Expert Investigation and Recovery of Telecommunication Charges

Hieu Le
Pacific Bell
2600 Camino Ramon
San Ramon, CA 94583

Gary Vrooman, Philip Klahr, David Coles, Michael Stoler
Inference Corporation
550 N. Continental Blvd.
El Segundo, CA 90245

Abstract

Pacific Bell has millions of phone calls that cannot routinely be charged. Investigators manually search for clues that allow unbilled charges to be correctly charged before the state regulated time period expires. In order to recover billable revenue Pacific Bell elected to bring additional automated support to the problem. As a result, Pacific Bell chose to implement an expert system, the Expert Message Correction System (EMCS). EMCS was deployed in November 1993, and is currently processing thousands of unidentified calls each day. The EMCS architecture involves access to multiple mainframe applications and databases, and incorporates a multi-process architecture that automatically selects and schedules multiple expert system processes to execute within a single workstation. This design provides a high degree of parallelism to enable processing the large volumes of calls and their associated mainframe data. In addition to this multi-process architecture, EMCS provides new uses of AI technology in the telecommunications industry, particularly in the areas of billing and investigative procedures.

Introduction

An *unidentified message* refers to any phone call ("message") which causes exceptions during normal billing procedures. These exceptions require investigation and then corrective action before the billing can be completed. Unidentified messages are investigated by the Pacific Bell billing organization.

The unidentified message volume is a reflection of both a large customer base and a wide range of customer flexibility in choosing features, services, discount programs and billing options. A knowledge-based system was proposed to automate the message investigation and recovery process for a broad range of error conditions. The knowledge-based system approach was deemed appropriate because it provided the most effective technology tools to those employees who are in the best position to make use of them.

Error Investigation Procedures

Unidentified messages are tagged with error codes by various Pacific Bell automated systems. Tagged messages are collected by a mainframe system, partitioned into cases, and assigned to investigators. Messages are grouped into cases by various criteria. Each case typically consists of one or two types of errors and perhaps 20 messages (although thousands of messages sometimes occur).

Cases are retrieved by the assigned investigator and processed. This may involve accessing multiple mainframe applications to obtain the necessary information required to determine the correct billing. Recommended actions concluded by the investigator are then sent back to the mainframe system for automated processing. Actions are specified by a transaction code and some number of additional arguments. History and comments concerning actions taken are inserted into the mainframe case record.

EMCS Application Description

The Pacific Bell *Expert Message Correction System* (EMCS) automates the investigation and billing procedures for the processing of unidentified messages. These exceptions now require EMCS investigation and corrective action. Typical results of an EMCS investigation might include billing the message as is, correcting a mistaken account number, holding a message until further information is available, or collecting all relevant data for the call from several different mainframe information systems and diverting the case for manual follow-up.

Error Message Processing

The principal requirement for the EMCS system was the automation of unidentified message investigation procedures. The necessary set of investigative procedures depends on the error codes found on the case.

For the EMCS expert system, one of three categories of actions is possible:

- Initiate a transaction to *charge* messages.
- Put the case in *abeyance*. Abeyance holds a case for some number of days. Case processing is resumed following abeyance.
- *Divert* the case to an investigator for manual investigation, for cases where manual handling is required.

Once case processing is complete, individual messages are directed to one of several target applications for completion.

High-Level Architecture

The overall throughput of the EMCS application can be affected by several conditions, such as the time required to obtain data from the various mainframe applications. In an attempt to model the system throughput requirements, total daily processing time estimates were performed on volumes of cases and messages. Actual case and message volumes were sampled for a typical single day and for a complete month. Error messages were summarized according to external host access dictated by the error procedures defined during the requirements analysis phase.

Some important results were evident from the resulting processing time analysis. Two components of mainframe access time were modeled: the mainframe application program response time (the average wait time while the mainframe was servicing other users plus the time necessary to process the data request) and the transmission delays in sending data over transmission lines of various speeds. It was discovered that both factors had a substantial impact on overall throughput of cases. The conclusion from this analysis was that a single expert system image could not possibly handle the necessary workload because data could not be retrieved from the mainframe quickly enough.

These observations resulted in a multi-process architecture, where multiple UNIX child expert system processes are maintained and supervised by a parent EMCS process (Figure 1). Child processes are run concurrently on the workstation, each with its own dedicated set of communications sessions to the necessary mainframe hosts. In the initially deployed system, five to seven of these child processes are run simultaneously on each workstation. Inference Corporation's ART-IM for Unix was utilized as the programming language for EMCS because it provided the necessary representational capability, speed (data-driven architecture and compiled rules), and flexibility to implement this architecture.

EMCS provides three primary modes of operation:

- A *test* operation mode allows users to step through case investigation steps, providing a running description of information extracted from case messages or external database systems. Test mode does not send any update transactions back to the mainframe application.
- An *interactive* operation mode provides display and interaction characteristics identical to the test mode, except that update transactions are enabled. Following user confirmation of proposed actions, transactions are submitted to the mainframe.
- A *batch* mode of operation provides continuous case processing. EMCS maintains a cumulative statistical summary of cases processed for the session. Batch mode is used to service most of the case load. The batch subsystem is designed to allow multiple EMCS batch processes to be running concurrently (limited by the number of host ports attached to the workstation).

The test and interactive subsystems of EMCS are designed as a single UNIX expert system process, while the batch subsystem is a multi-process architecture designed in a parent/child configuration. The parent process has a graphical Motif-based user interface. This parent process uses the UNIX *fork/exec* system calls to spawn a

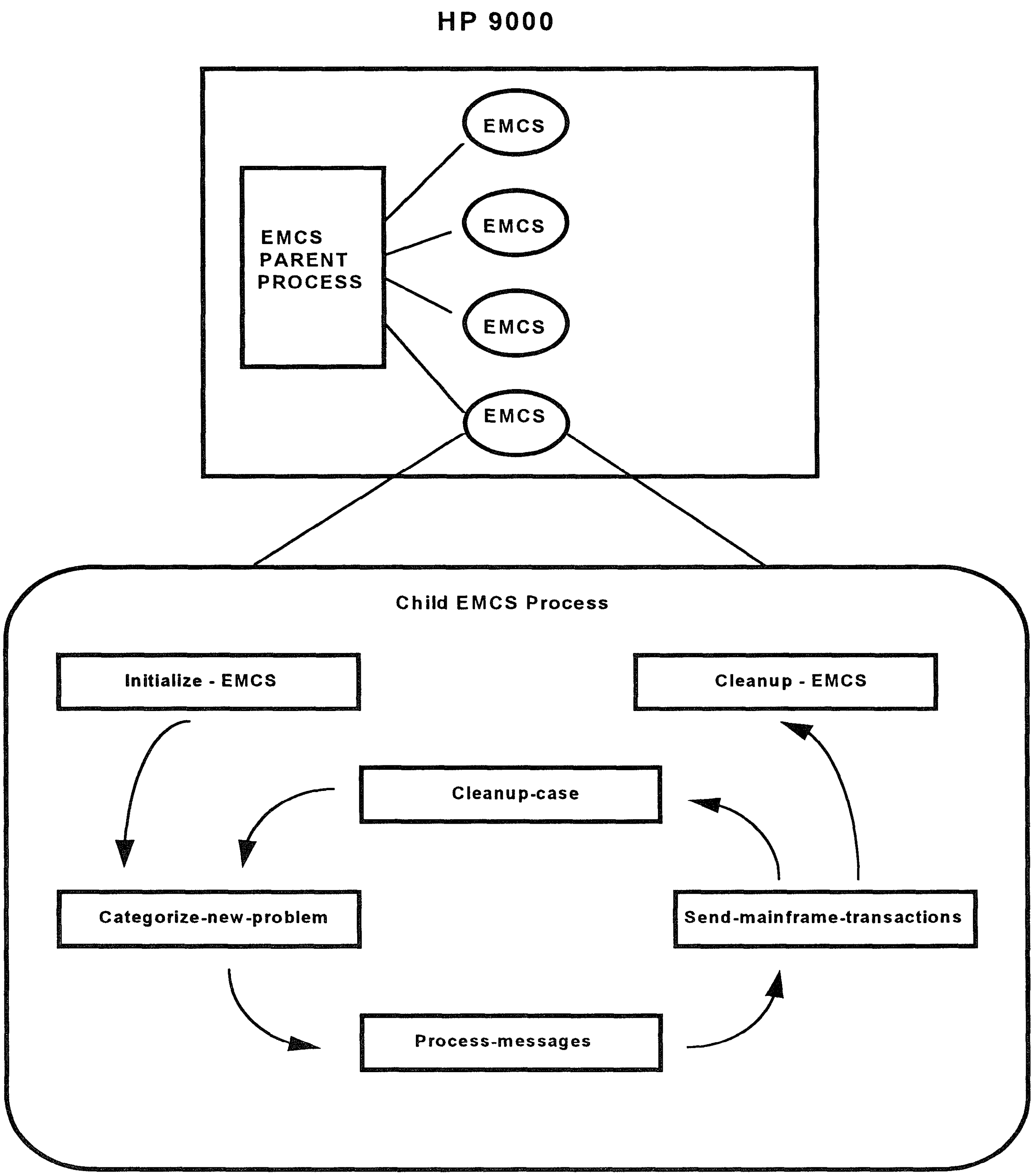

Figure 1. EMCS Multi-Processing

user-specified number of batch child processes. Each child process independently processes cases, using a local case ID file and record locking to determine the case ID of the next case to process. Each child process writes out to a file a set of summary statistics. This file is updated after each case is fully processed.

The parent process monitors the progress of its children, and allows an on-line user to view a summary of that progress. The parent process utilizes the XWindows X Toolkit XtAppAddTimeOut facility to automatically collect statistics at designated time periods. At the designated intervals, the timer expires and the parent process collect summary statistics by reading the output files generated by each child process. This information is summed and displayed in a Motif window. Once the batch system has been invoked, the system runs unattended, with summary information presented in a window on the screen. Statistics can also be updated via a menu choice when the user needs an updated summary. This menu choice causes the EMCS parent process to immediately total the statistics from all of its children and display this summary information in the window.

When a particular day's workload exceed EMCS' capacity, both the primary mainframe application and EMCS are designed so that any unprocessed cases will automatically be rolled over into the next day's work. Each batch process also automatically checks the time of day after processing each case. Just before the designated end of the business day, when the host systems are taken down for the day, each batch process stops case processing and logs off from the various host systems. Users can define the hours of operation for EMCS processing by setting the AUTOSTART and AUTOTERMINATE times in a dialog box available from the options menu. Once these times have been set, EMCS operates in self-scheduling mode.

Should any communication problem occur with one of the host systems during normal business hours, any batch EMCS processes requiring data from that host system will be unable to complete their processing. They will simply generate the appropriate error message and exit. Normally, the host system problem is quickly fixed by the appropriate Pacific Bell system administrators. Therefore, every few minutes (as the timer expires), the EMCS parent process checks on the status of its children. If any children have terminated and unprocessed cases are pending, the EMCS parent process automatically spawns a new set of replacement child batch processes. In this manner, once the batch subsystem of EMCS has been invoked, it can run unattended - EMCS processes are automatically spawned daily. If some abnormal event occurs such as the workstation crashing, simply re-booting the workstation (if necessary) and logging back into the EMCS account will restore normal EMCS processing.

The EMCS user interface was built in Inference Corporation's Motif Integration Library (MIL). MIL is itself implemented in ART-IM and C using an object oriented methodology built on top of XWindows and provides an OSF Motif-based graphical user interface. The basic EMCS user interface design relies on the Motif Integration Library for access to Motif primitives, and relies on the ART-IM schema system to build higher-level objects from those primitives.

EMCS Reasoning Kernel

The EMCS reasoning kernel can be invoked by the user directly from the application user interface (interactive or test mode) or invoked automatically at the start of a new business day (batch mode). The flow of control in the reasoning kernel is managed by a set of ruleset data structures. Each ruleset contains a set of rules which implement the detailed processing logic for each particular stage of the reasoning kernel processing. Just before it invokes the reasoning kernel, the calling C program asserts the ART-IM fact (*initialize-EMCS*). The presence of this fact causes certain initialization rules to fire in the *initialize-EMCS* ruleset.

Once initialization is complete, the reasoning kernel repeatedly calls the *get-case-summary* function to retrieve the next case to process. Next, the reasoning kernel calls *get-message-details* to get the necessary data for each message on the case. Then, depending on the error code and other factors, other functions are called to collect data from the various mainframe systems. When there are no further cases to process, the reasoning kernel returns control to the calling C program.

Representation of Declarative Knowledge

EMCS data structures have been defined so that the application directly models, in a natural way, the underlying data and the key relationships among the data items. To the investigator, for example, it seems natural to view the data on each bill as a database record. Using an expert system approach, this maps into a bill object with an associated set of properties. In this approach, most of the main panel types for each host application map directly into objects.

In ART-IM terminology, these panels map into *schema* definitions. A schema is a data structure which aggregates object properties into an associated set of slots and values. For each unbilled message, a *message-detail* schema is asserted at runtime.

When EMCS decides upon appropriate corrective action for a particular message or set of messages in a case, an instance of the *transaction-record* class is generated which contains the details of the necessary corrective action. At the completion of case processing, a series of mainframe transactions are submitted, one for each *transaction-record* schema.

Schemas are similarly constructed and dynamically asserted to hold the data returned by each of the different types of mainframe access functions.

Rulesets

In EMCS, collections of rules that implement a task are organized into rulesets (Figure 1). Each major stage in EMCS processing is defined as a ruleset:

1. *initialize-EMCS* ruleset performs miscellaneous initializations when EMCS is first invoked.

2. *categorize-new-problem* ruleset retrieves a new case and message detail records, and performs certain tasks to categorize the problem.

3. *process-messages* ruleset contains rules which deduce the corrective action for particular messages in the case. The rules in this ruleset pattern match against *message-detail* schemas and produce intermediate results, or they generate or update *transaction-record* schemas. Each of these schemas represent a particular corrective action to take and a list of the messages for which this corrective action is appropriate.

4. *send-transactions* ruleset is invoked after the messages of a case have been processed. For each *transaction-record* schema created by the *process-messages* ruleset, a mainframe transaction is created and sent.

5. *cleanup-case* ruleset contains a set of rules which perform cleanup actions at the end of processing an individual case. These cleanup actions re-initialize EMCS and prepare for processing the next case.

6. *cleanup-EMCS* ruleset contains rules which perform final cleanup, host logoff, and other termination actions at the completion of case processing for that day's workload.

Processing Logic

The *process-messages* ruleset contains the rules which perform the critical decision-making tasks. These rules

- decide which mainframe systems need to be accessed
- examine the fields from each host transaction
- analyze the information to understand the situation
- decide how to correct the problem.

An initial set of mainframe data is always retrieved when processing a new case. In the *process-messages* ruleset, a set of analysis rules fire which pattern-match against the initially retrieved mainframe data. Collectively, these rules decide which of the several other mainframe systems must be accessed in order to process the case. Due to the volume of cases EMCS processes, a large amount of mainframe data must be accessed daily. These accesses are computationally expensive and without proper care could have degraded mainframe response times and other critical applications requiring mainframe access. Therefore, EMCS rules are structured so that, for each stage of processing, only mainframe data critical to the decision at hand is retrieved. Furthermore, the analysis rules decide which of the available data access methods to invoke in order to retrieve the required fields at minimal costs.

The analysis rules express their decisions by asserting a set of ART-IM facts. As these rules assert facts specific to data needs, a set of data-gathering rules are invoked. These rules perform the necessary book-keeping to ensure that mainframe data is never retrieved more than once.

A set of ART-IM schemas have been defined which represent the various classes of mainframe system data. Each data retrieval, if successful, causes a schema to be created which is an instance of the appropriate mainframe system data class.

As mainframe data becomes available in this form, decision-making rules become eligible to fire. These rules tend to be rather complex, typically with many pattern matches against the slots of different instances of mainframe system data schema. Frequently these decision-making rules invoke one or more sets of procedural calls which perform specific additional tests on the accessed data. If the data meets all of the conditions specified in the pattern-matching constraints, plus meets the conditions specified in the procedural logic, then EMCS has deduced the nature of the problem. EMCS then builds a transaction schema whose slots specify the particular details of the mainframe transaction that must be submitted to correct the problem.

Mainframe System Access

Several external database systems must be consulted during case processing. Information extracted from these databases governs decisions concerning billing of case messages. Connection with external systems is made at EMCS initialization, and is maintained for the duration of system processing. The EMCS application runs on an HP 9000 and requires data from several different applications which reside on IBM mainframe computers.

For some of the above systems, mainframe connectivity is provided via UIA software from Apertus Technologies Incorporated. A total of 17 transactions were built which provide all the necessary data from these mainframe systems. The EMCS application can be running either on the same machine as UIA or on another networked workstation.

The UIA does not currently provide interfaces to two of the mainframe applications, so EMCS utilizes a second communications pathway to access data from these systems. This second pathway is based on the high-level applications programming interface (HLLAPI), which was originally defined by IBM and is now a de facto industry standard for mainframe connectivity using 3270-type screens. Inference's ART*SNA package was utilized to provide a higher-level interface for the EMCS application to access the various IBM hosts via HLLAPI calls.

Application History

The project encompassed a two-year timeframe from its initial conception at the end of 1991 to its deployment at the end of 1993. Approximate dates for the various phases of the project include:

Dec91-Jan92	Initial discussions on knowledge-based application
Jan92-Mar92	Requirements analysis/high-level design/project planning
Apr92-Jul92	Return on investment analysis
Aug92-Apr93	Detailed design & development
May93-Sep93	Integration testing
Oct93	Acceptance testing
Nov93	Initial deployment

Of great benefit was a testbed of cases supplied by the user during the development phase to provide an on-going testing environment. Unit testing could be performed on selected cases to verify code modules. This testbed included representative cases encompassing the full range of cases that EMCS was required to process. As such, it served as the initial test suite for integration testing. In addition, other previously unseen cases were added to the test suite for further integration testing and acceptance testing.

During the system integration and deployment phases, Pacific Bell software specialists joined the development team to become fully knowledgeable in the EMCS design and code. The original system developers continued to be involved in any bug fixes during the warranty period.

EMCS Benefits

There are many substantial business benefits obtained from the EMCS system:

- *Productivity Improvements*: EMCS is designed to improve the productivity of the billing group through a more efficient application of investigation procedures. Much of the effort involved in message investigation is tedious and time consuming, suitable for an automated approach.

• *Consistency and Completeness*: Investigation procedures can be quite complex. EMCS provides a consistent application of policy to errors within its scope, providing detailed case information for future audit or manual investigation requirements.

• *Improved Responsiveness*: Efficient EMCS processing of unidentified messages ultimately translates into improved responsiveness.

• *Additional Revenue Protection*: The most significant benefit of the system is the avoidance of lost revenues due to message expiry. Based on the first three months of production, the additional revenue billed from prevented message expiry is projected to be more than a million dollars a year.

Summary

EMCS provides an innovative architecture which allows multiple expert system processes on each workstation to simultaneously investigate and recover telecommunications charges. The processes are spawned and monitored by a parent expert system process. This high degree of parallelism was necessary to handle the overall processing workload, given the inherent speed limitations of the mainframe connections. EMCS incorporates a robust architecture that allowed the system to automatically schedule its own operations and provide automatic recovery from common mainframe connectivity problems.

In addition to this multi-process architecture, EMCS provides new uses of AI technology in the telecommunications industry, particularly in the areas of billing and investigative procedures. Rule-based processing was used to capture the expertise of human investigators. A process that had been done manually has been automated.

Acknowledgments

EMCS was a team effort involving many groups and individuals. Pacific Bell's Strategic Information Systems (SIS) provided overall project management and technical monitoring. SIS head Eric Firdman helped launch the project; Rebecca Barfknecht supervised the development and integration process; Craig Cottrell provided ongoing project management; and Hieu Le was the technical monitor. From CBS, Paula Bentley headed the user community; Judy Rogers and Cathy Flores headed the northern and southern regions, respectively, and Marilee Yamarone, Delpha Stutznegger and Susan Jenkins provided the expertise on investigative procedures. Other important team members included Joseph Tung from UNIX system software support and Terry Rosfeld for user application software development. From Inference, Dave Coles provided the upfront requirements analysis and project initiation, Gary Vrooman and Mike Stoler were the principal developers with support from Sugandh Mehta, and Phil Klahr provided EMCS project management. Apertus, who provided the mainframe application transactions, was represented by Elizabeth Converse-Wilson, Shawn Hunter, Craig Cameron, Dan Riesland, and Dave Fisher.