Expanding the Utility of Legacy Systems

Joseph McManus
AT&T Network Systems
Dept. 40440, 1600 Osgood St.
North Andover, MA 01845

Teresa Garland
Inference Corporation
4 Parklane Blvd., Suite 470
Dearborn, MI 48126

Abstract

Over time, the inflexibility of a large software system can inhibit the growth of the business it supports. This paper describes a situation encountered at AT&T's Merrimack Valley Manufacturing Plant. The problem domain is inventory parts allocation in a complex environment. Two systems were developed on PCs to provide front and back end processing of customer order and inventory data stored in a mainframe. Data driven rules access the user data via terminal emulation software, and a rule-based program provides in-depth problem analysis. The systems have been used successfully for eighteen months.

Introduction

This paper discusses two inter-related systems with the same technology approach. These systems are: MPSA - The Master Production Scheduler's Assistant, and COLES - The Customer Order Loading Expert System, both built for the Patriot Factory at AT&T's Merrimack Valley Works.

Inherent in these systems is a knowledge-based solution to a problem common to many businesses - how to prevent the large software systems that support the business processes from actually inhibiting business growth and evolution. Software is generally chosen or built to match business needs at a specific moment in time. Once deployed, these systems often develop an inertia that tends to define how the business is run. The system data entry requirements drive the day to day processes, and the databases and data availability in reports are used to monitor and control the day-to-day operations.

This solution demonstrates how the addition of an intelligent knowledge-based front-end running on a Personal Computer significantly enhanced the utility of a legacy COBOL program running on a mainframe, and also established the capability to easily respond to changing business needs. This solution enabled AT&T to preserve the considerable investment already made in the COBOL system, and to deploy a system that met all requirements at a relatively low cost. The innovative implementation of data-driven rule processing of IMS screen data allowed the system to be built with no involvement from the mainframe maintenance programmers, or mainframe downtime.

Background

The Merrimack Valley Works in North Andover, Massachusetts is one of AT&T's largest manufacturing sites. Like many large business operations, the plant relies on an IBM mainframe and a set of COBOL applications to support the business processes. The applications include multiple Material Resource Planning systems, a Part Purchasing system, and a Customer Order Entry system. Some of these were internally developed and maintained, others were purchased and maintained externally. These systems were implemented at a time when operations were simpler than they are today, and it became apparent that they could no longer effectively respond to more complex business needs. Faced with the necessity of enhancing these systems, AT&T decided to apply AI techniques to the problem, for a number of reasons.

Traditional DP procedural code responses to the changing needs of a business can be very expensive, or even impossible to implement. Many companies employ a large staff of maintenance programmers to extend program functionality to meet changing requirements. This approach can be ineffective for many reasons, as it involves a trade-off between businesses' very real need to change and the ability to deliver the complexity of the changes required, and issues such as the availability of staff capable of making the changes, the age of the system, or even the availability of the technology required to support the system.

A not uncommon situation is one in which a legacy software system, designed to support now outdated business practices, cannot be changed quickly enough to meet the very different requirements of current operations. Even if it can be modified to meet the needs of the present, a system that is not flexible or easily scalable will inevitably again fall short of the demands imposed by a dynamically evolving business.

Another traditional, if expensive, approach may then be suggested - that of replacing the old system with a new one that more closely models how business is now being conducted. Even less attractive approaches are to continue to run the business in the ways required by the old system, or to extend the functionality of the old system with manual procedures.

All of these traditional solutions seem to share one common feature: they are attempting to handle a basic inflexibility in development techniques through the application of those same techniques. AT&T recognized this failing and decided instead to use AI techniques as the approach to this large application system maintenance problem. The solutions developed at AT&T significantly enhanced an existing COBOL application by broadening the scope of data retrieval, processing, and presentation. This was accomplished by augmenting the existing, and unmodified, application with a cooperative process which does not reside on the same mainframe computer.

Patriot's Business Problem

The initial problem to be solved was that of managing the planning process for product manufacturing. The MVW plant builds various systems for communications transmission, switching and networking products. The building is internally divided into three logical factories or shops. The Patriot Shop, one of these factories, handles the more mature and stable systems. Patriot products consist mainly of electronic circuit boards, also called circuit packs. Product orders tend to be low volume and quite variable, with a complex product mix.

Supporting the three factories is a common, internally developed Material Resource Planning (MRP) system called IMPAC. IMPAC is used to track customer orders and inventory and to aid in planning and scheduling. It handles each of the factories as separate business units, monitoring the assembly parts needs for each. Materials planners and analysts use IMPAC to ensure that all critical parts are available when they are needed. Each day, new customer orders are entered into IMPAC and tentatively scheduled for production at some reasonable future date. The orders specify quantities of circuit packs or circuit pack sub-assemblies.

Before an order can be firmly scheduled, missing parts must be found and allocated from within the plant or obtained from a supplier. The Master Production Scheduler and his team of analysts and expediters have the job of prioritizing orders and chasing short parts to satisfy requirements. They locate parts using various IMPAC screens to determine availability of parts, projected future use of parts, and expected delivery of parts. One week before an order is scheduled to be built, IMPAC checks to see which (if any) circuit pack's components are missing and issues a report listing orders and shortages.

The three factories manufacture different products, often from common components. Ideally, IMPAC should look across all three sets of customer orders to determine the quantity of parts needed, and then check all three inventory stores to see what is available. When orders compete for parts, IMPAC should be used to decide which factory gets precedence. Sometimes, products created in one factory are sub components (i.e. parts) for products in another factory, so IMPAC should incorporate a representation of such relationships and be able to use this to allocate components.

However, when IMPAC was developed, MVW was run as a single factory and parts monitoring was much simpler than it is today. Now, processes are more complex than IMPAC can handle, and changes to the system software have not been sufficient to enable effective support of this activity. Since the three factories are separate business units and IMPAC handles them separately, the report on parts availability is not necessarily accurate. For example, IMPAC looks only in Patriot's storeroom for parts to fill Patriot orders. In fact, another factory unit may have the short parts in its storeroom or shop. It may be that another factory uses large quantities of these parts and so has agreed to supply Patriot's need.

To locate these parts, analysts must spend considerable time searching through IMPAC inventory screens for other factories, and also analyze storeroom and shop inventory data. Once they locate the parts, they can request that the parts be transferred to Patriot so that the order can be scheduled and built. The process is tedious, and analysts do not always have time to perform complete searches. Thus, decisions that have a critical impact on product scheduling, delivery, profitability, and customer satisfaction, are made with incomplete data.

Patriot's Requirements

The high level requirement was to create a system that would perform thorough searches that were not limited to one factory's inventory or production schedule, make decisions based upon findings and then use those decisions to update program data and make recommendations. This would reduce the workload of the Master Production Scheduling Analysts.

A constraint on design was that the system had to interact with IMPAC internals rather than simply with the package's data bases, as it needed derived data which was displayed on the screens but not stored in the data bases. One approach involved developing programmatic hooks to enable the system to make use of the data. However, the COBOL software developers already had long lists of desired enhancements and were not able to participate in the new system development. An alternative solution was necessary for the Master Production Scheduler's Assistant (MPSA).

Approach - MPSA

The first problem was to design a system that could be built without the participation of the mainframe maintenance programmers, and would not create a need for downtime on the mainframe in order to achieve integration with IMPAC. The adopted solution focused on the concept of *user emulation*. User emulation takes the process of terminal emulation one step further by creating an automated system which interacts with the mainframe software screens in the same way that the user does. The system reads the user reports, logs onto the mainframe with a used-id, looks at the same screens, makes the same decisions and enters the same data that an experienced and conscientious user would. It does all of this from an intelligent workstation, and never once invades the non-public territory of the IMPAC software package. Thus, it is a very clean solution.

Interface Issues

Thc original plan was to create a mainframe application to look for parts problems and make recommendations for resolving them. Initially it was expected that this application would read a flat file for initial order input, interact in real-time with IMPAC, and then output data to RAMIS, a mainframe report writer.

However, when it was recognized that a PC based system could interact more readily with IMPAC than a mainframe system, all aspects of the system were redefined. Principal issues were: when and how to transmit bulk input data (the flat file) to the PC, where MPSA output should go for reports, whether the user interface should conform to PC or mainframe standards (our users had not previously used PCs), and finally, how to break up the series of tasks users wanted performed.

System Components

MPSA is comprised of a number of modules:

Screen Access: Data driven rules request IMPAC screens, recognize the screens when they appear, access screen data and map the data into MPSA facts and objects. There is a small set of rules for the logon process and for each screen. A larger ruleset handles errors and exceptions.

Input: This module reads the IMPAC flat file, order by order. It then creates an object for each order and each missing part.

Classify Orders This module is a reduction to 15 complex rules of the tortuous task of dynamically determining status within the process. Allowable status conditions include "Complete" (no missing parts), "Short" (Missing Parts), and "Found" (Found Missing Parts). The status is determined by pattern matching on the order and parts objects. The Classify Module's rule set is activated twice for each order - first to determine if parts need to be found, and then to determine if the search for them was successful.

Parts Search This module activates the screen access rules for the purpose of accessing IMPAC inventory screens. It uses data from the screens to determine if there are extra parts available.

MPSA enters the missing part number into an inventory screen. IMPAC responds with a screen listing all plant locations of that part, and the quantity available. Other screens show the demands (there are several types) already existing for that part. MPSA analyses the demands and determines if there is an excess. If there is, the parts are allocated within MPSA to that order. If the parts are in a location controlled by Patriot, they can be immediately assigned to this order. If they are controlled by another shop, then IMPAC sends a request to that shop. The part analysis is stored in the missing part object, where the classify module can use it to determine order status.

Parts Substitution This module accesses a small PC based database of legal parts substitutes. It checks to

determine whether the part has a substitute, and if so, whether it is available. If more than one substitute is listed in the database, MPSA will retrieve and allocate substitutes in database order until the required quantity is filled.

Book Keeping. This module provides internal bookkeeping for MPSA, and its purpose is to prevent parts from being allocated twice. Whenever a search for a missing part is initiated, a record is kept of the number found, and the number remaining.

Output. This module stores results for each order in a flat file on the PC. The data includes the order input information, a list of the parts missing, and those that were located, along with their locations.

Control. This module contains the rules used to control the flow of the program. The program passes through seven sequential phases: Input, Classify, Parts Search, Parts Substitution, Classify (again), Bookkeeping, and Report. In addition, the Parts Search and Parts Substitution modules each have their own control sequencing.

A representation of MPSA processing and the inter-relationship of the modules is shown in Figure 1.

At the start of each day, users obtain a report from IMPAC showing its knowledge of parts availability for each customer order. That report was a logical starting point for MPSA. First, a small 4GL program was written to create an edited, flat-file version of that report and then a small PC batch file to transfer the file from the mainframe to the PC.

MPSA reads the report, performs its analysis and then stores the results in a flat file. This file is read by yet another 4GL program which merges it with data in a large PC database and produces the final user reports. (It was fortunate that the plant's favorite mainframe report writing tool, RAMIS, has a PC counterpart called PC RAMIS. This removed report writing from the task list.) Users immediately identified and created six separate reports for daily and weekly use. They also created new PC RAMIS data base files for storing and reporting day to day part expediting information.

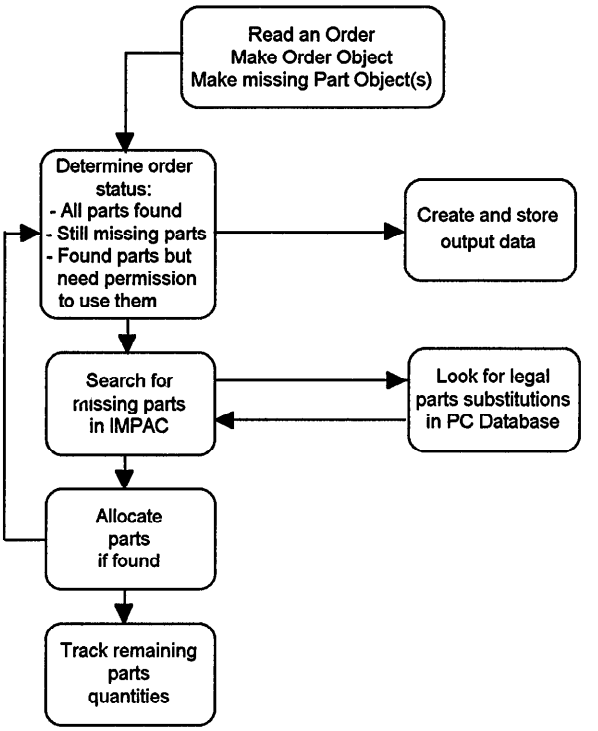

Figure 1: MPSA Components

Technological Approach

An object-oriented solution was chosen as probably the only way to handle the complexity of data. The variety of parts situations suggested that a pattern-matching rule-based approach was advisable.

Inference's ART-IM was selected because its data driven rules would pattern match on object information. ART-SNA was chosen for its ability to access (read and write) IMS screens, thus allowing access to the IMPAC system internals as displayed on the screens. The MPSA solution also includes a terminal emulation package and a HLLAPI package.

The system configuration is as illustrated in Figure 2.

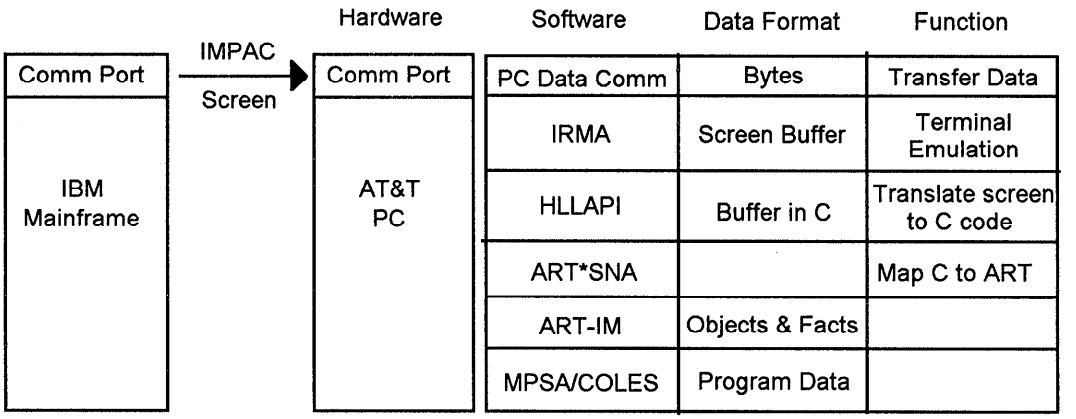

Figure 2: MPSA Hardware/Software

The Master Production Scheduler's Assistant (MPSA) runs on a PC connected to the mainframe. MPSA logs onto the mainframe, brings up the application package, performs its user tasks, then logs off. In addition it performs more traditional system tasks such as allowing user input through a user interface, problem analysis and report generation.

The HLLAPI package creates a buffer of the screen data and ART*SNA maps fields from the screen (the buffer) into ART-IM objects and facts, which in turn then activate ART-IM rules.

The data-driven, rule-based approach was especially effective for interacting with the real-time data access from IMS database screens.

MPSA Rule and Object Architecture. The MPSA Program rule base is segmented into logical sets of rules. At the highest level are control rules used to sequentially walk input orders through the necessary processing steps. The sequencing steps vary from order to order depending on whether or not there are short parts, and whether or not those parts can be located or substituted. The control is accomplished by means of phasing and makes use of ART-IM facts.

The individual processing steps make up the second layer of rules. These sets of rules handle such functions as determining order status, enumerating still missing parts, looking for spare parts, performing bookkeeping for each order, and bookkeeping for parts. Within these rules, objects are used to represent the customer orders, each order's short parts, and materials locations and substitutions. These objects are short lived; they are deleted as soon as they are processed and reported.

Another use for objects within MPSA is as storage for available part information. Once the search for a part is completed, its inventory status (available quantities and locations) is stored in bookkeeping objects. Other orders requiring the part will bypass the screens, and get data from the bookkeeping objects (and decrease the quantity available). These objects last the duration of the program.

Sets of task-specific rules dominate the bottom layers of rules. These include the screen access rules, reporting rules, and the like.

Communications and Interfaces. The configuration of all the many pieces of the emulation hardware and software was of major concern - it looked too complex to actually work. But in fact the team was pleasantly surprised by the amount of effort needed to make the communications functional. The hardest part was deciding which hardware and software to order!

The communications hardware chosen was DCA's IRMA 3270 terminal emulator board for the PC. It is installed in a 25MHz, 80386 PC (AT&T 6386/25 WGS) with a co-axial connection to the IBM cluster controller. A LAN gateway has been tested, and it is probable that the system will be moved to that configuration soon.

DCA was also chosen for the emulation and HLLAPI protocol software. The HLLAPI is used to make mainframe data packets available to a high level language (like C) on the PC. It consists of high level function calls to read and write mainframe screen data.

ART*SNA is the highest level of the communications link. It informs the data driven ART-IM code when new screens are present, and provides a simple rule-based, object-oriented programming interface between the screens and ART-IM.

Developing the screen access rules took much less time than was originally anticipated. After the first screen was coded, rules to process a non-trivial application screen were written in just a few hours.

It is worth describing the data-driven process which recognizes and handles the mainframe screens. Each screen is identified to ART-IM by its name and recognized by a unique set of characters on the screen - such as a screen ID. Whenever there is a screen change, it is announced (via a fact) to the inference engine. So, for example, when MPSA first connects to the mainframe, a fact is asserted in MPSA that the logon screen is currently active and displayed. Within that announcement is a pointer to a buffer containing the screen text and data. The announcement triggers any waiting logon specific rules so that the appropriate action can be taken. For example, MPSA has a rule that says, "If we are sitting at the logon screen and we want to start a session, then send the logon text and an 'enter' key". This action produces a new screen, which in turn activates other rules. In this manner, system logon, application logon, and then access of the inventory screens is accomplished. Simple mapping functions are available to pull interesting screen data from the screen buffer and store them into objects or facts.

****

Before discussing MPSA's deployment and benefits, the next section of this paper gives a brief overview of a second AT&T Patriot application which built upon MPSA's design and philosophy. It also fulfilled the goal of extending the utility of a legacy system.

The Customer Order Loading Expert System (COLES)

After MPSA was completed, the team turned its attention to another problem - one which could be solved using the same technological approach. The problem to be addressed was error handling in a Customer Order Entry system. The COBOL system maintainers were unable to implement effective error recovery procedures for many of the same reasons that the IMPAC system could not be upgraded. The system's inability to deal effectively with errors negatively impacted customer order processing. For various reasons, a percentage of orders would be rejected as errors. A clerk would then spend, on average, an hour a day re-entering these orders into the system. This problem was solved by introducing an intelligent error handler that enhanced departmental processes and saved significant amounts of manual input. COLES reads the error reports, determines the cause of the problem, and automatically enters the corrected order into the system. It eliminates the need to manually re-enter customer orders.

The system can be represented as below in Figure 3:

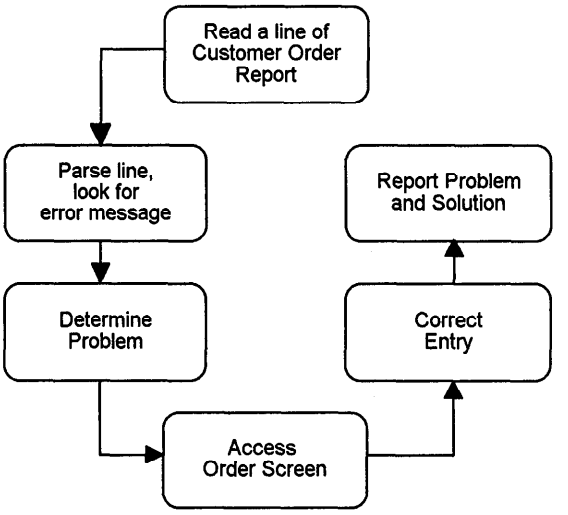

Figure 3 : COLES processing

The COLES system is a relatively simple application of rules that analyze customer order entry error conditions. On determining the cause of the error it then writes the correct data to the order forms on IMS screens via PC terminal emulation

The Development Team

A noteworthy aspect of the project was the degree of successful technology transfer accomplished. The development team consisted of three AT&T engineers each of whom had little or no rule-based programming experience, and an Inference Consultant who headed the project and trained the others. The AT&T team members included a mechanical engineer, an experienced COBOL programmer who knew IMPAC, and a PC "systems" programmer. The Consultant's responsibilities included designing the system, training the team and transferring ownership of the system to AT&T. At the end of the project, one team member was comfortably maintaining the system and all team members were actively pursuing other rule-based projects.

The largest factor in the technology transfer success was the freedom from rigid deadlines. The system could have been built more quickly, but AT&T management, committed to having a staff trained in the use of the ART application development tools, agreed to extend the project's time span to accommodate the technology transfer learning curve.

Another factor in this success was the structuring of the task assignments. Each person leveraged what they knew within their assignment. For example, the PC systems programmer, who loved tinkering with new and unusual PC products, learned to write rules by coding the ART-IM screen access. The COBOL programmer created rules to set up IMPAC parameters for nightly production processing. Each member of the team had a stable base of his/her own expertise to work from. This kept the frustration of learning a new technology lower, and the interest higher.

Deployment

The MPSA project was deployed within nine months of the initial scoping effort. The actual programming effort took six months.

The following serves as a testimony to the flexibility of the knowledge-based approach to MPSA. Just as the system was nearing completion, departmental processes shifted and the business problem changed. Several pieces deemed mandatory at the outset were suddenly unnecessary. The application was refocused in order to ensure it was not obsolete before it was completed. The front and back-ends to MPSA were re-designed and coded, and new layers of rules added to accommodate the data needs. Within 30 days, the system was in operation.

The program was tested and validated using traditional software practices: unit test and system test on standard, exception and boundary-condition data. The user performed the acceptance testing by hand checking MPSA results.

The initially delivered system required two hours of an operator's attention before reports were finally produced. Once MPSA was functionally correct, this aspect was automated. Batch commands were set up to logon to TSO, run a database query to create a flat file and then download the flat file to the PC where it is used as program input. MPSA output was then funneled to a PC-based data base for input to a report generation program. The operator now spends two minutes initiating the process, and then picks up the reports in 45 minutes.

The COLES project required just three months effort from concept to deployment, since much of the design and code was borrowed from MPSA.

Project Status

MPSA has been in daily use since June, 1991. COLES has been in use since November, 1991. The systems service two separate departments. The system operator is a senior clerk in the scheduling department. Reports are used by the Patriot shop management, and by the material planners.

System Maintenance

Most maintenance thus far has been minor: modification of input or report data, and an occasional addition of an order processing condition. The system is maintained by one of the original team members, who is currently in the process of handing off this task to the system user. Maintenance complexity has been minimized by using the same general rule and object architecture, control structure, input and output methods, and screen handling rules in all of the programs.

Benefits

MPSA has been an extremely successful application for AT&T in a number of ways. It was probably the only means by which this very necessary additional functionality could have been achieved within a reasonable time frame and cost. The original COBOL system, IMPAC, was built to satisfy the needs of a business very different to the one it is supporting today. MPSA was able not only to deliver expanded functionality to meet the requirements of today, but also, through its design and technology selection, to provide the flexibility to respond to future business evolution.

The system was built without using the time of busy maintenance programmers, and deploying MPSA on a PC platform allowed the system to be developed without tying up valuable mainframe time.

It has been calculated that, overall, savings of more than 200,000 dollars per annum have been made in personnel costs, product sales attributed to increased customer satisfaction, and better inventory utilization.

The quality of work produced and the level of job satisfaction have both increased. MPSA has helped to switch analysts from reactive to pro-active mode in doing their jobs. Several tedious work assignments in the planning and scheduling departments have been made obsolete. The quality of scheduling plans is far superior and more useful data is available. Several departments use the reports as the basis for their weekly meetings. The reports achieve an accuracy of data that was not available from IMPAC. The reports are also presented in a more useful form than previously.

The application development process provided the context for a very effective technology transfer to AT&T employees. This laid the foundation for the development of other applications without the involvement of outside consultants.

Acknowledgments

We would like to thank co-team members Jonathan Lofton and Arjun Harpalani, AT&T management: Maurice Henderson, Bob Menard and Mike Jones for their support, users Kyle Lynch, Ken Back and Sue Meyer for their constructive criticism and Norbert DeAmato, Glenn Coffin, Bill Steele and Don Bohnwanger for mainframe programming assistance.