrate of knowledge is then available as the common base upon which all problem solvers are implemented. Not only do the problem solvers heavily use the common knowledge about financial marketing that is available through the substrate, they also communicate with each other via this medium.

The FAME knowledge bases represent the common substrate. Captured within the knowledge bases is knowledge about the mainframe computer market that is available both publicly (through firms like Computer Price Watch) and privately (through a company's own marketing research). This knowledge is represented in K-Rep via concepts that are connected using either a subsumption relation or a role (attribute) relation. Thus, one can create explicit models of domain knowledge in the form of taxonomic organizations. The K-Rep knowledge-representation environment has been carefully engineered over the past few years to provide an efficient service for building and using the large-scale knowledge bases in FAME. In addition to powerful knowledge-presentation interfaces, the K-Rep environment has also extended the basic definitional classification-representation paradigm to support services such as truth-preserving functional value representations through caching, and functional constraint specifications through role-value maps.

Figure 3 is a high-level view of the organization of FAME. The innermost kernel is the K-Rep knowledge-representation formalism, tightly coupled with its programming environment. This core provides a domain-independent, object-centered knowledge representation service. Residing upon the core are the FAME knowledge bases and the FAME programming environment. The knowledge bases capture the domain-specific generic knowledge about financial marketing, while the programming environment is a toolkit geared toward supporting the FAME environment particulars (e.g., the user interface). We view this layered organization as a specialized shell. This is an application enabler for financial marketing. Using this enabler, we have been able to efficiently build a variety of problem solvers. These problem solvers do not necessarily need each other, although they all do operate in the domain of the underlying enabler.

Several types of objects modeled in the FAME knowledge bases make rapid prototyping possible. The basic entities that must be considered in our financial marketing scenario include mainframe computer vendors and their products, the customers who plan to acquire these products, the marketing proposals that vendors prepare for their customers, and techniques for financial analysis of the proposals. An expert in this field will have to understand and reason about the structure and function of these entities. Therefore, we chose to model a comprehensive collection of these types of entities in our knowledge bases.

Various types of objects that play a major role in financial marketing were identified, and their interconnections studied. Relations among various objects were developed using the K-Rep relations of subsumption and attribution. Classifying the domain in terms of objects

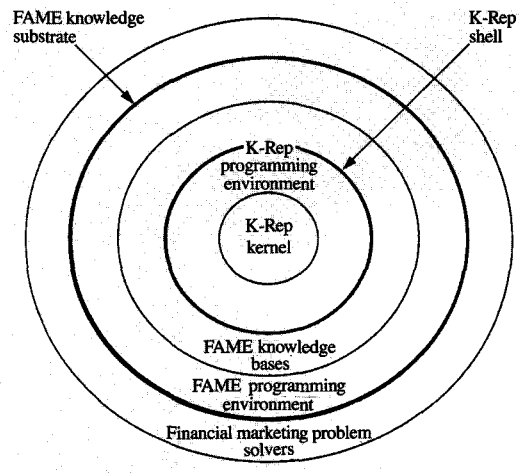

Figure 3

A high-level view of the organization of FAME.

allows us to quickly build taxonomies for different classes. These classes include, but are not limited to, financing mechanisms, manufacturers, products (past, present, and future), customers, financiers, etc., as relevant to the marketing of mainframe computers. Subsumption allows us to build abstractions of these classes. This categorization not only makes it easy to organize knowledge using structured inheritance networks, but also permits acquisition of such knowledge from experts, either via a knowledge engineer or by using computer-based acquisition tools. Figure 1 illustrates a partial representation of these classes using K-Rep semantics.

The knowledge bases provide a variety of services for these objects, including inspection and manipulation, abstraction and aggregation, prototype template filling, and financial analysis. Reuse of these entities goes beyond just data reuse. The knowledge bases provide a basic toolkit of structures that one would have to deal with, as well as generic problem-solving methods such as default and prototypical reasoning that go with these structures. In addition, the explicit object-centered style in which they are modeled makes it very efficient to directly examine and reuse the contents of such knowledge bases.

User interface

Expert systems typically attack complex problems, which may lead to a complex interface. An expert system that deals with hundreds of concepts and their

interrelationships in order to design problem solutions interactively with a user must balance the need for accuracy and completeness with simplicity of presentation. The system should guide, direct, and focus unobtrusively while allowing the user to control the interaction. FAME uses an approach to expert system user interfaces that is based on an object-centered knowledge base representation of data, problem structure, problem solvers, and problem-solving control. The knowledge base contains levels of abstraction to allow guidance and focus without obfuscation of details. All objects can be directly manipulated by the user. Problem solving is a structured, focused, and filtered walk through the knowledge base, instantiating individual objects and their relationships.

User interfaces can vary widely with respect to the degree and nature of user–machine interaction intended. They range from autonomous systems to systems in which the user plays the dominant role. FAME represents an intermediate case, in which the user and the system share control: Sometimes the system drives the interaction and sometimes the user dominates it. This balance depends upon the nature of the particular planning problem, the part of the plan being worked on, and the skill of the user.

There are good reasons for having this kind of interactive flexibility. FAME is intended to aid computer marketing representatives in preparing sound and justifiable plans for their customers. These plans are intended to augment the customer's mainframe computing capacity in the presence of growing workloads and to propose financial acquisition methods, according to the customer's own preferred financial criteria. The criteria and constraints for the equipment and the financial plans depend not only upon current knowledge of customer doctrine, situation, and preferences, but also upon the user's estimates of the customer's needs and upon the user's ability to justify his recommendations persuasively. No system now contemplated will have a complete enough knowledge base to carry out the various trade-offs autonomously. On the other hand, very few marketing representatives have the broad range of knowledge needed to prepare a detailed plan without human or machine assistance. In practice today, human expert consultants and relatively unintelligent machine aids are used—but productivity is limited by such an approach.

Our approach places an object-centered interface in a semantic network, which holds all major entities dealt with in the system. The semantic network enforces the strict semantics of a subsumption hierarchy through an automatic classification scheme and dynamic constraints called role value maps among attributes [1, 2]. All input and output of the system is done through a combination of presentation and acceptance of concepts. The presentation is driven by the information on the concept. Input and output are not just a side-effect of the computation (e.g., print statements inside rules) but represent a filtered and guided tour of the knowledge base, instantiating individual concepts to represent the current state of the problem. Concepts are redisplayed automatically whenever the displayed information associated with them changes.

A user's view of FAME We have observed that much of the problem-solving behavior of experts in financial marketing is sporadic. There is typically no "best" answer to satisfy customer capacity requirements while meeting applicable financial objectives. The question is not simply to optimize some financial objective. Of necessity, any proposed solution will be the result of balancing competing goals. In addition, the financial marketing domain is an open system (there will always be hidden variables) that changes dynamically. Therefore, we have striven to create a system whose control is highly interactive and flexible.

A user of our financial marketing advisory system sees a series of screens leading through a consultation session. Each screen is divided into separate panes for queries, output, icons, etc. Thus, the user can always distinguish between the input to, and the output from, the consultation system by its location on the screen. There is always a pending query presented to the user, although the user may "click" on any displayed object to redirect the problem-solving activity, browse the knowledge base, or ask for help. The system will always prompt the user for the information needed to perform the next task, but will also always allow one of the asynchronous escapes.

Figure 4 shows a screen which would be displayed at the start of the problem-identification process in FAME. The narrow pane on the left side is the *query pane*, the larger pane on the right is the *output pane*, and the long pane just above both is the *plan step pane* (or *Roadmap*). The plan step pane shows the state of the problem-solving plan's execution and can be used to redirect that execution. The output pane gives a filtered look into the concepts currently instantiated to represent the problem and proposed solutions. The display in the query pane represents a problem-solving control concept, which dynamically points to other concepts in the semantic network that are selectable to define a problem type. The recommended choice is highlighted, but the user is free to select any other choice. After the user identifies that the type of problem to be addressed is, say, an equipment-planning problem for building proposals from capacity requirements, one of the next screens (potentially, there are any number of intervening screens, depending on the context) asks for the details of the customer's current installed base along with expected growth (**Figure 5**).

Here, although the query pane has a pending question, the user may point the mouse to any object in the cluster (in the output pane) and change its attributes, copy it,

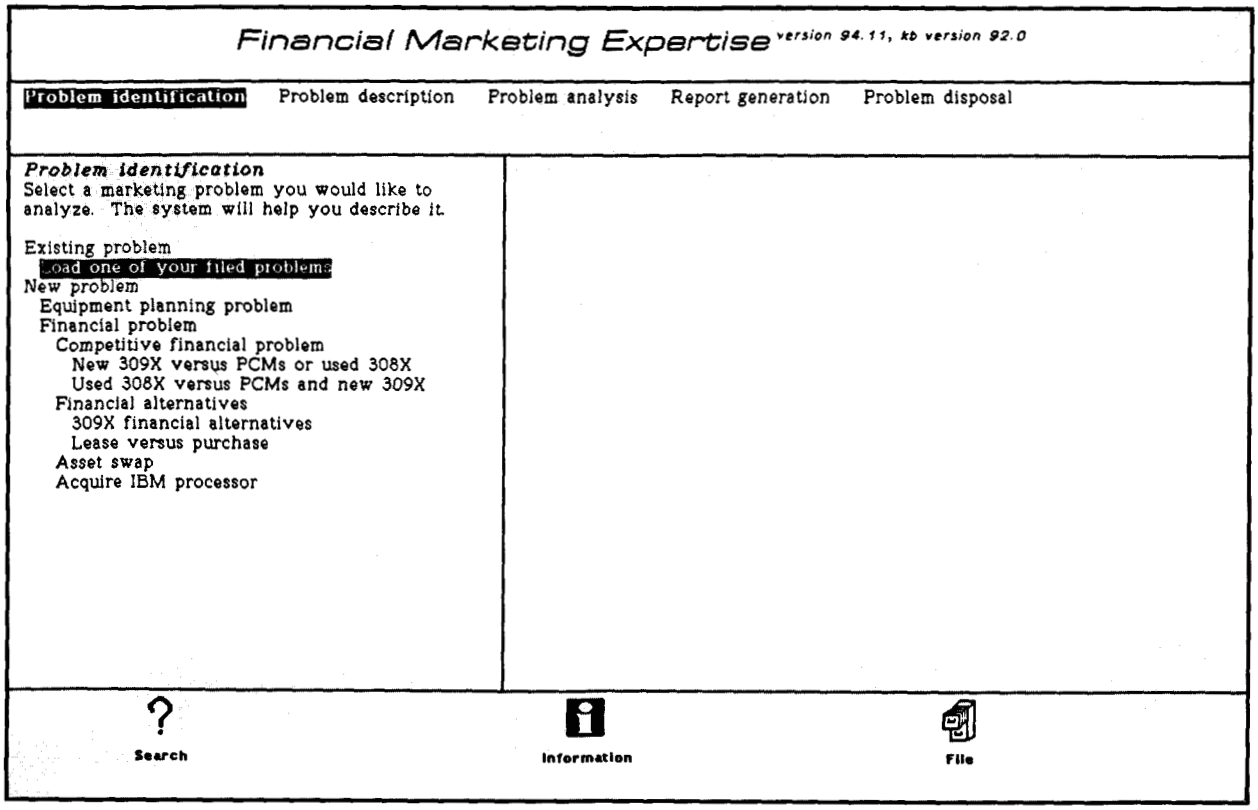

Figure 4

Problem identification.

create another object of the same type, move it, delete it, or ask for further information about it. This set of asynchronous actions is possible because each item on the screen is the representation of a concept in the current problem description, and the expertise to manipulate that concept is a property (possibly inherited) of the concept.

Notice that, after *Problem identification* and *Problem description* are completed, the plan step pane now contains several levels of detail. This is a map of the problem-solving control flow. The *Problem analysis* step consists of three subparts, *Equipment planning*, *Financial calculation*, and *Financial critique*. Each of these steps also has subparts. The user can use this information to 1) determine "Where am I?", "What do I do next?", or 2) actively redirect the problem solving by "clicking" on a plan step. This uses the same mechanisms as the manipulation of the problem description, since plan steps are also concepts.

The *Proposal generation* step then recommends an equipment plan (**Figure 6**) which may also be manipulated (**Figure 7**). Here, the financing method is being changed, and the proposal display in the output pane will be updated after the change is made.

By spreading the expertise of the system throughout the concepts in the system, it becomes possible to also spread the control structure, enabling a "mixed-initiative" interaction. The objects are responsible for their own correct behavior. The enforced semantics of the knowledge representation is used to guarantee well-behaved objects. A close interface is kept between objects manipulated in the system and objects input or displayed on the screen (**Figure 8**). The default displays are in context and at a level of detail designed to anticipate the user by providing defaults and suppressing irrelevant concepts. With a parsimonious set of input/output styles ubiquitously available via inheritance, the user can easily navigate through the screens to customize the problem.

Implementation of the interface We use our semantic network, K-Rep, pervasively throughout the FAME system for representing both problems and problem-solving control, and for communicating between problem solvers. K-Rep structures, built dynamically during consultation, represent input from the user, databases, problem solvers, intermediate results obtained by the system, and

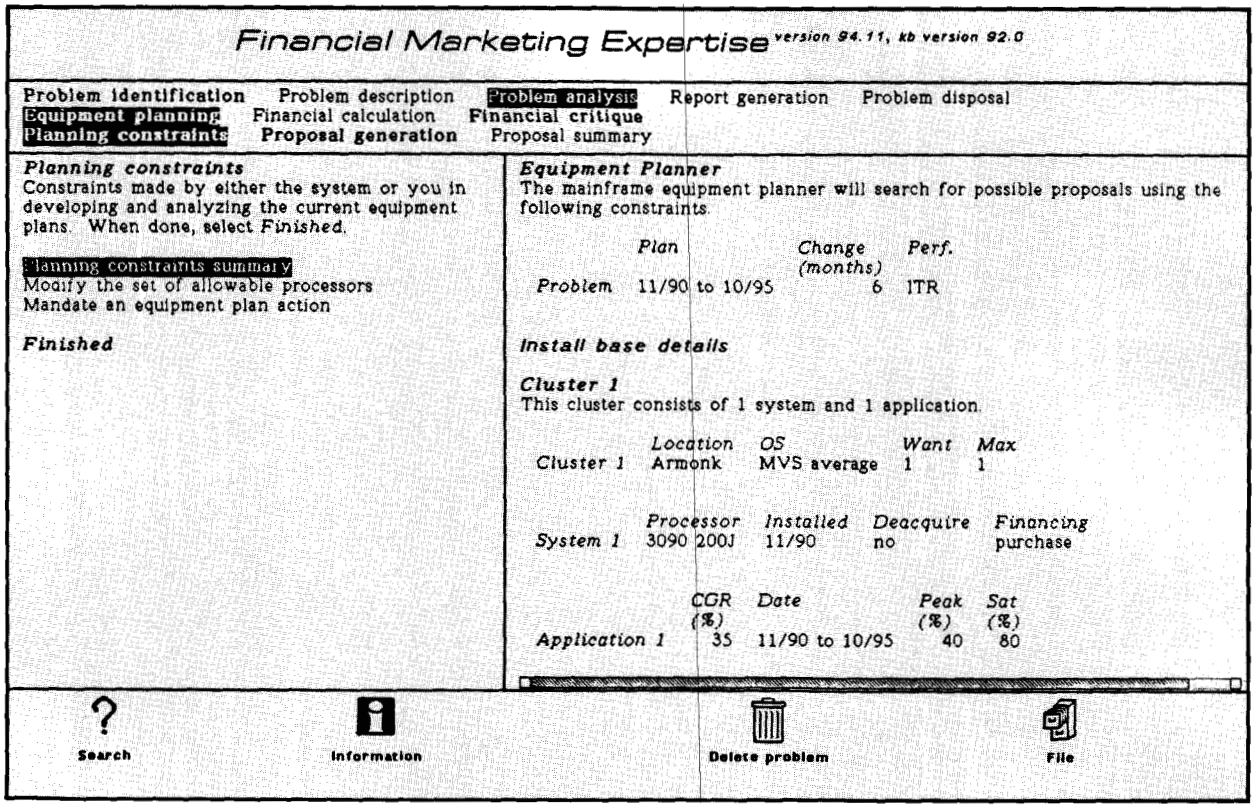

Figure 5

Planning constraints summary.

conclusions. Unlike static traces of program activity, K-Rep concepts are dynamic objects heavily using inherited roles, functional attachments for inferred roles, and developer-defined facets.

We use an object-centered approach for both presentation to the user and internal representation of the problem-solving activity. Each query presented to the user is represented internally as one or more K-Rep concepts. The items of a menu correspond to subconcepts of a concept or to the roles of a particular concept. The presentation of a query to the user is implemented as the context-dependent prompting for an instantiation of the corresponding concept. The interface is driven by the information obtained from the concept or concepts being displayed. The dynamic creation, modification, and deletion of such concepts leaves a representation of the problem-solving state in the semantic network and a filtered display (text, tables, graphics, etc.) on the screen.

The direct correspondence between internal representation and external presentation provides many benefits. We have actively pursued the goal of mirroring the knowledge used by human experts explicitly in our underlying knowledge representation. Our users have found such displays useful in explaining the problem-solving behavior of the system. This has been extremely useful in the elicitation of knowledge from experts, the validation and refinement of that knowledge, and the maintenance of knowledge in the constantly changing domain of financial marketing.

Knowledge acquisition Tightly coupled with knowledge engineering is the requirement for some form of computer-based support for knowledge acquisition. The success of fully automated knowledge acquisition (e.g., self-learning) tools has been quite limited to date. Our experience with the FAME system indicates that even semiautomatic support in the form of advanced graphical interfaces for browsing and editing knowledge bases can be a powerful medium for both managing and use of very large knowledge bases. Other experiences also indicate this to be the case. In this spirit, we have been evolving an approach to develop object-centered interfaces on advanced workstations that allow a structured retrieval of all knowledge stored in the knowledge bases that support FAME. This technique eases tasks such as maintaining very large knowledge bases and using them as an

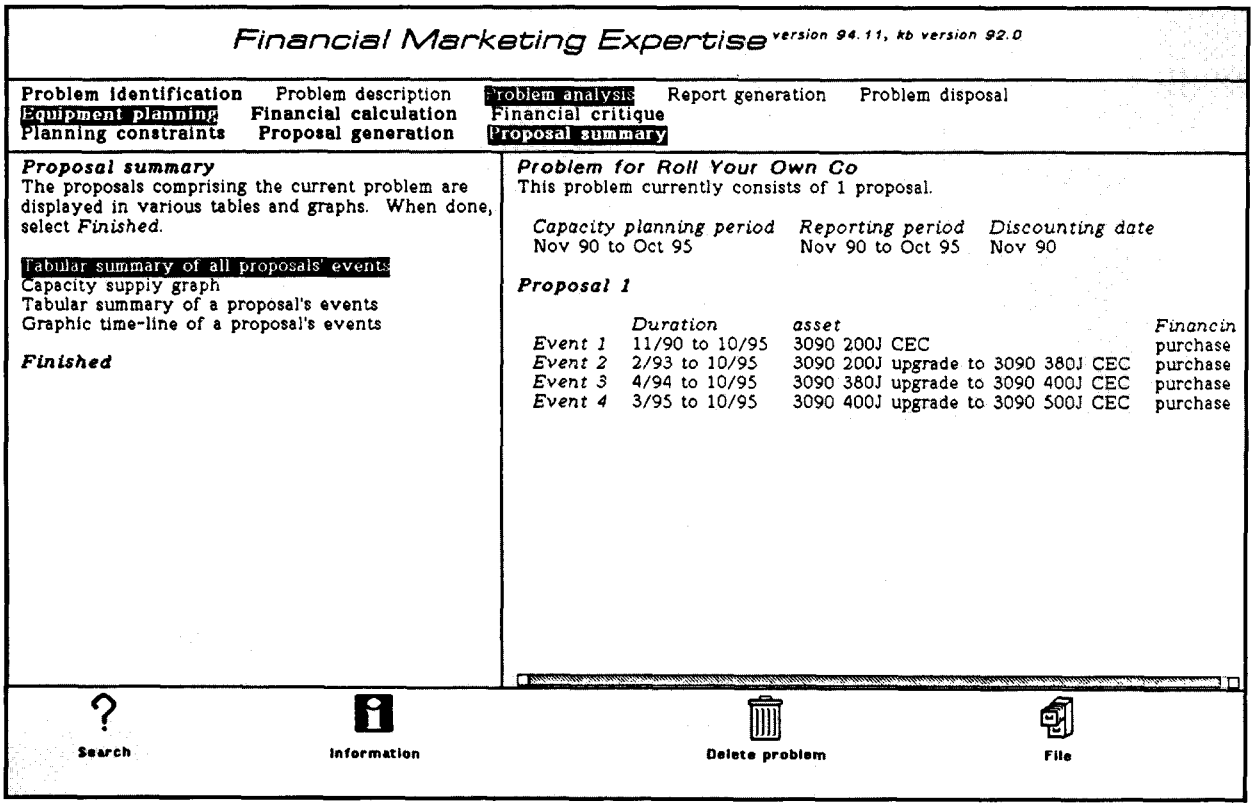

Figure 6

Proposal summary.

educational medium. In this activity, we are developing and extending capabilities similar to those reported in hypertext systems and systems such as XCON-in-RIME [6].

Knowledge acquisition and maintenance are important issues for knowledge-based systems, and become even more so when we deal with very large knowledge bases. Hypertext systems have traditionally taken the approach that by fully exploiting the more advanced features that are available on today's high-resolution bit-mapped monitors and fast desktop computers, one can build fairly powerful interfaces to large amounts of stored data. These interfaces have the goal of providing access to large databases with very much the ease and flexibility one has when using large dictionaries and encyclopedias. We take the view in FAME that providing such interfaces to large knowledge bases not only makes them easy to use, but also makes maintenance and modeling of the knowledge in them very tractable.

We have successfully put in use an object-centered interface to the FAME knowledge bases for both knowledge engineer and end user. This interface has a very clear and domain-independent syntax. It can present objects from a knowledge base on a screen. The interface uses certain presentation mechanisms that allow knowledge-base objects, even when appearing on a screen in many different ways, to be selectable by a user as a unit object. By selecting objects on a screen, the user (whether a knowledge engineer or an end user) may be allowed to carry out a variety of activities, all controlled essentially by where the object is located in the underlying knowledge base. Thus, the user may traverse or browse through the knowledge base by reaching out through the object's subsumption and attribution links.

This methodology provides a single coherent interface to knowledge bases that relies on the semantics of the underlying object-role network, rather than on some forced artificial interaction. As Figure 7 illustrates, such interfaces allow users to open windows at random into the knowledge base, thereby permitting a structured retrieval of all related knowledge that supports the objects in the initial window.

This object-centered browser is used to good effect when knowledge is being acquired and entered into the FAME knowledge bases. It is extremely easy to identify sub-areas in the knowledge bases in which objects must be

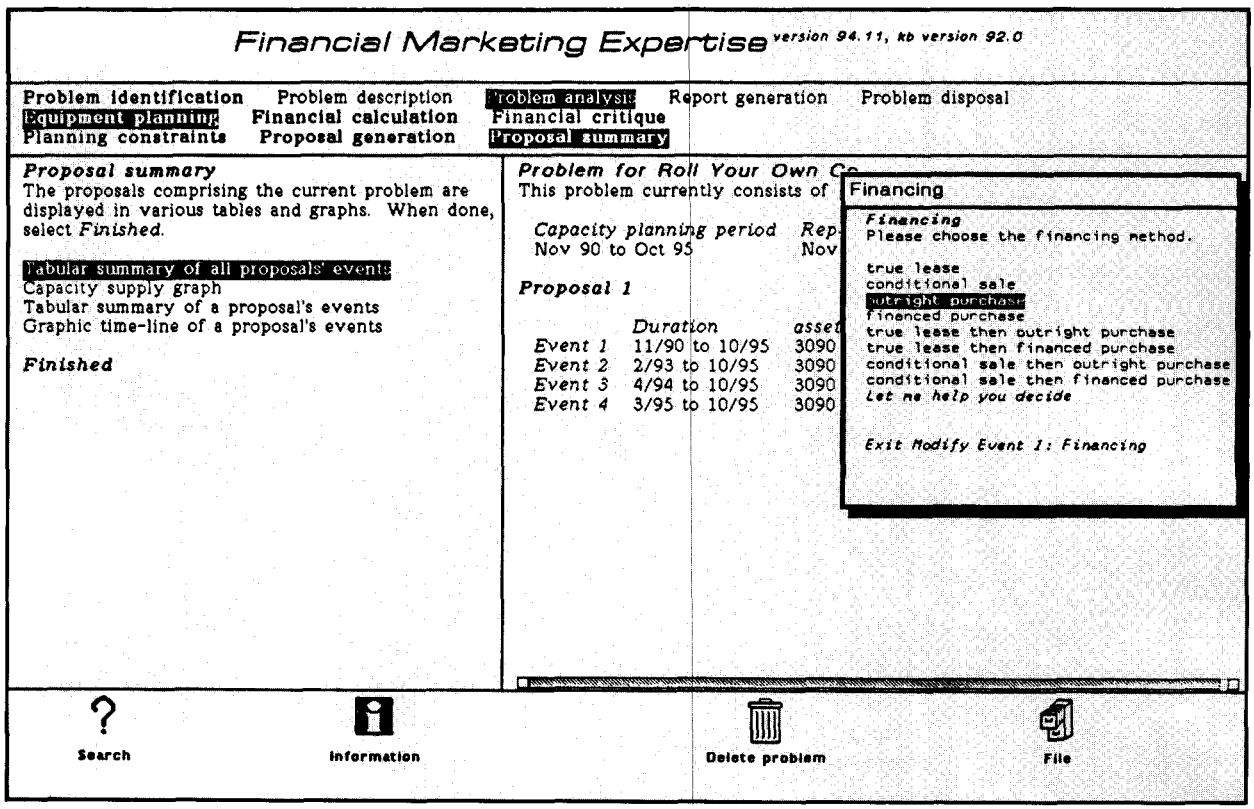

Figure 7

Financing.

added, modified, copied, or deleted. Specialized methods associated with these objects have been provided for these operations that can be dynamically invoked by selecting objects on a screen.

In field trials, users expressed delight with the interface. Most difficulties with the interface arose from the choices of display filter, defaults, plan steps, and language, not from the style of distributed expertise and mixed-initiative control. Field studies also revealed, however, that the computational overhead of animated displays may cause unacceptable delays in an interactive system.

Mainframe Equipment Planner

While the primary application for FAME is the financial analysis and sensitivity analysis of marketing proposals, it was recognized that such analyses would inevitably encourage the iterated generation of modified proposals. For this reason, FAME could not adequately improve the productivity of marketing teams without also providing tools to facilitate the generation of good proposals. Many such tools are feasible, varying in focus, time horizon, and degree of product coverage. Also, various trade-offs between computational cost and accuracy may be used.

The Mainframe Equipment Planner (MEP) is one such tool. It is designed to find medium- to long-term plans for providing enough mainframe computer throughput capacity, in the presence of demand growth, while satisfying a variety of auxiliary constraints. The goal of the MEP is to find a least-cost proposal that satisfies all requirements. Typical MEP planning periods range from one to five years. The longer-term plans provide some assurance that a technically and financially acceptable growth plan exists. However, all manufacturers modify their product lines or prices frequently, and customers' perceived needs are also subject to change. Any plan is therefore likely to be revised at frequent intervals. Only the earlier actions in the plan are actually proposed. The only products currently considered by the MEP are mainframes, and the only capacity requirement it deals with is for processor throughput. More comprehensive capacity planning will require additional tools and knowledge.

Implementation of the MEP takes the form of a heuristic search program [7], which is best described as a variant of branch-and-bound. It is designed so that, for simple problems, the least-cost solution is obtained. In the case of more complex problems, a trade-off between optimality

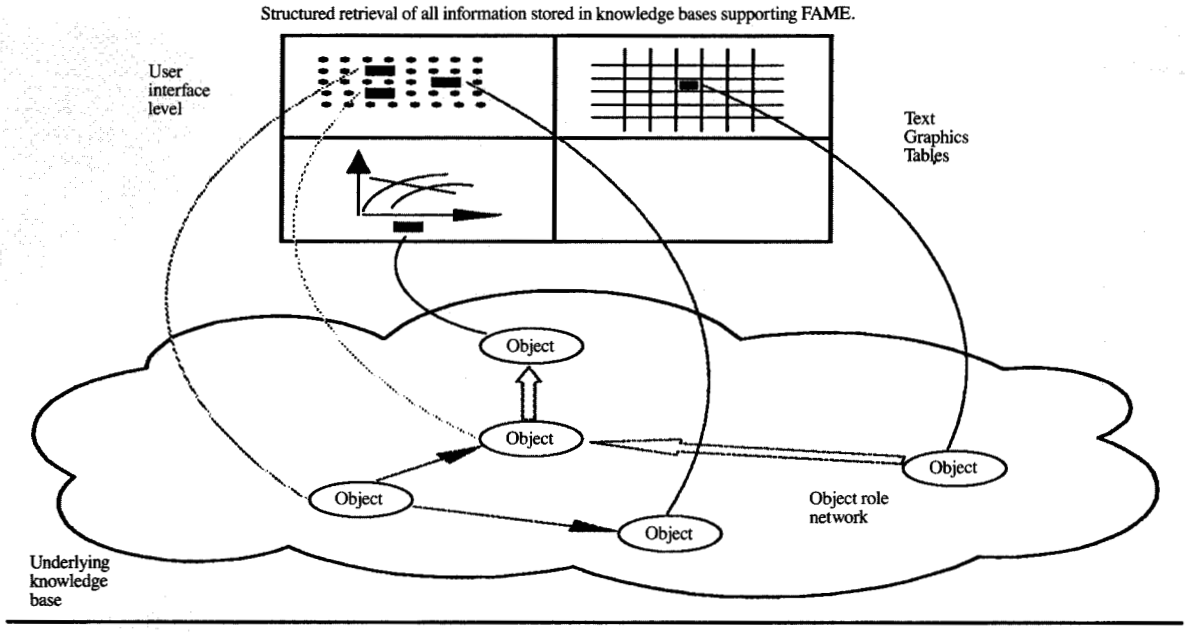

Figure 8

Object-centered interfaces for browsing the knowledge bases.

and performance is essential. Field trials indicate a strong desire on the part of users to obtain results almost instantly. While this is not possible, we are able to produce good plans within a few minutes. The computational time per node expansion is variable, depending upon the number of successor nodes, that is, the branching factor. Typical times, however, are from one-half to one second per node expansion. On the basis of this observation, we selected a default range of 300 to 600 node expansions for completion of a MEP search.

If a search has expanded 300 nodes of the search tree without completion of a plan, a nonadmissible search technique is invoked that guarantees solutions within no more than 300 additional node expansions. In such cases, the resulting plan is not guaranteed to be least-cost, but the search is designed to produce good results. By this means, we have been able to produce mainframe equipment plans in less than five minutes for nearly all of the test problems.

FAME can override the default limit of 600 node expansions. If a problem is complex, and if the user is patient, the MEP can be instructed to conduct much larger searches. The fundamental limitation is the paging space of the host computer, and tens of thousands of nodes can be expanded when time and space are available. Fortunately, a small search usually suffices.

Unlike many other types of artificial intelligence tools, search programs for potentially large problems must be designed from the start for efficient performance. Without this precaution, extensive testing would not be possible. The MEP uses a set of special data structures, other than the existing knowledge base, to enhance its performance. These are the vectors, arrays, and defined structures which are frequently accessed or created during the course of the search. In addition, a set of special access functions is employed to obtain price and performance data from the K-Rep knowledge base. All of the constraints and parameters needed by the MEP are passed to it by the system interface, and many static data needed repeatedly by the search program are precomputed and stored. An example of this is the vector of monthly financial discount factors, used in discounting the cash flows of each partially completed plan.

The basic computational entity modeled by the MEP is the *cluster*. This is a set of mainframes that share a common computational load, that is, some set of applications. For each application, we specify a) the initial capacity needed, b) the compound annual growth rate of

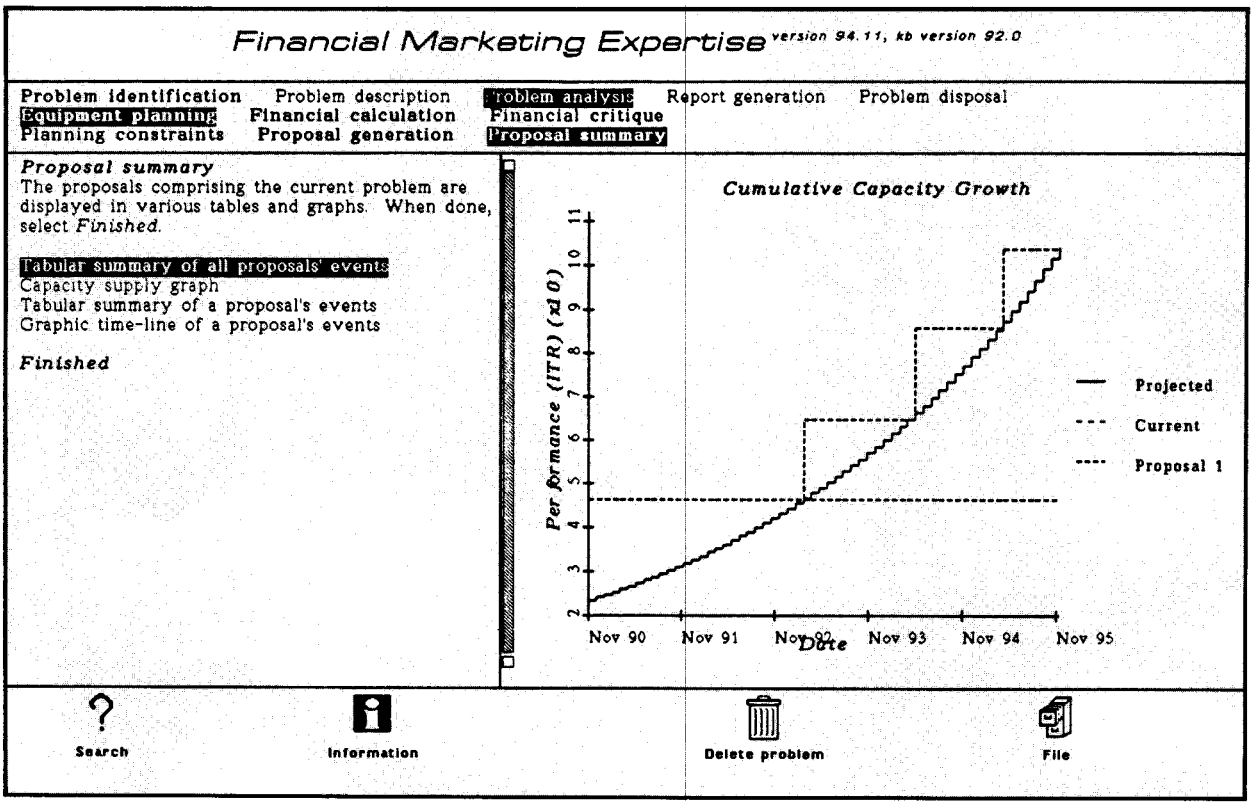

Figure 9

Capacity graph.

the needed capacity, and c) the maximum processor percent utilization that is tolerable, usually, for meeting response-time requirements in interactive applications. These three values determine the need in each application for processor capacity over the planning period. The capacity requirement for the cluster, during each month of the planning period, is taken to be the sum of the applications' requirements. The MEP will augment the cluster's processing capacity whenever the demand curve rises to the level of the existing capacity. This gives the search a temporal parameter: the last month for which the existing capacity meets requirements. At each node expansion, this parameter strictly increases for all of the successor nodes. This behavior is illustrated in **Figure 9**, in which the planned capacity (staircase) of a single cluster is increased whenever the required capacity reaches the same level.

Two measures of throughput capacity are available in the MEP: *millions of instructions per second (MIPS)*, and *internal throughput rate (ITR)*. IBM measures ITR by benchmarking a variety of application types and operating systems on each mainframe model. Since the relative performance of two different models depends upon the software being benchmarked, ITR offers a more detailed comparison of processor throughputs.

It is important to note that a given problem may include just one cluster or multiple clusters. In the latter case, some aspects of site consolidation or the rearrangement of mainframes and tasks among the clusters may be included in the constraints on the MEP.

The MEP takes into account for any customer all computation costs that are supported by the knowledge base. These include new or used price, lease rate, power, floor space, and operations. It is expected that operating systems will ultimately also be included. The total cost of a plan also includes the recapture of residual values at the end of the planning period. Future costs and residual values are estimates, based upon history, and are intended for planning purposes only. In comparing alternative plans, all costs are discounted to the start of the planning period. While the financial analyzer may use a variety of criteria, the MEP uses only one, which is pretax-discounted cash flow. This may lead to small discrepancies between the MEP and the financial analyzer in the comparison of

alternatives, but it is employed for performance reasons. There is no other barrier to using a variety of financial criteria within the MEP.

The expansion of a search node in the MEP involves the use of macro-operators based upon a small number of primitives. The primitives involve acquiring, upgrading, deacquiring, or rearranging mainframes (in the case of multiple clusters). All of these must be considered in order to meet the constraints. In addition to meeting the capacity requirements, the proposals must satisfy constraints which limit the number of mainframes for each cluster and which place a lower bound on the number of months between augmentations of each cluster. Additional constraints limit the computer models which may be added within any plan and may specify precisely the months in which any changes are permitted.

Another class of constraints is the class of *mandated actions.* The MEP user may specify an acquisition, upgrade, or deacquisition in any cluster at any month. He may also mandate the merging of any two clusters, the transfer of a mainframe from one cluster to another, or the transfer of a task from one cluster to another at any month. The MEP will carry out any mandate that is possible. If it is not possible—for example, the mandated deacquisition of a mainframe that has already been deacquired (in a given node being expanded)—the MEP will simply ignore it. The mandated actions offer the user a great deal of flexibility to try his ideas and to have the MEP surround them with a near-optimal comprehensive plan. Feedback on the consequences of each idea is available within a few minutes. While the merging of clusters must be mandated to the MEP, the subsequent addition of a site consolidation planner to FAME could also automate this function.

Figure 5 shows some of the planning constraints that appear on the FAME screen for a very simple problem. These can be edited on the screen by the user.

The task of implementing the MEP as heuristic search was complicated by the extreme differences in the sizes of the search trees for different problems. In the case of a single cluster with one processor, each capacity augmentation may involve only three or four possibilities for upgrade or replacement. If only, say, four such augmentations are needed to complete the plan, less than 100 node expansions will exhaust the possibilities. On the other hand, if we have four clusters with two mainframes in each, we estimate that there are more than 2500 arrangements of the mainframes in the clusters at any one time. In addition, the possible numbers of ways of augmenting the clusters are multiplicative. In order to complete the search within an acceptable time, it must be judiciously limited. Only a few of the most promising rearrangements of mainframes can be considered. Also, the search must employ a nonadmissible technique in order to expand only the most promising nodes of the search tree. Although these methods of pruning the search tree preclude a guarantee of optimality, they can be tailored to nearly always give very good results.

Financial analyzer

Financial analysis and planning is one of the major problem-solving substeps in FAME [8]. Financial analysis is essentially the calculation of the quantitative effects of a computer acquisition on a variety of customer financial parameters. These include cash flows (before tax, after tax, and discounted), profit and loss statements, and budgets. How a computer acquisition affects each of these depends upon how the computer acquisition is being financed, i.e., whether it is being leased, or purchased, or a combination of the two, or some special case of one of these. Since these methods of financial analysis are generic enough to be widely applicable, we considered them to be part of the financial marketing consensus reality and chose to model them within the FAME knowledge bases. We describe in some detail here how these models are constructed.

In general, when a corporation acquires a capital item, it can choose from a variety of financing methods for executing the acquisition. The corporation may pay for the item entirely in cash by drawing upon part of its liquid assets. This is a common form of financing an acquisition that we all know about, *outright purchase.* Alternatively, the corporation may borrow money from a lending institution to finance the acquisition, or arrange for some kind of an installment payment agreement with the manufacturer, which is generally known as a *financed purchase.*

A popular method of financing acquisitions by corporations is leasing, which is essentially a long-term rental agreement between a lessor (typically the manufacturer or a third party) and the lessee (the corporation that will actually be using the machine). However, depending on various factors, including the item's first ship date and its monthly lease payment, the Internal Revenue Service and the FASB (Financial Accounting Standards Board) may view a lease agreement either as an *operating lease* or a *capital lease.* A model of financing methods that is available at any given time for a given kind of a product can be built and captured in an object-centered representation very succinctly. **Figure 10** illustrates how this is done in the FAME system.

An abstraction hierarchy is modeled on the basis of commonalities and disjunctions among the various financing methods that are generally available for financing computer acquisitions. The types of roles that are attached to this class of concepts are attributes that can be used to calculate the effects of a financing method on the corporation's cash flows, profit and loss statements, and

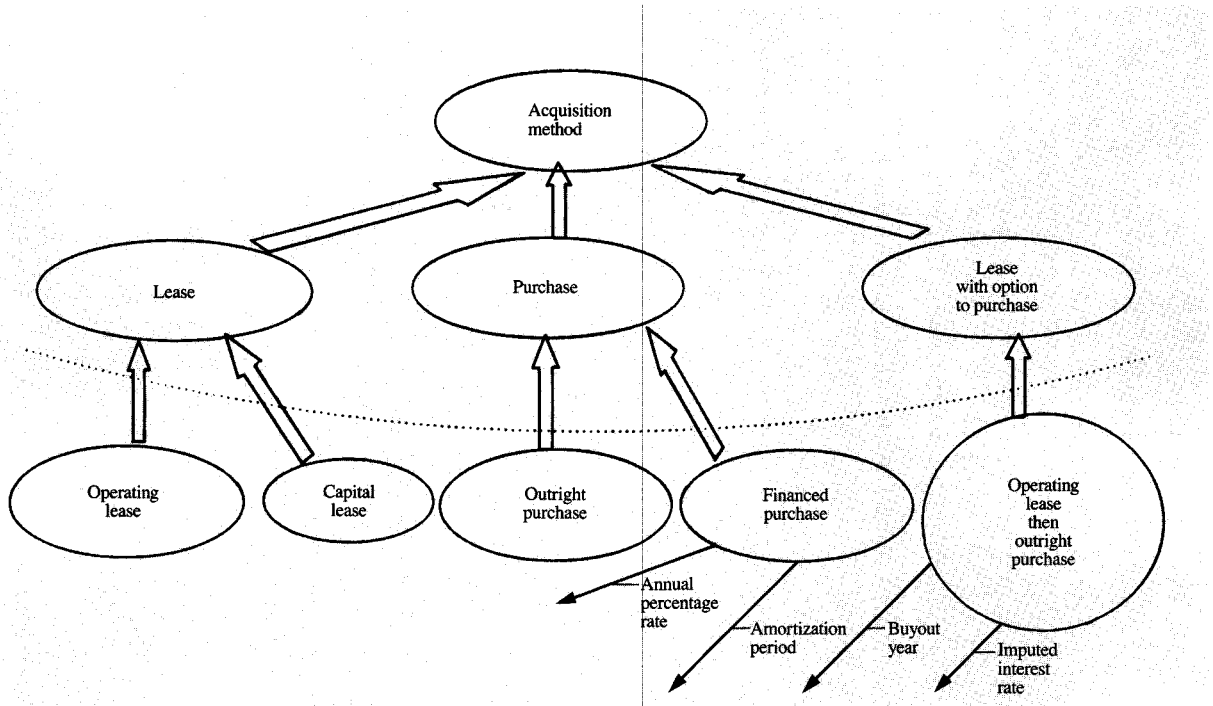

Figure 10

Modeling mainframe computer financing.

various financial ratios. For example, the *financed purchase* object would have roles including *annual percentage rate* and *amortization period*, while *operating lease then outright purchase* would have roles including *buyout year* and *imputed interest rate.*

The next step in building the model is to relate the computer financing mechanisms to some other relevant objects in the domain. In the computer mainframe market, the mainframe manufacturer or marketer will typically present proposals tailor-made to satisfy some customer's requirements for mainframe computing. The financing method therefore needs to be related in two ways. The first relationship would be with the marketing proposal, which would link the financing to a specific mainframe product, thereby allowing instantiation of financing terms and conditions. The second relationship would be with the customer, which would allow instantiation of objects/roles that denote how the specific proposal affects a variety of the customer's financial numbers, some of which may be of concern to the customer and some not. A detailed illustration of this tie-in is indicated in **Figure 11**. Note that this picture is now that of a more comprehensive model that encompasses a seller of mainframe computers, a buyer of mainframe computers, and the resulting financial ramifications for the buyer. Another powerful feature of object-centered representation that is well put forth in this illustration is that of "self-containedness." That is, all relevant financial analyses and other related financial computations are present in this object-role network in the form of function value-restrictions on roles of appropriate objects. This makes the model very self-descriptive.

The cash-stream computation step of the financial calculation does the financial analysis on all proposals in the analysis. The financial analysis computes the impact of a proposal on the customer's budget, cash flow (before tax), cash flow (after tax), and the P&L statement. The impacts are done on a monthly, annualized (on a fiscal year basis), and cumulative basis. Once cash-stream computation is done, the results are available for inspection in a variety of formats and available to other problem solvers for further reasoning. An example of the results of a financial analysis of the equipment plan in Figure 9 being presented to a user is shown in **Figure 12.**

Cash-stream computation for a proposal is necessarily performed only if it is the first time it is being computed. Later, if some input parameter to the computation has changed since the last time the computation was done, only those parts of the computation which depend on this

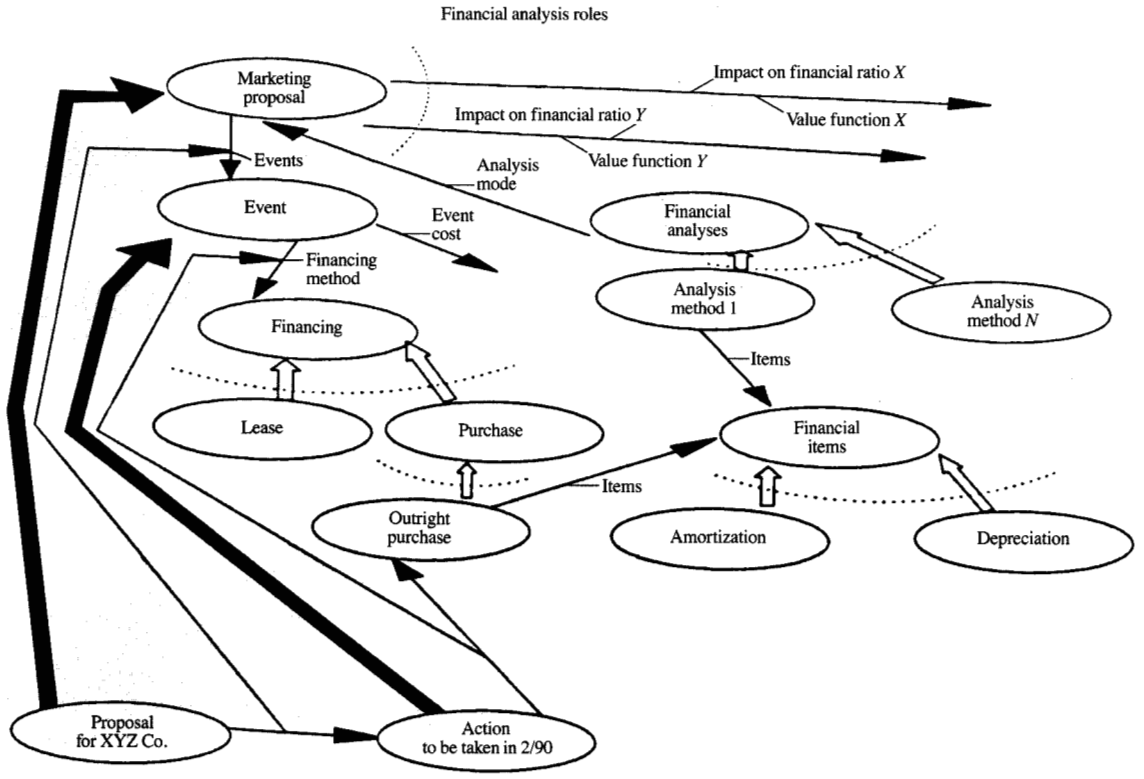

Figure 11

Modeling financial ramifications of a mainframe acquisition.

parameter are recomputed. This feature exploits the K-Rep caching mechanism for truth preservation. A proposal's cash streams are computed by computing individual cash streams for each of the proposal's events and then combining them. An event's cash streams are computed by selecting all the included TCC (total cost of computing) items for that event, computing their monthly cash streams, and then performing financial analysis on those streams.

Impact analyzer

Impact analysis as a problem-solving technique is widely used in engineering and business problem solving. This technique is particularly useful when mathematical models control some aspect of overall domain behavior. One may think of these models as black boxes which exhibit some kind of observable behavior and are controlled through some set of defining parameters. Impact analysis in such domains may be thought of as mechanisms for experimenting with key defining parameters for achieving desired model behavior. A human expert at impact analysis is usually very knowledgeable about the mathematical characteristics of quantitative models that play a role in a domain. This expertise permits the conducting of experiments on defining parameters in a controlled fashion.

Using **Figure 13** as an illustration, assume a domain model with a defining parameter set X. The observable behavior of this domain model may be identified with some function $F[X]$. The problem typically confronted in the course of problem solving is, given that $F[X]$ is to exhibit a particular required behavior, to what should the defining parameter set X be set? In a sense, the expertise to solve this problem is what one would like to model and provide in a computer-based tool.

The FAME impact analyzer uses a heuristic-based goal-directed algorithm that provides this specific problem-solving expertise. The desired output behavior of a numerical model is a goal for the automated reasoner. The reasoner uses explicitly coded heuristics to guide its

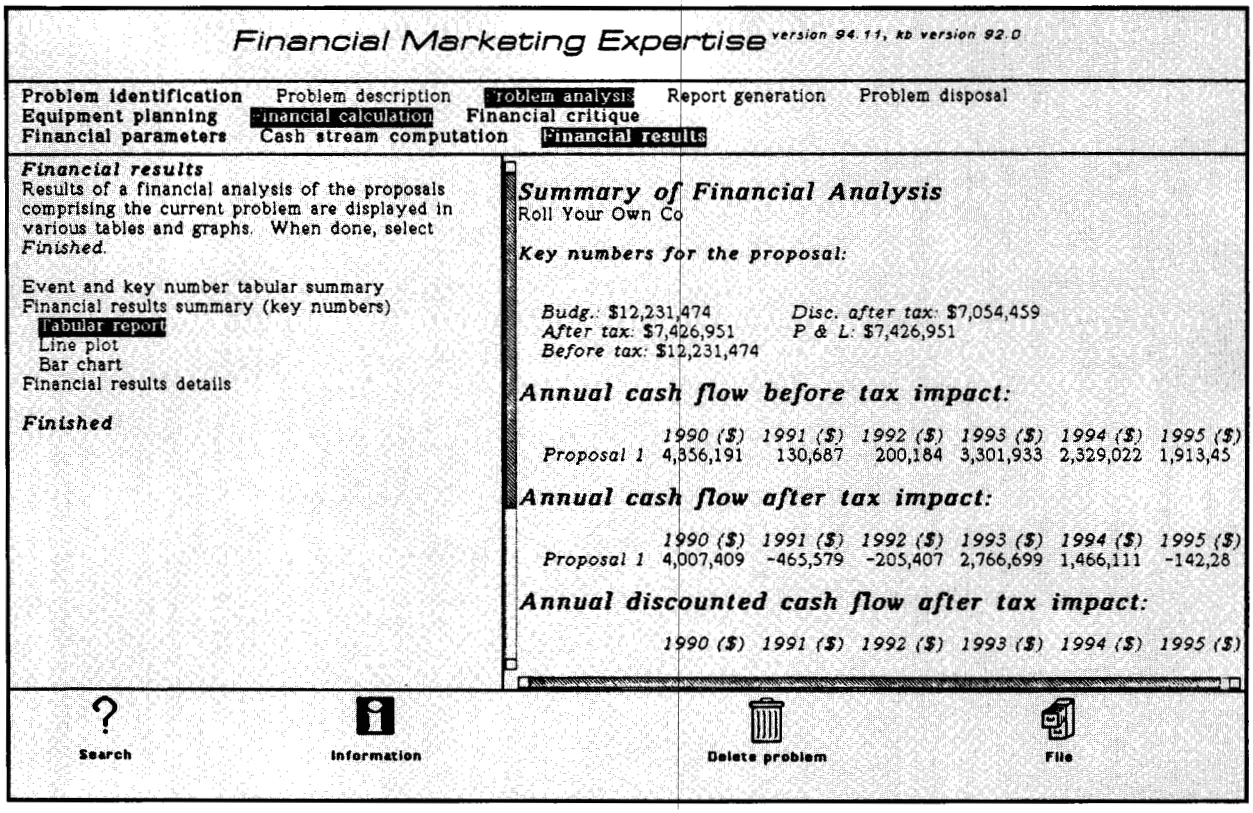

Figure 12

Financial analysis presentation.

search, which is done using a conventional numerical root-finding technique (the secant method). The heuristics that guide the secant method are *a priori* enumerated and specified for all possible quantitative models that FAME may employ in the course of its problem solving.

We describe in brief the secant method with the aid of **Figure 14**. (The secant method is described in textbooks on numerical analysis, e.g., [9].) The x-y plot in the figure is that of a function $y = f(x)$. Assume that for a given value of x, say x_0, the function produces a value y_0. Assume that the function f is unknown, i.e., a black box. The problem then is, if the function is required to produce a value y_*, what value of x, say x_*, will satisfy the relation $f(x_*) = y_*$? The secant method proceeds as follows. Increment x_0 by a small amount δ to x_1, and plug this into the function to obtain y_1. Plot a straight line using the coordinates x_0, y_0 and x_1, y_1 (the *secant* between x_0, y_0 and x_1, y_1). Find the value of x that satisfies y_* on this straight line, say x_2. Substitute the obtained x_2 in the function f to obtain the corresponding y_2. If y_2 and y_* are within desired limits of tolerance, we have obtained our answer ($x_* = x_2$). Otherwise, we plot a straight line using the coordinates x_1, y_1 and x_2, y_2 and repeat the cycle. The example in the figure shows this iteration. This method has been shown to work effectively on monotonic functions, with guaranteed rapid convergence.

This approach is used effectively in the FAME impact analyzer [10, 11]. All of the quantitative models in FAME are monotonic in behavior. For each of these models, we have broken apart the defining parameter set X into a set of mutually exclusive parameters. The impact analyzer invokes the secant method individually for each specific defining parameter. The results of the impact analyzer are in terms of what each individual defining parameter should be to achieve a desired $F[X]$. Domain knowledge (heuristics) is used for *a priori* partitioning and identification of the defining parameters and observable behavior into individual parameters.

This analysis is very useful when performing financial analyses of computer acquisitions and computing their impact on corporate tax books, profit and loss statements, and department budgets. The complexity of these financial analyses is quite overwhelming, since many different financial methods exist for acquiring computer mainframes.

Each financial method corresponds to a mathematical model with its own peculiarities and differing defining parameters. The impact analyzer allows a FAME user to examine how critical (and minimal) adjustments to a marketing proposal will allow it to be within some customer-desired budgetary, cash flow, or profit and loss impact levels. **Figure 15** illustrates the use of this component of FAME.

The impact analyzer described here is based upon a univariate approach. The univariate approach offers insight into the changes required to key input parameters in an individual, isolated fashion. In practice, goal values are rarely obtained by just modifying one individual input parameter at a time. Rather, required changes are obtained by making simultaneous changes to one or more input parameters. For a given output value, there are generally an infinite set of input changes that will produce the output. Given that $[X]$ is a collection of continuous valued variables, it is usually impractical to determine all the possible values of $[Y]$ that will satisfy a relation $y = F[X]$. Instead, users are normally interested in obtaining one preferred change to $[X]$. Under the assumption that the preference is for the least costly (in some sense) simultaneous change to $[X]$, we have devised and are currently experimenting with more general methods for mechanizing goal-directed analyses for numerical decision models [12].

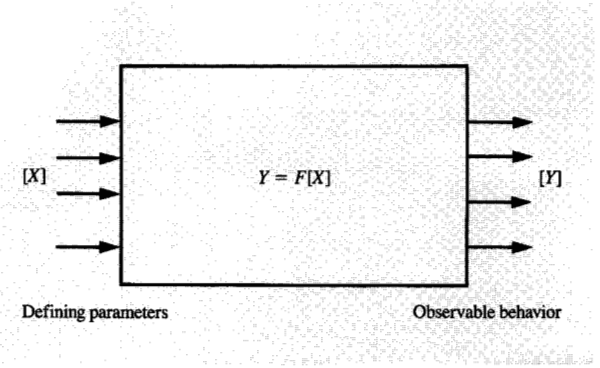

Figure 13

Reasoning about a quantitative black box.

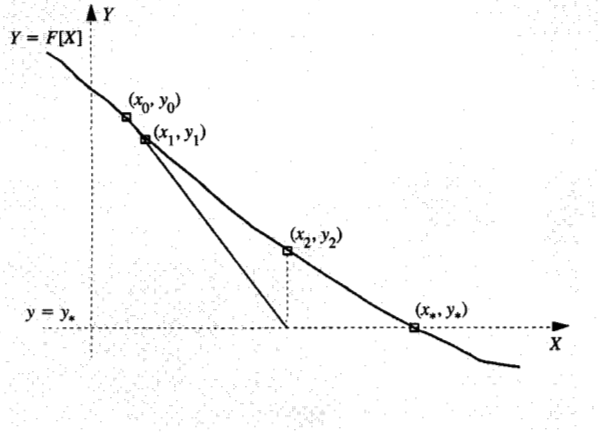

Figure 14

Impact analysis using the secant method.

Marketing incentives support service

Marketing incentives can be provided to a customer in conjunction with a possible mainframe sale. There is considerable leeway in developing an incentive offering for a particular situation, and a marketing representative works closely with an expert to design a particular incentives offering. Support for the marketing incentives service is incorporated within the FAME environment via a specialized problem solver [13]. We call this problem solver MISS (Marketing Incentives Support Service).

One concern of companies that purchase computer mainframes is how to finance the acquisition of these capital-intensive goods. Accurate evaluation of the trade-offs a firm must make in this choice is of prime importance because the decision regarding investment in such capital goods often has significant long-term effects. It is, therefore, not surprising that customers exhibit significant interest in financing plans proposed by a mainframe vendor.

FAME supports a user not only in identifying the most appropriate mainframes but also in formulating an attractive financial plan to acquire or upgrade a particular system. FAME captures the specialized nature of the computer mainframe market so that a user can take account of competitors' actions to generate competitive financial plans. When a financial plan generated by FAME seems uncompetitive or unfavorable to the customer, MISS provides a way to make a mainframe marketing proposal more attractive in the marketplace. Marketing representatives can derive conclusions for eligible customers regarding the acquisition of mainframes by comparing plans with and without the incentives suggested by MISS.

The MISS implementation consists of three components: knowledge modeling, cooperative problem solving, and a knowledge-updating utility. The problem solver captures the specialties of the marketing incentives techniques. The

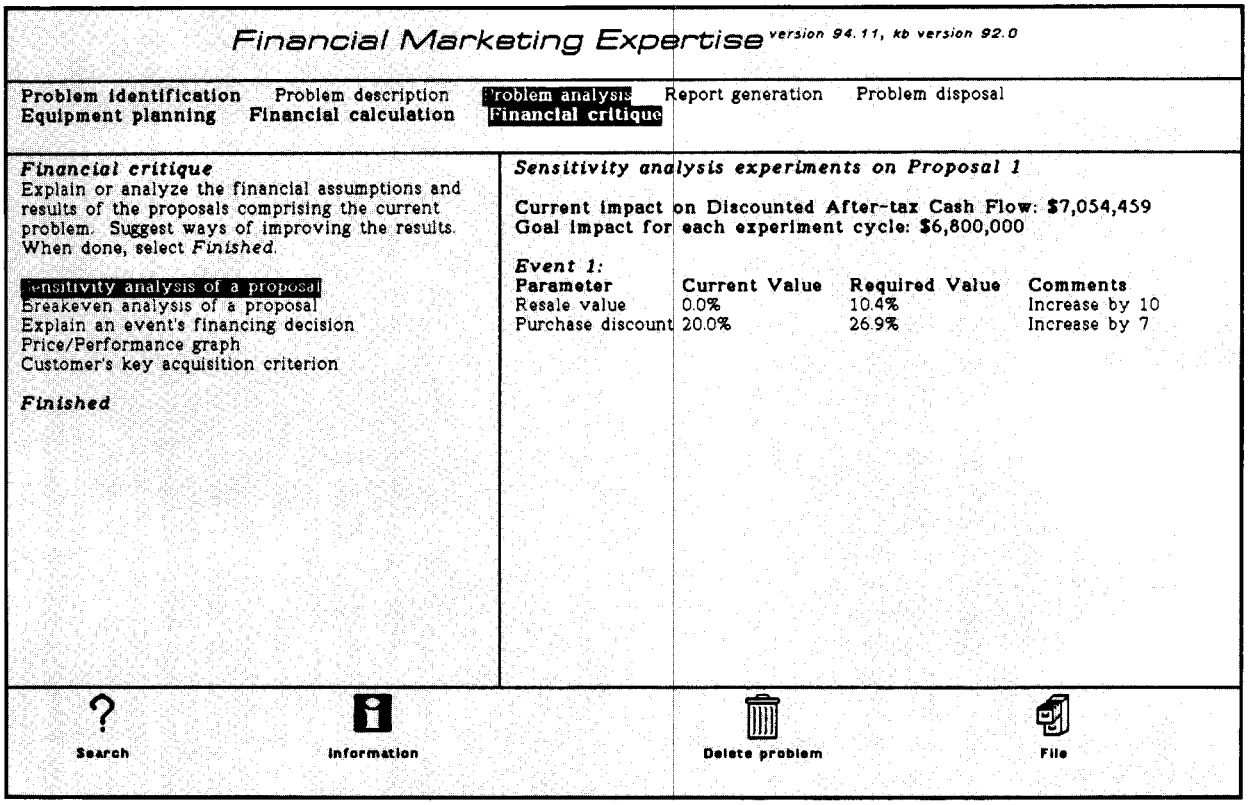

Figure 15

Using the impact analyzer in FAME.

knowledge base for MISS extends the FAME knowledge base substrate by modeling techniques for current marketing incentives. The knowledge-updating utility allows a user to easily upgrade specifics about the marketing incentives techniques when they change (which could be very frequent).

In essence, MISS must support a three-part problem-solving strategy. First, on the basis of various characteristics about the customer and the marketing proposal under consideration, an evaluation is made as to what incentives the customer may be offered and with what limits. Second, an outline of how the customer may choose to use the incentives is designed. Finally, two versions of the cash-flow analysis are performed, to illustrate the ramifications of the marketing proposal without and with the incentives included.

Knowledge modeling MISS stores its own specific knowledge in a knowledge base that is built on top of the FAME common substrate of knowledge. Underlying concepts, rules, and heuristics of MISS have been compiled from domain experts and modeled either declaratively or procedurally. The key concepts of MISS regarding the eligibility of incentives, for instance, are primarily represented as K-Rep concepts or objects. A mainframe sale that may be eligible for incentives might be represented in K-Rep as shown in **Figure 16.**

This chunk of knowledge defines that, for a customer to be considered as a potential candidate for an incentive offer, he must purchase from *eligible-processor-family* defined in the FAME knowledge base substrate. In addition, it defines that the offered incentive may be used in a manner defined by the *current-incentive-offerings* family of concepts.

Once it is determined that a customer is eligible for certain incentives, MISS computes the incentives. The prime factors upon which the evaluation is based include qualifying operating systems and versions, the processors from which the workload is being migrated, types of new applications on qualifying processors/upgrades, first installation of qualifying systems in a qualifying location, and the installation of designated qualifying hardware products.

For example, knowledge specifying those installations which use operating systems that will result in incentives qualification is illustrated in **Figure 17.** Knowledge that, if

OS-A is installed in a qualifying system as an operating system, the size of an incentive package increases by Y units is also represented in a concept, as shown in **Figure 18.** Basic knowledge of how and where to utilize the incentive is also modeled in a similar manner.

Cooperative problem solving Problems in domains such as financial marketing may not have optimal solutions. Very often the choice criteria are subjective. It is almost impossible for a problem solver to capture such subjective criteria with ease. MISS takes an alternate paradigm, that of cooperative problem solving between a user and a problem solver. Through an interactive mode, a user can productively guide the course of problem solving.

The design and implementation of MISS touches upon two related issues in cooperative problem solving. One issue concerns the extent to which a user and a problem solver participate in problem solving. The coordination of such participation is another issue addressed in MISS. One extreme case of cooperative problem solving is to have a user take the lead completely in problem solving. This mode of operation undermines the fundamental reason for developing a knowledge-based system. In addition, this approach can be an inefficient and time-consuming way of solving a problem. The other extreme case is that in which a problem solver is completely automated. Most existing knowledge-based systems have been developed, implicitly or explicitly, using the latter approach. This approach, however, tends to be incapable of listening to a user's insightful advice on a direction toward a solution. Ignorance of productive advice can be prohibitively expensive when the size of a search space grows at an exponential rate. MISS, like the MEP, has a search space of an exponentially large number of possible utilization patterns. It therefore employs a mixed method, or a cooperative problem-solving approach.

An instance of this use is in MISS's support for designing outlines of how eligible incentives may be used. A potential customer may use eligible incentives in a variety of ways. Instead of trying to find an optimal way to exploit the incentives, which is highly subjective, MISS lets a user construct the utilization pattern based on his or her own preference, while enforcing constraints during the process. MISS initiates the process of constructing a utilization pattern by generating a feasible pattern based on greedy heuristics. A user then may either accept the pattern or modify it according to his or her preferences. The process of modifying the utilization pattern continues until both the user and the problem solver accept the solution.

Knowledge-updating utility Criteria for determining eligibility of incentives are stored in the MISS specialized knowledge base. This knowledge is subject to frequent

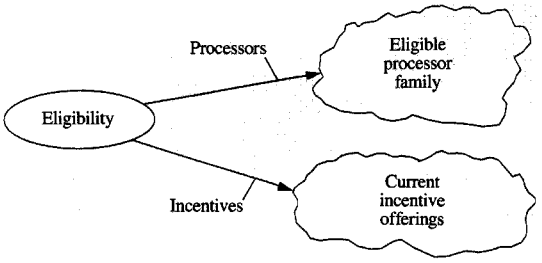

Figure 16

The notion of eligibility.

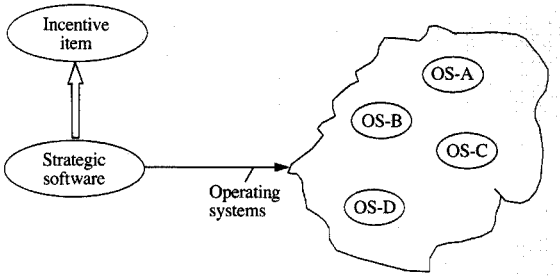

Figure 17

Operating systems qualifying for incentives.

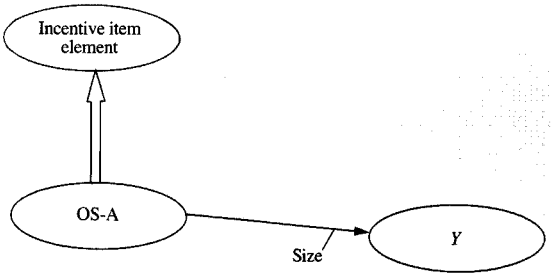

Figure 18

Incentive sizing.

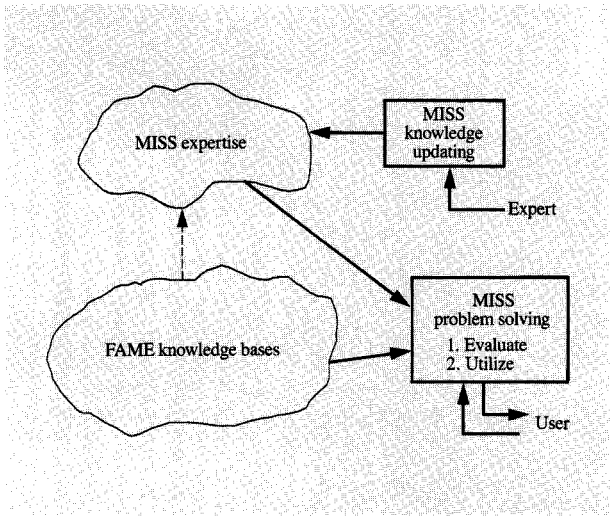

Figure 19

Knowledge flow between FAME and MISS.

change in response to changes in prevailing market factors such as price or actions taken by competitors. MISS provides a knowledge-updating component so that such changes can be reflected in the MISS knowledge base with ease in an intuitive manner.

Knowledge used by MISS is represented in a declarative style using K-Rep, much like the knowledge in the rest of the FAME knowledge base(s). Even though representing knowledge in a concept-based language facilitates access to, and retrieval of, knowledge based on semantics, the language is seemingly unnatural to ultimate MISS users. Looking at matters through users' eyes, the knowledge-updating component of MISS provides to them a front-end interface that communicates in the language of the domain and hides the syntactic details of the underlying representation. The knowledge-updating component accepts knowledge in domain terminology using a menu-driven appearance. It converts the knowledge into a K-Rep formalism, so that MISS can use up-to-date knowledge in its problem solving.

Figure 19 illustrates our notion of the structural relation of MISS to the existing environment of FAME. A solid line represents knowledge flow between FAME and MISS, while a broken line indicates that the knowledge base of MISS expertise is built on top of the FAME substrate of knowledge. The large volume of domain knowledge stored in the FAME common knowledge bases, such as the mainframe product knowledge, the structure and function of a marketing proposal, and financial analysis techniques, made the rapid prototyping of MISS proceed with maximum efficiency.

Project issues

- *Performance issues*

For FAME to be acceptable to the intended user community, it had to be easy to use, provide function beyond what was available with existing tools, and do so at interactive speed.

It should be pointed out that much of the discussion during the course of the project concerning performance was, in truth, about "perceived performance." Users often expressed a desire for improved performance. However, what they were usually comparing were modest computations that they were used to performing, as opposed to the use of FAME to generate one or more equipment plans and to price out these multi-event plans over a five-year horizon. The comparison might better have been between the way they were currently solving such problems (which might involve trying to contact experts at headquarters locations, or using tools which were not integrated and, hence, involved having to copy data from the output of one step to serve as the input of another step, etc.) and the time spent doing problem solving using FAME. One of those who evaluated FAME in a field trial stated that he "just spent three weeks doing [an] analysis that took 30 minutes [on FAME]."

The system was modified to ease the sense of disquiet on the part of the users during somewhat longer computations that "nothing was happening." Progress bars were added to show that the system was in different stages of a problem solver, and some computations were forced to be carried out at earlier points in a session.

One of the early results of our attention to performance was the inclusion of the caching mechanism in K-Rep, the knowledge representation service. As an example, consider a situation in which a base processor, followed by a subsequent upgrade, is being proposed, and the five key financial numbers have been calculated. If the user then wishes to consider the effect of using a different discount for the base processor, the use of the caching mechanism allows the required recomputations to be done in a fraction of the time required for a full recomputation of all values.

- *Validation and testing*

The task of testing the FAME system presented a considerable challenge to the project members. From the earliest stages of the project, it was clear that during the life of the project there would be continuing incremental development, intermixed with a constant requirement to demonstrate capability. The requirement for continuing development made it difficult to adhere to a constant test regime. Furthermore, a tight relation had to be maintained between the evolving system and its test suite. On the other hand, we were greatly assisted by the enforced semantics of the K-Rep network itself, which supports a

knowledge-engineering discipline and emphasizes consistency and completeness.

Both K-Rep itself and the FAME application built on K-Rep had to be exercised. In addition to a test suite of application scenarios, a test suite which exercised the various facilities of the knowledge representation service was constructed and has been used on a regular basis.

In early 1988 FAME began to be deployed to sites other than Yorktown. The system was demonstrated and test-used on a frequent basis, while still undergoing significant enhancement. Prior to deployment, interactive exercising of FAME was instituted. This exercising not only compared answers (typically, any generated equipment plans, and five key financial numbers: cumulative cash flow before tax, cumulative cash flow after tax, cumulative discounted cash flow after tax, cumulative profit-and-loss impact, and budgetary impact), but, of equal importance, established that the problem-solving control flow remained correct.

As the scope and complexity of FAME increased, in addition to interactive testing, it became necessary to exercise the system automatically and carry out regression testing in batch mode. We had been accumulating marketing scenarios from the experts we interviewed and the users who tested our system. Among some newly constructed test cases were ones whose financial answers had been certified by being derived using FINPAK*, the mainstay financial tool which is currently used by IBM marketing personnel. Most of the scenarios were available as stored cases which could be loaded by the FAME system. As improvements were made to FAME, or changes were made to the structures in its knowledge base, some of these cases required minor changes. An example of such a change was a modification in the representation for dates.

For the automated test facility, some 25 rather complex and varied test cases were collected. An initial iteration over the test cases produced for each case a single file which combined the stored case and the set of tests to be performed. The accumulation of the set of tests to be performed made use of the facet capabilities of the knowledge structures. When certain roles of concepts were assigned *test facets*, the reference values of those roles were computed and saved as part of the case. Then, when a test case was loaded, and the values of these roles were computed, they were compared to the reference values. Thus, the test facility took advantage of the knowledge base structure to construct test cases, and could maintain a tight relation between test cases and knowledge base models.

For some cases, finer-grain testing was used, in which not just final values but also intermediate values were checked. The test facility also had the ability to check the result of any function. Thus, when any discrepancy was detected by these particular cases, more detail was provided to determine the cause. During the period in which this automated test facility has been running, a number of discrepancies have been pointed out, some of which have indicated problems in the underlying knowledge base, or in the role-value functions involved.

```
7/27/1989-at-16:54:36 fame version 84:20  krep version 38:11  kb version 75.2
processing case CASE00 name Ray's Follies
processing case CASE01 name Fame Test 1
  . . .
processing case CASE09 name Different Financial Alternatives 3090  200S

  Discrepancy in Test Cumulative Cash Flow Before Tax of
  Proposal True Lease Then Financed Purchase
  Current value     6,643,107
  Reference value   6,766,700

  Discrepancy in Test Cumulative Cash Flow After Tax of
  Proposal True Lease Then Financed Purchase
  Current value     4,033,695
  Reference value   4,108,740

  . . .
processing case CASE10 name XYZ Inc.
  . . .
```

Figure 20

Excerpts from a test run log.

Our first phase of testing has focused solely on the activities of the financial analysis problem solver. A run of the automated tester causes the 25 test cases to be cycled through to test various parts of the financial analyzer. This testing regime is executed automatically each night and has been in use since April 1989.

Figure 20 shows an abbreviated log of a test run.

The log shows that two cumulative cash flows, before tax and after tax, are now different for the True Lease Then Financed Purchase proposal of Case 9. The grain size for the nightly tests can be varied but we typically are only comparing final results. On review of this log, we often compare intermediate results leading to the cash flows in question.

In addition to validating FAME, the testing activity also gathered timing information. Thus, testing was also able to point to either improvements or degradation in performance caused by recent modifications to the knowledge base and to problem-solving routines.

In contrast to the financial analyzer, whose answers can be monitored and verified through the comparison of five

*FINPAK Users' Guide, IBM Corporation.

key numeric values with reference results, the equipment-planning problem solver, being a design tool rather than an analysis tool, can produce multiple equipment plans using its criterion of attempting to achieve plans to meet anticipated growth at near-minimum cost. Our current approach is to compare a generated equipment plan for a test case with a previously developed reference equipment plan. The actual comparison is made by comparing two arrays of numbers—the month-by-month capacities provided by the two equipment plans—as well as by comparing the five key financial numbers associated with the plans.

• *Field evaluation*

Since one of our goals was to have the system which evolved from our research efforts actually be used, and be a part of a technology transfer, it had to undergo evaluation by potential users. Input from the potential beneficiaries of the project and from the eventual owners of the project would be needed to support a business case. To this end we engaged the services of an organization within IBM Business Systems that tests new products in model marketing branch offices in the field. This group planned a series of tests of FAME in the field with representatives of the actual user community. The group within IBM that would eventually take over the maintenance and development of the project helped conduct the field tests and provided cost estimates for a business case. The FAME system was deployed for evaluation at the IBM Chicago Area office in November and December 1988, and at two branch offices in Chicago and two branch offices in Los Angeles in the first quarter of 1990. During these two periods, more than 100 users evaluated the system. The users were given a short questionnaire at the end of a half-day session with the system, and their responses were collected for the business case.

In addition to helping build the business case, these tests were also critical to the success of FAME. Evaluation by members of the intended user community, individuals responsible for marketing IBM's large mainframe product line, was a key input to recent system enhancements. Feedback from the November–December 1988 tests occupied the developers for the following 12 months, and led to considerable improvement, and a complete redesign of the control aspects of the system.

The 1990 field tests confirmed the 1988 tests and indicated that, when fully deployed, the FAME system could enhance considerably the abilities of a large-systems marketing representative or systems engineer. Again, the use of the system on "live" cases gave both an indication of the benefits to be derived from FAME, and, of equal importance, feedback to the developers from a user's point of view.

• *Integration issues*

The weakest link in most systems is the interface to the outside world. In our case, this included 1) the data obtained from various sources that drove the analyses, 2) the interfaces among the various problem solvers and to other code such as the operating system, network, print spooler, file system, and knowledge base, and 3) the user interface which provided interactive menus and mouse handling along with tabular and graphical output. The interface issues were at the same time the most attractive and the most troublesome aspects of the system.

Another attractive attribute, and perhaps the greatest use of expertise in the system, was the distribution of heuristics peppered throughout the knowledge base on all of the objects representing required data that would fill the gaps and smooth the inconsistencies. Thus a user could complete an analysis literally without any knowledge. The analysis could be provided with specifics where known, and the other knowledge would be filled in automatically on demand. This enabled the user to get a "ballpark" estimate of the problem in a few minutes without worrying about the details. One could always go back and provide details to override the defaults to later refine the analysis.

Because we had partitioned the domain, and the various problem solvers for each partition were developed relatively independently and at the same time as the base knowledge representations, the job of integration was extremely difficult. Quite often the interface code was initially much larger than the problem-solving code. However, as the integration continued and the expressive power of the base knowledge representation increased, we naturally found accommodation. Making the integration appear seamless to the user provided stimulus for synthesis of knowledge techniques. Sometimes objects were merely represented by the collection of attributes required by each of the problem solvers. But more often, multi-use attributes gradually evolved which made the representations much more generally useful.

We revised the user interface more often than any other part of the system. We were admittedly naive about our user community, and soon found that the users did not read instructions on the screen and would only skim the user guide, had little patience, were easily frustrated by an interface dramatically different from those to which they were accustomed, and typically had unrealistic expectations about the capabilities of an expert system.

• *Adversities*

Once the system showed potential as a tool that could be used by marketing representatives, the issue of attracting support for the continuing development of FAME and its eventual transfer became important. It took two rounds of field trials and strongly favorable evaluation reports before the final acceptance of the project began to take shape to

	1985	1986	1987	1988	1989	1990
Focus	IDV	Concentration on task, tools	'87 Prototype	'88 Prototype	'89 Prototype	
Tools	Mainframe based Rules Character graphics and windows	Workstation based Semantic net DB considerations	Objects + rules Sequenced menu windows	Object-centered interface	Animated spreadsheet windows	
Enhancements			Value caching Financial analyzer Financial calculator	Equipment planner Impact analyzer Information browser Testing	Role-value constraints Roadmap	MISS
"Selling"		Demos at marketing rep conference		Field evaluation (one city)	Field evaluation (two cities)	
Domain coverage	Shallow		Finance only	Finance + equipment planning	Broader finance + equipment planning	Previous capabilities + MISS

Figure 21

FAME project time line.

the point of organizing a group to receive the system for deployment engineering. Support from the user organization waxed and waned. Ultimately, because of resource constraints, FAME was not accepted as a marketing tool.

Had we tackled an application that was of greater urgency in the minds of the users' upper management, an earlier partnership might have resulted. That would also have brought on more pressure to finish earlier in some form, and would naturally have imposed more checkpoints and planned progress. In a way, the lack of massive user-group interest and the corresponding pressure may have been a mixed blessing. Although most of the team members found the constant need to dress up the system for demonstrations and field trials to be quite irritating, we might not even have had a chance to achieve technical advances had an eager user been waiting.

Among some of the more specific difficulties that beset the project were the following: lack of expertise, no permanently assigned experts, constant requirements to have the system ready for demonstrations, new faces in the decision chain in marketing management, and upgrades in the workstation environment with respect to both hardware and software.

Project history

A time line for the years 1985 through 1990 is shown in **Figure 21**. The dates associated with the listed tools and enhancements indicate when specific facilities were judged to be available for use. However, the evolution of these facilities continued long after the stated dates.

In the initial stages of the project, we felt a need to become more familiar with the domain, and at the same time, to demonstrate to our partners and eventually to marketing management the efficacy of an expert systems approach to financial marketing. Our feeling was that the best way to do this was to create a limited version of what the ultimate system might look like. This effort, involving four people, was intentionally limited to a six-month period. The resulting version was called FAME/IDV (Initial Demonstration Version) [14]. This version of the system would correspond to the demonstration prototype stage described by Waterman [15].

The IDV system ran on a mainframe and consisted of more than 700 OPS5 rules, about 40 Lisp functions, and a color graphics interface. It was written as a set of relatively independent OPS5 rule groups. Each rule group covered a specific area of expertise and computation and communicated with the other groups through a set of protocols. This grouping of the rules was originally intended to divide the system into small, easily maintained, and relatively independent subsystems, but this approach later led to computational bottlenecks, with developmental bottlenecks also being foreseen.

The IDV system solved a small subset of the problems in the financial marketing domain, and not in as much detail and depth as the system that replaced it. The domain

coverage was spotty and limited. It addressed problems of only a small subset of large mainframe processors, their upgrades, and replacements, and could only compute the financial impact of straight lease and outright purchase acquisition methods. Even though severely limited in its scope, the initial prototype system generated an enthusiastic response from our marketing organization and was a key factor in expanding the project size and scope. In spite of the limitations exhibited by the IDV system, our partners looked on it as a considerable success and suggested it be the base for a quick development. Fortunately we were able to convince them that we needed a better way.

We therefore set about to utilize a structured inheritance network system, K-Rep, to represent the shared structures of the domain, often found during the development of the initial prototype system. The inheritance mechanisms would allow us to share computation by doing computation on abstract and aggregate objects rather than on each instance. Also, a unique feature of the K-Rep classification facility was that as concepts were added, modified, or deleted, the concept network would be restructured automatically to maintain the consistency of the knowledge base. This reclassification would provide us with a powerful facility for automatically dealing with aggregate objects and multiple levels of abstraction.

With K-Rep providing a representational basis for FAME from late 1986 onward, three stages of development can be distinguished, roughly corresponding to the end of 1987, 1988, and 1989, respectively. These three versions can be equated to increasingly enhanced field test prototypes using Waterman's terminology:

1. '87 prototype: The system made use of a fairly standard window interface. A financial analyzer was available as the principal problem solver. A financial calculator was also provided. The user stated a problem through a rather fixed sequence of menus. Control was exercised through a state transition model which was simulated using the KROPS facility [16]. KROPS provided a rule-writing facility in which the rules were sensitive to both elements in a working memory and objects of the K-Rep network. This version was demonstrated to executive management in September 1987, and was installed in a headquarters office of financial experts in early 1988. This was the first time the system received a test by a user group.
2. '88 prototype: A mainframe equipment planner, impact analyzer, and information browser were added as problem solvers. Mouse-sensitive window-management facilities were added so that displayed objects could be modified or queried wherever they appeared. More systematic testing was introduced. This version was used in the first field evaluation in Chicago in late 1988.
3. '89 prototype: This version contained a "roadmap" window to allow the user to better navigate through and control the problem-solving process. A new capability of maintaining constraints between related roles was added to K-Rep. The deployed systems used color monitors. This version was used in the second field evaluations in early 1990. The '89 prototype, together with the Marketing Incentives Support System, constitutes the current version of FAME and has been described in detail in the second section. A more detailed discussion of the project history of FAME through 1991, and of the issues related to the project, can be found in [17].

FAME is implemented in the Common Lisp language and runs on a Lisp workstation. The source files contain about 100 000 lines, of which the K-Rep kernel is about 7600 lines, the declarations of the application knowledge for the KBs are about 43 000 lines, and the rest is problem-solver specific. About 40 000 lines of code and knowledge base declarations are devoted to the interaction and presentation services. FAME consists of more than 2000 concepts averaging 38 roles (five local and the rest inherited), with a typical case containing 150 to 400 additional concepts. The knowledge base contains configuration information (e.g., console, channels, memory, power, and cooling units), pricing information (e.g., new, used, and sale prices over the last several years), and marketing information (e.g., promotions, discounts) on more than 180 IBM large processors and their competition. It also includes information on more than 1200 processor upgrade paths.

Conclusions

The FAME system is now at a point where it represents a fully functional prototype of a field-deliverable system. We have conducted a series of evaluation studies at various branch offices within IBM U.S. Marketing and Services. The results of our studies were quite positive, indicating substantial potential for productivity and competitive enhancement. The FAME system, originally developed on Lisp workstations, has now been transferred to the IBM RISC System/6000™ environment.

The continually changing marketing domain has required constant modifications in the system. Keeping the system open to change of an unpredictable nature has been a constant challenge. We have found considerable advantage in building layers of domain complexity on a relatively stable knowledge substrate. Instead of directly implementing the expert knowledge in the form of situation and appropriate response, we have concentrated on providing layers of problem-solving specialists that communicate through a common conceptual model. The knowledge bases provide an abstraction hierarchy, so that

particular pieces of information and their corresponding problem solvers can be spliced in or taken out at a convenient level without totally disrupting the system.

The construction of large multi-use and reusable knowledge bases to support knowledge-intensive problem solving is incumbent upon corporations that wish to preserve strategic expertise. The size and complexity of today's computer-based decision-support applications increasingly demands not just data, but knowledge linked with associated problem-solving capabilities. Embedding problem solving in a common conceptual model populated with the required information is an extremely effective technique, as demonstrated by this experiment. Designing, developing, and delivering such systems must still be considered a difficult task, however.

Acknowledgments

The FAME project is the result of efforts on the part of many individuals. The project has had sustained leadership since its inception from Se June Hong of the Research Division and Joseph Parzych of Advanced Marketing Education. Robert Weida of Columbia University has made substantial contributions to K-Rep. Yeona Jang of MIT designed and implemented much of the Marketing Incentives Support Service. We would also like to acknowledge others who have been involved in the FAME effort: Paul Alter, Charles Chae, Tim Daly, Teri Gomez, Stan Hurwitz, Gail Irwin, Connie MacIntyre, Chuck Mason, Richard Min, Mark Rich, Alistair Stone, Eugene Surowitz, and Yoshio Tozawa. Executive support to date has come from Shmuel Winograd and Marshall Balter.

References

1. E. Mays, C. Apté, J. Griesmer, and J. Kastner, "Experience with K-Rep: An Object Centered Knowledge Representation Language," *Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications*, March 1988, pp. 62–67.
2. E. Mays, C. Apté, J. Griesmer, and J. Kastner, "Organizing Knowledge in a Complex Financial Domain," *IEEE EXPERT* **2**, No. 3, 61–70 (Fall 1987).
3. R. Brachman and J. Schmolze, "An Overview of the KL-One Knowledge Representation System," *Cogn. Sci.* **9**, 171–216 (1985).
4. E. Mays, S. Lanka, B. Dionne, and R. Weida, "A Persistent Store for Large Shared Knowledge Bases," *IEEE Trans. Knowledge & Data Eng.* **3**, No. 1, 33–41 (1991).
5. D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd, "Cyc: Toward Programs with Common Sense," *Commun. ACM* **33**, No. 8, 30–49 (August 1990).
6. E. Soloway, J. Bachant, and K. Jensen, "Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-Base," *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, 1987, pp. 824–829.
7. J. Pearl, *Heuristics: Intelligent Strategies for Computer Problem Solving*, Addison-Wesley Publishing Co., Reading, MA, 1984.
8. C. Apté and J. Kastner, "An Object Centered Representation for Financial Analysis," *Expert Systems with Applications: An International Journal* **3**, No. 1, 19–25 (1991).
9. G. Dahlquist and A. Bjorck, *Numerical Methods*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.
10. C. Apté and R. Dionne, "Building Numerical Sensitivity Analysis Systems Using a Knowledge Based Approach," *Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications*, March 1988, pp. 371–378.
11. C. Apté and S. J. Hong, "Using Qualitative Reasoning to Understand Financial Arithmetic," *Proceedings of the National Conference on Artificial Intelligence*, American Association for Artificial Intelligence, Philadelphia, August 1986, pp. 942–948.
12. C. Apté and S. J. Hong, "Preference Directed Reasoning with Numerical Relations in Decision Support Expert Systems," *Research Report RC-18259*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1992.
13. Y. Jang and C. Apté, "Rapid Prototyping Based on Common Substrate of Knowledge," *Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications*, Miami Beach, February 1991, pp. 155–159.
14. J. Kastner, C. Apté, J. Griesmer, S. J. Hong, M. Karnaugh, E. Mays, and Y. Tozawa, "A Knowledge-Based Consultant for Financial Marketing," *AI Magazine* **VII**, No. 5, 71–79 (Winter 1986).
15. D. Waterman, *A Guide to Expert Systems*, Addison-Wesley Publishing Co., Reading, MA, 1986.
16. T. Daly, J. Kastner, and E. Mays, "Integrating Rules and Inheritance Networks in a Knowledge-Based Financial Marketing Consultation System," *Proceedings of the Hawaii International Conference on System Sciences, HICSS-88*, January 1988.
17. J. Griesmer, S. J. Hong, and J. Kastner, "How To Achieve FAME," *Managing Expert Systems*, Efraim Turban and Jay Liebowitz, Eds., Idea Group Publishing, Harrisburg, PA, 1991, pp. 389–421.

Received November 20, 1990; accepted for publication February 18, 1992

Chidanand V. Apté *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (apte@watson.ibm.com).* Dr. Apté is a research staff member at the Research Center. He received a B.Tech. in electrical engineering from the Indian Institute of Technology, Bombay, in 1976, and a Ph.D. in computer science from Rutgers University in 1984. Since joining IBM Research in 1984, Dr. Apté has worked with the Expert Systems group in the Mathematical Sciences Department. His current technical interests include knowledge-based problem-solving applications and classification methods for knowledge acquisition and data analysis. Dr. Apté is an Associate Editor-in-Chief of the IEEE Computer Society's *EXPERT* magazine, and has served on numerous program committees for AI-application-related conferences and workshops. He is a senior member of the IEEE and a member of the Association for Computing Machinery and the American Association for Artificial Intelligence.

Robert A. Dionne *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (dionne@watson.ibm.com).* Mr. Dionne is a staff programmer at the Research Center. He received a B.S. in mathematics from the University of Lowell in 1980, and an M.S. in applied mathematics from Michigan State University in 1982. In 1983 he joined the IBM National Accounts Division, where he worked on development of large business systems; his work at IBM Research has been primarily in expert systems. Mr. Dionne's current interests include term subsumption languages, functional programming, and semantics of programming languages. He is a member of the Mathematical Association of America.

James H. Griesmer *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (griesmr@watson.ibm.com).* Dr. Griesmer is a research staff member at the Research Center. He received a B.S. in mathematics from the University of Notre Dame in 1951, and a Ph.D. in mathematics from Princeton University in 1958. Since joining IBM Research in 1957, he has carried out research activity in the areas of game theory, switching theory, circuit fault diagnosis, error-correcting codes, computer algebra, and, currently, expert systems. He has also served as a member of the technical staff of the Director of Research, manager of the Research Computing Center, manager of a project which developed the SCRATCHPAD interactive computer algebra system, and manager of education and development programs at the Research Center. Dr. Griesmer is a senior member of the Institute for Electrical and Electronics Engineers, and a member of the Association for Computing Machinery, the American Association for Artificial Intelligence, the Mathematical Association of America, and the Consortium for Mathematics and Its Applications. He is a member of the Review Board for the journal *Systems Automation: Research and Applications (SARA).*

Maurice Karnaugh *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (karno@watson.ibm.com).* Dr. Karnaugh received his B.S. degree from the City College of New York in 1948, and his M.S. and Ph.D. degrees from Yale University in 1950 and 1952. He has been a member of the research staff of IBM since 1970. His current fields of interest are heuristic search and knowledge-based systems. Prior to 1979, Dr. Karnaugh worked in the field of digital telecommunications research at the Bell Telephone Laboratories and at IBM. He was elected a Fellow of the IEEE in 1975 for contributions to the understanding and application of digital techniques to telecommunications. He is a Governor Emeritus of the International Council for Computer Communication, which he served previously as Secretary General and Vice President. Dr. Karnaugh is a Distinguished Adjunct Professor of Computer Science at Polytechnic University, where he teaches artificial intelligence.

John K. Kastner *Enterprise Software Corporation, Marina Del Rey, California 90292.* Dr. Kastner joined IBM as a member of the Expert Systems group in the Mathematical Sciences Department at the Thomas J. Watson Research Center in 1983, and was named manager of the group in 1989. He received the B.A. degree in physics from Occidental College and the B.S. degree in engineering from the California Institute of Technology, both in 1974, the M.S. degree in computer science from the University of California at Los Angeles in 1977, and the Ph.D. degree in computer science from Rutgers University in 1983. He is a senior member of the IEEE, and a member of AAAI and ACM. His current research interests include artificial intelligence specializing in expert systems design. Dr. Kastner left his assignment at IBM in March 1992 to join the Enterprise Software Corporation of Marina del Rey, California.

Meir M. Laker *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (meir@watson.ibm.com).* Mr. Laker is an advisory programmer at the Research Center, where he is pursuing research in expert systems design, knowledge representation, and user interface issues. He joined IBM in 1982, initially working on the HANDY course-authoring language for PC-based multimedia education and the TQA natural-language query system for databases. Since 1986, he has been working with the FAME project, a financial marketing advisory system for IBM marketing representatives. He received his B.A. degree in computer science from Yeshiva College in 1979 and his M.S. degree in computer science from the Courant Institute at New York University in 1982. Mr. Laker is a member of the American Association for Artificial Intelligence and the Computer Society of the IEEE. He has lectured at the IBM European Information Systems Center in La Hulpe, Belgium.

Eric K. Mays *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (emays@watson.ibm.com).* Dr. Mays has been a research staff member at the Research Center since receiving the Ph.D. in computer and information science from the University of Pennsylvania in 1984. His research interests are knowledge representation and knowledge base management.