INCA—An Innovative Approach to Constructing Large-Scale, Real-Time Expert Systems

R. Phelps, F. Ristori, D. Mukherjee, L. Thomae, and J. Steinier

This chapter describes the intelligent network controller assistant (INCA), an innovative system that shows that large-scale, real-time expert systems can be developed as a matter of course, on schedule, and with the same degree of reliability expected of conventional data processing systems.

In developing this system, several more detailed innovative aspects were introduced. First, real-time performance is necessary for network control, which was achieved using standard (rather than specifically real-time adapted) hardware and operating systems. Second, the fault tolerance of the system is essential for such real-time control; so, INCA incorporates an online backup.

Finally and perhaps most importantly, INCA is a new expert system application for a company's core business, with all the risks and visibility that this responsibility exposes INCA to. Therefore, reliability, efficiency, and usability were mandatory; this project could not hide behind descriptions such as "prototype" or "demonstration of principle." Other expert systems for network control have been developed (Goyal and Worrest 1988), but none of them directly confronts the core business issue where failure can be catastrophic.

The application of AI and, in particular, expert system techniques in real applications is now relatively widespread (for example, as shown at the third conference on AI sponsored by the Institute of Electrical and Electronic Engineers). However, most developments have characteristics that set them apart from more conventional data processing developments and that can give the impression that AI remains a specialist niche only capable of producing small-scale gains in isolated areas of a company's activities.

In particular, it is still unusual for AI systems to be truly integrated into an organization's existing computing infrastructure; although interfaces from expert system shells to some databases have become common, real-time interfacing is still a rarity, as is the networking of AI applications into distributed systems. Furthermore, applications are rarely developed in critical areas of a company's business; understandably, they tend to be used where a fallback to existing or manual methods is possible without serious disruption to the business.

INCA tackled this problem by developing this large-scale AI application in a way similar to non–AI system development. A software development methodology was developed and followed to ensure tight quality control (reflecting the sensitive area into which the software would be introduced and a tight schedule) and, at the same time, allow the flexibility needed for an AI development where the incremental extraction of knowledge from the experts means that rigid linear development is not possible. In addition, the methodology had to allow for the controlled introduction of the software into the live environment to minimize the chance of a serious failure.

Of necessity, the traditional *waterfall* approach to software engineering, where the development proceeds in consecutive order from specification to code, could not be applied. Because the functions to be performed by the system were implicitly specified by the reactions made by experienced operators to (initially unspecified) event streams, it would have been necessary to first obtain a complete knowledge dump from the operators of all significant event patterns and all reactions before proceeding with system construction. This approach was impossible both because of time constraints and, perhaps more importantly, because experience in knowledge elicitation indicates that a recursive prompting of experts by reaction to prototype rule implementations is necessary for the full extraction of available knowledge. This latter point was found to be true, and it even surprised the experts.

The standard alternative promoted within AI is the prototype-and-refine cycle. In its basic form, this approach was also unsuitable for the introduction of software into a critical aspect of company operation. Full testing of the software is time consuming, and final evaluation can only be done within the live network environment, with its rich possibilities for interacting events and situations. However, it was imperative that disruption to our network be avoided: The introduction of INCA onto the network was possible only once; no interactive introduction as part of successive refine cycles could be contemplated. Therefore, a modular prototyping approach was developed to overcome these problems (Phelps, Ristori, and Steinier 1989).

Schedules were drawn up before the approval and start of the project (drawing on the experience of a quick prototyping phase) and adhered to. Although the hardware and software used were AI Lisp workstations and expert system shells, they were typical products in their field and did not require specialist real-time skills. The client group was involved throughout the project and was responsible for signing off the system modules as they passed their quality control testing and for approving live installation. Training plans for users were drawn up and implemented in parallel with system development. User procedures were documented, and fault tolerance was built in.

All these points helped to ensure that system development proceeded on time and that its quality was in accordance with the stringent requirements of the clients. The implementation of INCA replaces network operators. The potential for a network control system to cause network downtime should it malfunction is clearly very great; in our network environment, downtime of even a few minutes has a severe impact on financial transactions worldwide. Therefore, building the system to the highest possible standards was mandatory. We believe that INCA has shown that expert system technology is now mature enough for large-scale projects to be introduced on a routine basis even in critical software environments.

Network Control

The Society for Worldwide Interbank Financial Telecommunication (S.W.I.F.T.) network provides automated international message-processing and message transmission services between financial institutions. The S.W.I.F.T. network covers all continents, with banks in more than 60 countries, totaling more than 3000 users (mostly banks). The S.W.I.F.T. network now handles more than 1 million messages a day and has become an indispensable financial tool worldwide.

Figure 1 provides an architectural overview of the S.W.I.F.T. network.

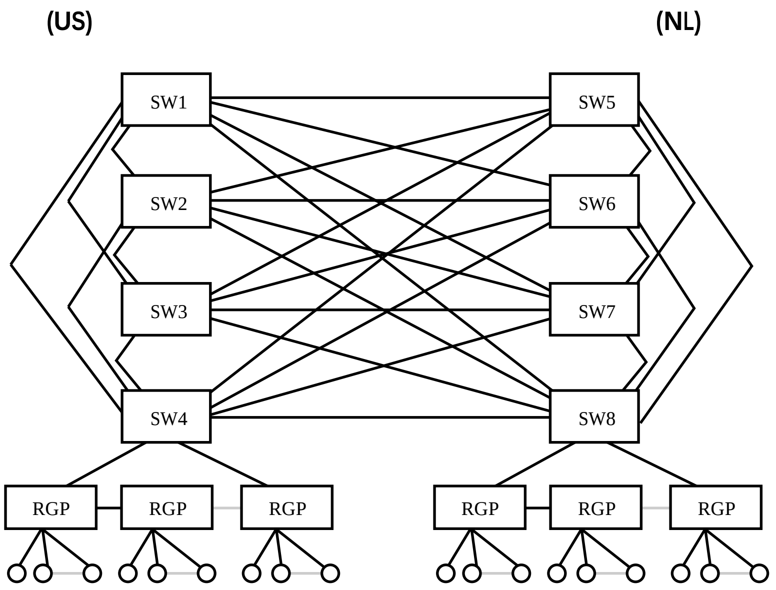


Figure 1. S.W.I.F.T. Network.

It consists of three layers: the control layer, the relay layer, and the user layer. The *control layer* refers to the two operating (control) centers, located in The Netherlands and the United States, that are the hub of the network. They house eight mainframes called *switches*, or active systems, that supervise the S.W.I.F.T. network operations, such as system monitoring, message archiving, and reporting. The *relay layer* consists of several regional processors (RGPs), typically, one per connected country. RGPs are the computers serving as relays from banks to the operating centers. Finally, the *user layer* refers to the banks' computer-based terminals (CBTs). In terms of the physical structure, all bank CBTs are connected through dedicated links or telephone lines to RGPs, which are then connected to the active systems. Several RGPs might be interconnected. communications between active systems are accomplished either through direct physical connection or through satellites, such as for transatlantic communications.

Day-to-day control of the network is currently provided by teams of operators working around the clock at each of the eight active systems. Each active system has a printer (LSP) attached to show network events received and a monitor (VDU) to enter system commands. It is the job of the operators to detect event patterns indicating network malfunction and to issue appropriate commands. This setup is shown in figure 2.

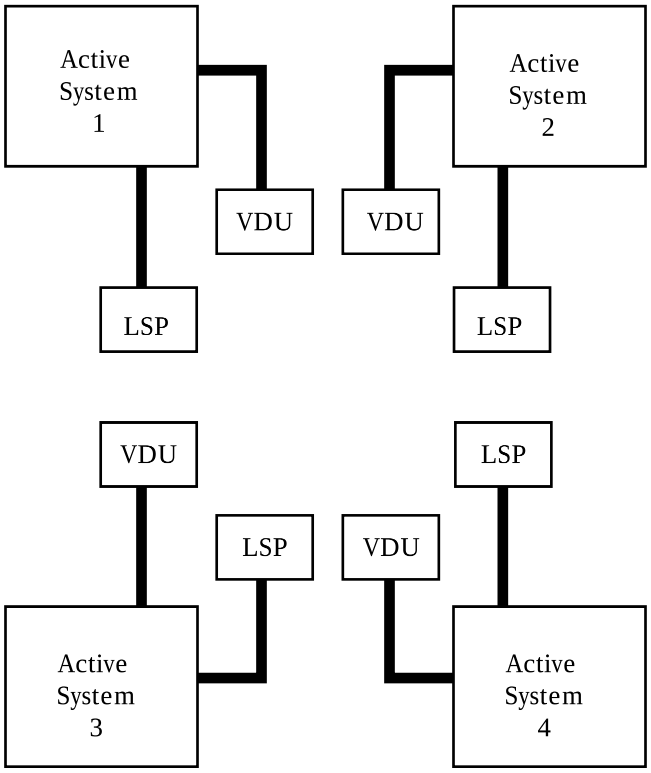
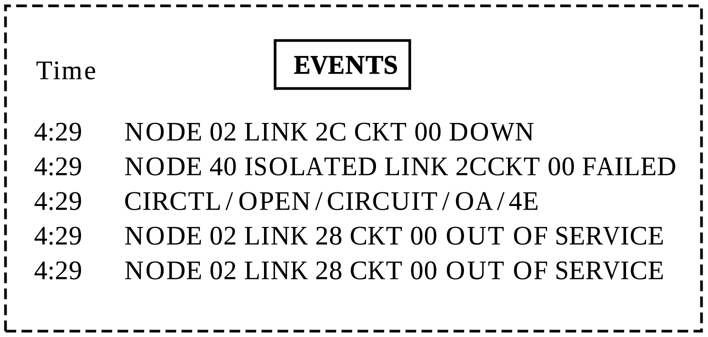


Figure 2. Operating Center.

Automated Event Handling

Figure 3 gives an example of events arising from an active system. From the time stamp, it can be seen that a great many events can occur together, which is typical if a network problem occurs, with the basic fault triggering further error reports as its effects arrive at other network points. INCA performs two functions: First, it filters these incoming events, diagnosing which combinations indicate problems and which do not. Second, it displays problems requiring attention as “tickets” in a window. The windowing interface is shown schematically in figure 4.

Other windows on the operator’s display give additional background information about the problem, and another window gives a choice of possible actions that the operator might want to take. If the operator chooses to perform an action, it is automatically checked for validity against built-in S.W.I.F.T. procedures. Many problems are, in fact, automatically responded to by the system without the operator needing to act at all. In this case, the system allows the operator



```
Time                  EVENTS

4:29      NODE 02 LINK 2C CKT 00 DOWN
4:29      NODE 40 ISOLATED LINK 2CCKT 00 FAILED
4:29      CIRCTL / OPEN / CIRCUIT / OA / 4E
4:29      NODE 02 LINK 28 CKT 00 OUT OF SERVICE
4:29      NODE 02 LINK 28 CKT 00 OUT OF SERVICE
```

Figure 3. Active System Events.

to see the full stream of incoming events, if so required.

The LSP window is not strictly necessary but allows the operator to gain confidence in INCA. As an example, consider the following sequence of actions typifying the way events are handled by INCA, when an event is received stating that the data communications processor (DCP) buffer saturation is greater than 50 percent. This buffer holds messages to be sent from the associated active system. INCA initiates an interrogation process to check the saturation level once a minute. If it drops below 50 percent, the problem is solved. However, if it rises above 75 percent, then commands are sent to temporarily prevent banks from sending messages to the active system, and the operators are advised to disable any other internal network links that are unstable. A time-out is set for 10 minutes so that if buffer saturation remains above 75 percent through the 10 minutes, then another DCP is reloaded (that is, processing is swapped to another, standby DCP). If an operations time-out is received within the 10 minutes, which indicates total saturation, then another DCP is immediately reloaded.

To reload another DCP, INCA first checks if one is on standby. If not, it will either be on standby for a different active system or be unavailable. In the first case INCA reallocates this DCP to stand by for this active system; in the second case, the operator is asked through a screen display if it can be used. After reloading, the buffered messages are resent by the active system. If another DCP is not available, the original one is reloaded. The reloading produces a system dump that will be looked at by the system support group and can solve a DCP problem if it is not simply the weight of traffic causing the saturation. For reloading, the operator is requested to physically switch the connections to the new DCP using a t-bar switch.

After reloading, the network status must be checked. INCA does this

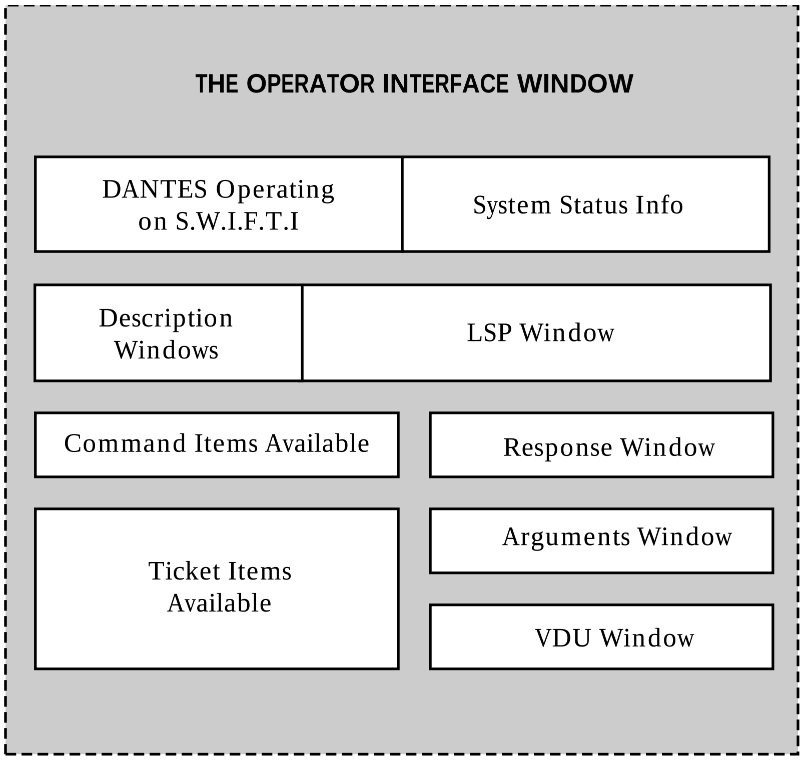


Figure 4. Inca Operator Interface.

checking by reconciling (interrogating) its connected regional processors and lines for their statuses; INCA knows that if any RGP has been isolated for more than 20 minutes because of the DCP problem, it also must be reloaded. This decision to reload an isolated RGP then causes other INCA RGP reload procedures to come into action.

In addition, INCA keeps a counter history for DCP; should the problem recur in less than an hour, it displays a ticket to the operator advising an immediate reload, shortcutting the previous procedures.

The expert system is built using the Dantes tool, which was specifically developed for network applications (Mathonet et al. 1987). It uses an object-oriented paradigm supporting rules. It is functions similarly to the many other expert system tools of this sort now available. A diagram of part of an INCA rule is shown in figure 5.

The rules were obtained from experienced S.W.I.F.T. senior operators who acted as the project's experts and formed part of the project team. The system performs pattern matching on the antecedents of these rules to fire the consequents that typically consist of sequences of

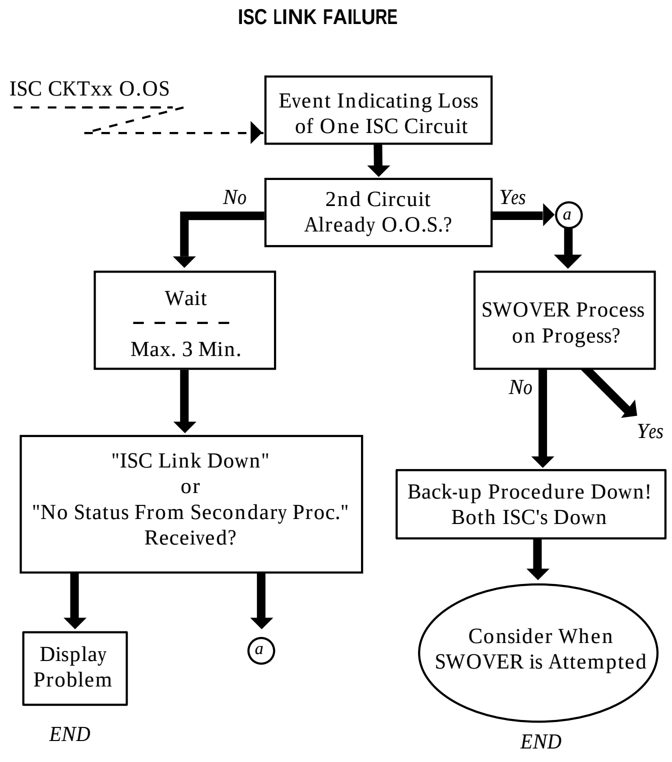

ISC LINK FAILURE
ISC CKTxx O.OS
Event Indicating Loss of One ISC Circuit
2nd Circuit Already O.O.S.?
No
Yes
a
Wait
Max. 3 Min.
SWOVER Process on Progess?
No
Yes
"ISC Link Down" or "No Status From Secondary Proc." Received?
Back-up Procedure Down! Both ISC's Down
Display Problem
a
Consider When SWOVER is Attempted
END
END

Figure 5. Part of an Inca Rule.

gathering additional information from the network and sending network commands (for example, disconnecting and reconnecting a link or putting displays on the screen).

The Dantes rules are defined for specific classes of objects and comprise three parts. The *trigger* specifies the network events or system actions that can cause the rule to fire. The trigger focuses the control of the inference procedure and avoids unnecessary search. The *state* is the value that an object must have for the rule to be fired. The *body* of the rule contains Lisp forms and is generally an If condition–Then action form (Mathonet et al. 1987). Temporal information plays a large part in the evaluation of events and in the rules.

The filtering and the automated responses, coupled with a simple mouse and menu user interface, reduce the amount of human attention needed for the incoming events. Thus, in place of one team of operators for each active system, one INCA team controls four active systems. Examples of the network problems that INCA can handle are

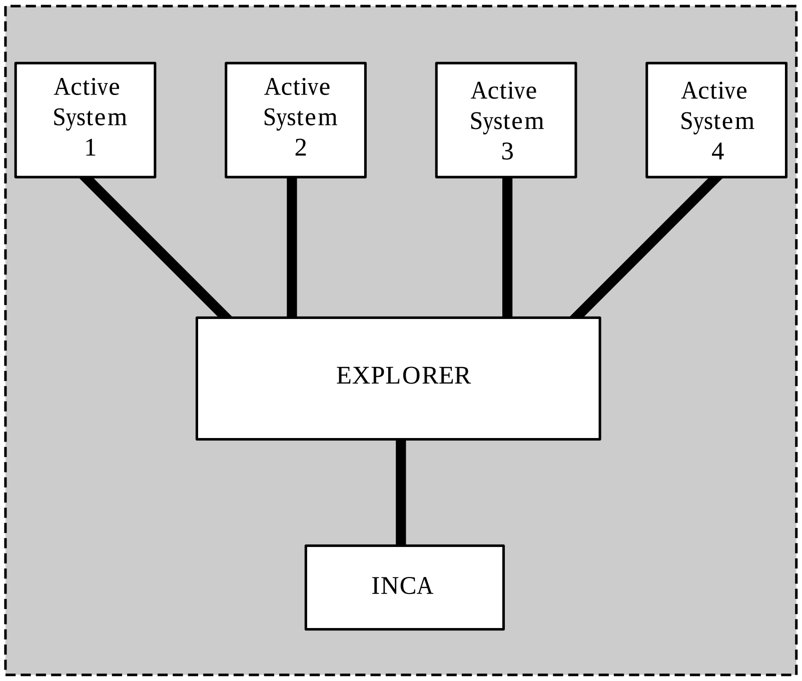


Figure 6. Inca in the Operating Center.

regional processor reloads, regional processor swaps, regional processor isolation procedures, link instabilities, and user line problems.

In addition to dealing with incoming events, INCA also has a dynamic network model builder that enables it to automatically update its model of the S.W.I.F.T. network by interrogating the system components. The dynamic network model builder allows it to automatically adjust to changes in the network configuration that regularly occur. Also, reporting facilities were written to allow the selection of previous events received and to keep backup copies of the event stream.

Embedding in the Network

INCA runs on two Explorer II Lx workstations, one in each S.W.I.F.T. operating center, each controlling half of the network. This division into halves both allows for phased changes from current operations and cuts the risk from INCA failure in half. The one multiwindowing Explorer is directly connected to four active systems, as shown in figure 6; thus, the original eight display and command devices have been replaced by just one terminal. To ensure continuity of operation, each

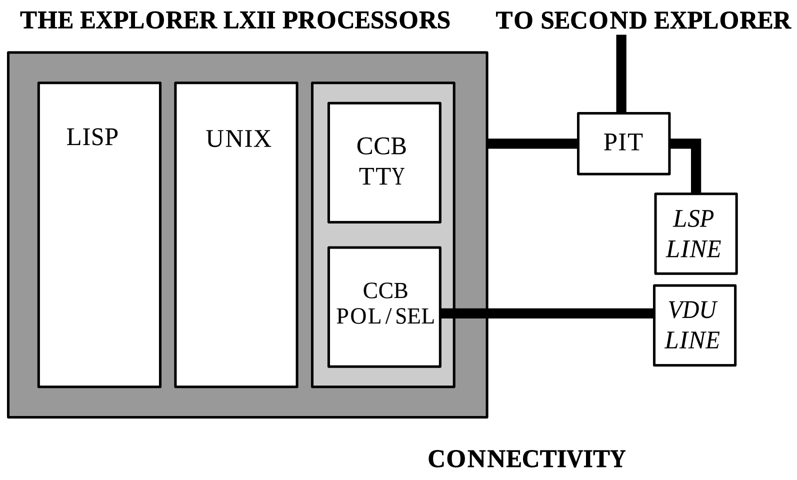


Figure 7. The Explorer Connectivity.

Explorer is connected to a backup Explorer that can immediately take over control in the event of a failure in the active machine. Connection to the active systems is through communications boards and programmable interface translator boxes to deal with the different protocols understood by the two types of machine and to enable signal splitting for the backup Explorer (figure 7).

The hardware is a Texas Instruments Explorer II LX machine running Unix. Although these workstations use Lisp, they do not involve the use of proprietary real-time operating systems or special real-time memory management but ordinary, nonreal-time processes such as garbage collection. In keeping with AI development, we used the Dantes shell as the programming environment. Dantes is an object-oriented environment using rules and is based on Lisp.

The backup is designed so that in backup mode, the Explorer receives all the events from the network that the active Explorer does, but it does not send commands to the network in response. Thus, the backup machine is kept fully updated on network events and can function immediately when switched into live mode. This backup design raises the issue of protection against software errors as well as hardware failures. It would be possible to have either a different software release on the backup machine or an activation of the backup with a slightly changed internal state by only allowing the backup to make deductions from a limited time window before becoming live. In either case, the idea is to avoid a software error that causes both machines to go down simultaneously. This possibility is the subject of continuing research.

Schedules and Implementation

A brief prototyping exercise was carried out during the first quarter of 1989. The INCA project officially started in April 1989. INCA has been working in one control center since October 1989; its processing functions were introduced in phases to minimize risks. INCA became completely operational at S.W.I.F.T.'s Netherlands Operating Center network control in February 1990 and at the U.S. center in May 1990. Experience to date has shown it is working well and is well accepted by the operations staff.

At the time of this writing (July 1990), the system has stabilized in use after we fixed a number of bugs that escaped the rigorous testing procedures. None of these bugs was serious enough to cause significant problems, and INCA was never withdrawn from operational use. On-call maintenance was provided for this initial period by S.W.I.F.T. knowledge engineers from the development group and the operations experts from the group. Maintenance will be turned over in stages to S.W.I.F.T.'s internal system support group, which is receiving training to deal with this new technology.

Reaction from users has been good: The operators have learned that INCA can be relied on to deal with the vast majority of network events. In fact, 97 percent of all events are automatically handled by INCA; of those remaining, the majority require manual actions, for example, changing a disk pack. Regular review meetings are held with the operators to ensure their views are understood. Various minor amendments have been made, for example, reducing the number of mouse clicks needed to send some commands, but no serious changes have been requested or needed.

In one respect, the reaction has been too good. The quick, on-schedule development of this complex system prompted a series of requests for additional features and functions to deal with other network aspects besides problem troubleshooting. After evaluating resource requirements and cost effectiveness, a number of these enhancements were successfully implemented. Procedures to discuss and control the growth of wish lists have had to be introduced. It is already clear that the project goal of substantial staff member saving can and will be achieved.

The development team consisted of a core of five people, three from S.W.I.F.T.'s corporate research group and two from operations. In addition, two knowledge engineers from Texas Instruments were attached for the first half of the development process. The total expenditure of time was six person-years. The system was phased into operation over the period from October 1989 to February 1990.

The criteria for successful implementation were that INCA should achieve staff savings, should be accepted by the client group for the functions it performs and its quality, and should improve response times and limit network downtime compared with the old system. Payoff to S.W.I.F.T. will be in the form of reduced network operator labor needs, estimated by the clients to be a reduction of 50 staff members.

Conclusion

We showed that a professional and structured approach to designing and implementing large-scale expert systems can lead to these systems being produced on time and within budget and being successfully accepted even in a most critical environment. We hope that this experience will help to remove some of the mystique surrounding expert system development and show that they can now be treated as one more element in the software engineer's tool kit.

It is possible to apply the techniques used in this project to areas other than network management. For example in financial dealing room systems, a similar environment exists: Many financial events are being reported, and the benefits of intelligent filtering and an automated response to common situations such as arbitrage opportunities are evident. The techniques we applied in INCA could also be applied here.

Research and development in many aspects of expert systems are still needed. However, the basic rules and object paradigm should now be accepted in the same way that, for example, the development of large databases is accepted as part of commercial information technology.

References

Goyal, S. K., and Worrest, R. W. 1988. Expert System Applications to Network Management. In *Expert System Applications to Telecommunications,* ed. J. Liebowitz, 3–44. New York: Wiley.

Phelps, R.; Ristori, F.; and Steinier, J. 1990. Managing AI System Development in Critical Large-Scale Applications. In Proceedings of the Sixth Conference on Artificial Intelligence Applications. Washington, D.C.: IEEE Computer Society.

Mathonet, R.; Van Cotthem, H.; and Vanryckghen, L. 1987. Dantes: An Expert System for Real-Time Network Troubleshooting. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 527-530. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence