Appears in *Knowledge-Based Intelligent Techniques in Industry* (chapter 8), L. C. Jain, editor, CRC Press, 249-275, 1998.

INTELLIGENT TELECOMMUNICATION TECHNOLOGIES

Gary Weiss, John Eddy, Sholom Weiss
Network & Computing Services
AT&T Labs
AT&T Corporation
United States

Telecommunication networks are extremely complex systems requiring high reliability and availability. The effective management of these networks is a critical, but complex, task. To help with this problem, the telecommunications industry has heavily invested in intelligent technologies. This chapter describes intelligent technologies and applications used within the telecommunications industry. Section 1 describes expert systems and data mining technologies. Section 2 describes several intelligent telecommunication applications and the intelligent technologies they employ. The next two sections provide detailed descriptions of representative modern telecommunication applications. Section 3 describes the ANSWER expert system, which monitors and maintains all of the 4ESS switching elements in the AT&T network. This system is noteworthy for the fact that it combines object and rule-based technologies, to gain advantages from each. Section 4 describes the use of data mining to predict extremely rare telecommunication equipment failures. This telecommunication application, and several others described earlier in the chapter, highlight the increasing use of data mining technology in the telecommunications industry to *automatically* acquire useful knowledge from the large quantities of data which are routinely available.

1 Intelligent Technologies

For many years the telecommunication industry has relied on intelligent solutions to help manage telecommunication networks. Building such applications involved acquiring valuable telecommunication knowledge from human experts and then applying this knowledge, typically by embedding it in an expert system. This knowledge acquisition process is so time-consuming that it is referred to as the "knowledge acquisition bottleneck". Data mining techniques are now being applied to industrial applications to break this bottleneck, by replacing the manual knowledge acquisition process with automated knowledge discovery. Telecommunication networks, which routinely generate tremendous amounts of data, are ideal candidates for data mining [1]. This section will describe expert system and data mining technologies and how they are evolving to solve complex industrial problems.

1.1 Expert Systems

Expert systems are programs which represent and apply factual knowledge of specific areas of expertise to solve problems [2]. Expert systems have been applied extensively within the telecommunications industry, but not without problems. Early expert systems required a knowledge engineer to acquire knowledge from the domain experts and encode this knowledge in a rule-based expert system. These rules were very "ad-hoc" and as the number of rules increased, the expert system became more difficult to understand and modify. The 4ESS-ES expert system, which is described later in this chapter, is an example of such a "first generation" expert system. Second generation expert systems attempted to solve these problems by using stronger methods, such as model-based and functional reasoning. Model-based reasoning is of interest to the telecommunications industry since model-based approaches can represent the structure and behavior of the telecommunication network components declaratively and then reason from first principles. While these expert systems seem preferable to first generation systems, they have not seen the same level of commercial success. In the field of telecommunications, this is because it is often too difficult to specify a behavioral or functional model at a sufficiently high level to make the model practical and yet have it be useful.

The design of telecommunication expert systems needs to recognize that virtually all telecommunication equipment incorporates self-diagnostic capabilities and hence it is not necessary to fully model the telecommunications network [3]. However, this does not mean that we should return to the previous approach of writing ad-hoc rules to reason based on the results of the self-diagnostic tests. Section 3 describes a hybrid approach which uses "affective" relations and object-oriented technology to define an abstract non-behavioral model for modeling devices. Another related approach, described in this chapter, is to use data mining to automatically build a causal, or Bayesian, network to explain a telecommunication network's behavior.

1.2 Knowledge Discovery and Data Mining

1.2.1 Overview

Knowledge discovery is a field which has emerged from various disciplines, including artificial intelligence, machine learning, statistics and databases. The knowledge discovery process involves identifying valid, novel, potentially useful and ultimately understandable patterns in data [4]. Data mining, the most researched topic in this process, involves finding interesting patterns in the data via data analysis and discovery algorithms. The knowledge discovery process, which has been described in detail in several recent papers [4, 5], is comprised of the following steps:

1. *data preparation:* selecting, cleaning and preprocessing the data (e.g., filling in missing values) and transforming it so that it is suitable for data mining
2. *data mining:* finding patterns in the data
3. *interpretation and evaluation*: interpreting and evaluating the patterns produced by data mining

A key motivation for knowledge discovery is that it can replace or minimize the need for the time-consuming process of manually acquiring knowledge from a domain expert. Knowledge discovery is especially attractive to the telecommunications industry since:

- Telecommunication networks are typically too complex to build complete simulation models
- Huge quantities of data are routinely available
- Domain experts often are not aware of subtle patterns in data and hence automated knowledge discovery can acquire new, previously unknown, knowledge

Many of the applications in this chapter utilize knowledge discovery. The step in the knowledge discovery process which typically requires the most work for telecommunications applications is the transformation step, which involves identifying useful features to represent the data. This step is complicated by the fact that telecommunication networks produce *sequences* of alarms, where it is not the individual alarms which are of importance but the behavior over time of the network. Since most data mining methods do not directly operate on temporal sequences, these sequences must be transformed so that these methods can be used. The Scout application in Section 2.1.4 and the forecasting application in Section 4 both take this approach. An alternative approach is to develop a data mining method which can reason about temporal relationships. Such a system, the telecommunication network alarm sequence analyzer, is described in Section 2.1.3.

1.2.2 Data Mining

Data mining finds interesting patterns in data. It is an example of inductive learning, since it is based on generalizing from past instances. A typical data mining application from the telecommunications industry is to predict the failure of a network component based on past alarm history. Data mining can be used to solve many tasks, including the following:

- *classification:* learning to map an example into one of several classes
- *clustering*: partitioning examples into categories which are not predefined
- *dependency modeling*: finding a model that explains dependencies between variables
- *sequential and temporal pattern detection*: discovering sequential or temporal patterns between/among examples

These tasks can be associated with real telecommunication problems. All of the applications described in this chapter can be considered classification tasks (e.g., is a network element faulty or not). The Trouble Locator and APRI systems, described in Section 2.1.2 and 2.2.1, respectively, build dependency models. The TASA application described in Section 2.1.3, and to a lesser degree the forecasting application described in Section 4, are both examples of temporal pattern detection. NYNEX has used clustering to target the most lucrative direct mail markets and to tailor sales messages to produce the maximum impact.

There are many data mining methods for solving the various data mining tasks. These methods vary in several ways, including: the time they require for learning, their tolerance of noise, the expected format of the data and the concepts they are capable of expressing. Rule induction and Bayesian, or causal, networks are two data mining methods used extensively within the telecommunications industry and by applications described in this chapter. Rule induction systems generate rule sets capable of classifying new examples, while Bayesian networks learn probabilistic concepts to account for the observed data. Both rule induction systems and Bayesian networks are particularly appropriate for telecommunication applications. Rule induction systems are appropriate because rules are easy to understand and can easily be incorporated into existing rule-based expert systems. The Swap-1 rule induction system [6] is used by two applications in this chapter. Bayesian networks are also appropriate since many telecommunication problems, such as isolating a faulty hardware component, are best handled probabilistically, due to a lack of complete simulation models. Many other data mining methods, such as decision trees and neural networks, can also be used to solve data mining tasks.

2 Intelligent Applications

This section describes several representative intelligent telecommunication applications and the intelligent techniques they employ. A more comprehensive listing, but with only a brief description of each, can be found in [7].

2.1 Network Management Applications

Network management applications, which involve the monitoring, diagnosis, and maintenance of telecommunication networks, are the most important applications in the telecommunication industry. Five such applications are described in this section.

2.1.1 Max & Opti-Max: Locating Problems in the Local Loop

The Max (Maintenance administrator expert) system [8], developed by NYNEX, diagnoses customer reported telephone problems in the local loop, the final segment of the telephone network that connects the customer to a central office. Max is a rule-based expert system which diagnoses problems based on results of an electrical test on the customer's phone line, specific knowledge of the customer's phone line and general equipment knowledge. Max determines where the trouble lies and selects the type of technician to solve the problem. Max was deployed in 1990 and has been able to reduce the number of incorrect technician dispatches over previous methods. One problem with Max is that its performance is affected by the local characteristics of each site and thus numerous rule parameters must be tuned to optimize its performance. This tuning process is time consuming and for this reason a system called Opti-Max [9] was created to automatically tune these parameters to appropriate values. Opti-Max takes as input a set of training examples, each of which includes a problem description and a diagnosis assigned by an expert, and then uses a hill-climbing search to find a set of parameter values which perform well on these examples. Opti-Max performs a type of automated knowledge discovery.

2.1.2 Trouble Locator: Locating Cable Network Troubles

Pacific Bell has an intelligent system which determines the location of troubles in a local telephone cable network [10]. This system uses data generated by a nightly automated test to help narrow down potential cables or network equipment which may be faulty; however, the test results are not sufficient to determine the exact cause. The Trouble Locator uses a Bayesian network and Bayesian inference [11] to solve this problem. The system begins by generating a local plant topology graph and then from this generates a Bayesian network, where each node in the network contains state information (belief of failure) of a plant component. This network also takes into account historical information about the components and the data from the overnight test. The belief of failure is then propagated throughout the network until equilibrium is reached, at which point a ranked list of faulty components can be generated. This system is used by preventative maintenance analysts as a decision support system.

2.1.3 TASA: Finding Frequently Occurring Alarm Episodes

The Telecommunication Network Alarm Sequence Analyzer (TASA) [12] is a system for extracting knowledge about the behavior of the network from a database of telecommunication network alarms. The goal of this system is to locate regularities in the alarm sequences in order to filter redundant alarms, locate problems in the network and predict future faults. TASA operates in two phases. In the first phase, specialized algorithms are used to find rules that describe frequently occurring alarm episodes from the sequential alarm data [13]. An episode describes a set of alarm sequences over a given time period and this set can include alarm sequences in which the specific order of alarms does not matter. An example rule describing an alarm episode is: *if alarms of types A and B occur within 5 seconds, then an alarm of type C occurs within 60 seconds with probability 0.7.* In the second phase, collections of episodes are interactively manipulated by the user so that *interesting* episodes from the original set can be found. TASA supports this process by providing operations to prune uninteresting episodes, order the set of episodes and group similar episodes. Pruning and ordering of the rules is accomplished by specifying the values of various attributes of the rules (e.g., the types or severities of the alarms or the confidence or frequency of such rules). TASA has been tested with telecommunication alarm data and interesting rules have been found and incorporated into existing alarm handling systems.

2.1.4 Scout: Identifying Network Faults via Data Mining

AT&T's Scout system proactively identifies recurring transient faults [14]. It operates by mining historical telecommunication data using machine learning and correlation techniques. In one approach, Scout identifies patterns of chronic problems directly from the data by examining the network behavior over periods of days and weeks. To do this, features which summarize time-varying historical data are extracted from the data so that standard machine learning algorithms can be used (i.e., algorithms which are not capable of explicit temporal reasoning). This featurization is accomplished by using two fixed consecutive time windows, W_1 and W_2. The objective is to use the measurements from W_1 to predict problems in W_2. One way of summarizing these measurements is to count the number of times each feature occurs within the window. Scout then used Swap-1 [6] to learn rules that predict recurring transient faults.

Another version of Scout uses a topological map of the network in order to improve its performance. The addition of this topological knowledge allows Scout to learn using time intervals of minutes and hours instead of days and weeks—something which is extremely important if acute failures are to be predicted. This knowledge also allows Scout to effectively identify problems with the same root cause. Scout has been deployed in AT&T's Network Service Centers since 1992 and has successfully increased technician productivity by identifying root cause problems and recurring transient problems.

2.1.5 4ESS-ES: Network Management for 4ESS Switches

Most of the switching capacity for domestic long distance traffic in the AT&T network is provided by 4ESS switches. Until very recently, the 4ESS-ES (4ESS Expert System) was responsible for managing these switches: for monitoring them, running diagnostic tests and filtering alarms. The 4ESS-ES is a first generation rule-based expert system, implemented in 1990 using C5, a C version of the popular OPS-5 [15] rule-based language. It consists mainly of shallow ad-hoc rules acquired over a period of many years from domain experts and suffers the same problems as most first generation systems—it is difficult to modify and maintain. This system was redesigned and re-implemented in 1996 using a hybrid, object-oriented, rule-based paradigm, which is described in detail in Section 3.

2.2 Other Intelligent Applications

In addition to network management applications, the telecommunication industry has developed many other types of intelligent applications, including: marketing applications, product configuration applications, help-desk applications and fraud detection/security applications. Since these applications are not unique to the telecommunication industry, only a single application will be described.

2.2.1 APRI: Predicting Uncollectible Debt

The telecommunications industry incurs billions of dollars of uncollectible debt each year. The Advanced Pattern Recognition and Identification (APRI) system was developed by AT&T's Consumer Laboratory to predict the probability of uncollectible debt based on historical data, including data of past uncollectibles [16]. The output of APRI is fed into a decision support system which can take a variety of actions, including blocking a call from being completed. APRI automatically constructs Bayesian network models for classification problems using extremely large databases. Bayesian networks were chosen for this problem due to the inherently probabilistic nature of the prediction problem and due to the highly unequal misclassification costs (i.e., the cost of mistakenly labeling an account as uncollectible vs. mistakenly labeling it as collectible) and skewed class distributions.

3 ANSWER: A Hybrid Approach to Network Management

This section describes ANSWER (Automatic Network Surveillance with Expert Rules), the operation support system responsible for maintaining the 4ESS switches in the AT&T long distance network. This system replaces the 4ESS-ES, the “first generation” expert system described in Section 2.1.5. ANSWER is noteworthy for the fact that it utilizes both rule-based *and* object-oriented technologies, by employing a rule-based extension to the C++ object-oriented programming language. The inclusion of object oriented technology facilitated the design and implementation of ANSWER by providing a principled way to model the 4ESS as a hierarchical collection of devices.

This section is organized as follows. Section 3.1 provides a functional overview of ANSWER. Section 3.2 describes the object model used by ANSWER to model the 4ESS switch. Section 3.3 provides a brief description of R++, the rule-based extension to C++ which was used to implement the expert system component of ANSWER. Section 3.4 discusses the types of reasoning performed by ANSWER and illustrates this with several examples. Finally, Section 3.5 summarizes the significance and contributions of the approach taken by ANSWER.

3.1 Functional Overview

Surveillance technicians at AT&T’s two network control centers use ANSWER to help them monitor and maintain the 4ESS switches, which handle the majority of calls in the AT&T long distance network. Their goal, and the goal of ANSWER, is to minimize the number of service affecting incidents and the number of blocked or lost calls. ANSWER helps to achieve this goal by observing thousands of 4ESS alarm messages and notifying technicians of conditions which are serious and most likely will require human intervention.

ANSWER is a complete telecommunications operation support system and includes many components. The central and most important component is the knowledge base, which is responsible for all of ANSWER’s intelligent behavior. The basic input/output functionality of ANSWER, from the perspective of its knowledge base, is shown in Figure 1.

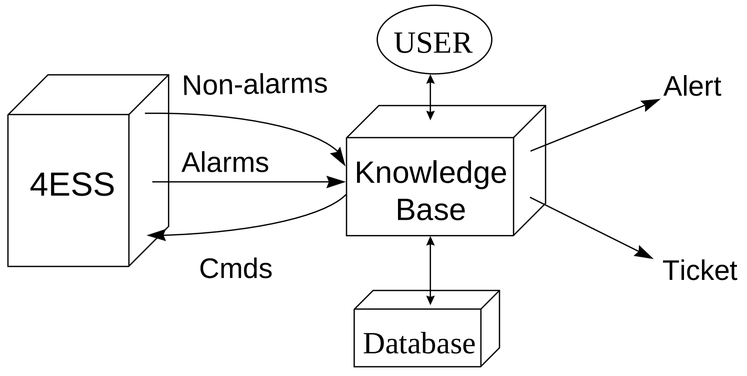


Figure 1: Functional View of ANSWER

The key inputs to the knowledge base are alarm and non-alarm messages from the 4ESS switches. Alarm messages provide information about anomalous conditions which occur on a specific device within the 4ESS, while non-alarm messages provide status information or the results of diagnostics previously requested by the knowledge base. The key outputs of the knowledge base are commands to the 4ESS (e.g., to request diagnostics to be run) and notifications to the technicians of problems which require further attention. These notifications come in the form of alerts and tickets. Alerts are sent to the surveillance technicians at the network control centers, who then decide what action to take. They may either ignore the alert, defer processing of the alert pending additional information, or manually create a “work order” ticket and dispatch it to the on-site work force. Members of the on-site work force are located at each 4ESS switch and can make physical equipment repairs such as replacing a faulty circuit pack. In cases where the need for physical human intervention is clear, the knowledge base may autonomously create the ticket and directly dispatch it to the on-site work force, bypassing the centralized surveillance technicians.

The main task of the knowledge base is to decide when to generate an alert or ticket. These notifications are not generated for each alarm received by the knowledge base, and hence one of the key functions of the knowledge base is to perform *alarm filtering*. For example, in many cases an alert or ticket is only generated if a threshold number of alarms is exceeded within a specified period of time. The knowledge base is just one part of ANSWER. In addition to the inputs and outputs just described, the knowledge base is also connected to a database and to users, via a graphical user interface. The database is used by ANSWER to provide persistent storage of alarms, alerts and other information, some of which is placed there by the knowledge base and some of which is placed there by other parts of the system. Users can also interact with the knowledge base via the graphical user interface to retrieve information and to set various options to customize the behavior of the knowledge base.

3.2 The Object Model

3.2.1 Overview

One of the key advantages and distinguishing characteristics of ANSWER's knowledge base is that, in addition to using rule-based programming, it also uses object-oriented technology—a technology now in widespread use in industrial applications. For object oriented technology to be useful in this context, there must be a way for the knowledge base to model the 4ESS as a collection of objects. Such an object model is shown in Figure 2 using Rumbaugh's Object Modeling Technique [17].

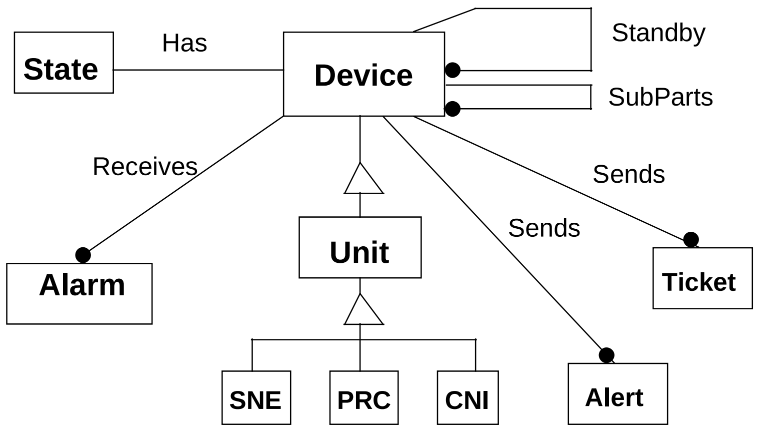


Figure 2: The 4ESS Object Model

The 4ESS is readily viewed as a collection of devices and consequently *device* is the central object in the 4ESS object model. The object model is described by the following:

- subparts: a one-to-many relation on device that allows any device (including the 4ESS itself) to be viewed as a hierarchical collection of devices
- standby: a one-to-many relation on device that specifies the devices able to take over for a device if it fails
- state: the state of a device (in-service, out-of-service, etc.) and the time it entered that state
- alarm: each device has an ordered list of alarms, which contains the alarms generated by the 4ESS on behalf of the device
- alert and ticket: these objects are associated with a device and are created when the knowledge base decides the device may require human intervention

In addition, Figure 2 specifies an inheritance hierarchy for the device class. Device is intended to represent a relatively abstract object. The device class has a subclass named Unit which represents a generic 4ESS device. Finally, the unit class has many subclasses, of which only three are shown in the figure. These subclasses represent either a very specific 4ESS hardware component (e.g., PRC represents the main processor) or a specific class of equipment that shares many common characteristics. Inheritance allows us to share and/or specialize device behavior, as appropriate. By having an object-oriented language integrated with a rule-based language, we can inherit not only methods and data members (i.e., functions and variables), but also rule-driven behavior. For example, ANSWER's PRC class only contains rules which specify behavior unique to processor devices—generic device behavior is inherited from the unit and devices classes.

3.2.2 The Device Model and Model Instantiation

Devices are the key object in our model and therefore it is important to understand how these objects are used, and in particular, how they are created. A key requirement for our model is that it be *dynamically* built from the information sent to it from the 4ESS. There are two reasons for this requirement: flexibility and efficiency. The flexibility of a dynamic model arises from the fact that no up-front configuration information is required—which is essential since each 4ESS switch is unique and components are continually added and removed. The second advantage of a dynamic model is that it permits us to model only the components which have abnormal activity, thereby reducing the size of the model and thus realizing time and space savings.

The knowledge base is driven primarily by two types of events: 4ESS alarms and one-second timer ticks. Each 4ESS alarm refers to a device by specifying up to three levels of device information (i.e., the 4ESS device hierarchy is at most 3 levels deep, excluding the 4ESS itself). At each level in the hierarchy, a device type is specified, along with an integer-valued device identifier, to ensure that each device within a 4ESS switch is unique. The timer ticks update the expert system's internal clock and drives all of its time-based behavior. A device model, which is a view of the devices in the object model at some instant in time, is shown in Figure 3. This figure will also be used to describe the device creation process.

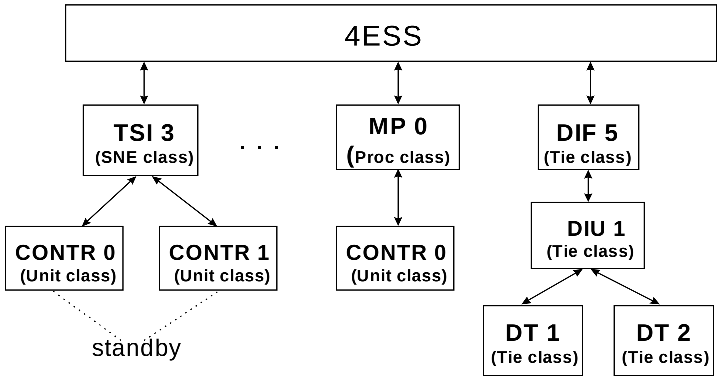


Figure 3: A Device Model

When the knowledge base receives an alarm and the device specified in the alarm doesn't already exist in the model, then it is created. The class of the device object is determined by a table lookup. For example, when a TSI:3 device is created, an object of class SNE is created. The model requires that all "ancestors" of this device (i.e., devices that contain this device) be present in the model, so if they do not already exist in the model then they are created. For example, when the knowledge base receives an alarm for the device "DIF:5 DIU:1 DT:2" it will first attempt to create the DT:2 object. However, if the DIF:5 or DIU:1 of which this DT:2 is a part do not already exist in the model, they will be created first. When devices are created, data driven rules check for "invariant" relations and create them in the model if appropriate. For example, if "TSI:3 CONTR:1" already exists in the model, then the creation of "TSI:3 CONTR:0" will cause a standby relation to be formed between these two objects. Device deletion is driven by timer events; a device is deleted if it is in the "in-service" state and has been in that state for a specified period of time.

3.3 R++: A Rule-Based Extension to C++

In order for ANSWER to make effective use of the 4ESS object model and still use rule-based inference to implement the "reasoning" in ANSWER, an implementation language was needed that supports both the object-oriented and rule-based programming paradigms and provides for the tight integration of rules and objects (e.g., so that rules can be triggered based on changes in the data members of an object). An evaluation of existing programming languages indicated that no satisfactory language was available, so R++, a rule-based extension to the C++ object oriented language, was developed.

R++ rules are considered another type of C++ member function and share the object-oriented properties of C++ member functions: inheritance, polymorphism and dynamic binding. R++ is implemented as a preprocessor which runs before the C++ compiler is invoked. Figure 4 shows the declaration of ANSWER's Device class. Note that each device's *sub_parts* and *standby* devices are specified by a set of device pointers. The optional *monitored* keyword in the class declaration identifies data members which may trigger rule evaluation. The declaration also includes the declaration of the link_standbys rule, which ensures that the standby relation is always kept up-to-date.

```
class Device {
protected:
   String                          type;          // type of device (e.g., TSI)
   int                             number;        // identifies device uniquely
   monitored State                 *state;        // ptr. to device's state information
   monitored Alarm                 *new_alarm;    // ptr to newest alarm received
   monitored Device                *part_of;      // points to device this is part of
   monitored Set_of_p<Device>      sub_parts ;    // set of devices that are sub-parts
   monitored Set_of_p<Device>      standby;       // pointer to set of standby devices

   rule link_standbys;
};
```

Figure 4: Declaration of Class "Device"

Rules have a special *if-then* syntax, where the if (antecedent) and then (consequent) parts are separated by an arrow (⇒). The rule in figure 5a can be translated as: *if there is a new alarm for this device and the state of a standby of this device is out-of-service, then send an alert.* R++ also provides the ability to write rules which operate on container classes, like sets and lists. Using an R++ feature called "branch binding", a rule can be applied to each element of the container (the at-sign "@" is the branch-binding operator). The rule in Figure 5b relies on the fact that there is a list called *Devices* that contains all of the devices in the model. For those not familiar with C++, the variable *this* always refers to the object itself—in the case of the rule in Figure 5b, the device on whom *link_standbys* is being performed. The antecedent of the rule in Figure 5b can be translated as: *for each device dev in the set Devices, check if dev is the standby of this device.* The consequent then updates the standby field of each device to reflect the new standby relationship.

```
// alert if get alarm and standby out of service
rule Device::alert_if_standby_out_service
{
   new_alarm &&
   Device *stby = standby &&
    State *st = stby→state &&
    state→name == "out_of_service"
⇒
   send_alert("standby_oos, this, stby);
};
```

(a)

```
// link standby devices together
rule Device::link_standbys
{
   Device *dev @  Devices  &&
   is_standby(dev)
⇒
   this→add_standby(dev);
   dev→add_standby(this);
};
```

(b)

Figure 5: Two R++ Rules from ANSWER

The key difference between rules and ordinary C++ member functions is that changes to data members in the antecedent of the rule automatically cause the rule to be evaluated and, if the antecedent evaluates to TRUE, then the consequent is executed. Thus, rules are *data-driven*. In the example in Figure 5a, whenever a new alarm is received on a device or the state of one of its standby device changes, the rule will be automatically reevaluated. The key difference between R++ rules and rules in other rule-based languages is that R++ rules are *path-based*. This means that the antecedent of an R++ rule can only include data members which the class has access to—typically through a pointer reference. Even though R++ rules are therefore less expressive than rules in other languages (because they cannot reference arbitrary objects), this is an advantage because it ensures that R++ rules *respect the object model*. For those interested in a more in-depth understanding of R++, see [18].

3.4 Reasoning Model

The 4ESS object model provides a principled way of reasoning about the failures in the 4ESS switches. It implicitly contains information about the structure and behavior of the 4ESS. For example, if a standby relationship exists between two device objects, A and B, then device A's standby field will point to device B, and vice versa. Much of the reasoning in ANSWER is accomplished by using *affective relations.* Affective relations define a highly abstract non-behavioral representation for modeling devices and are named for the fact that one component affects another in a diagnostically important way [19, 20]. These relations are too weak to simulate device behavior, but serve to organize the domain knowledge in a coherent way; ad-hoc heuristics are replaced by a smaller set of general principles based on affective relations. Affective relations express aspects of the design at a level of abstraction that expert troubleshooters use to link symptoms to faults, and hence are easily acquired.

Two important affective relations used by ANSWER are the standby and sub-part relations. These relations are very general and can potentially apply to any device. Rules which maintain and use the standby relation were shown earlier in Figure 5. For example, the rule in Figure 5a shows how the standby relation is important for monitoring the state of the 4ESS, since this rule will cause a "warning" alert to be sent out when an alert comes in on a device whose standby is already out-of-service. The sub-part relation is very important for diagnosis, since it can be used to isolate faults. For example, ANSWER has a rule which says that if many of a device A's sub-parts fail, then the failure is most likely with device A.

It is worthwhile to compare the reasoning in ANSWER with that of its predecessor, the 4ESS-ES. In the 4ESS-ES, the reasoning was not based on affective relations, due to the lack of a general model of the 4ESS; instead, the reasoning in that system was based on many (overly) specialized ad-hoc rules. There were many cases where a single general rule was not quite adequate due to small differences in device behavior. The solution to this problem in the 4ESS-ES was to have a completely separate, nearly identical, rule for each device; the solution in ANSWER is to have a single general rule and specialize it using inheritance for the few devices which require special handling.

3.5 Discussion

ANSWER's hybrid approach of using rule-based and object-oriented technologies has proven to be highly effective. By providing an abstract device object with diagnostically motivated affective relations, a simple form of model-based reasoning was able to be applied to a domain normally too complex for such methods. Thus, this approach has led to a middle-ground between model-based reasoning and heuristic (ad-hoc) expert systems. The use of object oriented technology also provided a principled approach for designing, implementing and organizing the expert system, and is responsible for ANSWER being a more comprehensible and maintainable system than its predecessor, the 4ESS-ES. Another significant advantage of our approach is that by using a slight extension to a "mainstream" programming language for implementing the knowledge base, the entire ANSWER system could essentially be implemented using a single language. This permitted the tight integration of the knowledge base with the rest of ANSWER. In the 4ESS-ES, interface routines had to be written to allow the knowledge base's C5 code to communicate with the rest of the system. This prevented the C5 rules from being triggered by changes in other parts of the system, and this led to several artificial and sub-optimal architecture and design decisions.

ANSWER is fully deployed and is monitoring and maintaining all of AT&T's 4ESS switches. Additional operation support systems are now being implemented using R++ and the approach described in this section; in fact, some of the design and implementation work done for ANSWER is being reused on these new projects.

4 Forecasting Telecommunication Equipment Failures from Time Series Data

The previous section described how combining object-oriented and rule-based technologies improved the development and maintainability of telecommunication-based expert systems. The example described in this section shows how data mining can improve the quality of the domain knowledge to be incorporated into such systems and also minimize the manual effort required to acquire this knowledge.

Reliability is a critical factor in the design of telecommunications networks. Errors may occur during the transmission of data over the network, but these errors can be detected and the data rerouted through alternate paths. The effect of the failure of a single component is limited due to the redundancy in modern large-scale telecommunications networks. The failure of a singular, major component, like an entire switch or a major component in a switch, is a very rare, but catastrophic event. In order to help diagnose and prevent problems before they occur, modern telecommunication equipment contains self-diagnostic testing capabilities. When any of these tests fail, an alarm message is sent to a centralized site, where it may be handled by a human or by an expert system (such as the ANSWER expert system described in the previous section). Many of these alarm messages are caused by temporary transient problems. The main objective of the data mining effort described in this section is to identify patterns of alarms that can help predict catastrophic equipment failures. These rare events may not be forecast with 100% accuracy, but their effect is so serious that identifying any increase in risk of failure is of great value.

We describe a general approach for transforming time-series data into the classical case-based representation. Standard machine learning classification methods, such as neural nets or decision trees, can then be applied to the transformed data. Predictive performance is maximized by varying a sampling period window and a prediction period window during the data transformation. We applied this technique to time-series data with 176 features over tens of thousands of cases, with the goal of forecasting a rare catastrophic failure that can occur over a massive communications network.

4.1 The Problem: Mining Time Series Data

Data mining techniques have been successfully applied to classification tasks. Many classification techniques have been researched over the past twenty years, resulting in many useful algorithms for finding predictive patterns in data. The data used in classification tasks are modeled in a standard case-based representation, where each case includes a set of feature variables and a single class variable. The prediction task is then to predict the class variable based on the feature variables. To take advantage of existing techniques, it is necessary to model the data using the case-based representation. Unfortunately, the data sets from many tasks are not naturally expressed in this standard format. Time series data, for example, are not naturally expressed in the standard case-based format and cannot be trivially converted into this format. Time series may contain one or more variables; the telecommunication alarm time series discussed later in this section contains several variables, but the simple time series presented in Figure 6 contains only a single variable.

TIME	MESSAGE
509	B
601	A
601	C
607	X
702	D
.	
.	
.	
952	A
953	C
953	X
967	B

Figure 6: Prototypical Time Series Data

In the time series in Figure 6 the number represents a time unit, perhaps seconds, and captures the ordering and positioning of each datum on a continuous time line. The message that occurred at that time is uniquely identified by a message letter (A, B, C, etc.). A general goal of our data-mining effort is to discover the pattern of alarms that identifies the pending arrival of some distinctive event. In our domain, the distinctive events are network and equipment faults, which themselves cause an alarm to be generated. Therefore, our goal is to identify a pattern of alarms that identify the pending arrival of the *target* alarm—the alarm to be predicted. In Figure 6 the target alarm is represented by the letter X. Our pattern of alarms can be thought of as the precondition of a rule, where the consequent of the rule is the prediction of the target event. A simple rule that might be induced from the data in Figure 6 is:

IF C occurs, *THEN* X occurs next.

For most real world time series and associated prediction problems, there are seldom such simple and obvious rules. Instead, a great deal of processing power, data modeling and machine learning are required to induce the rules. For these time series, not only do we not know what pattern of messages are strongly predictive of an event, we also do not know how far back in time to search for the predictive patterns. Similarly, we do not know how far into the future we can successfully predict the occurrence of the interesting event. A data-mining technique is needed that systematically varies window sizes in order to maximize the predictive performance of the rules uncovered by the classification method. The techniques that we describe have been applied to a very large database of time series data that represents alarm messages that are emitted by equipment in a large telecommunications network. There were 176 different message types with nearly one million separate messages. Our objective was to predict *extremely* rare, network critical failures.

4.1.1 Models for Representing Time Series Data

A standard model for representing data for classification tasks is shown in Figure 7. In this spreadsheet format, each row represents a single case. For each case, the columns represent the values of features, F_1 through F_n. The example data shown in the figure are numeric, but nominal or categorical features (such as true and false) are also possible. In fact, for our particular task of predicting network equipment failures, the features were categorical and included features such as the type of alarm and the name of the equipment reporting the problem. The final column in the figure, C, is special because it contains the class of the case—that is, the value the classification task is attempting to predict. While multiple values (such as, child, teen, adult, senior) or numeric ranges (such as, Age > 35) are possible, the class variable is usually binary (e.g., 1 indicates a fault will occur and a 0 that a fault will not occur).

	F_1	**F_2**	**F_3**	**. . .**	**F_n**	**C**
S_1	1	3	20	. . .	9	1
S_2	0	6	15	. . .	8	0
S_3	1	7	8	. . .	10	0
S_4	1	4	6	. . .	7	1
.	.	.	.	. . .	.	.
.	.	.	.	. . .	.	.
.	.	.	.	. . .	.	.
S_n	0	1	2	. . .	5	1

Figure 7: Standard Case-Based Format for a Classification Task

Our goal is to model time-series data using the standard classification format shown in Figure 7. One area of research that has successfully dealt with time ordered data is time series analysis. Figure 8 shows a typical time-series analysis problem involving unit sales data of a fictitious product. The figure shows that the unit sales for today is 56. It also shows the sales of the product in the previous 5 days, t-1 through t-5.

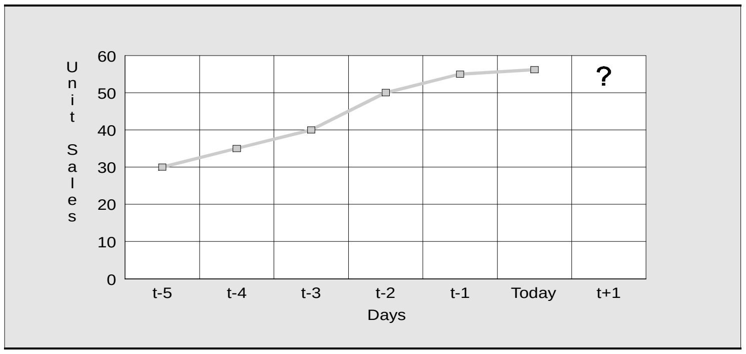


Figure 8: An Ordered Time Series of Unit Sales for a Fictitious Product

A time-series analysis of unit sales might attempt to predict the unit sales for tomorrow (time t+1) by looking at the prior unit sales. The following is a hypothetical equation for forecasting future sales:

$$S_{t+1} = (w * S_t) + \left((1-w) * \left(\left(\sum_{i=1}^{n} S_{t-i} \right) / n \right) \right)$$

Equation 1: A time-series analysis equation to forecast sales

The equation shows that tomorrow's unit sales is a weighted function of the current sales and the average of the previous days sales, where the weight is w and the number of previous days is n. This "linear" approach is the classical forecasting method. The classical time-series approach cannot be readily applied to certain complex data-mining tasks. In Figure 8, the measure of sales was a continuous variable, whereas we are interested in predicting a discrete, categorical, class variable—the event corresponding to a catastrophic failure of switching equipment. Furthermore, the features we use to make this prediction, the fields in the alarm messages, are also categorical. Classical time-series analysis requires numerical features. Also, time-series analysis has been most successful when the measured variables are univariate. In the sales example, there is only one measure, unit sales each day. In fact, what this equation suggests is that today's sales is a strong indicator of tomorrow's sales. It is a very different situation when the raw data are time-stamped messages that are used to forecast rare catastrophic events. Events are discrete, that is, they either occur or they do not. Also, these data involve many different variables.

It is fascinating to see how time-series analysis repeatedly uses the same data over and over to make predictions. Our approach borrows a few concepts from classical linear time-series analysis to learn rules that will forecast a future event. At any point in time, we look backward a fixed unit of time and observe the count of each message type. Then we look forward some other fixed unit of time and observe whether the crucial event occurred. As a simple example, suppose that there are only three types of messages: A, B and X messages and it is the X message we want to predict. As shown in Figure 9, it is possible to look back from any point in time and determine the frequency of A and B messages (and possibly X messages) and then look forward into the future to determine if any X message has occurred.

```
                 PAST            CURRENT          FUTURE
S0  |--------A--------B----B-----------A-------|------------------------X-------------|
              Monitor Window                  Prediction Window
```

Figure 9: Using the Past to Predict the Future

Each case is represented by a monitor window, M, and a prediction window, P. The monitor window contains the messages that are monitored in order to make a prediction about the future and the prediction window is scanned for the alarms that we are trying to predict (i.e., determines the class of the case).

Figure 10 demonstrates the process of generating cases from a time series by repeatedly shifting the window forward by one prediction window. The top of the figure shows the alarms on a continuous time line and below that are the separate cases, S_1 - $S_{n.}$

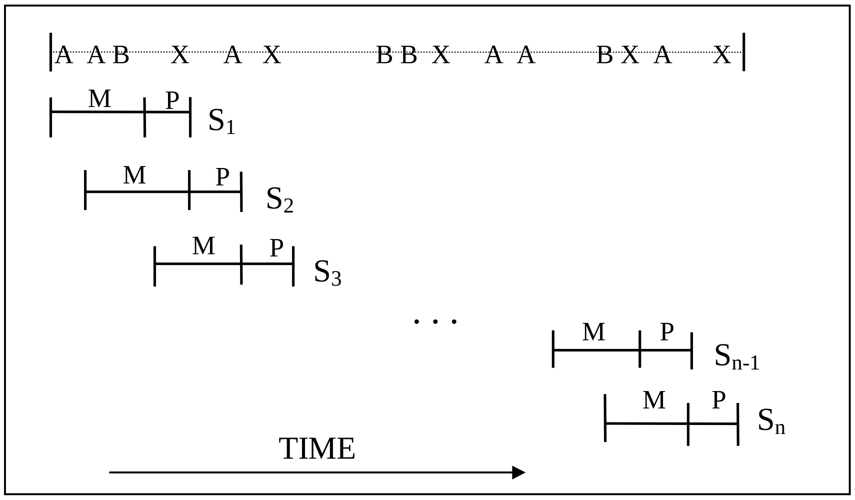


Figure 10: Samples of Monitor and Prediction Windows Across Time

From the time series we can now generate a case-based representation of the data. Each message type (A, B and X) corresponds to one of the features in Figure 7 and the value of each feature is the frequency count of that message in the monitor window. The value of the classification variable (the C of the last column in Figure 7) becomes either the frequency count of the target alarm in the prediction window or a 1 (or 0) to indicate the presence (or absence) of the target alarm in the prediction window. In this chapter we use the later encoding for the class variable, since we are only concerned with whether the target alarm occurs or not. The case-based representation of the time-series data in Figure 9 is shown in the first row of Figure 11 and the first three samples from Figure 10, S_1 - S_3, are shown in the following three lines. The entry for S_1 in Figure 11 indicates that there are two messages of type A and one message of type B within the monitor window, no X alarms within the monitor window and that the target message, X, does occur in the prediction window. This method of modeling time-series data is very general and can be applied to any data regardless of number of variables, window sizes and period of time.

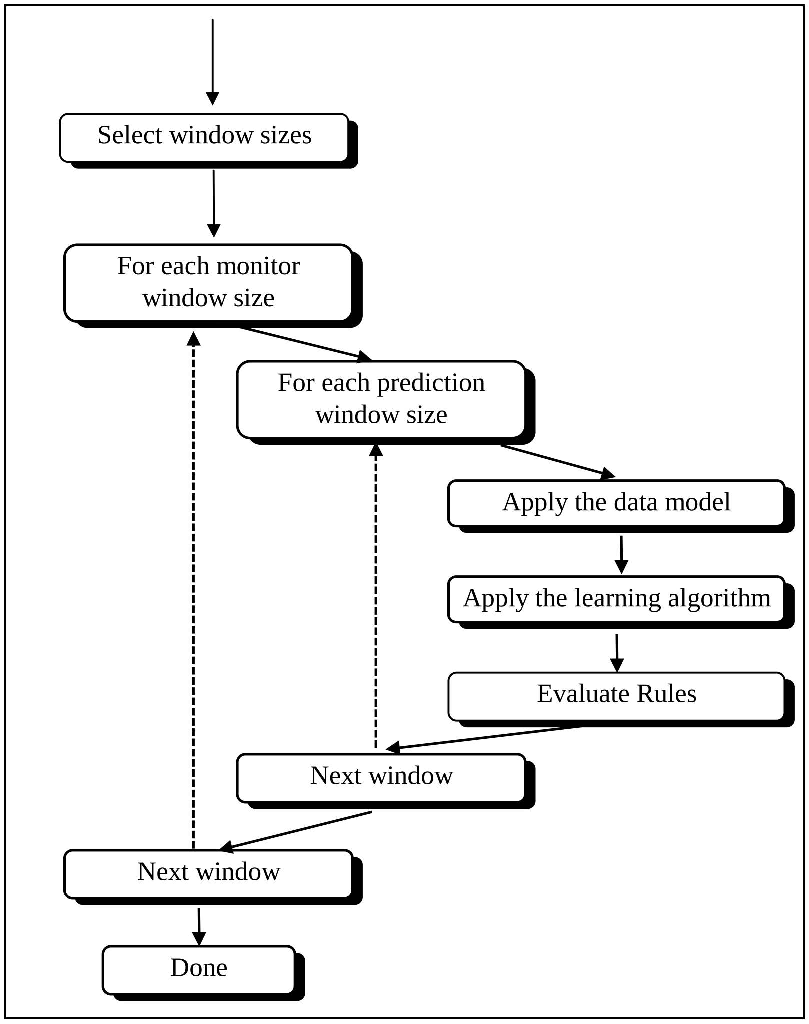
	A	B	X	C
S_0	2	2	0	1
S_1	2	1	0	1
S_2	1	1	1	0
S_3	1	0	1	1

Figure 11: Case-Based Representation of Time-Series Data

4.1.2 Exploring Window Sizes

Selecting the monitor window size is difficult when predicting the occurrence of rare events in communication networks. The messages in this domain are tracked to the second. There is nothing in the domain that suggests how to help decide how far back into the past to look for a pattern of messages. The problem is to discover where in time the critical information lies that indicates an upcoming catastrophic event. Is it important to monitor the messages that occurred in the previous fifteen minutes? In the previous hour, day, week, or even month? Without some special understanding about this domain it is not possible to know a priori what size monitor window is appropriate. The data must be mined by exploring various sizes of the monitor window.

Similarly, there must be a fixed size for the prediction window of time for forecasting the predicted event. It is useless to merely say that sometime in the future the event we are looking for will occur. In many domains it is always the case that the event will eventually occur. It is necessary to say an event will occur in the next hour, the next day, or some other time unit. Again, it is necessary to mine the data by exploration to discover what prediction window works for a particular task. While a priori we do not know anything about the sizes of the windows that will help us forecast events, we do know some desirable characteristics of the window sizes. For the monitor window, we prefer that the window is small rather than large, since, for example, it is simpler to keep track of the messages from the previous fifteen minutes than from the previous hour. The prediction window requires more of a balancing act. Too short of a prediction window will lead to predictions which are not useful. For example, when predicting a failure in a telecommunication network, it is necessary to predict the failure far enough in advance so that there is enough time to repair or reroute around the failure before it occurs; however, if the prediction window is too large, then the predictions may be meaningless (e.g., predicting that a failure will occur sometime in the next two years is not typically a useful prediction). Given the amount of time that it takes to execute most learning algorithms on large amounts of data, it is impractical to explore all possible window sizes. Figure 12 describes a reasonable procedure for mining the data with a pre-selected list of window sizes.



Select window sizes
For each monitor window size
For each prediction window size
Apply the data model
Apply the learning algorithm
Evaluate Rules
Next window
Next window
Done

Figure 12: Algorithm for Data Mining with Various Window Sizes

4.1.3 Testing the Model on a Large Database

The massive AT&T communication network is made up of high-capacity communication links connected by telecommunication switches. The total number of switches is surprisingly small, fewer than 150, even though the network spans the entire continental United States. The complete failure of a switch is a potentially catastrophic event. AT&T uses many strategies to minimize the effects of a complete switch failure on network traffic. Network traffic can immediately and automatically be routed through other switches by sophisticated software that observes the network traffic flow as a whole. The rerouting occurs on excess network capacity that is specifically provided to handle any congestion problems. AT&T also has many years of experience in designing and building fault tolerant control software that is embedded in the switches. The result is that a catastrophic switch failure is extremely unlikely and its effects are handled by excess network capacity so loss of network call traffic is minimized. Nonetheless, while the failure of a switch is rare, when it does occur the consequences are serious. Ideally, it would be best to forecast the occurrence of a failure and correct it before it occurs or at least minimize its effects by rerouting the calls prior to the failure.

We studied a historical database that contains alarm messages of over 50 switches for a period of two months. Switches generate nearly 100,000 alarm messages a week. Our database was filtered to somewhat less than 1 million alarm messages with fewer than 50 failures. Furthermore, many of these failures were bunched in time, indicating that there was only a single failure. While the goal is to forecast a catastrophic event, the paucity of cases for the failure events would require extremely strong predictors to achieve high accuracy forecasts. An alternative strategy is to identify some patterns that are highly predictive, but forecast only a subset of the catastrophic events (i.e., to trade-off recall for higher precision). Experiments were performed with varying time periods for the monitor and prediction window. The SWAP-1 rule induction system was used and learned prediction rules of the form: "If A and B occur in the monitor period, then a failure will occur in the prediction period." The best results were achieved when the monitor and prediction windows were both set to 15 minutes. Not all failures could be predicted accurately, but two patterns were identified that occurred prior to failures but never prior to normal periods. These results indicate that further study is warranted. However, before a more comprehensive study can be undertaken, more data must be obtained. In our case the problem isn't so much the lack of data, but rather the scarcity of failures within the data.

5 Conclusion

This chapter has described several intelligent telecommunication technologies and applications. Two technologies were highlighted: expert system technology and data mining technology. Expert system technology is the more mature technology, with many commercial successes over the past two decades. This chapter's description of ANSWER shows how object technology—a technology which is enjoying great commercial success—can be integrated with rule-based technology to make it easier to develop and maintain an expert system.

Both rule-based expert system and data mining technologies help the telecommunications industry deal with the large quantities of available data. For example, the ANSWER expert system processes thousands of alarm messages daily and from these is able to diagnose equipment problems. In this case, the knowledge contained within the expert system was manually acquired from domain experts. Data mining technology, however, allows useful knowledge to be automatically acquired directly from the data. This was demonstrated by several of the applications in Section 2 and by the data mining application in Section 4, which is an attempt to supplement the manually acquired rules in ANSWER with rules automatically acquired via data mining. While it is unlikely that data mining will totally eliminate the need to acquire knowledge from human experts, data mining technology is making significant contributions to the telecommunications industry and we expect these contributions to accelerate as advances are made in this relatively new technology.

References

[1] Weiss, S., Indurkhya, N. (1998), *Predictive Data Mining: A Practical Guide*, Morgan Kaufmann, San Francisco.

[2] Hayes-Roth, F., Waterman, D., Lenat, D. eds. (1983), *Building Expert Systems*, Addison-Wesley, Reading. MA.

[3] Crawford, J., Dvorak, D., Litman, D., Mishra, A., Patel-Schneider, P. (1995), Device Representation and Reasoning with Affective Relations, *Proceedings of the Fourteenth International Conference on Artificial Intelligence* (IJCAI-95), pp. 1814-1820, Montreal, Quebec, Canada.

[4] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996), From Data Mining to Knowledge Discovery: An Overview, *Advances in Knowledge Discovery and Data Mining*, Fayyad, U., Piatetsky-Shapiro, G., Smyth, P, Uthurusamy, R., eds., MIT Press, pp. 1-34.

[5] Brachman, R., Anand, T. (1996), The Process of Knowledge Discovery in Databases, *Advances in Knowledge Discovery and Data Mining*, Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds., MIT Press, pp. 37-57.

[6] Weiss, S., Indurkhya, N. (1993), Optimized Rule Induction, *IEEE Expert*, Vol. 8, No. 6, pp. 61-69.

[7] Hedberg, S. (1996), AI's Impact in Telecommunications—Today and Tomorrow, *IEEE Expert*, Vol. 11, No, 1, Feb., pp. 6-9.

[8] Rabinowitz, H., Flamholz, J., Wolin, E., Euchner, J. (1991), NYNEX MAX: A Telephone Trouble Screening Expert, *Innovative Applications of Artificial Intelligence 3*, Smith, R. & Scott, C., eds., AAAI Press, Menlo Park, CA, pp. 213-230.

[9] Merz, C., Pazzani, M., Danyluk, A. (1996), Tuning Numeric Parameters to Troubleshoot a Telephone-Network Loop, *IEEE Expert*, Vol. 11, No. 1, Feb., pp. 44-49.

[10] Chen, C., Hollidge, T., Sharma, D. (1996), Localization of Troubles in Telephone Cable Networks, *Innovative Applications of Artificial Intelligence*, Vol. 2, AAAI Press, Menlo Park, CA, pp. 1461-1470.

[11] Pearl, J. (1988), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco.

[12] Hatonen, K., Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H. (1996), Knowledge Discovery from Telecommunication Network Alarm Databases, *Twelfth International Conference on Data Engineering* (ICDE'96), New Orleans, Louisiana.

[13] Mannila, H., Toivonen, H., Verkamo, A. (1995), Discovering Frequent Episodes in Sequences, *First International Conference on Knowledge Discovery and Data Mining (KDD '95)*, pp. 210-215, Montreal, Canada, AAAI Press.

[14] Sasisekharan, R., Seshadri, V., Weiss, S. (1996), Data Mining and Forecasting in Large-Scale Telecommunication Networks, *IEEE Expert*, Vol. 11, No. 1, Feb., pp. 37-43.

[15] Cooper, T., Wogrin, N. (1988), *Rule-Based Programming with OPS5*, Morgan Kaufmann, San Mateo, CA.

[16] Ezawa, K., Norton, S. (1996), Constructing Bayesian Networks to Predict Uncollectible Telecommunication Accounts, *IEEE Expert*, Vol. 11, No. 5, Oct., pp. 45-50.

[17] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. (1991), *Object-Modeling and Design*, Prentice Hall.

[18] R++ home page: http://www.research.att.com/sw/tools/r++.

[19] Singhal, A., Weiss, G. M., Ros, J. (1996), A Model Based Reasoning Approach to Network Monitoring, *Proceedings of the ACM Workshop on Databases for Active and Real Time Systems* (DART '96), Rockville, Maryland, pp. 41-44.

[20] Mishra, A., Ros, J., Singhal, A., Weiss, G., Litman, D., Patel-Schneider, P., Dvorak, D., Crawford, J. (1996), R++: Using Rules in Object-Oriented Designs, *Addendum Object-Oriented Programming Systems, Languages, and Applications* (OOPSLA).