Design of High Performance Help Desk Application and Its Implementation Results

Charles S. Moon
IBM Corporation
11400 Burnet Rd.
Austin, TX 78758 USA
cmoon@vnet.ibm.com

Thomas A. Shore
IBM Corporation
Highway 52 N.
Rochester, MN 55901 USA
tomshore@vnet.ibm.com

Gary Brophy
IBM Corporation
Highway 52 N.
Rochester, MN 55901 USA
gbrophy@vnet.ibm.com

Dennis Koski
IBM Corporation
Highway 52 N.
Rochester, MN 55901 USA
dkoski@vnet.ibm.com

Abstract

The primary purpose of this paper is to discuss the design and implementation of an automated help desk application called "Remote Expert System To Optimize Repair Efficiency" (RESTORE) currently deployed in the IBM Corporation at Rochester, Minnesota. It was designed in 1992 mainly to enhance the support of AS/400 business computer systems. In order to familiarize those readers with the topics being discussed, an overview of the product support strategy will be provided as a part of the introduction.

The RESTORE application is one of the most successful implementations of Knowledge Based Systems (KBS) and Case-Based Reasoning (CBR) technology in IBM history. It is currently deployed and used very successfully in AS/400 product support, saving tens of millions of dollars in support cost each year. This complex system was designed with a blackboard architecture to enable the use of multiple knowledge sources. These knowledge sources include several rule-based subsystems and a large case-based subsystem, all tied together with a blackboard control module. This innovative design takes advantage of the strength of KBS and CBR by using the concept of blackboard architecture that allows these knowledge-based subsystems to coexist and contribute. This paper will discuss the design of the overall RESTORE system along with its successful implementation

Introduction

This application was initially conceived as an expansion of an older "rule-based" application that was already successfully deployed. This deployed application was initially designed and implemented with minor emphasis on rule maintenance. It was limited by the increasing maintenance required to keep rules and data current. In addition, development time to add rules for new knowledge domains was prohibitive. This new application called RESTORE (for Remote Expert System to Optimize Repair Efficiency), was designed to allow quick and relatively easy expansion of the knowledge base to cover new domains and provide for continuous improvement.

This RESTORE application provides a mechanism for locating similar problems and/or solutions and includes rules to govern some segmenting of the database and selection of "problem type" pathways. This application also provides a semi-guided approach to gathering initial information for input (which can be unique for each of many different supported devices) as well as requesting specific detail information for diagnosing the problem based on earlier input.

Repair service and support

Repair process overview. Service support for computer system servicers generally follows a process similar to what is described here. (Some portions are handled subconsciously.) If the support person handles many different types of devices, he must first determine which type he is dealing with on a given problem. Other incident specific data must then be obtained such as device model, hardware level, software level, microcode level, and general area of failure (processor, DASD, etc., if known).

Other problem specific data must be gathered, such as failure messages, error codes, incorrect operation, lack of some function, etc.. This information will generally lead the experienced support person to seek additional information based upon his recollection of similar problems or knowledge of the area of the failure.

The search now begins for a resolution to the problem using the manufacturer supplied documentation, or previously solved problems if an appropriate database is available. The support person would then evaluate all available data and determine which solution he feels is most appropriate. If that solution did not correct the problem, then the solution for the next most similar problem may be attempted. If the problem remains unsolved, then the next level of support would generally be invoked.

Challenges of repair service support. Many challenges face those who perform repair service support. One of the primary challenges is locating helpful information related to the many different devices they support. If access to helpful information is not a problem, dealing with multiple

disparate databases is sure to be an inhibitor. The servicer must remember: what information is in which database, which search methods and syntax to use, and how information is stored in the database to enable effective searching for data similar to the way in which it was initially stored. Knowing or remembering where to search for specific types of data and knowing what is relevant can be major obstacles. Another significant challenge, especially for new or inexperienced servicers is knowing what initial data about the device or failure symptoms should be gathered before the search for the solution even begins

Use of AI technology

The RESTORE system is designed utilizing three distinct AI technology components: Knowledge Based System (KBS), Cased-Based Reasoning (CBR), and Blackboard. The following sections describe these technologies in more detail. The KBS discussed in this paper refers specifically to a rule-based system.

KBS Technology. The rule-based KBS represents the expert knowledge as data or rules within the computer. These rules and data can be called upon to solve the problems when needed. By utilizing the captured expertise (rules), the KBS processes information similar to experts of their domain. In the case of the project described in this paper, the experts are the actual developers and support personnel of AS/400 hardware systems who have a very in-depth knowledge of the system level hardware. They can quickly diagnose the problem and apply fixes to correct the problem. The expert rules were derived from their knowledge of the AS/400 system. The rule-based KBS technology was chosen for this project for the following reasons:

1. Rapid proof of concept and prototyping
2. Ability to handle complex diagnostic routines
3. Easy to maintain and update

There are several modules in RESTORE designed and implemented using KBS technology. All of these modules contribute to solving a common problem.

CBR Technology. All CBR systems include two key ingredients: (1) algorithms for indexing, searching, and modifying cases, and (2) representation of a case. Cases are groups of features with associated problem descriptions that make each case unique and previously implemented problem resolutions.

A casebase is similar to a database, and cases are similar to records in a database. Cases are generally represented by one or more case specific features, which again, are similar to fields of records in a database. However, unlike the database, one advantage of CBR technology is that the case searched and retrieved from the case base need not exactly match the search criteria. The search engine will retrieve the best matching cases according to the user-defined preciseness.

Blackboard Technology. Blackboard-based problem solving is a powerful means of flexibly combining individually developed software systems and modules into a single integrated application. The blackboard approach is based on the idea that a collaboration of several experts are more beneficial in solving a complex problem than a single expert. This concept also reflects the typical AS/400 computer system failure diagnostic process, where the expertise and collaboration of several hardware and software specialists may be required to diagnose a problem. In some cases, this expertise has been captured and represented as rules in knowledge-based systems. In other cases, their expertise is documented as problem/fix tables for each model of AS/400 systems and are updated with new models as they are announced. This technology was ideal for this implementation because of its ability to integrate modularized and independent knowledge sources, and allow them to all contribute to solve the problem.

Basically, the advantages of a blackboard architecture include separation of knowledge processing into independent modules with each module being free to use the appropriate technology to arrive at the best solution with the most efficiency. This is the important component that tied together all the subsystem components of the RESTORE system.

Design of RESTORE

RESTORE Functional Overview

As the name suggests, the RESTORE application is a knowledge-based system, that contains analysis and repair expertise from past experience and uses this information to optimize the future repair efficiency. In the CBR subsystem, the expertise will be in the combined form of development, manufacturing, and field repair experiences structured as historical cases. Using proper rules and case retrieval methods, the system can not only identify the historical cases that exactly match the current case, but also identify the ones that are similar. Based on the user-defined sensitivity, the retrieval mechanism can present solutions to the current problems that are known fixes. RESTORE is a hybrid system that combines the flexibility of a rule base and the maintainability of a casebase. The remainder of this paper will discuss the technology, design and the implementation results primarily focused on the case base module in RESTORE.

Figure 1 illustrates the overall service support process used by the AS/400 hardware support center. It shows how problems can flow to the support center personnel who in turn use the RESTORE application and other corporate databases to solve the reported problem. It also shows how the various portions of the RESTORE application are used

to locate different types of information to solve the problem.

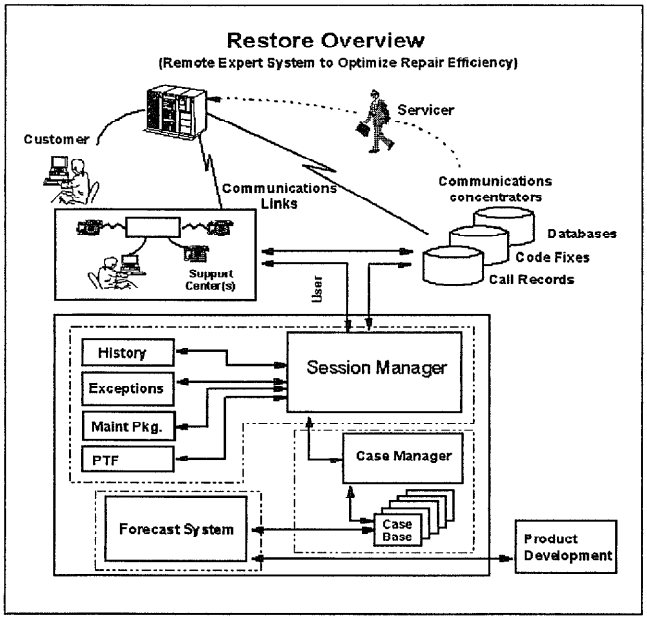

Figure 1. AS/400 Support and RESTORE Overview

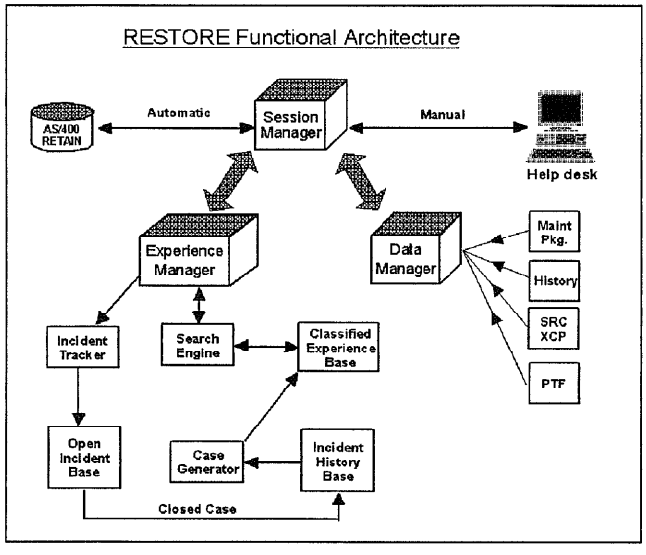

Figure 2. RESTORE Functional Architecture Diagram

Design details

Figure 2 illustrates the functional architecture of RESTORE. The Session Manager includes the blackboard control module. It also contains additional functions to access AS/400 RETAIN database where all the field repair call information is stored. The Experience Manager contains the CBR module with all necessary components for that knowledge base. The Data Manager contains several rule-based KBS modules. The following sections describe these modules in more detail.

Blackboard Control Module. The blackboard architecture in RESTORE has three major components.

1. Centralized global workspace or memory called a blackboard which saves the solutions generated by the knowledge sources.
2. Collection of Knowledge Sources (KS) made up of AS/400 diagnostic specialists that generate independent solutions on this blackboard using KBS, CBR, and problem/fix tables.
3. Blackboard control module which reviews the knowledge sources and recommends the most appropriate solution for a given problem.

This control module plays an important role in managing the knowledge sources in RESTORE. It keeps track of the knowledge source modules and their messages, including the output messages containing solutions. It also knows how to resolve the conflicts in case of inconsistent solutions or contradicting solutions placed on the blackboard by the KS modules.

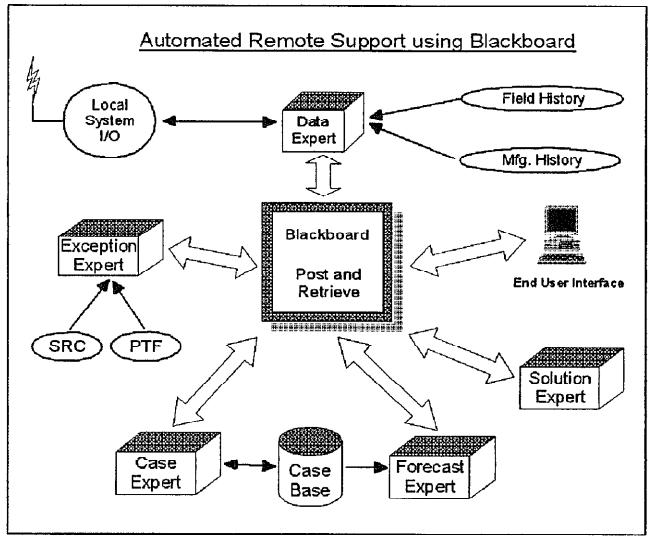

Figure 3. Automated Support Using Blackboard Diagram

Figure 3 illustrates the blackboard architecture in RESTORE. All knowledge sources are identified as experts: Data Expert, Exception Expert, Case Expert, Forecast Expert, and Solution Expert. These knowledge sources post and retrieve information using the blackboard. The control module manages the information in and out of the knowledge sources.

KBS Module. RESTORE was developed as a follow-on to an existing AS/400 rule-based KBS and several other information-access tools. RESTORE was designed to incorporate much of the existing functionality of these other tools, to minimize the development effort, ease the transition to RESTORE, and also facilitate the acceptance of this new tool.

One of the key features of RESTORE is that it performs multiple searches through a variety of different knowledge sources, and gives the user a concise summary of its findings. The rule-based KBS in RESTORE contains four major modules: Data Expert, Exception Expert, Solution Expert, and Forecast Expert. Some of these sources were designed as direct replacements of the previous rule-based KBS. In addition, some were designed to integrate with existing diagnostic tools used by the service personnel.

Data Expert:
The Data expert was enhanced to incorporate new rule-based logic for interpreting, analyzing, and inferring knowledge from the existing information entered. For instance, in the domain of AS/400 repair service and support, the following information could be entered by a user of RESTORE: AS/400 system type = 9406, model = F80, and reference code = 9337D001.

Based on the AS/400 domain, the following additional information can be inferred:

Area of system = Disk, Type = 9337

Each time a user of RESTORE enters new information, the current set of information is analyzed through the KBS rule processing, and any additional knowledge is added to the current domain. This reduces the amount of obvious or redundant information that must be entered by the user, and narrows the search domain, producing more concise and accurate search results.

The additional KBS experts also provide several independent and highly specialized tasks. A couple of these tasks include a lookup into the AS/400 On-line Maintenance Documenation, and a search for relevant and the most current AS/400 Software patch information. Each of these functions will analyze the set of information that has currently been provided either by the user or inferred by another KBS module.

The key information used includes the AS/400 system CPU type and model, operating system release level, and valid system error codes and message identifiers. RESTORE also has access to a machine Serial number cross match database to allow much of this additional information to be gathered. Based on the available information, the search will be either initiated or skipped, if not enough information has been provided.

The Documentation lookup will locate information from the relevant sections of the Maintenance Documentation. The Software Patch lookup, which can consist of multiple software releases, will locate relevant patch numbers, any supersedes or replacements, and an optional Cumulative patch that consists of multiple patches for a given software release.

Exception Expert: The predecessor Expert System stored action plans that were discovered, and not part of the original AS/400 maintenance documentation. Use of the data represented as "exceptions" had been so extensive that RESTORE needed to continue to reference it in its original form due to the time required to convert the large amount of data to another form. Ideally, this information should be eventually converted to stored cases.

Solution Expert: The solution expert is really a logical container within the blackboard module which allows the user to select all or portions of the solutions identified by the various modules and export them to be used by other tools or sent to another person for implementation of the solution.

Forecast Expert: The forecast expert was intended to be a data mining application that would alert the user to potential situations such as a high number of problems on a specific device (by serial number) or alert the product developers of the most common problems reported through the RESTORE application. This module has not yet been completed.

CBR Module. The core knowledge source in RESTORE is the Case Expert and was designed using the Case-based reasoning technology.

Basic Concept: The RESTORE design incorporates the common basic steps of CBR. This section describes how these steps are used in the design of the CBR module.

- ***Create a casebase shell to hold future cases.*** This is the most critical step in building a CBR system. The efficiency of the search engine will be governed by the data structure of the cases and their features.
- ***Add unique cases to the case base using unique features.*** Features are attributes of cases. Adding a unique feature to the case base involves specifying the type of matching which should be applied to the feature and the match and mismatch weights. These feature weights specify how much a match or mismatch on a particular feature will affect the overall score. In some cases, a set of case specific weights may be used to emphasize the importance of a specific feature match or mismatch. Every case must be verified for its uniqueness before adding it to the case base.
- ***Create an index using a data structure that will optimize the search efficiency.*** Although, it is possible to use the case base as-is to search for possible matches, it is not very efficient since not all the searches are looking for exact matches. In order to optimize the efficiency by the search engine, the case base must be systematically packed to load faster and occupy less memory once it's loaded. The method used to pack the index will depend on the scanning method used by the search engine.
- ***Load the index into memory prior to search.*** The time and size of the memory required to load the index will depend on the method used to pack the index. There is no direct correlation in the size of the packed index and the performance of the search engine.

- ***Specify global match parameters such as maximum number of cases to retrieve and the preciseness of the match required.*** These global casebase parameters allow the search engine to retrieve only the top ranked matches with the score higher than the threshold value. These parameters are very critical when this case-based system is used in remote/automated mode.
- ***Preprocess incident parameter to optimize search.*** Without some form of text preprocessing, the search engine would be overwhelmed with "noise" text that may accompany the text based features. Some preprocessing is obvious, such as, convert to upper case, remove word separators, remove ignored words, etc.
- ***Define local match parameters such as match, mismatch, and absence score.*** Once an incident is ready to be presented to the search engine, the final local weight parameters must be defined. One important weight most often ignored is the absence weight. If the incident case contains a feature that is not represented in a stored case, the final match score should reflect a deduction of the absence weight.
- ***Perform search.*** The search engine scans every stored case, compares it to the incident case on the closeness and scores each case accordingly, based on matches, mismatches, and absence. Then, it will rank all the cases on the final match score, and retrieve the top ranking cases above a specified threshold.
- ***Receive pre-ranked matched cases.*** These pre-ranked cases must be identified and stored into memory since there may be multiple searches on multiple case bases.
- ***Redefine local parameters and repeat search.*** The entire search and rank process must be dynamic. If the highest ranked match is not satisfactory, a modification for the incident case feature is needed before repeating the search. If all else fails and the search engine does not return a matched case, then a new case must be added to the case base.

Basic Requirements: The CBR module in RESTORE was specifically designed to optimize the search of a product maintenance database. With proper data structure and index scheme, it was possible to implement this module with only the following key functions:

1. Create compressed index of the casebase to optimize the search
2. Preprocess cases (parse and create search objects, properties and descriptions)
3. Highly optimized search engine
4. Identification of best match
5. Casebase update

The above functions only allow the user to build and retrieve matched case histories manually. Since RESTORE will be eventually operating automatically and remotely, many other functions are needed in order to automatically provide accurate property information on current incidents, track all open incidents and close the ones with verified successful resolution, and finally, provide a case-base maintenance facility.

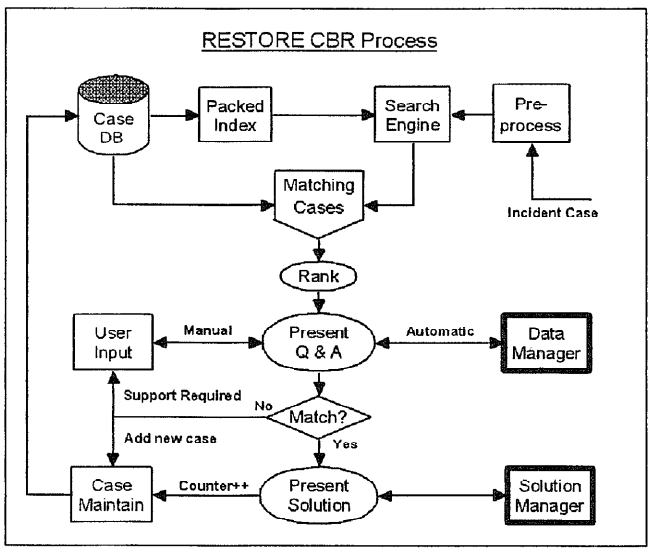

Figure 4. RESTORE CBR Process Diagram

Figure 4 illustrates the process of the CBR module. All the key functions identified in this section are important parts of this process. The "Present Q & A" and "Present Solution" functions are also key parts of this process. They manage the input and output of the case-related information from the CBR module.

Some of these functions rely on "product specific" rules. Many of these functions are being constructed as separate modules to allow for easier separation of the product specific rules from the generic "logical direction" rules. This will allow for adaptation to other problem domains with minimal program changes to the base application. The domain specific knowledge can be added primarily in the form of cases in the case database and external sources for other helpful information such as on-line documentation.

Integration of technologies

The priority in implementing the RESTORE application was to focus the development effort on the core technology - the CBR module. Around this core CBR technology, the remainder of the application would be implemented with standard off-the-shelf technologies for the deployment platform of choice. We chose to alter the blackboard portion of the original design to be a "driver" of the overall session flow rather than a "reactor" to the data presented. Because the RESTORE application was dealing with a somewhat limited domain (servicing a specific family of IBM computers), this would improve the efficiency of the searches. This "Session Manager" as was designed, contained high-level program flow and served to analyze

all data that was presented. This was accomplished through use of both rules and a special set of "cases" in the case base used only by this session manager. Together, these functioned to determine which sources of knowledge were appropriate to search using that data appropriate for the specific knowledge source, and in some logical order.

A DB2 relational database was chosen to store the case data and solutions. The database tables were designed in order to allow efficiency in:

- Creating and updating the specialized CBR indexes
- Identifying and transforming new incidents or queries into new cases
- Allow standard SQL queries into the case and incident data to track problem trends, etc.

Uniqueness of design

One unique feature of this system is the design of its high-performance CBR kernels including the search engine. This new design emphasizes the search performance and its accuracy in a CBR implementation for a help desk application. The resulting performance is discussed in later sections of this paper.

Another unique feature of this system is its use of the blackboard technology in its design. This design allowed the RESTORE application to take advantage of the strength of this technology in the following way:

1. Modularize the development of independent knowledge sources
2. Integrate the new knowledge sources with the existing ones
3. Allow the system maintainers to simplify the management of the updates

The last advantage is particularly important because the product support strategy requires the maintenance team to update many independent knowledge sources for all models of products as they are released. In this domain, there are currently four system types, more than ten models and more than ten software release levels.

Implementation

An iterative step approach was used to implement this project, partially because a large amount of information had already been gathered and represented in a simple rule-based knowledge base. It was already deployed to the target audience. We chose to incorporate that information as-is into this project to speed building the knowledge and provide a base level of user familiarity. As new functions were completed, they were given to a subgroup of the target audience for 'live' testing. This provided early feedback and helped build a core of users who could act as advocates for the new tools.

Application development

As the RESTORE development team and project were pulled together, the first task was choosing where and how to develop the application, taking into account the following:

- Skills of the development team
  - Considerable VM, REXX and Pipelines expertise
  - C/C++ experience
- Application requirements
  - Optimal Search engine performance
  - Rapid development/prototype for User interface routines
  - Support and access multiple and independent knowledge sources
- Database requirements
  - Relational database - DB2, SQL access
- Deployment requirements/restrictions
  - Worldwide deployment
  - Support multiple types/level of user workstation (i.e. Dumb terminals)
- Development Tools/Environment
  - Access control/checkout
  - Build/integration

Development process

Project Plan: The ideas for this project began in late 1991 as a method for improving the effectiveness and coverage provided by the existing knowledge base. After evaluating different knowledge processing methods and commercial tools (coupled with our desire to utilize the existing base of knowledge), we decided to develop a new application to allow us to merge several types of knowledge processing.

These ideas quickly grew to become a full-fledged KBS project and grew in proposed function and scope. The basic concepts and high level design were put together over the closing months of 1991. This began as a pet project that we developed in addition to our assigned duties. Throughout the first half of 1992, we developed the ideas and fleshed out the concepts while building a project plan and business case. Much of this time was spent raising awareness of the concepts and defining benefits to the people who would be most affected.

The business case was first presented to management in June 1992. Within three months a development team was assembled and the design completed. By early January 1993, a limited function prototype was undergoing usability and 'proof of concept' tests with the targeted users. The first full function version was placed in production in March 1993. Worldwide roll-out began in June 1993.

The resource required for this initial version amounted to approximately 4 person-years effort. This includes the low-level designs, coding, test and support of the initial roll-out. Since our development and deployment utilized

the existing corporate processing and communications infrastructure, no costs other than the development personnel were planned or realized.

Justification of this cost: Cost justification for development of this tool was based primarily on the theory that with the proper information readily available to the support center personnel, the duration of the calls should be reduced and the rate at which those problems are fixed on the first call should improve. Additional benefits were expected by reducing the unnecessary replacement of parts.

Some intangible savings, though not quantifiable were also expected. Customer satisfaction with service was expected to rise due to handling problems quicker, and improving the rate at which they were solved on the first call. In addition, feedback of specific field failure data captured through this application would enable product development teams to recognize and correct problems sooner than was historically possible.

Benefits and Validation. Validation was performed using real calls to the support center and was intended to measure two points. First and foremost was validation of the time savings per problem. A secondary point was the parts savings based on use of the knowledge provided by this tool.

After actual problem calls to the support center were handled, a follow-up evaluation was performed by a different support person, using only the information contained in the call record. During this follow-up evaluation, this knowledge tool was used as the first source for information. Actual time required to locate the proper solution was measured in both cases. Parts replacements were also tracked.

A range of savings was identified based on the conservative and potential values of several variables, such as usage rates, effectiveness based on partial population of the knowledge domain, etc.. This validation was performed when only a few hundred cases were present in the casebase. Solving problems for which information existed in the casebase was dramatically improved. The time required to locate the proper information was reduced and consistency in locating the proper information was greatly improved. Since the plan was to add all new solutions (those not found in the casebase when searching for a solution), the overall effectiveness would naturally improve as the casebase becomes more populated. With the relatively few cases at the time of this evaluation, the effectiveness was expected to be quite low. In reality however, since the first cases that were added were those for which many calls to the support center are received, the effectiveness was higher than expected.

The results of this trial showed time savings for the support center person of eight to nine minutes per problem call. This represents a significant portion of the overall time spent on each call. In addition, the instance of parts replacement was reduced by 15 to 20%. Overall, these results represented worldwide savings of between 33 and 40 million dollars (US) in the first year alone. This is the total benefit, including that from the original knowledge base data. This was projected to grow in subsequent years based on increasing the knowledge base (domain coverage) and usage.

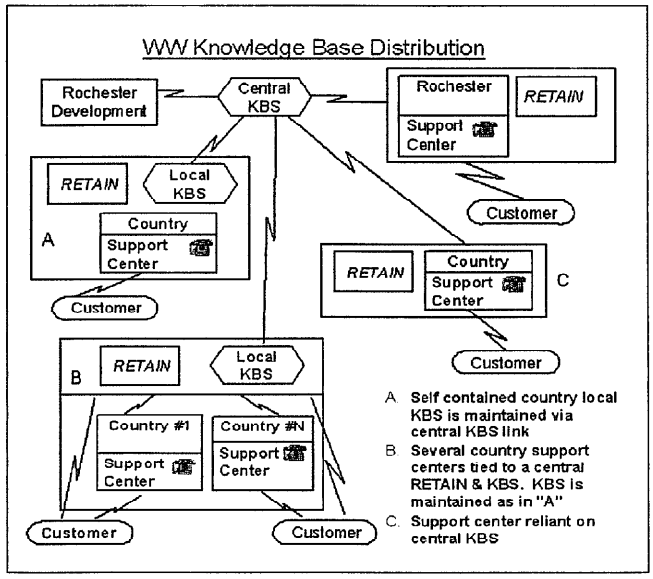

Figure 5. Worldwide Knowledge Base Distribution Network

Application deployment

This application was deployed worldwide and is regularly used by over 2000 service and support personnel. As mentioned earlier, RESTORE is a follow-on project that had the following requirements:

1. Provide a superset of the functions of the earlier Expert System.
2. Deploy worldwide as effectively as the predecessor.

Figure 5 shows a high level view of the knowledge base distribution network utilized to distribute the application set and databases. The remote nodes are then automatically updated worldwide.

Deployment process. This process is used to distribute all executable programs, data files, SQL data for stored cases, and SQL data for incidents (usage information). The programs and data files are automatically distributed worldwide via interconnected VM service machines. The stored cases are developed at a master location, and distributed to remote nodes.

(i) Updates: All data and program updates are made to the master distribution account in Rochester MN, and the service machines automatically update the other world-wide service locations. This process is used for all RESTORE programs, control files, and data files. In

practice, it takes less than three minutes for all the worldwide machines to be updated, once the master machine receives an update. It takes less than 10 seconds for each site to receive and store the update files, so the bulk of these three minutes is due to network delays, primarily between the US and Europe or Japan.

(ii) Project plan: The deployment plan consisted primarily of three parts. The first was training and use by the local (Rochester) hardware support team. This team was already using a simple rule-based system to assist them in solving problems involving a machine failure code. This was a very narrow and finite domain. Our challenge here was to modify their thinking to encompass a much broader scope and multiple problem domains to enable them to realize the benefits this application could provide. This consisted of specific training and demonstrations showing how the flow of this application mirrored the flow they generally followed in solving a problem.

The second was training a subset of the users to create new cases and populate the casebase. Our initial plan was to enable all users to perform the initial building of new cases for any solutions they found that were not already in the casebase. Then a select few (basically, the leaders of the sub teams) would review the cases for technical accuracy and assure adherence to our case structure guidelines. This was soon modified to a scheme where all users are expected to identify new solutions to be added to the casebase, but a core of two or three people actually build the case. The sub-team leaders are still responsible to ensure technical accuracy, but the few case builders are to ensure the case guidelines are followed.

The third major part was training and use at the remote support centers around the world. A User's Guide was assembled and distributed to the leaders of these remote support centers. A demonstration and presentation of the features and flow of the application was made at a meeting of these support center leaders. They in turn would then work with their respective teams to ensure proper understanding and operation of the application.

(iii) Other costs: One cost that is sometimes overlooked is the cost of maintenance of the actual data or knowledge contained in the tools. We chose to have the technical data maintenance performed by the more knowledgeable support people who would also be the users of the tool.

Platform. RESTORE is currently deployed on the VM platform, and updates are automatically handled over the VM RSCS network. This decision was based on the reliability and availability of the existing data and knowledge distribution methods, as well as the VM code development skills that were available at the time.

The main consideration for choosing the deployment platform was the need for worldwide distribution and access. The RESTORE application would be used by AS/400 service personnel throughout the world. VM was the logical choice since the network was available worldwide, and tools were already in place with which to distribute the programs and the data.

Training. RESTORE was developed as an AS/400 Product Support Center Representative (PSC Reps) assistance tool. These PSC Reps have limited knowledge of VM, but are highly specialized in their area of AS/400 expertise. Team leaders for these PSC Reps were closely involved in the design meetings of the RESTORE tool, and this simplified the training process.

The RESTORE developers and the Support Center Team Leaders performed all the training of the PSC Reps. Classes for RESTORE users typically took one hour, and 2 more hours of training were needed for case-builders.

Maintenance

Stored-case data. If a user runs RESTORE, and a useful action plan is not found, the user can enter the action that fixed the problem, and flag the incident as 'case-required.' A service machine automatically detects these incidents, and notifies the case-builders. The case-builders typically take 10 to 15 minutes to research and build a case, and to forward it for approval. Currently, about 8 to 12 new cases per week are being added.

On-line documentation. This data is refreshed with each major AS/400 Operating System release, typically a couple times a year. Due to this low frequency, this is a fairly manual process.

Software patch information. Due to the complexity of the operating system, the number of supported releases, and the number of supported devices and IBM Program Products new patch information is generated every day. Currently, this information is automatically processed and forwarded worldwide, via VM service machines.

Exception information documentation. The maintenance associated with this (predecessor) type of data is very minimal. About twice a year, one support center rep spends a day removing any obsolete data. Any new data of this type is entered as stored cases instead.

Implementation Results

Total cost of implementation. The total cost associated with this project amounted to about 1.2 million dollars (US). This includes the design, coding, testing, training, and maintenance support for three years. This also includes user training and setup and maintenance of four installations worldwide with their associated maintenance including code level and database synchronization. Since we chose to place the knowledge/data maintenance responsibility with the users, the time invested in such maintenance is a relatively low incremental amount above what they were spending to document problems and solutions in notes and other non-knowledge based means.

Functional Results. Overall, the choice of using C and REXX for coding has worked out very well. This has enabled very quick redesigns that became necessary as cases were developed. It is very important that the users of the tool were highly involved in the early design decisions. This, coupled with the planned 'Beta' deployment and the significant usability rework of the final product, meant that no significant functional changes have been required. The primary changes were made between the Beta and final stage. Other than that, several short-cut methods have been implemented to reduce keystrokes.

Knowledge Bases. The transition from the predecessor Expert System to RESTORE dictated that the new tool had to provide all of the existing knowledge, and more. This incremental requirement clearly defined our starting point, and where improvement was needed. All of the data refreshes with a high frequency of change have been automated. As a result, our users have come to expect extremely current data, and the Support Center efficiency is very high.

There are currently over 2000 active stored cases in RESTORE, and new cases are being written at a rate of 10 to 12 per week. These cases have an average of 4 to 6 properties each, and we have found our initial performance projections to be accurate.

Performance: In a live help desk environment, the speed of response can have a high impact on customer satisfaction, as well as morale and productivity. Certain design modifications were made to reduce keystrokes, provide shortcuts, and to speed up the data searches.

One interesting point is the reliability of moving large files over the IBM Remote Spooling Communications Subsystem (RSCS) network. Some of the patch data files are over 100 MB long. A file this large can take up to four hours to transmit across the network, due to scheduling constraints beyond our control.

The tradeoff we have reached is to pack this data to 20 percent of its original size, and add about 0.6 seconds to the real-time data search process. Our users have agreed that this is a worthwhile compromise.

The RESTORE users are concerned with the following performance factors:

1. Minimize the number of screens. Screen refreshes can take 5 to 10 seconds for remote users at some worldwide locations.
2. Minimize screen-to-screen delays. Part of this is due to interactive network delays, as mentioned above. At the master site, most screens refresh within 2 seconds, and two take about 5 seconds to refresh.
3. Minimize keystrokes if possible.

RESTORE design focused on these three performance areas. Aside from interactive network delays, which is a continual challenge to improve, we have been very successful. During the longest screen-to-screen delays, we have added progress-indicator blips, which show the brief description of the task being performed. This has almost universally improved the perceived performance of RESTORE.

(i) Case search engine: The case search design has been very effective. Several minor changes have been implemented, and they could best be characterized as application-specific. The scoring algorithms for properties with multiple value answers were expanded and generalized in order to produce the results expected by the case builders. The scoring algorithm for one particular property, the textual problem description, was fine-tuned several times in order to produce expected scoring results.

(ii) Performance of the design prototype: The case search was designed to be an extremely high-efficiency process, and our evaluations have not changed. Some of the data-preparation work, prerequisite to the search, needed to be modified in order to improve the apparent performance of the case-search.

In light of the application environment, this issue was very important from the start. A target was established of 0.5 seconds to score the initial set of cases, projected to include upto 10,000 cases.

It was possible to easily stay within the target of 0.5 seconds for the ranking of the initial set of case data. Thus, the case-based design will not need further enhancements to satisfy our performance requirements.

(iii) Performance of the final implementation: The case-search is so efficient that performance has never been an issue. The case-search interface screen takes 3 to 5 seconds to be built initially, and even the most sophisticated case search takes less than 0.3 seconds.

RESTORE was initially designed to run as a blackboard, performing data searches in parallel as needed. As it turned out, true parallel processing is available with some VM operating systems, but implementing it would have impacted, and not improved performance. A user typically spends an elapsed time of 3 to 5 minutes running one RESTORE consultation, from start to finish. This includes the time required to gather the pertinent data and enter it into the tool. The tasks that could be performed in parallel currently take 5 to 7 seconds, total, when they are run sequentially.

Lessons learned

Fairly early in the process of building and distributing actual case data, it was determined that some aspects of the RESTORE CBR function needed to be redesigned. These were predominantly related to usability and requests for enhancements. However, the only aspect of the case-search design that was changed was adding additional multiple answer scoring algorithms.

During the development process, the designs were thoroughly discussed, documented, and reviewed. Due to

the variety of input that was received, it was concluded that certain areas of the original design needed to be implemented with the intent to easily modify them later. This added initial resources for the implementation, but dramatically reduced the time required to make design enhancements as a part of maintenance.

Summary

The RESTORE application was designed to solve a product support and help desk problem for the AS/400 business computer systems. It was designed using AI technologies, such as KBS, CBR and Blackboard to solve a very complex problem that is not easily solved otherwise. Although not all of the initial design components were implemented fully, this system has already demonstrated the benefits of using these technologies.

One of the unique features of this system is its high performance CBR subsystem. This CBR module design applies to a specific application domain such as the help desk application for AS/400 systems. Furthermore, the rule-based and case-based hybrid implementation greatly increases the effectiveness of RESTORE to operate remotely and automatically.

Another unique feature of this system is the blackboard architecture used in its design. This design allowed RESTORE to take full advantage this technology by modularizing the development of independent knowledge sources and integrating these new knowledge sources with the existing ones. It also allows the system maintainers to simplify the management of the updates. This is particularly important because the product support requires the maintenance team to update many independent knowledge sources for every model of product as they are released.

Implementation began with a skeleton simulating the various functions and user user interface. This allowed for as much user interaction and user input to the design of both the user interface and functions as possible. New functions were added as soon they were operational to give the user the ability to provide early feedback. This approach has allowed the ability to revise items based on user input and actually provide some functions ahead of schedule. However, this shifted the focus to some domain specific functions and delayed the building of cases. Significant case building has been ongoing for the past 2 years.

References

Charles Moon, Tom Shore, and Gary Brophy, "Design of High Performance Case Base Module for a Help Desk Application and Its Implementation Results", IBM Technical Report, 1997.

Daniel Corkill, "Blackboard Systems", AI Expert 6(9):40-47. September, 1991.

Chuck Williams and Bruce Clayton, "Case Base Retrieval", White paper, Inference corporation, 1994. (http://m5.inference.com/products/cbrwp.html)

Christopher K. Riesbeck and Roger C. Schank, "Inside Case-Based Reasoning", Lawrence Erlbaum Associates, 1989.

David S. Prerau, "Developing and Managing Expert Systems", Addison-Wesley, 1990