AN EVALUATION OF CASE BASED REASONING FOR FAULT DIAGNOSIS

by

Robert Cookson

TR97-113, April 1997

This is an unaltered version of the author's
MCS Thesis

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
www: http://www.cs.unb.ca

AN EVALUATION OF CASE-BASED REASONING FOR FAULT DIAGNOSIS

by

Robert L. Cookson

BSc.(C.S.), University of New Brunswick, 1989

A Thesis Submitted in Partial Fulfilment of
the Requirements for the Degree of

Master of Science in Computer Science

in the Graduate Academic Unit of Computer Science

Supervisor:	Bradford G. Nickerson, Ph.D., Computer Science
Examining Board:	John M. DeDourek, MS, Computer Science, Chair Colin Ware, Ph.D., Computer Science
External Examiner:	David J. Bonham, Ph.D., Mechanical Engineering

This thesis is accepted.

.....

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

April, 1997

ABSTRACT

Recalling past cases and their associated diagnoses to apply to a current fault situation is often an important step in the diagnostic process. In this thesis, a model for fault diagnosis in telephone networks has been developed, along with a prototype that implements the model using case-based reasoning. The prototype, called TETAA (TElephone Trouble Analyst's Assistant), aids in analysis and diagnosis of telecommunications faults by retrieving and presenting past fault cases and their diagnostic information. The diagnostic categorization model serves as an analytical tool to aid in fault reduction. The TETAA prototype was evaluated over a two month period at a New Brunswick Telephone Company (NBTel) repair center to determine its performance and potential in diagnosing and reducing network faults. The prototype demonstrated that case-based reasoning is applicable to fault diagnosis for a large telecommunications network and demonstrates potential for aiding in fault reduction.

ACKNOWLEDGMENTS

The author would like to thank his supervisor Dr. B.G. Nickerson for his guidance throughout the course of this work. His insight, commitment, and patience have made the completion of this research possible. The author would also like to thank Dr. Nickerson for the use of the Artificial Intelligence lab facilities, which have made much of this research reality.

Special acknowledgment is extended to NBTel personnel, Sean McHatten, Robert Costello, Ernest Jones and Ben Leblanc. Their input and assistance was instrumental in the development and evaluation of TETAA.

Special thanks to Kirby Ward, whose technical assistance and advice was very helpful throughout the development phase of this research.

The author would also like to thank the Natural Sciences and Engineering Research Council of Canada for their funding of this work.

The author extends special acknowledgment to his parents for their prayer and support throughout his university years. Their example, beliefs, and values have taught him more than any of his academic undertakings.

To the author's wife, who shared in the burden of completing this thesis, thank you for your support, and for understanding.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Overview of Case-Based Reasoning

1.1.1 Knowledge-Based Systems

Artificial intelligence can be defined as the branch of computer science that is concerned with the automation of intelligent behavior [Luger and Stubblefield, 1989]. The problem with this statement is that it implies that computer scientists currently understand and can define the process of human intelligence well enough to automate it. Instead, research in artificial intelligence serves as a tool for exploring and evaluating theories about the mysteries of intelligence. Knowledge-based systems comprise one of the more prominent areas of this research.

Knowledge-based systems, or expert systems, focus on knowledge in a specified problem domain that is represented symbolically, and perform very specialized and difficult tasks at expert levels of performance. In recent years, these types of systems have become one of the hottest topics in computer science and artificial intelligence, and have been implemented in a broad range of problem domains from medical diagnosis to mineral exploration, to telecommunications [Leibowitz, 1988].

An integral component of any knowledge based system is the domain specific knowledge that it utilizes for problem solving, or, in other words, the knowledge-base itself. The knowledge base can appear in two different forms, with the first and most common being a set of domain specific rules that emulate the knowledge of an expert in an area, allowing the knowledge-based system to solve problems in that domain with competence often comparable to an expert. The second knowledge base representation, and the focus of this work, is a set of domain specific examples with known solutions, called a case-base, with an associated reasoning process called case-based reasoning.

1.1.2 Rule-Based Reasoning

The traditional approach to implementing a knowledge-based system is through the use of rule-based reasoning (RBR). The premise behind rule-based reasoning is that the knowledge of an expert can be represented by a set of rules. This idea was first encountered in Newell and Simon's pioneering work on the General Problem Solver (GPS) [Newell and Simon, 1972], one of the first artificially intelligent programs, and was implemented at Stanford University in the DENDRAL project, a mass spectrometer analyzer, and one of the first rule-based expert systems [Riesbeck and Schank, 1989].

The rule-based reasoning model is made up of three basic components: available information about the current state of the problem domain (i.e. fact base), the rule-base, which represents domain specific, expert level operational knowledge, and the inference engine or rule interpreter that combines the current state of the domain and the actions inferred by the rule-base to traverse the reasoning chain to reach decisions and conclusions.

The rules of most rule-based systems can be represented by production rules of the following form:

IF <some conditions are met>

THEN <some actions are taken>

The "IF" clause of the rule, referred to as the left hand side (LHS), specifies certain conditions that, when met, will cause the rule to "fire", triggering the actions specified in the "THEN" clause or right hand side (RHS).

The use of rules to represent knowledge is undoubtedly the most popular approach to knowledge representation today. Rules are a natural and intuitive mode of representing many types of domain knowledge [e.g. Ignizio, 1991]. As well, the additivity property [Riesbeck and Schank, 1989], or the characteristic of adding or modifying knowledge through the addition or modification of rules in the knowledge base, allows relatively easy maintenance of rule-based expert systems.

There are two approaches that can be used to implement rules-based reasoning, each of which perform better in certain problem domains. These approaches are forward chaining and backward chaining.

1.1.2.1 Forward Chaining

Forward chaining is often referred to as data driven reasoning. The reasoner constructs a chain of inferences from an initial set of facts (i.e. data) to a final conclusion. This type of chaining is most effective in interpretation or data analysis problem domains which give all or most of the data in the initial problem statement [Luger and Stubblefield, 1989].

1.1.2.2 Backward Chaining

Backward chaining is often referred to as goal driven reasoning. In this case the reasoner, starts with a hypothesis (or goal) that can solve the problem, finds the rules that can produce the hypothesis, and chains backward through successive rules and subgoals to the given facts of the problem. This type of chaining is most effective in problem domains where a goal or hypothesis is given in the problem [Luger and Stubblefield, 1989]. The domain of diagnosis, which is examined in this thesis, is best implemented using backward chaining by considering potential diagnoses in a systematic fashion.

1.1.2.3 Combining Forward and Backward Chaining

A forward chainer can conduct backward chaining by forward chaining from goals. The goal is asserted to describe a fact that is needed to match the LHS of the rule (i.e. a fact about a fact). The goal is generally the first clause in the LHS of a backward chaining rule. A match between a goal and the LHS of a rule causes the backward chaining rule to be fired, generally asserting a fact that satisfies the goal fact. This new fact will instantiate the original goal. This process is referred to as the "Message and Reply Model" [Inference Corporation, 1991].

1.1.3 Case-Based Reasoning

1.1.3.1 What is Case-Based Reasoning?

Case-based reasoning is the technique of comparing a current case to a library of similar experiences with known solutions. It is particularly useful in tasks where a formal set of domain-specific rules for generating solutions are difficult to obtain, but examples of correct solutions are readily available [Riesbeck and Schank, 1989].

At its simplest, case-based reasoning is based on the premise that when we solve a problem we often base our solution on one that worked for a similar problem in the past. An example would be driving to work. When you get in the car in the morning you don't intentionally plan your route, you take the route you usually take. If you meet a traffic jam you may remember how you avoided a similar jam in the past. If you take an alternate route to avoid a jam and it solves the problem, you will remember it and use it again in similar circumstances in the future.

Case-based reasoning is a relatively simple problem solving process that involves matching your current problem against problems that you have solved successfully in the past. The process can be broadened by adapting solutions so they more closely match the current problem.

1.1.3.2 The Development of Case-Based Reasoning

The roots of case-based reasoning in artificial intelligence is found in the work of Roger Schank on dynamic memory and the central role that a reminding of earlier situations or cases, and situation patterns (scripts, MOPs) has in problem solving and learning [Schank, 1982].

The first system that might be called a case-based reasoner was the CYRUS system, developed by Janet Kolodner [Kolodner, 1983], at Yale University in a group headed by Schank. CYRUS was based on Schank's dynamic memory model and MOP theory of problem solving and learning [Schank, 1982]. It was basically a question-answering system with knowledge of the various travels and meetings of former US

Secretary of State Cyrus Vance. The case memory model developed for this system has later served as the basis for several other case-based reasoning systems (including MEDIATOR [Simpson, 1985], PERSUADER [Sycara, 1988], CHEF [Hammond, 1989], JULIA [Hinrichs, 1992], and CASEY [Koton, 1989]).

Another basis for case-based reasoning, was developed by Bruce Porter and his group [Porter, 1986] at the University of Texas. Their research on the problem of concept learning for classification tasks eventually lead to the development of the PROTOS system [Bareiss, 1989], whose emphasis was on integrating domain knowledge and specific case knowledge into a single knowledge base. The combination of cases with domain knowledge was pushed further in GREBE [Branting, 1991], an application in the domain of law. Another early significant contribution to case-based reasoning was the work by Edwina Rissland and her group at the University of Massachusetts, Amhearst. With several law scientists in the group, they were interested in the role of precedence reasoning in legal judgments [Rissland, 1983]. In this system, cases were not used to produce a single answer, but to interpret a situation in court, and to produce and assess arguments for both parties. This resulted in the HYPO system [Ashley, 1990], and later the combined case-based and rule-based system CABARET [Skalak, 1992]. Phyllis Koton at MIT studied the use of case-based reasoning to optimize performance in an existing knowledge based system, where the domain (heart failure) was described by a causal model. This resulted in the CASEY system [Koton, 1989], in which case-based reasoning was combined with deep model-based reasoning which describes the actual behavior of the modelled system.

In Europe, research on case-based reasoning began to develop a little later than in North America. One of the earliest results was the work on case-based reasoning for complex technical diagnosis within the MOLTKE system, done by Michael Richter together with Klaus Dieter Althoff and others at the University of Kaiserslautern [Althoff, 1989]. This lead to the PATDEX system [Richter, 1991], with Stefan Wess as the main

developer. At IIIA in Blanes, Enric Plaza and Ramon Lopez de Mantaras developed a case-based learning apprentice system for medical diagnosis [Plaza, 1990], and Beatrice Lopez researched the use of case-based methods for strategy-level reasoning [Lopez, 1990]. In Aberdeen, Derek Sleeman's group studied the use of cases for knowledge base refinement. An early result was the REFINER system, developed by Sunil Sharma [Sharma, 1988]. Another result is the IULIAN system for theory revision [Oehlmann-92]. At the University of Trondheim, Agnar Aamodt and colleagues at Sintef studied the learning aspect of case-based reasoning in knowledge acquisition. Research on combining domain knowledge and cases lead to the development of the CREEK system and integration framework [Aamodt, 1991], and to continued work on knowledge-intensive case-based reasoning. In the area of cognitive science, early work was done on analogical reasoning by Mark Keane, at Trinity College, Dublin, [Keane, 1988]. Keane used analogical reasoning as a form of case-based reasoning to derive implications, or form analogies, from cases with missing data and incomplete specification of parameters. In Gerhard Strube's group at the University of Freiburg, the role of knowledge representing events over a period of time in the form of episodes for knowledge aquisition or cognitive models was investigated in the EVENTS project [Strube, 1990], which lead to the group's current research profile of cognitive science and case-based reasoning.

Currently, research on case-based reasoning research is becoming more prominent throughout the world. In an internet query performed in the process of researching this subject, over one hundred recently published papers on case-based reasoning or related topics were found. As well, several highly regarded conferences that focus on case-based reasoning in both national and international forums have been established. For example the American Association for Artificial Intelligence(AAAI) has sponsored a series of case-based reasoning workshops at its annual conference, and a new international

conference, The International Conference on Case-Based Reasoning, held for the first time in 1995, has been organized.

Case-based reasoning has also begun to achieve recognition in commercial fields. Research has spawned several commercial implementations of case-based reasoning from different companies and numerous software firms are developing and shipping commercial case-based reasoning products worldwide. In the customer support and service market, all of the top eight suppliers offer case-based reasoning problem identification and resolution technology along with their problem management and call tracking products.

In total, case-based reasoning has now been licensed and is in use by hundreds of corporations and hundred of thousands of users worldwide. It has been implemented in over a dozen different languages, proving its adaptability, and has been successfully implemented in a host of commercial applications [Tierney, 1995].

1.1.3.3 Why the Interest?

As has been previously described (see section 1.1.3.2), much interest in the field of case-based reasoning exists, and continues to grow. The reason for this expanding research is the potential that case-based reasoning shows in helping to understand human intelligence and how human experts reason which is the main objective of research in artificial intelligence.

Case based-reasoning has been proposed by its promoters to be a more psychologically sound model of the reasoning of an expert than the more common rule-based reasoning that is the basis of most knowledge-based systems in existence today[Riesbeck and Schank, 1989]. The purpose of a knowledge-based system is to effectively model the knowledge of an expert in a given area. Therefore, when determining the effectiveness of the two reasoning approaches, a question that must be asked is whether that expert reasons using rules or by referencing previous experiences and situations. Most experts can supply to knowledge engineers some rules that they use

to solve problems, but they will also tell you that it is their actual experience that makes them experts. If we believe that experts reason using rules, then we must also believe that their experience is encoded in their memories as a set of rules. Since experts will also tell you that it is not unusual for a rule that they follow to have a number of exceptions, and in these situations frequently fall back on the experience of previous cases that the current case reminds them of. The implication of this process is that reminding plays an important part in expert reasoning [Riesbeck and Schank, 1989].

Case-based reasoning is at the base of how human reasoning works. People reason from experience. They use their own experiences if they have a relevant one, or they make use of the experience of others to the extent that they can obtain information about such experiences. An individual's knowledge is the collection of experiences that he has had or that he has heard about [Riesbeck and Schank, 1989]. This discovery is the reason for the increase in research and commercial use of case-based reasoning, and also the reason for the success achieved.

1.1.3.4 The Case-Based Reasoning Cycle

The basic cycle of a case-based reasoner is to input a problem, find a relevant old solution, and adapt it. Figure 1.1 explains the general case-based reasoning model put forward by Riesbeck and Schank [Riesbeck and Schank, 1989] in more detail.

The first step in the CBR process is to determine which situations that are present in the case base are similar to the presented case. To accomplish this, the features of the relevant case base cases must be organized and labeled so that the features of the presented case can be used to find them. Determining the relevance of a case is usually not just a function of matching presented features with case base features, but also involves using relationships between features, absence of features, and so on. Therefore, the case-based reasoner must determine what features, or indices of a presented case are relevant for finding similar cases. The "indexing problem", as it is sometimes called, is

the problem of determining what extra, non-obvious, non-input features are needed for a particular domain.

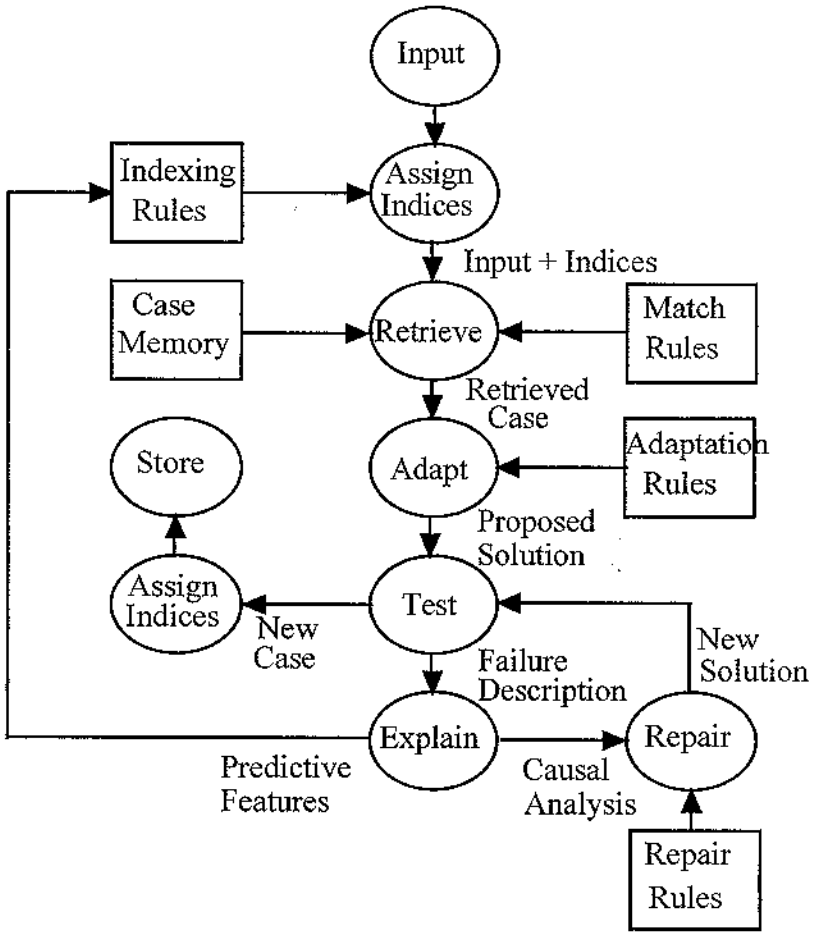


Figure 1.1 Case-Based Reasoning Cycle [from Riesbeck and Schank, 1989].

Once the indices have been finalized, the next step in the CBR process is to determine a best match for the presented case. In doing this the case-based reasoner must retrieve a set of cases from the case base which most closely resemble the presented case, reject the cases that differ too much from the presented case, and determine which of the remaining cases most resemble the presented case. In determining the similarity of cases there are two aspects that are important: how well the cases match on each feature or index, and how important the features or indexes are.

The adaptation step of the CBR cycle adjusts the best match case to fit the presented situation. This involves determining what is different about the retrieved and presented case, and modifying the solution of the retrieved case so that it is also applicable to the presented case. The adaptation process applies usually complex rules designed to bridge the differences between the retrieved case and the presented case. The degree of adaptation depends on the differences between the retrieved case and the presented case. Sometimes little or no adaptation will be needed, while at other times there will be such a high degree of mismatch that it will be of no use to the user. In this case, the best approach is to ask a human expert for the solution, and save this as a new case in the case base.

1.1.4 Rule Versus Case-Based Reasoning

When comparing rule-based reasoning and case-based reasoning, a useful starting point is to consider the analogy given by Riesbeck and Schank [1989]. Many of the differences between case-based reasoning and rule-based reasoning are similar to the human problem solving of an individual with experience in resolving problems in a specific situation, and an individual with textbook knowledge of how to solve the same problem. The experienced individual will apply case-based reasoning, while the individual with the textbook knowledge will apply the rules that he has learned using rule-based reasoning.

In this situation a number of tradeoffs will occur. The individual utilizing case-based reasoning will, based on experience, be able to give an approximate answer almost immediately. The individual applying rule-based reasoning will reach an exact solution after taking a much longer period of time. The case-based reasoner will only be able to provide a solution for situations exactly like or similar to situations that he has experienced in the past, but it will be efficient, and the solution will be known to have worked previously. The rule-based reasoner will be flexible and provide an exact answer but will not necessarily be efficient and will be prone to error. These characteristics

favour the rule-based reasoner in simple problem domains, but as the problem domain becomes more complex, so does the rule-base and the amount of knowledge engineering required. This situation characterizes one of the major problems traditionally experienced with rule-based reasoning called the knowledge engineering bottleneck [Hayes-Roth et al, 1983].

Case-based knowledge engineering requires the somewhat easier task of acquiring past cases with known solutions and identifying the relevant case features. This characteristic has a number of advantages including allowing case-based reasoning systems to be implemented faster than rule-based systems, as well as allowing case-based systems to be implemented with a partial case-base. Indeed the case-base is never complete; as new cases become available, the case-base will continue to grow. This characteristic of case-based knowledge engineering eliminates the problem (encountered with rules) of knowing when a knowledge base is complete. By the same token, case-based reasoning allows for implementation in problem domains where no rule based model is available, but past cases and solutions are readily available

In a complex problem domain utilizing a complicated rule-base, long and involved reasoning chain can be encountered, as the knowledge is broken into a number of separate pieces (i.e. rules). This characteristic adds to the inefficiency that rule-based implementations can introduce. It seems logical that knowledge would be better represented if related items were "near" each other [Smith et al, 1978] which is the case with case-based reasoning.

Maintenance of a rule-base is generally considered to be a straight forward process. The addition or modification of a piece of knowledge can be easily translated to the addition or modification of rules in the knowledge-base. This assumption is true of simple rule-bases, but as the rule-base grows in size and complexity, maintenance can become a complex debugging task. Case-based knowledge representation requires little maintenance; in fact, a case-base can maintain itself. When a search fails to locate a

similar case, the presented case itself becomes the basis for a new case. In effect, the case-based reasoner learns from experience and can keep pace with a changing environment.

Finally, a major disadvantage of rules is that they are not a good way to represent experiences [Riesbeck and Schank, 1989]. The knowledge gained from an experience is representable in rule form, but characteristics of an experience, such as a sequence of events, cannot be easily represented. Case-based reasoning solves this problem by using events to describe the knowledge represented by the rule base, as well as effectively representing an experience.

The following case-study supports the claim that case-based reasoning systems can be implemented faster than model-based systems. A study conducted by Cognitive Systems stated that it took two weeks to develop a case-based version of a system that took four months to build in rule-based form [Goodman, 1989]. More recently, developers at Digital Equipment Corporation confirmed that a rule-based system called CANASTA took more than eight times longer to develop than CASCADE, a case-based system with the same functionality [Simoudis et al., 1993]. They also claim that the maintenance of CANASTA is continual whereas CASCADE needs almost no maintenance. Related claims are provided by Hennessy and Hinkle [1992] concerning CLAVIER. Instances such as these may not be indicative of all comparable rule and case-based systems. We should also be aware that for well understood domains, rule-based systems can be very effective and are a relatively mature and well understood technology.

1.2 Overview of Fault Diagnosis in Knowledge-Based Systems

Diagnosis has been one of the major subjects of research in the Artificial Intelligence and Knowledge-Based community since the early 1970's. Defining diagnosis in the operational sense: "Diagnosis is the process of fault finding in a system (or

determination of disease state in a living system) based on the interpretation of potentially noisy data" [Stefik et al., 1982]. The fault diagnosis problems has been addressed in many different problem domains, from troubleshooting in electric appliances and circuits, to diagnosis of complex mechanical or physical systems, to medical diagnosis [Torasso and Console, 1989].

Many diagnostic expert systems have been developed since the early 1970's and many problem solving approaches and inference processes have been implemented. To date, the most widely adopted approach to fault diagnosis has been to use production rules for knowledge representation, form hypotheses about possible fault causes, and evaluate these hypotheses as goals in a backward chaining system to reach a correct diagnosis (see section 1.1.2). Starting with MYCIN [Buchanan and Shortliffe, 1984], many other systems have been built based on this approach. This approach demonstrated some success, but problems with knowledge engineering and maintenance gave rise to other approaches such as causal reasoning [Torasso et al, 1989], and model-based reasoning [Hamscher et al, 1992]. Another approach to improve diagnostic performance is the combination of various types of reasoning, such as the combining heuristic (rule-based) reasoning and causal reasoning as in the CHECK system [Console et al, 1993].

As with many other fields of Artificial Intelligence research, case-based reasoning has recently emerged as one of the leading approaches to diagnostic problem solving. Some of the earliest case-based diagnostic systems were in the medical field, specifically the area of heart failure [Koton, 1989; Aghassi, 1990]. Since then, diagnostic research in the area of case-based reasoning has continued to evolve with the combination of case-based reasoning and other reasoning approaches, such as model-based reasoning [Pews and Wess, 1993; Torasso and Portinale, 1995]. This type of research has provided some of the best results, with accurate diagnosis, but also showing major efficiency gains over other reasoning approaches.

1.3 The Problem

With the rapid advancement of telecommunications in recent years, a need for better understanding the new technologies has developed. The emergence of knowledge-based systems as an important research topic in computer science has provided a way to effectively capture (and put to widespread use) knowledge about telecommunication operations [Leibowitz, 1988]. Several successful examples include: the Automated Cable Expertise system (ACE), which assists telephone company engineers in maintaining the local loop, a switch maintenance analysis and repair tool (SMART) for diagnosis of analog switch problems, and an expert system for screening and diagnosing telephone troubles (MAX) [Rabinowitz et al, 1991].

The investigation of the opportunities for knowledge-based systems in the telecommunications industry was undertaken in conjunction with the New Brunswick Telephone Company Limited (NBTel). NBTel is the major supplier of telecommunication services in the province of New Brunswick. Their primary goal is to provide communications solutions efficiently and effectively to the expressed satisfaction of their customers. This thesis is a result of NBTel's interest in on-going research in knowledge-based system applications in telecommunications, and how these applications can benefit their business.

Maintenance of the telecommunication network is a significant problem for any telephone company. The task of diagnosing and repairing customer-reported faults has been made more difficult in recent years by technological advances in network components, new kinds of customer premise equipment, and nonstandard equipment that was not anticipated by most current diagnostic systems. With these technological advancements, it has become more and more difficult for any individual, or group of individuals, to be experts in all of the areas that troubles are typically encountered, thus making the diagnosis process much more difficult.

Work has been done in the area of using rule-based reasoning tools for telephone fault diagnosis which shows that this technology is beneficial in increasing the effectiveness of the network maintenance process, although difficulties in knowledge engineering still remain [Rabinowitz et al, 1991].

It is hypothesized that case-based reasoning may provide a better solution in this problem domain than the typical rule-based approach. The investigation of the feasibility of using case-based reasoning for fault diagnosis, and specifically telecommunications fault diagnosis, is the objective of this thesis.

2.0 ADAPTING CASE-BASED REASONING TO FAULT DIAGNOSIS

2.1 A Diagnostic Model

In the course of this research, while data gathering, knowledge engineering and conducting interviews with NBTel personnel, it was discovered that there is much interest in, and a large amount of work being done, in the area of trouble report reduction, and trouble prevention. This interest is not specific to NBTel alone, but to any telecommunications or other product delivery company that is interested in remaining competitive. The reason for this is obvious, in that it will allow the company to provide better service, and reduce maintenance costs, and in doing so benefit both the customer and the company.

The inspection and analysis of past trouble reports is one starting point for accomplishing the goal of trouble prevention, and effective trouble analysis requires tools that provide the analyst with pertinent information in a useful format. It was also discovered through interviews with the NBTel managers who are responsible for trouble report rate reduction that an analysis tool for the categorization of troubles and their diagnoses would be very useful. This section presents a model that accomplishes this categorization. Figure 2.1 shows a condensed overview of the model structure, while the entire model is presented in detail in Appendix I.

2.1.1 Model Structure

Following analysis of trouble report records, and the information contained therein, it was evident that a model for categorizing troubles and their diagnoses should be structured hierarchically, as illustrated in Figure 2.1, with each descending level of the resulting tree representing a further step in the trouble classification and diagnosis process. Each level of the hierarchy represents a specific piece of diagnostic information, whose significance is explained below.

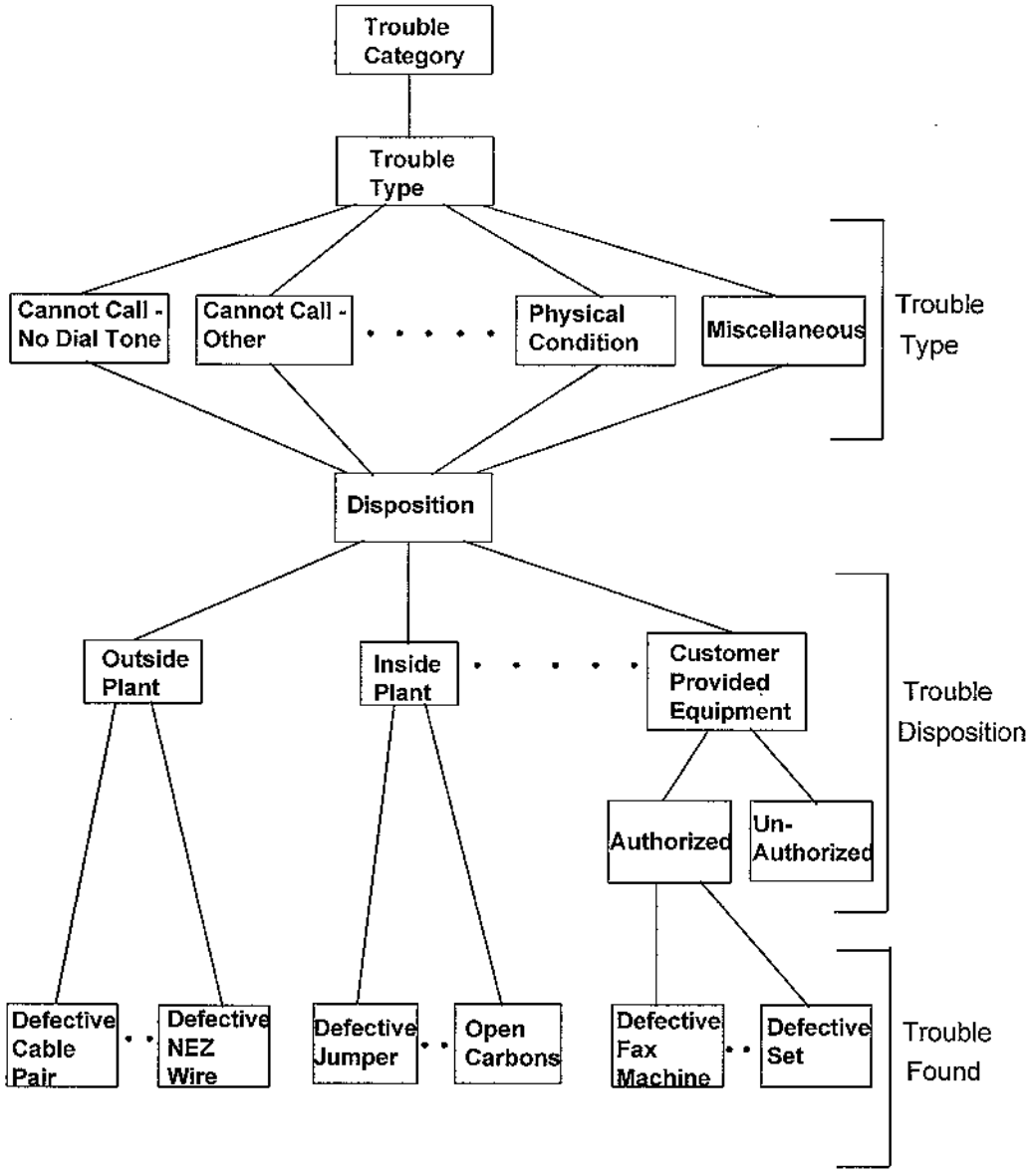


Figure 2.1 Overview of Diagnostic Model

2.1.1.1 Trouble Type

The top level of the model hierarchy is represented by the trouble type, which is the starting point for the trouble diagnosis. The trouble type is indicative of the problem reported by the customer, and in diagnostic terms, represents the symptom of the fault.

The trouble types are broken down into eight broad classifications; all possible customer reported trouble symptoms will fit into one of these classifications. These classifications, whose names are self explanatory, are listed in Table 2.1.

Table 2.1 Trouble Types

Trouble Types (Symptoms)
1. Cannot Call - No Dial Tone
2. Cannot Call - Other
3. Transmission Noise
4. Cannot Be Called
5. Memory Service Failure
6. Data Failure
7. Physical Condition
8. Miscellaneous

2.1.1.2 Disposition

All levels of the model hierarchy that are between the top and leaf levels are represented by the trouble disposition. Generally, the disposition corresponds to only one level of the hierarchy, but in some cases sub-dispositions are identified. In these cases the disposition represents at most two levels of the model.

The main purpose of the disposition classification is to indicate, either directly or indirectly, the physical location of the fault on the telecommunications network. This is the logical next step in the fault diagnosis process after identifying the symptoms. The disposition classifications also allow for special cases where the fault that was reported cannot be found, or the trouble must be referred to an outside party. The trouble disposition classifications are listed in table 2.2.

Table 2.2 Trouble Dispositions

Trouble Dispositions
1. Outside Plant
2. Inside Plant
3. Customer Premises
3.1 Station Set
3.2 Station Wiring
3.3 Other Station Equipment
4. No Problem Found
4.1 Test OK
4.2 Found OK In
4.3 Found OK Out
5. Referred Out
6. Subscriber Carrier
7. Customer Action
8. Customer Provided Equipment
8.1 Authorized
8.2 Unauthorized

2.1.1.3 Trouble Found

The final step in the diagnosis of a fault is the determination of the specific problem. This data is represented at the leaf level of the model hierarchy by trouble found. This piece of information describes what was found to be causing the problem, and what was repaired by the installation and repair craftsperson.

The list of trouble found classifications is unique depending on the disposition that it appears under. The reason for this is that only certain faults are encountered in different regions of the network, and the types of faults that correspond to these regions are reflected in the trouble found classifications under each disposition. The trouble found classifications for each disposition are listed in Tables 2.3 through 2.9.

Table 2.3 Trouble Found Classifications - Outside Plant

Trouble Found Classifications- Outside Plant
1 Defective Cable Pair
2. Open Cable Pair
3. Incorrect Cable Pair Assignment
4. Defective Cable
5. Open Jumper in Jumper Wired Interface
6. Defective Binding Post Lead
7. Binding Post Lead Open
8. Defective NEZ Wire

Table 2.4 Trouble Found Classifications - Inside Plant

Trouble Found Classifications - Inside Plant
1. Defective Jumper
2. Open Jumper
3. Defective LLE
4. Defective SLR
5. Switch Hardware Problem
6. Switch Software Problem
7. Incorrect Cable Pair Assignment
8. Defective Carbons
9. Open Carbons

Table 2.5 Trouble Found Classifications - Customer Premises

Trouble Found Classification - Customer Premises		
Station Set	**Station Wiring**	**Other Station Equipment**
1. Defective Receiver Cord	1. Defective Jack	1. Defective SX200 Equipment
2. Defective Hand Set	2. Corroded Wiring	2. Defective Norstar Equipment
3. Defective Dial / Touch-tone Pad	3. Defective Outside Wiring	3. Defective SP1 Equipment
4. Defective Set	4. Ground In Wiring	4. Defective V12 Equipment
5. Defective Base Cord	5. Defective Connecting Block	5. Defective Tye Equipment
6. Defective Receiver Unit	6. Defective Inside Wiring	6. Defective Button / Buzzer
7. Defective Transmitter Unit	7. Defective Protector	7. Defective Transformer
	8. Defective Carbons	8. Defective Extension Ringer
	9. Defective Protector	
	10. Ground Not Connected	
	11. Defective Ground Clamp	

Table 2.6 Trouble Found Classifications - Referred Out

Trouble Found Classifications - Referred Out
1. Assigning
2. Central Office
3. Technical Assistance Centre
4. Business Office
5. Engineering
6. Data
7. Construction

Table 2.7 Trouble Found Classifications - Subscriber Carrier

Trouble Found Classifications - Subscriber Carrier
1. Defective Subscriber Carrier

Table 2.8 Trouble Found Classifications - Customer Action

Trouble Found Classifications- Customer Action
1. Receiver Off Hook
2. Calling Party Hold
3. Damage By Customer

Table 2.9 Trouble Found Classifications - Customer Provided Equipment

Trouble Found Classifications - Customer Provided Equipment	
Authorized	**Unauthorized**
1. Defective Fax Machine	1. Defective Unauthorized Customer Provided Equipment
2. Defective Modem	
3. Defective Set	

2.1.2 Trouble Categorization

For the diagnostic model to be useful as an analysis tool it is important that the troubles with the same diagnosis be classified into the same distinct category. This can be accomplished by treating a complete path from the top, down to the leaf level of the hierarchy as a trouble category. Therefore there will be as many trouble categories represented by the model as there are paths in the tree.

A category can be easily represented by a list of integers that individually represent the branch chosen at a given level of the model. To do this, numbers must be assigned to the nodes of the tree in a way such that all the children of a specific node have a unique number. These number designations are shown in Tables 2.1 through 2.9.

As an example, let us consider a trouble report that has been determined to have a trouble type of cannot call - no dial tone, a trouble disposition of inside plant, and the trouble found being an open jumper. From Table 2.1, Table 2.2, and Table 2.4 we see that the corresponding numbers are 1, 2, and 2 respectively. Therefore, the trouble

category of cannot call - no dial tone, inside plant, open jumper can be represented by the string "1.2.2".

The current model structure contains 8 trouble type classifications, 13 trouble disposition classifications, and 58 trouble found classifications. This structure allows for representation of 488 unique fault categories.

2.1.3 Disposition Classifications

The disposition classification terms, listed in Table 2.2, describe the physical locations, and special cases of the types of troubles that occur on a telecommunications network. Some of the terms used are not self explanatory, and are described below.

2.1.3.1 Inside Plant

The term inside plant refers to any components of the network that are contained within a building. The building referred to is generally the Central Office, which houses the local switching equipment, and are dispersed as needed throughout the network. Inside plant generally refers to the switching equipment, which is responsible for directing calls to their proper destinations, and the frame, which connects the switch to the outside world. The trouble found categories for this disposition are listed in Table 2.4.

2.1.3.2 Outside Plant

The outside plant is the part of the network that connects the Central Office (Inside Plant) and the customer's premises. This includes feeder cables, which are large cables with many pairs that feed directly from the switch, distribution cables, which have a smaller number of pairs that are distributed into the populated areas, and the jumper wired interfaces, which connect the feeder cables to the distribution cables. In general, the relationship between feeder cables and distribution cables is one-to-many.

2.1.3.3 Customer Premises

The disposition classification of customer premises is fairly self explanatory in that it represents any trouble whose cause is at the customers home or place of business. This category can be broken down into three sub-dispositions:

Station set refers to a telephone set at the customers premises that may be causing a problem.

Station wiring refers to any of the telecommunication related wiring at the customers premises in which there may be a fault.

Other station equipment refers to any other telecommunication equipment, such as signaling equipment, cabinets or booths, that are the cause of the trouble.

2.1.3.4 No Problem Found

No problem found is a special case of disposition classification that describes a trouble that was reported by the customer but cannot be detected by a line test or re-created by the technician. This classification can be broken down into three sub-dispositions:

Test OK applies to a trouble that was reported by the customer, but when a line test does not indicate trouble, and no other trouble-causing condition can be determined.

"Found OK in" describes a trouble that was thought to be in the central office, but when investigated by a technician, no problem was found, a line test did not indicate a problem, and no other trouble-causing condition can be determined.

"Found OK out" refers to a trouble that was thought to be outside (i.e. outside plant), but when investigated by a technician, no problem was found, a line test did not indicate a problem, and no other trouble-causing condition can be determined.

2.1.3.5 Referred Out

Trouble reports referred to other technician groups such as toll offices, crossbar tandem offices, or Plant Service Centres, are classified as referred out. Trouble reports with this disposition are not closed until the referred to party has corrected the problem.

2.1.3.6 Subscriber Carrier

Trouble reports caused by switching equipment failures are referred to as Subscriber Carrier. The failure of line cards which terminate the feeder pair at the central office are also included in this disposition.

2.1.3.7 Customer Action

Customer action is again a self explanatory term that describes any trouble that has occurred because of the action that a customer has taken. This classification usually refers to problems whose cause is at the customers premises.

2.1.3.8 Customer Provided Equipment

Customer provided equipment refers to troubles that are a result of equipment that was not provided by the telephone company, but was bought by the customer from a third party. This classification is made up of two sub-dispositions:

Authorized, which describes equipment that the telephone company does not provide, but is under contract with the third party to maintain.

Unauthorized, which refers to equipment that that the telephone company does not provide, and does not maintain. When a trouble of this disposition occurs, the customer is responsible for any repairs.

2.2 Diagnosis Using Matching

The diagnostic task is to determine why a correctly designed system is not functioning as it was intended, and to explain the faulty behavior by specifying what is at variance with the design [De Kleer et al., 1992]. The concept of case-based reasoning is based on the ability of the reasoner to find the best match for the presented case. This implies, that the effectiveness of fault diagnosis using case-based reasoning in resolving and explaining the fault in the system is dependent upon determining which of the features of the fault cases are the most important in determining the faulty behavior, and the ability of the diagnostic system to match on these features and retrieve the case with

the correct diagnosis. In a diagnostic system, these features can be characterized as symptoms of the fault, and will vary according to the fault diagnosis problem domain. To be able to retrieve the stored case from memory that will provide the most applicable solution, the case-based reasoner must be able to match on the features or symptoms of the fault that are most relevant to the diagnostic problem domain. We will refer to these symptoms as the diagnostic case features.

To determine the best match, and therefore the most applicable diagnosis of the fault, the case-base must contain three kinds of knowledge [Hammond, 1989].

2.2.1 Case Library

The case library provides descriptions of unique fault situations and their correct diagnosis for the diagnostic domain being examined. These cases include the diagnostic case features that have been determined to be most relevant in determining the cause of the fault. These features are indexed to ensure that matching is efficient, and that the appropriate cases are retrieved. The diagnostic case features provide the foundation for the diagnostic matching process and are the basis upon which the other two types of knowledge in the case-base are brought to bear.

2.2.2 Partial Matching

The case-base must include a way to determine and evaluate the degree of partial matching on diagnostic case features. When a diagnostic system cannot find a case that exactly matches the symptoms of the presented case, it must have a means of finding a case which partially satisfies the symptoms. A case based diagnostic system also needs some idea of which symptom descriptions are similar to each other. It can then use the partial satisfaction of presented case symptoms, along with the similarity measurement to retrieve the most suitable fault diagnosis.

2.2.3 Value Hierarchy

The value hierarchy determines the relative importance or weight of the diagnostic case features in the matching process. The case-based reasoner integrates this knowledge

with the degree of matching on the identified fault symptoms to determine the value or score of a stored case with respect to the presented fault symptoms, and retrieves the case that will most likely provide the correct diagnosis.

2.3 Explanation

The explanation process is another integral aspect to the study of human intelligence, and plays an important role in diagnosis and case-based reasoning. An important component of the diagnostic process is, upon determination of the cause of the faulty behavior in the system, to also explain what has caused the problem [Torasso and Console, 1989]. Explanation has a number of applications when applying case-based reasoning to diagnosis.

2.3.1 Failure Explanation

Failure of a case-based reasoning system to find an existing case that matches the presented situation brings about the need for the explanation and repair of the failure (see Figure 1.1). These two steps are interrelated in that one is based on the other, depending on problem domain in which the failure has occurred. In a planning system the explanation is needed to correctly complete the repair of the plan, but in a diagnostic domain, the correct diagnosis, and subsequently the repair, is needed before a failure can be explained; therefore the explanation is dependent on the repair.

Currently the appropriate reasoning technique for implementing the failure explanation process is a matter of research and debate. The explanation of a failure can be represented by a chain of inference which implies that the reasoning should be rule based, but the complexity of the failure explanation process suggests that it is more often case-based [Riesbeck and Schank, 1989].

2.3.2 Explanation Facility

The second application of explanation in diagnosis using case-based reasoning is the implementation of an explanation facility. An explanation facility in a knowledge-

based system provides justification for the solution from the system to the domain experts. This concept is important in a diagnostic system as it is usually a domain expert who is seeking the diagnosis, and the result must be justifiable by the system. Generally this justification is given as the answer to the questions "Why?" and "How?". Why is the system asking me for this piece of information? How did the system come to that solution? In a case-based reasoning diagnostic system, it can be argued that the case itself can provide the explanation for why it was selected as the solution to the problem [Riesbeck and Schank, 1989] because of the premise that expertise is most likely a library of past experiences and better supports knowledge transfer. In fact, a significant amount of research is being done in using case-based reasoning to implement explanation systems in a broad range of problem domains [Schank et al, 1994].

The traditional approach to implementing explanation facilities in knowledge-based systems is to use the inference chain that has taken place in the reasoning process through "firing" of the rules in the knowledge-base. MYCIN was one of the first knowledge-based systems to offer this innovation [Buchanan and Shortliffe, 1984].

An explanation facility for a diagnostic case-based reasoner can be implemented by recalculating the diagnosis match scores (see section 3.2.2.5 for match score calculation) for the cases being proposed as solutions [Taylor, 1994]. This process would determine the contribution made to the final match score by each diagnostic case feature, and be explained back to the user in textual format, answering the question how the case came to be given as a solution.

The traditional approach to implementing an explanation facility was not followed in this thesis and is left to future work (see section 5.2), but a non-traditional explanation facility was implemented. During the course of research, it was found that the cases with known solutions available in technical domains, specifically fault diagnosis, are usually stored in systems built for tracking and analyzing these cases. In these cases many features are stored as codes or numbers which represent a textual description. This

approach is beneficial in conserving storage space, but causes a problem when building a case-base. The case-based reasoner must be able to answer the question "what?" and explain to the user the meaning of the codes and sequence numbers. In this thesis, this knowledge is included in the knowledge base and presented to the user during the diagnostic process (for details see section 3.3).

3.0 A KNOWLEDGE BASE FOR TELECOMMUNICATIONS TROUBLES

3.1 ART-IM Expert System Shell

Expert system shells are software products that aid users in developing knowledge-based systems. These shells provide knowledge representation, data structures necessary for representing knowledge, an inference engine which provides the ability to reason about that knowledge, as well as user and external system interface capabilities.

Before examining the knowledge base required to diagnose telecommunication troubles, it is first necessary to describe the expert system shell used in its development, and the components that were used to host, represent, and manipulate the knowledge base. This will allow the knowledge base to be described in terms of the expert system shell.

The Automated Reasoning Tool - Information Management (ART-IM) is an expert system shell developed by Inference Corporation that provides an effective tool kit for development of knowledge-based systems. ART-IM contains the ART-IM language as well as several special purpose components for developing user interfaces and linking to external procedural languages.

ART-IM employs a modified Rete match algorithm within its inference engine, a memory intensive pattern matching algorithm, which provides the matching in ART-IM, and drives the inference engine through pattern matching on forward chaining rules (see section 1.1.2.1). The extent and inclination of the modifications that Inference has made to the Rete match algorithm are not known, as this implementation information is considered proprietary and is kept confidential [Taylor, 1994].

ART-IM is available on a number of platforms, and is continually evolving and improving its user interaction and reasoning modules. Recently, Inference Corporation

has focused on case-based reasoning, with its CBR2 family of products, providing case-based reasoning tools in a number of commercial fields, such as help desk applications.

3.1.1 Schema Structure

A schema is a data structure, made up of frames, that is used to represent a collection of information about a particular object [Inference Corporation, 1991]. Schemas in ART-IM are the fundamental data structure and consist of a schema name and a number of constituents called slots. The slots represent individual pieces of information describing characteristics of the schema or its relationship with other schemas. A slot is a data structure used to represent some characteristic of an object. A slot consists of a slot name and zero or more values.

Schemas have a number of powerful characteristics. Hierarchies of schemas can be created using the schema characteristic of inheritance. Inheritance allows some schemas to represent a class of objects and other schemas to represent individual objects. Objects belonging to a class share characteristics or slots, which allows these characteristics to be shared with other objects or schemas belonging to that class.

Schemas can be retrieved from memory though the use of a key slot whose value uniquely identifies the schema, and is used as the parameter for retrieval.

3.1.2 Case-Based Reasoner

ART-IM provides a retrieval mechanism for ART-IM objects for implementing case-based reasoning. This mechanism can be used in conjunction with other ART-IM features to build case-based systems. This retrieval mechanism searches a collection of stored cases, represented as ART-IM schemas, to find and retrieve the cases which most closely match the presented case. The degree of match depends on the degree of similarity. Once a case is retrieved, it can then be examined to determine if it provides a solution to the presented problem.

In ART-IM, the objects used to represent cases are schemas. A case base structure is defined by a case schema whose slots represent the features of the stored case, and

which acts as the access method for the application to the case-base. Any operation performed on the case-base must be through the case schema, including adding cases, retrieving cases, and maintaining the feature values and index slot information that, along with the Rete match algorithm, provide ART-IM's matching and retrieval functionality.

3.2 The Case Base

3.2.1 Case Library

3.2.1.1 Source of Telecommunication Trouble Cases

A reliable set of cases with known solutions are a requirement for an effective case-based reasoning system. When dealing with the telecommunication network diagnosis domain the logical place to obtain this information is from a telecommunications company, who will have a ready supply of network troubles with associated resolutions. This process simplifies the knowledge engineering task to a great extent, which is an important characteristic of case-based reasoning

At the New Brunswick Telephone Company Ltd. (NBTel), telecommunication network facility information and related subscriber information is stored in the TELephone FACilities System (TELFACS). When a customer reported network trouble is received, the pertinent trouble information is entered by a repair clerk into the Trouble Reporting System (TRS) which is a sub-system of TELFACS. Here the trouble information is associated with the related facility and subscriber information. Once the trouble has been resolved, and the cause and resolution are recorded, the trouble, facility, and subscriber information are extracted and stored in the Trouble Report Analysis Capability Enhancement (TRACE) system.

The TRACE system operates on an IBM 3090 mainframe computer, running IBM's MVS operating system. The system is implemented using the COBOL programming language, and sequential files. TRACE data is used to provide batch analysis reports to indicate the performance of NBTel's repair personnel.

TRACE groups troubles into monthly files that are maintained for 90 days in the system. This data provides complete and reliable instances of faults and their characteristics, along with solutions, in a defined telecommunications network.

3.2.1.2 Case Preprocessing

Once the trouble information has been extracted from the TRACE system, the data must be optimized through a number of processing steps before it can be used as entries in the case base. It would be redundant to have cases with the same symptoms, diagnosis and descriptive information, so duplicate cases must be removed. This step is accomplished by a stand alone C procedure that sorts the case records on all fields, and sequentially compares the entire case records, removing any exact duplicates. This process removes only exact duplicate cases, as cases with any different feature values may improve the diagnostic performance of the case base.

As well as removing duplicate cases, unused (blank) fields that are included in the TRACE record must also be removed. This function is performed by the rule base as the cases are initially loaded into the case base (see section 4.4.3.1) The removal of unused fields reduces the size of the case base and results in helping to improve the efficiency of matching.

A third processing step that must take place is the generation of fields that do not exist in the original data, but are derived from existing fields to provide possible symptoms or diagnostic case features that are implied in the data but not easily indexed. This function is also performed by the rule base when creating the case base (see section 4.4.3.1). The addition of derived knowledge to the trouble case provides extended capabilities in using case-based reasoning to diagnose the trouble (see section 3.2.1.4).

3.2.1.3 Telecommunication Trouble Case Record Format

After preprocessing, the trouble case record contains 71 fields of information (see table 3.1) which describe trouble characteristics, subscriber information, and outside plant equipment and location information. Included in this information are trouble symptoms

and diagnostic information, which makes this data appropriate for the case base of a diagnostic system.

```
(defschema trouble
 (ACC-INFO)
 (ADDRESS)
 (APP-DATE)
 (APPNTMT)
 (AUX-LINE)
 (BATCH-DATE)
 (CABCNT-REM)
 (CABLE-LOC)
 (CABLE-NUM)
 (CAI)
 (CARRIER)
 (CATEGORY)
 (CAUSE)
 (CAUSE-SUB)
 (CLASS-OF-SERVICE)
 (CLOSE-DATE)
 (CLOSED-BY)
 (CO-EQUIP)
 (CO-GROUP)
 (COS-SUB)
 (DIS-DATE)
 (DIS-TIME)
 (DISP-SUB)
 (DISPOSITION)
 (DIST)
 (DPAIR)
 (DPAIR25)
 (DPAIR50)
 (EQUIP-TYP)
 (EST-DATE)
 (EST-TIME)
 (EXTEN)
 (FCI)
 (FEEDER)
 (GSA)
 (MCSIRBAN)
 (NET-NUM)
 (NTR-IND)
 (NUMBER-KEY)
 (OP-INIT)
 (PAIR)
 (PAIR25)
 (PAIR50)
 (PHONE-REM)
 (PRITY-IND)
 (REC-BY)
 (REC-DATE)
 (REF-DATE)
 (REF-IND)
 (REF-ORG-IND)
 (REF-TIME)
 (RPM-INIT)
 (SERVICE-CENTRE)
 (SR-CNT)
 (SR-IND)
 (SRI)
 (STUDY)
 (SUB-NAME)
 (TC-REM)
 (TEL-NUMBER)
 (TESTED-BY)
 (TF-REM)
 (TI-REM)
 (TRBL-REM)
 (TROUBLE-TYPE)
 (TSTR-INIT)
 (TT-CODE)
 (TT-DATE)
 (TT-IND)
 (TT-TIME)
 (WC))
```

Figure 3.1 Trouble Case Schema Structure

Also included in the trouble case record is the implied trouble categorization defined by the previously described diagnostic model (see section 2.1). This model requires the "trouble type", "disposition", and "trouble found" values for categorization of troubles. This information exists in the trouble case record, with the exception of the "trouble found" value which provides the specific diagnosis. This "trouble found" field is available in NBTel's Trouble Reporting System (TRS) (see section 3.2.1.1) but has not been carried through the extraction of data to the TRACE system, and therefore the data is not available for the case-base. The absence of this piece of information does not affect the usefulness of the case base in providing diagnostic information, as a diagnosis

description is available in the mandatory "trouble found remarks" field, but a proper categorization of the trouble is not possible without it. The inclusion of the "trouble found" field in the TRS extract is left to future work (see section 5.2).

A trouble case in ART-IM is represented using the trouble case schema (see figure 3.1), with a schema slot defined for each trouble case feature. This trouble case schema is always used when accessing or manipulating information in the case base.

Table 3.1 Trouble Case Data Fields

	Trouble Feature Name	Trouble Feature Slot Name	Trouble Feature Description
1.	Access Information	ACC-INFO	Trouble Access Point Description
2.	Address	ADDRESS	Trouble Address
3.	Appointment Date	APP-DATE	Repair Appointment Date
4.	Appointment Time	APPNTMT	Repair Appointment Time
5.	Auxiliary Line	AUX-LINE	Auxiliary Line Indicator
6.	Batch Date	BATCH-DATE	TRACE Batch Date
7.	Cable Count Remark	CABCNT-REM	Unformatted Input Describing Cable Count
8.	Cable Location	CABLE-LOC	Trouble Cable Location
9.	Cable Number	CABLE-NUM	Cable Serial Number
10.	Customer Advised Indicator	CAI	Indicates Customer was Advised of Closed Trouble
11.	Subscriber Carrier	CARRIER	Subscriber Long Distance Carrier
12.	Reported Category	CATEGORY	Category of how Trouble was Reported
13.	Trouble Cause	CAUSE	Classification of Trouble Cause
14.	Trouble Cause Sub Code	CAUSE-SUB	Used for Special Studies
15.	Class of Service	CLASS-OF-SERVICE	Class of Service
16.	Trouble Close Date	CLOSE-DATE	Date Trouble was Closed
17.	Trouble Closed By	CLOSED-BY	Identifier of Person who Closed Trouble
18.	Central Office Equipment	CO-EQUIP	Type of Central Office Equipment
19.	Central Office Group	CO-GROUP	Central Office Group Classification
20.	Class of Service Sub Code	COS-SUB	Used for Special Study
21.	Trouble Dispatch Date	DIS-DATE	Date Trouble was Dispatched
22.	Trouble Dispatch Time	DIS-TIME	Time Trouble was Dispatched
23.	Trouble Disposition Sub Code	DISP-SUB	Used for Special Studies
24.	Trouble Disposition	DISPOSITION	Trouble Physical Location Disposition
25.	Distribution Cable	DIST	Distribution Cable Number
26.	Distribution Cable Pair	DPAIR	Distribution Cable Pair Number
27.	Distribution Cable 25 Pair Range	DPAIR25	Derived Field to Make Cable Fault Matching More Efficient
28.	Distribution Cable 50 Pair Range	DPAIR50	Derived Field to Make Cable Fault Matching More Efficient
29.	Equipment Type	EQUIP-TYP	Switching Equipment Type
30.	Trouble Established Date	EST-DATE	Date Trouble was Established
31.	Trouble Established Time	EST-TIME	Time Trouble was Established
32.	Extension	EXTEN	Extension Identifier
33.	Frame Check Indicator	FCI	Frame Check Indicator
34.	Feeder Cable Number	FEEDER	Feeder Cable Number
35.	Geographical Serving Area	GSA	Geographical Serving Area
36.	Multi-Purpose Field	MCSIRBAN	Describes Various Characteristics of Trouble
37.	Network Serial Number	NET-NUM	Network Serial Number
38.	No Test Required	NTR-IND	No Test Required Indicator
39.	Sequence Number Key	NUMBER-KEY	Derived Field to Provide Unique Key
40.	Operator Initials	OP-INIT	Initials of Operator Involved in Trouble
41.	Feeder Cable Pair	PAIR	Feeder Cable Pair Number
42.	Feeder Cable 25 Pair Range	PAIR25	Derived Field to Make Cable Fault Matching More Efficient
43.	Feeder Cable 50 Pair Range	PAIR50	Derived Field to Make Cable Fault Matching More Efficient
44.	Phone Remarks	PHONE-REM	Unformatted Input Describing Phone Trouble
45.	Priority Indicator	PRITY-IND	Classification of Priority of Trouble
46.	Received By	REC-BY	Identifier of who Received Trouble Report
47.	Trouble Received Date	REC-DATE	Date Trouble was Received
48.	Referred Date	REF-DATE	Date Trouble was Referred to another Office
49.	Referred Indicator	REF-IND	Indicates Trouble was Referred
50.	Referred Organization Indicator	REF-ORG-IND	Indicates What Group Trouble was Referred to
51.	Referred Time	REF-TIME	Time that Trouble was Referred
52.	Repairman's Initials	RPM-INIT	Initials of Person who Repaired Trouble
53.	Service Center	SERVICE-CENTRE	Provincial Service Center
54.	Subscriber Report Count	SR-CNT	Indicates Number of Subscriber Reports
55.	Subscriber Report Indicator	SR-IND	Indicates if Subscriber Reported Trouble
56.	Set Repaired Indicator	SRI	Indicates if Phone Set was Repaired
57.	Special Study	STUDY	Trouble Included in Special Study
58.	Subscriber Name	SUB-NAME	Trouble Subscriber Name
59.	Trouble Cause Remarks	TC-REM	Unformatted Input Describing Trouble Cause
60.	Trouble Telephone Number	TEL-NUMBER	Trouble Telephone Number
61.	Trouble Tested By	TESTED-BY	Tester Identifier
62.	Trouble Found Remarks	TF-REM	Unformatted Input Describing Trouble Found
63.	Trouble Test Information Remarks	TI-REM	Unformatted Input Describing Test Results
64.	Trouble Remarks	TRBL-REM	Unformatted Input Describing General Trouble Remarks
65.	Trouble Type	TROUBLE-TYPE	Description of Reported Trouble
66.	Tester Initials	TSTR-INIT	Initials of Trouble Tester
67.	Trouble Type Code	TT-CODE	Trouble Type Code
68.	Trouble Test Date	TT-DATE	Date Trouble was Tested
69.	Trouble Test Indicator	TT-IND	Indicates Trouble was Tested
70.	Trouble Test Time	TT-TIME	Time Trouble was Tested
71.	Wire Center	WC	Provincial Wire Center

3.2.1.4 Diagnostic Case Features

Although experts are not needed for creation of the knowledge base when implementing a case-based reasoner, expert knowledge and experience are still required to optimize the performance of the system. Of the 71 fields of information available about each trouble case, 9 were identified by repair personnel experienced in isolating and diagnosing telecommunication network troubles, as being typical symptoms or diagnostic case features of troubles with similar diagnoses. Table 3.2 describes the trouble diagnostic case features, along with their schema slot names, and ART-IM match types. The experts also determined that the ability to search for cable pair ranges would be useful to allow trouble trend analysis on groups of cable pairs. This requirement initiated the addition of four derived index fields (see table 3.2, slot names DPAIR25, DPAIR50, PAIR25, PAIR50). These indices, whose values are derived from the feeder and distribution cable pair values, allow matching on each stored case for inclusion in a cable pair ranges of 25 or 50 pairs. These index features contain values indicating the first cable pair in the specified cable ranges of 25 or 50 pairs, allowing matching on each stored case for inclusion in the cable range of a presented case.

In a telecommunication network, the feeder and distribution cable pairs are numbered sequentially for a given cable. If the feeder cable pair associated with a specific trouble is number 37, then the feeder cable pair feature (PAIR slot) of that case will contain the value "37". In this case, the value "26" will be placed in the PAIR25 slot to indicate the first cable pair in the 25 pair range that the feeder cable pair value is included in (i.e. [26, 50]). Likewise, the value "1" will be placed in the PAIR50 slot to indicate the first cable pair in the 50 pair range that the feeder cable pair value is included in (i.e. [1, 50]). These value can then be matched when inclusion in a specific range is requested.

Input from trouble diagnosis experts also indicated that additional diagnostic case features, such as line test values, would be beneficial in utilizing the case base to

diagnose troubles. This data is available in the TRS system, but is not included in the TRACE extract. The inclusion of the line test results in the case base as a diagnostic case feature is left to future work (see section 5.2).

Table 3.2 Trouble Diagnostic Case Features

Feature Name	Feature Slot Name	Matching Type
Subscriber Address	ADDRESS	Character
Distribution Cable Identifier	DIST	Character
Distribution Cable Pair	DPAIR	String
Distribution Cable 25 Pair Range	DPAIR25	String
Distribution Cable 50 Pair Range	DPAIR50	String
Feeder Cable Identifier	FEEDER	String
Geographical Serving Area	GSA	String
Feeder Cable Pair	PAIR	String
Feeder Cable 25 Pair Range	PAIR25	String
Feeder Cable 50 Pair Range	PAIR50	String
Subscriber Name	SUB-NAME	String
Telephone Number	TEL-NUMBER	String
Wire Center	WC	String

3.2.2 Partial Matching

One of the principles of case-based reasoning is to find and retrieve similar cases to the case being presented. To accomplish this in a diagnostic case-based reasoner, partial matching on the diagnostic case features (i.e. some features match and some don't) must be possible. As well, specialized types of matching can also provide partial or "fuzzy" matching on an individual diagnostic case feature. ART-IM provides a number of match types and case scoring algorithms to accomplish partial matching.

It should be noted that not all of the match types described in the following sections are used in the case base implemented as part of this thesis, but are included to give a complete description of the types of matching that ART-IM provides, as well as indicate the types of matching that could be used if more diagnostic case features were added to the case base.

3.2.2.1 String Matching

String matching is the simplest match type. Either a string feature of the presented case is the exact match of a string in a stored case or it is not. The string match, when applied to a case feature, requires the specification of a match weight and a mismatch weight. If the strings match, the match weight is used to increase the case's score. If the strings do not match, the mismatch weight is used to decrease the cases score. In the case base implemented as part of this thesis, the match weight was set by the user, and could range between 0 and 100. The mismatch weight could range between -50 and 0 (see section 3.2.3 for calculation of the mismatch weight).

The user may also specify absolute matching on strings. If "select" is specified as the match weight, and a match is generated, the stored case is given a perfect score of 1.0. If "reject" is specified as the mismatch weight, and a mismatch is generated, the stored case is given an absolute mismatch score of -1.0.

The following formula is used to determine the string feature weight that is contributed to the overall score of the stored case [Inference Corporation, 1991]:

$$feature\_weights(f)i = \begin{cases} fmatchi, if & f = p \\ fmismatchi, if & f \neq p \end{cases} \quad (1)$$

where

$fmatchi$ = the match weight of feature f for the i^{th} case,

$fmismatchi$ = the mismatch weight of feature f for the i^{th} case,

f = string value for the i^{th} case,

p = string value of the same feature of the presented case.

This equation represents the feature-weight value for feature f of the i^{th} stored case which is being compared to the presented case.

3.2.2.2 Word Matching

In word matching, all of the words of the presented case's text feature are compared to all of the words in the stored case's text feature. The degree of partial matching is based on the proportion of words in the presented case's text feature that match words in the stored case's text feature. The order of the words is unimportant.

This type of matching provides a higher degree of partial matching than string matching in that it allows for partial matching of sentences or phrases. The following formula is used to determine the word feature weight that is contributed to the overall score of the stored case [Inference Corporation, 1991]:

$$feature\_weights(f)i = \frac{(wx - wm)}{wx} fmismatchi + \frac{wm}{wx} fmatchi \qquad (2)$$

where

wm = the number of words in common between the presented case and the i^{th} case,

wx = the number of words in the presented case (after text preprocessing),

$fmatchi$ = the match weight of feature f for the i^{th} case,

$fmismatchi$ = the mismatch weight of feature f for the i^{th} case.

This equation represents the feature-weight value for feature f of the i^{th} stored case which is being compared to the presented case.

3.2.2.3 Character Matching

Character match is by far the most powerful of the text matching types provided. It is similar to word matching but provides a much higher degree of partial matching. Instead of using words, character matching uses consecutive trigrams as the match unit. A trigram is a 3-character sequence. For example the word "speaker" has 9 trigrams embedded in it (i.e. "__s", "_sp", "spe", "pea", "eak", "ake", "ker", "er_", and "r__").

Character matching, like word matching, provides partial matching, but again at a much higher degree. This type of matching will still generate a partial match if words are misspelled, whereas with string or word matching, a misspelled word would generate a total mismatch.

In character matching, the trigrams of the character feature of the presented case are matched against the trigrams of the character feature of the stored case. The degree of partial matching is based on the proportion of the trigrams in the presented case that match trigrams in the stored case. The order of the trigrams is unimportant.

The following formula is used to determine the word feature weight that is contributed to the overall score of the stored case [Inference Corporation, 1991]:

$$feature\_weights(f)i = \frac{(tx - tm)}{tx} fmismatchi + \frac{tm}{tx} fmatchi \qquad (3)$$

where

tm = the number of trigrams in common between the presented case and the i^{th} case,

tx = the number of trigrams in the presented case (after text preprocessing),

$fmatchi$ = the match weight of feature f for the i^{th} case,

$fmismatchi$ = the mismatch weight of feature f for the i^{th} case.

This equation represents the feature-weight value for feature f of the i^{th} stored case which is being compared to the presented case.

3.2.2.4 Numeric Matching

String, word and character matching are very useful for matching textual features. These types of matching do not perform well if the feature value of the presented case is a number. Number matching uses the distance between two numbers to determine the score of the feature. As well as supplying the match and mismatch weights as were

required with the other types of matching, number matching also requires that a minimum value, maximum value, match deviation and minimum precision be specified. Figure 3.2 depicts the graph used in calculating the match score of the numeric feature of the stored case. The match deviation specifies the width of the base of the triangle. If the value of the numeric feature of the stored case is outside the triangle, the mismatch weight will be applied. If the value of the numeric feature of the stored case is within the triangle, the score will be determined by the matching coordinate on the y-axis. If the value of the numeric features of the presented case and stored case are a perfect match, the match weight will be applied.

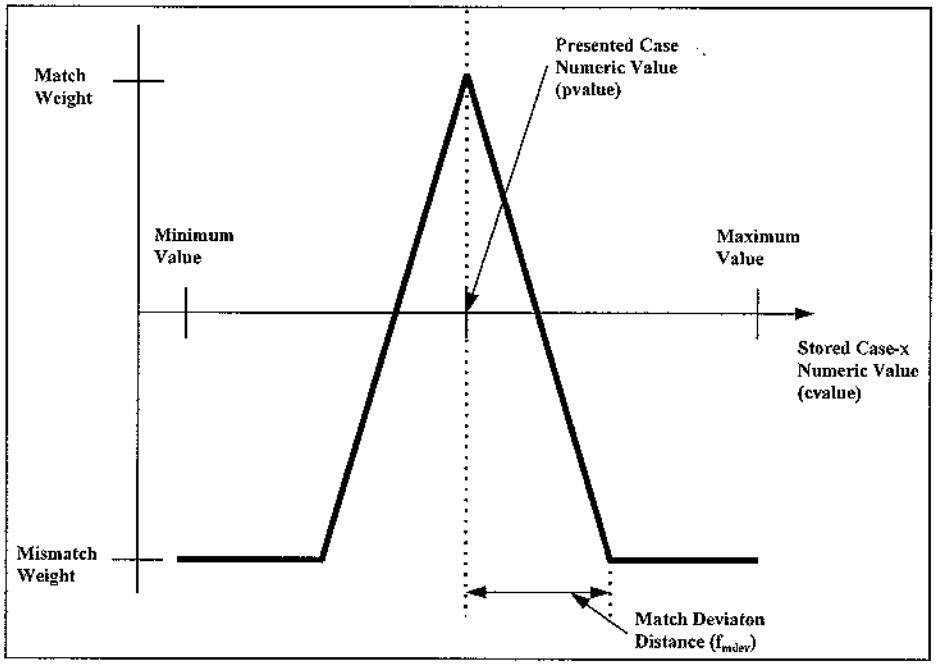


Figure 3.2 Numeric Scoring Function

The following formula is used to determine the numeric feature weight that is contributed to the overall score of the stored case [Inference Corporation, 1991]:

$$feature\_weights(f)i = \begin{cases} fmatchi - s(fmatchi - fmismatchi)\ if\ s < 1 \\ fmismatchi,\ if\ s \geq 1 \end{cases} \quad (4)$$

where

$$s = \frac{|cvalue - pvalue|}{f_{mdev}},$$

$cvalue$ = the i[th] case's value for the feature,

$pvalue$ = the presented case's value for the feature,

f_{mdev} = the match deviation,

f_{matchi} = the match weight of feature f for the i[th] case,

$f_{mismatchi}$ = the mismatch weight of feature f for the i[th] case.

This equation represents the feature-weight value for feature f of the i[th] stored case which is being compared to the presented case.

3.2.2.5 Case Scoring

Once the *feature_weights(f)i* have been calculated for the indexed case features, or in our case diagnostic case features using Equations 1 through 4, the variable percentage contribution weight for the i[th] case can be calculated using Equations 5 through 7 [Inference Corporation, 1991]. Normally, the importance of a feature to a case score is relative to the weights and degree of matching to the other features of the stored case. These features are referred to as variable percentage.

The vp_maximum_weight for a stored case is the sum of the greater of the feature match-weights and mismatch weights for the variable percentage features of the i[th] stored case, and can be calculated using the following formula:

$$vp\_maximum\_weight\,i = \sum_{\forall f \in Casei} max(fmatchi, fmismatchi) \quad (5)$$

where

$Case_i$ = the set of all features, except the fixed percentage slot (see equation 8), that are found in the i^{th} case.

The vp_weight is the sum of all feature weights for the ith case's variable-percentage features that appear on the presented case, and can be calculated using the following formula:

$$vp\_weight_i = \sum_{\forall f \in Pcase, f \in Case_i} feature\_weights(f)_i \qquad (6)$$

where

Pcase = the set of all features, except the fixed percentage slot, that are found in the presented case,

$Case_i$ = the set of all features found on the presented case, except the fixed percentage slots.

The contribution of the variable-percentage slots can be calculated using the following formula:

$$vp\_contribution_i = \frac{vp\_weight_i + vp\_absence\_weight_i}{vp\_maximum\_weight_i} \qquad (7)$$

where

$vp\_absence\_weight_i$ = the sum of the absence weights (i.e. penalty weights for cases missing specific slots) for all variable percentage slots of the i^{th} case .

Fixed percentage features provide the ability to set the contribution of a feature in the stored case to contribute a fixed percentage to case scores, regardless of the number of other features. The fixed percentage contribution is calculated using Equation 8, as follows:

$$fp\_contribution_i = \frac{fp\_weight_i + fp\_absence\_weight_i}{fp\_maximum\_weight_i} \quad (8)$$

With the calculation of the variable and fixed percentage slot contribution complete, the raw score can now be calculated using equation 9 as follows:

$$raw\_score_i = min(1, fp \times fp\_contribution_i + (1 - fp) \times vp\_contribution) \quad (9)$$

where

fp = the fixed percentage value given a single feature slot.

This raw score for a case falls into a range $(-\infty, 1]$.

Equation 10 can now be used to calculate the case score, mapping the raw score to the range [-1, 1], allowing for easy in-order retrieval of cases similar to the presented case.

$$case\_score_i = \frac{2}{2 - raw\_score_i} - 1 \quad (10)$$

The mapping relationship between the raw score and the case score is shown in Figure 3.3. This demonstrates how the case score values approach -1, 0, and 1 for the corresponding raw score values of $-\infty$, 0 and 1 respectively.

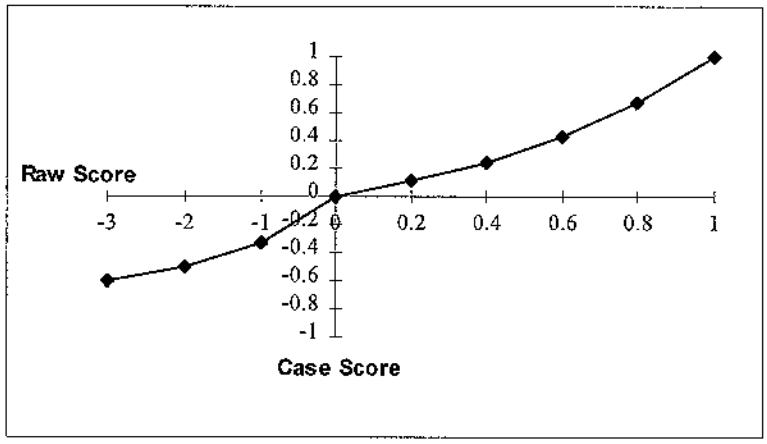


Figure 3.3 Case Score vs. Raw Score

To fully illustrate the scoring equations that are defined in this section, an example consisting of a presented case, and a case base containing two cases is given. Each case has two slots. One contains the subscriber address, which uses word matching, while the other contains the Geographical Serving Area, which uses string matching. The associated match and mismatch weights are 80 and -40 for the GSA and 50 and -25 for the address. The presented case and two case base cases are shown in Table 3.3, and the trigrams that are contained in the Subscriber Address values, that are needed to calculate the feature weight, are specified in Table 3.4.

Table 3.3 Score Calculation Example Cases

	Presented Case	**Case 1**	**Case 2**
Subscriber Address	P1 RL231	P4 RL231	P7 RL223
Geographical Serving Area	HVS	HVS	HVS

Table 3.4 Subscriber Address Value Trigrams

Presented Case	**Case 1**	**Case 2**
"__P"	"__P"	"__P"
"_P1"	"_P4"	"_P7"
"P1_"	"P4_"	"P7_"
"1_R"	"4_R"	"7_R"
"_RL"	"_RL"	"_RL"
"RL2"	"RL2"	"RL2"
"L23"	"L23"	"L22"
"231"	"231"	"223"
"31_"	"31_"	"23_"
"1__"	"1__"	"3__"

The following calculations are performed to determine which case is the best match for the presented case.

from Equation 3 $\quad$ $\text{feature\_weight(Address)}_1 = ((10-7)/10)*(-25)+(7/10)*50 = 27.5$
similarly $\quad$ $\text{feature\_weight(Address)}_2 = ((10-3)/10)*(-25)+(3/10)*50 = 17.5$

from Equation 1 $\quad$ $\text{feature\_weight(GSA)}_1 = 80$
similarly $\quad$ $\text{feature\_weight(GSA)}_2 = 80$

from Equation 5	$vp\_maximum\_weight_1 = 80 + 50 = 130$
similarly	$vp\_maximum\_weight_2 = 130$
from Equation 6	$vp\_weight_1 = 27.5 + 80 = 107.5$
similarly	$vp\_weight_2 = 17.5 + 80 = 97.5$
from Equation 7	$vp\_contribution_1 = (107.5 + 0)/130 = 0.8269$
similarly	$vp\_contribution_2 = (97.5 + 0)/130 = 0.75$

This assumes that a fixed percentage slot is not used.

from Equation 8	$raw\_score_1 = \min(1, 0 + (1-0) * 0.8269) = 0.8269$
similarly	$raw\_score_2 = \min(1, 0 + (1-0) * 0.75) = 0.75$
$\therefore$ from Equation 10	$case\_score_1 = (2/(2 - 0.8269)) - 1 = 0.7049$
$\therefore$ similarly	$case\_score_2 = (2/(2 - 0.75)) - 1 = 0.6$

Therefore, case1 is a better match for the presented case than case 2.

It should be noted that the scores presented to the user for this thesis have been multiplied by 1000, and had the decimal places truncated. This was done to allow the value to be read easily, and also provide adequate accuracy of the score.

3.2.3 Value Hierarchy

The value hierarchy defines the relative importance of the diagnostic case features and are used extensively in determining the contribution of a case feature to the score of a stored case. In ART-IM the value hierarchy is represented by the match and mismatch weights assigned to the diagnostic case features. In the trouble diagnosis case base default match and mismatch weights are assigned which give all of the indexed features the same relative importance.

There are two approaches to determining the appropriate value hierarchy for a diagnostic case-based reasoner; (1) the value hierarchy to be used in various diagnostic situations can be based on statistical study and analysis of the value hierarchy which produces the best diagnosis results, or (2) allow the user of the diagnostic case-based reasoning tool to determine the value hierarchy that they feel is appropriate based on their experience or preference. There are pros and cons for both of these approaches. The statistical approach may produce better results, at least until the user has had enough

experience to realize which value hierarchies will perform best in various situations, but allowing the user to have control of the diagnostic process by establishing the value hierarchy will most likely insure more use, and therefore improved success, of the diagnostic tool in practical situations.

An important feature of the value hierarchy implemented in ART-IM is that match and mismatch weights can be changed at run-time. The trouble diagnosis case base utilizes this feature to allow the diagnostic case feature value hierarchy to be changed depending on the situation by applying the specified match weights to the appropriate indexed features. For example the user may determine that outside plant features are important in diagnosing the current trouble (i.e. wire centre, cable, cable pair). The user may then change the relative importance of these diagnostic case features, and have them reflected in the subsequent match.

The mismatch weight is calculated using equation 11.

$$mismatch\_weight = -\left(\frac{match\_weight}{2}\right) \tag{11}$$

3.3 Explanation Facility

An important part of the knowledge base for telecommunication fault diagnosis is the explanation facility. Instead of the traditional explanation facility, answering the questions "why?" and "how?", this facility answers the question "what" by providing a description look-up utility that matches field codes contained in the case base with their associated textual description and provides this description to the user in place of the cryptic codes and numbers on a case-by-case basis.

```
(DEFSCHEMA type-fld0
 (instance-of look-up)
 (field-name "type")
 (field-code "0")
 (desc "Cannot Call - No Dial Tone"))
```

Figure 3.4 Sample Description Look-up Schema

The description knowledge is stored in the "desc" slot of instances of the "look-up" schema representing a specific field and code (see Figure 3.3). The name of the individual look-up schemas is derived from the case feature that it applies to (e.g. type) and the value used to represent the description contained in the schema (e.g. "0") The description is accessed by matching on the "field-name" and "field-code" and retrieving the value in the "desc" slot. The description retrieval is performed by the knowledge base as part of the case retrieval process. When the features of a specific case are being retrieved, the descriptions associated with that case are also retrieved. This is accomplished through the use of a retrieval rule and ART-IM user defined functions which are described in sections 4.4.3.1 and 4.4.3.2.

The description look-up knowledge base consists of the retrieval rule, the functions, and 81 instances of the look-up schema containing description information for the possible values of 8 trouble case fields. For a complete list of the look-up schemas, see Appendix III.

3.4 Integration of Rule and Case-Based Reasoning

As we have seen, rules play an important part in the case-based reasoning cycle when dealing with certain problem domains such as diagnosis (see section 1.1.3.3). Combining rule and case-based reasoning can broaden the domain knowledge and in turn, improve the accuracy of diagnostic knowledge-based systems [Golding and Rosenbloom, 1995]. The following sections describe two natural applications for the integration of rules with case-based reasoning in diagnosis.

3.4.1 Adaptation

In the past few years, two classes of case-based reasoners have emerged: the precedent case-based reasoner, and the problem solving case-based reasoner [Portinale et al 1993]. In the precedent case-based reasoner, previous solutions of cases similar to the current one are used as justification for the solution of the current case with very little or no adaptation. The problem solving case-based reasoner retrieves solutions to previous similar cases, but the solutions need to be adapted to fit the current situation. In the first case, pure case-based reasoning seems an adequate solution, but in the second area another problem solving approach, such as rule-based reasoning, needs to be combined with case-based reasoning. The diagnostic problem domain falls into the problem solving category of case-based reasoner, and would therefore benefit from the integration of adaptation rules.

In a diagnostic case-based reasoner, the adaption process must be able to find gaps in the diagnosis explanation and fill in the missing problem causes. The adaptation rules act as mini-problem solvers in this process using domain and task specific knowledge to produce a more correct diagnosis than was available from the case base.

The advantage of case-based reasoning using adaptation, is that the rules can be much simpler than those required by a strictly rule-based system if a strong enough case base is supplied [Hammond, 1989]. An analogy will help to explain how a bigger case library allows the use of significantly weaker adaptation rules in case-based reasoning and still obtain strong results [Riesbeck and Schank, 1989]. The process by which most people learned to calculate logarithms (before calculators) was to look up the number in a table of logarithms. If the number was in the table the process was complete. If the number was not in the table, the logarithms of the two numbers closest to the target number were used to approximate the logarithm by applying a ratio formula.

In this analogy the table of logarithms represents the case library, and looking up the closest numbers is case retrieval. Applying the ratio formula is the adaptation rule.

The rule is much simpler and more efficient that calculating the logarithm from scratch, but is only effective if the table is complete enough to have two numbers close to the target number. If the table is incomplete, bad results are obtained. Thus we see that the adaptation process, in domains such as diagnosis, is made much simpler and more effective with a strong case library.

3.4.2 Answering Questions

One approach to implementing the case-based reasoning in a diagnostic system is to allow the user to enter symptoms of the problem in natural language, and have the system ask the user questions about diagnostic case features to continue to narrow down the list of similar cases. This concept is especially prevalent in some commercial applications. A clear and natural extension to this concept is to use rules to help the user answer questions during a case-base search [Tierney, 1995]. Rules can be used to extract implied information from the symptom description, by matching on specific words or phrases, and apply this information to pending questions. The rules serve to augment and magnify the information in the symptom description, moving the diagnostic session along to a conclusion very rapidly.

Rules can also be used to enforce logical links between answered questions and unanswered ones, by having low-level questions imply information about context and domain to higher-level, more general questions. The knowledge that defines these implications is contained in rules.

The effect of rules on a diagnostic case-based reasoner when applied to question answering, is to augment the search criteria automatically by making explicit the implications of the user's description and answers [Tierney, 1995]. This obviously speeds up the diagnosis process, and cuts down the number of iterative searches required to reach a conclusion, because it supplies the early searches with more explicit criteria. In effect, rules implemented in this way accelerate the process of searching, but do not alter the outcome. The user can get to the same conclusion without rules, just not as quickly.

3.5 Trouble Diagnosis Rule Base

The rules included in the trouble diagnosis knowledge base are used to control the three primary processes of the case-based reasoner (i.e. case-base creation, dynamic weight, and matching and retrieval)(see section 4.4.3.1). This is done by triggering subsequent rules, as well as calling the ART-IM user defined functions associated with each process. For details of the trouble diagnosis rule base see section 4.4.3.1.

The applications for integrating rule and case-based reasoning described above have not been implemented in the trouble diagnosis knowledge base. Adaptation rules could be useful in improving the accuracy of fault diagnosis, and is suggested as future work (see section 5.2).

4.0 IMPLEMENTATION

4.1 A Prototype Diagnostic Aid Tool

The customer reported network troubles at NBTel are currently screened by a repair clerk, whose first objective is to diagnose where the trouble is in the network; i.e. in the customer premise equipment, the customer's wiring, the cable facilities, or the central office (i.e. the switch, or frame). The primary tool for determining this is trouble characteristics and previous experience of the diagnostic expert. If it is determined that the problem is with the subscriber loop, the trouble can be referred to a tester who will perform a metallic test to acquire an electrical profile of the wire pair. Once the diagnosis is complete, the troubles are then dispatched to technicians in the field or in the central office.

The TElephone Trouble Analysts Assistant (TETAA) prototype system was implemented as a tool to aid in this process, as well as evaluate the feasibility of the use of case-based reasoning for fault diagnosis. TETAA was designed to assist in analysis and diagnosis of customer reported fault reports in a telecommunications network, specifically in the outside plant and customer premise areas. The prototype was developed and evaluated with the assistance of the New Brunswick Telephone Company Limited (NBTel), who provided the access to their fault diagnosis experts, the trouble case library, and a real-world testing ground for evaluation of the tool.

4.2 Functional Architecture

The functional architecture of a system defines, and describes the interaction between the processes and data entities that make up an application. Figure 4.1 gives an overview of the TETAA architecture. This diagram describes the input GUI processes and how they interact with the case data and other components of the knowledge base. This interaction through the case-based reasoner facilitates the retrieval of matching cases and triggers the output GUI processes. The diagram also describes how the GUI interacts

directly with the match weights entity of the case base for user adaptation to specific search situations. This is shown in the diagram by an arrow passing through the match weights component.

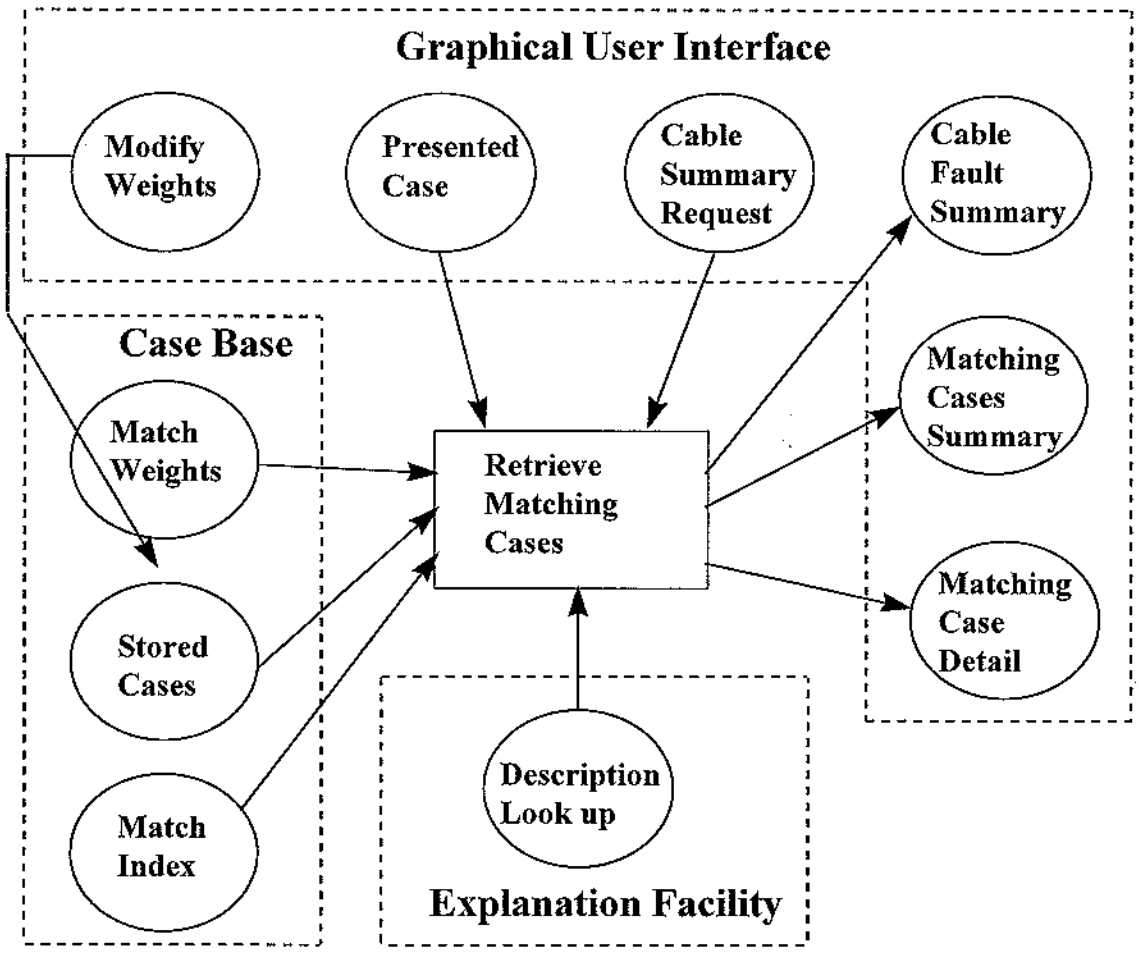


Figure 4.1 Overview of TETAA Functional Architecture

The TETAA functional architecture was designed to provide a template for implementing diagnostic case-based reasoners that could apply to any problem domain. The design includes the basic required functions for any diagnostic case-based reasoner. These include the ability to modify match weights, to specify the diagnostic case features of the current case, and to retrieve a matching case summary and detailed individual case information. The TETAA architecture does include some domain specific functionality,

such as the cable fault summary, and is not meant to be a complete architecture for all problem domains, but it can serve as a generic framework for implementing diagnostic case-based reasoning systems.

4.3 Technical Architecture

The technical architecture of a system describes the applications physical configuration as well as the technical design. Figure 4.2 illustrates an overview of the technical architecture for the TETAA prototype.

4.3.1 TETAA Technical Design

The technical aspects of the TETAA system were designed to provide a generic framework for implementing diagnosis aid tools in other problem domains. To provide this flexibility, the technical design is made up of three layers: the presentation layer, the knowledge access layer, and the knowledge layer. These layers are described in Figure 4.2, by the Graphical User Interface, the Diagnosis Application Programming Interface (API), and the Expert System Shell, respectively. The layered approach to technical design and implementation minimizes the impacts of changes in any one of the layers on the other two layers. For example, if changes are required to an application screen, only changes to the presentation layer are needed, unless the changes require additional diagnostic information. If no additional information is needed on the screen, the other two layers of the application are not affected. If additional information is required, the changes in the other layers are minimal, as the changes can be followed logically from layer to layer. This characteristic of the TETAA design makes maintenance of the application less complex and therefore less expensive, as well as making the application much easier to adapt to other diagnosis domains.

Another advantage to the three layer design approach is that it lends itself naturally to a distributed three-tier client/server configuration. If it were beneficial to have the TETAA application available to multiple users on a network, the layered design approach

could be easily implemented with the presentation layer running on the client, the knowledge access layer running on an application server, and the knowledge layer running on a knowledge base server. This would require the development of communication mechanisms between the GUI and knowledge access level, and the knowledge base, and probably the porting of some of the application code to another platform. To improve performance, it would also require that the expert system shell be supported on another platform (e.g. UNIX) for the knowledge base server. The detailed design and implementation of this architecture is suggested as future work (see section 5.2)

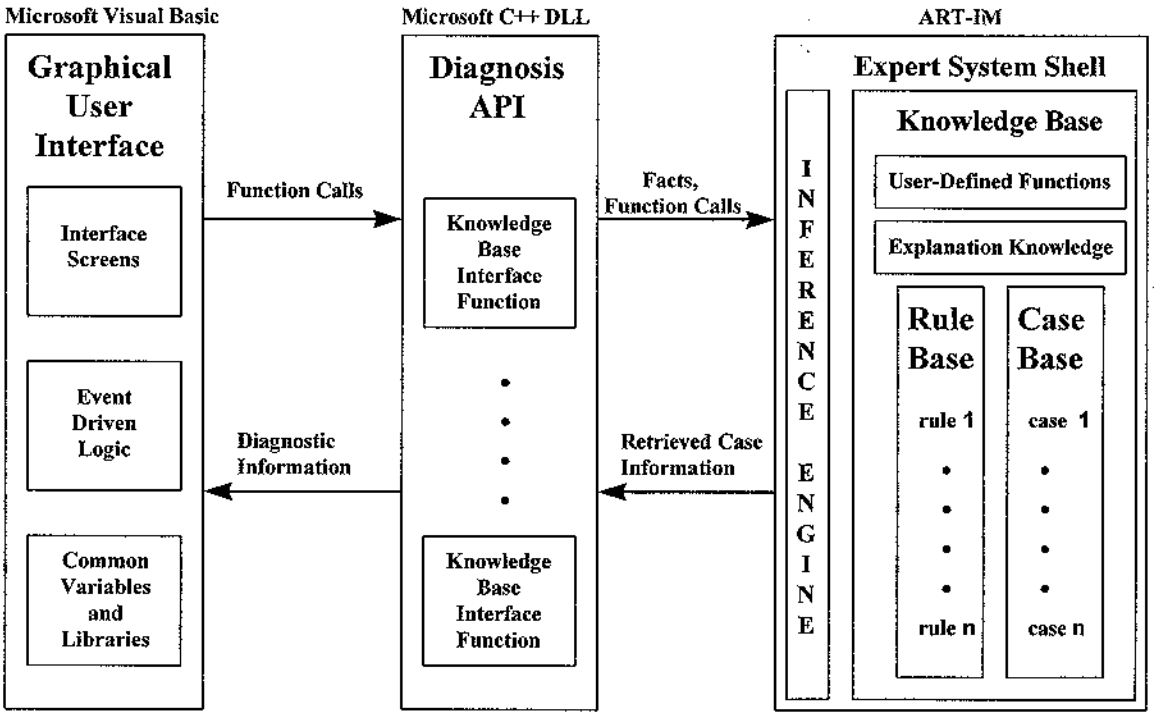


Figure 4.2 Overview of TETAA Three-Layer Technical Architecture

4.3.2 TETAA Physical Configuration

The TETAA prototype was developed, tested, and run on an Intel 486/DX based PC with 16 MB of RAM which ran the Windows 3.1 operating environment.

The software configuration required a different software platform to support each of the three layers. The Graphical User Interface (GUI) was developed using Microsoft Visual Basic 3.0. Visual Basic provides a graphical development environment which allows the user to develop applications using the event driven programming paradigm. Visual Basic allows rapid application development with and flexibility in interfaces to external libraries, such as the diagnosis API, systems and databases.

When completed, the TETAA GUI consisted of approximately 4200 lines of Visual Basic code.

The Application Programming Interface (API) that provides interaction between the GUI and the knowledge base, was developed using Microsoft C++ compiled into a Dynamic Link Library (DLL). The DLL is made up of knowledge base interface functions that can be called from the GUI. When completed, the Diagnosis API was made up of approximately 600 lines of C code.

The knowledge layer was implemented using the inference engine, case-based reasoner and supporting logic representation and data structures of the ART-IM expert system shell, version 2.5 (see section 3.1). Upon completion the ART-IM rules and functions contained approximately 650 lines of ART-IM code.

4.4 Implementing the Architecture

Developing the TETAA prototype involved developing the three layers described in the technical design (see section 4.3.1) to deliver the functionality described by the functional architecture (see section 4.2). The following sections describe the details of the implementation.

4.4.1 The Graphical User Interface

The Graphical User Interface (GUI) is a very important component of any application. The GUI provides the interaction layer between the user and application.

The usability of the interface will greatly affect the acceptance and success of the application, no matter how beneficial the underlying functionality may be.

In the TETAA prototype, the GUI was based on consultations with trouble diagnosis experts as to the interaction flow for retrieval and presentation of diagnostic information that would be the most natural and beneficial to them. The TETAA interface provides the user with a number of options for diagnosis and analysis of trouble reports. These include allowing for modification of match weights for retrieval of specific situations, and summaries of reports on specified cable and pair ranges.

Upon entering the TETAA application, the user is presented with the initial screen, shown in Figure 4.3, which allows them to launch the specific actions or modules of the prototype. The additional screens and their functionality are described in the following sections.

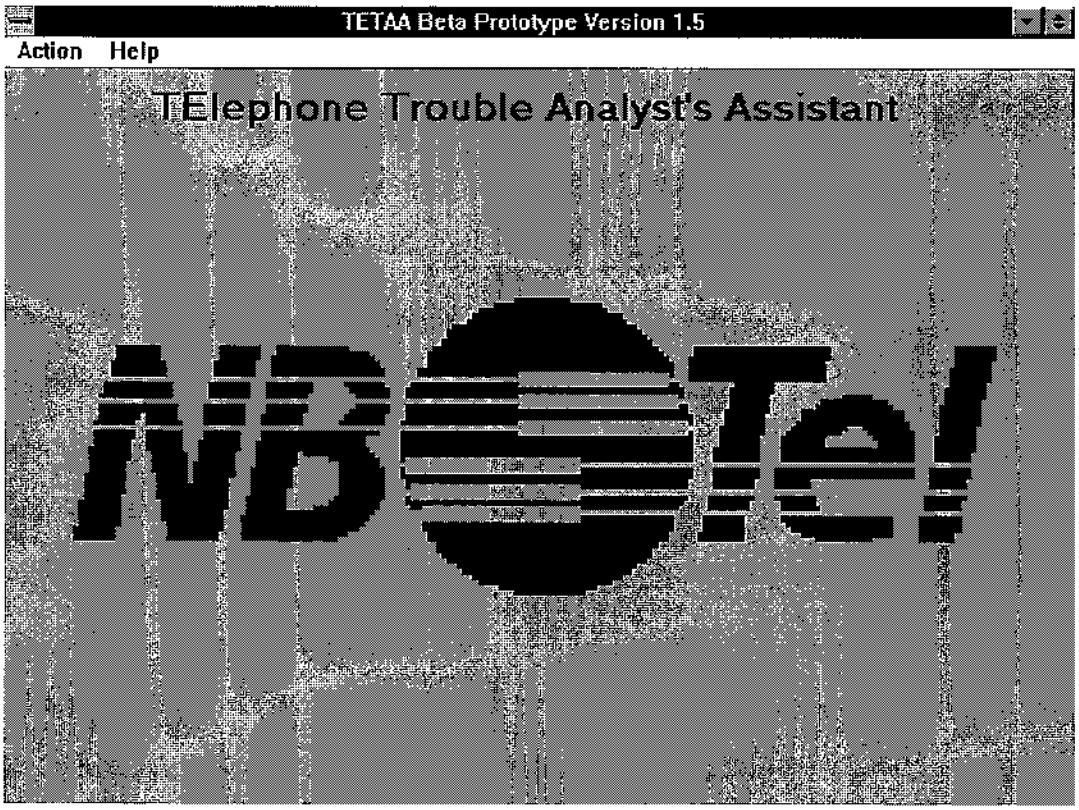


Figure 4.3 TETAA Initial Screen

4.4.1.1 Dynamic Match Weight

The Dynamic Match Weight screen, shown in Figure 4.4, allows the user to define the value hierarchy or relative importance of the specified diagnostic case features for subsequent retrievals of matching cases. This screen can be accessed and the match weights changed at any time during the diagnostic session to reflect different diagnostic situations where certain features are more or less important in the diagnostic process.

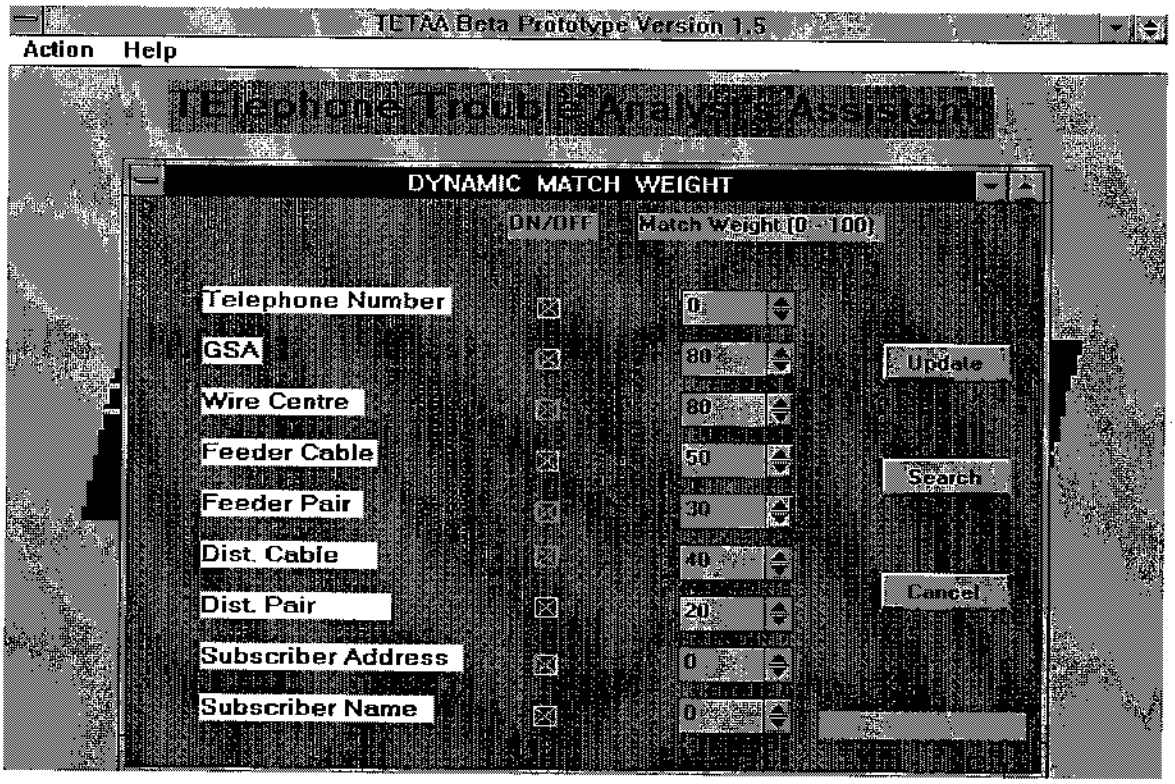


Figure 4.4 TETAA Dynamic Match Weight Screen

The Dynamic Match Weight screen allows the user to turn the matching for individual features on or off, and provides controls to allow the match weights to be set with values ranging from 0-100. In this example, the match weights were chosen as follows: 0 for Telephone No., 80 for GSA, 80 for WC, 50 for Feeder Cable, 30 for Feeder

Pairs, 40 for Dist. Cable, 20 for Dist. Pairs, 0 for Subscriber Addr., and 0 for Subscriber Name. Match results shown in the following sections are based on these match weights.

4.4.1.2 Presented Case

The presented case input screen, seen in Figure 4.5, allows the user to enter the known symptoms of the current case, and to search on these symptoms. The interface provides flexibility to allow the user to enter any number of features, and utilizes applicable matching to allow for misspellings or missing words.

The current case shown in Figure 4.5 represents the typical diagnostic case features that may be used to diagnose an outside plant trouble. Other features or combinations of features may be used in different diagnosis situations.

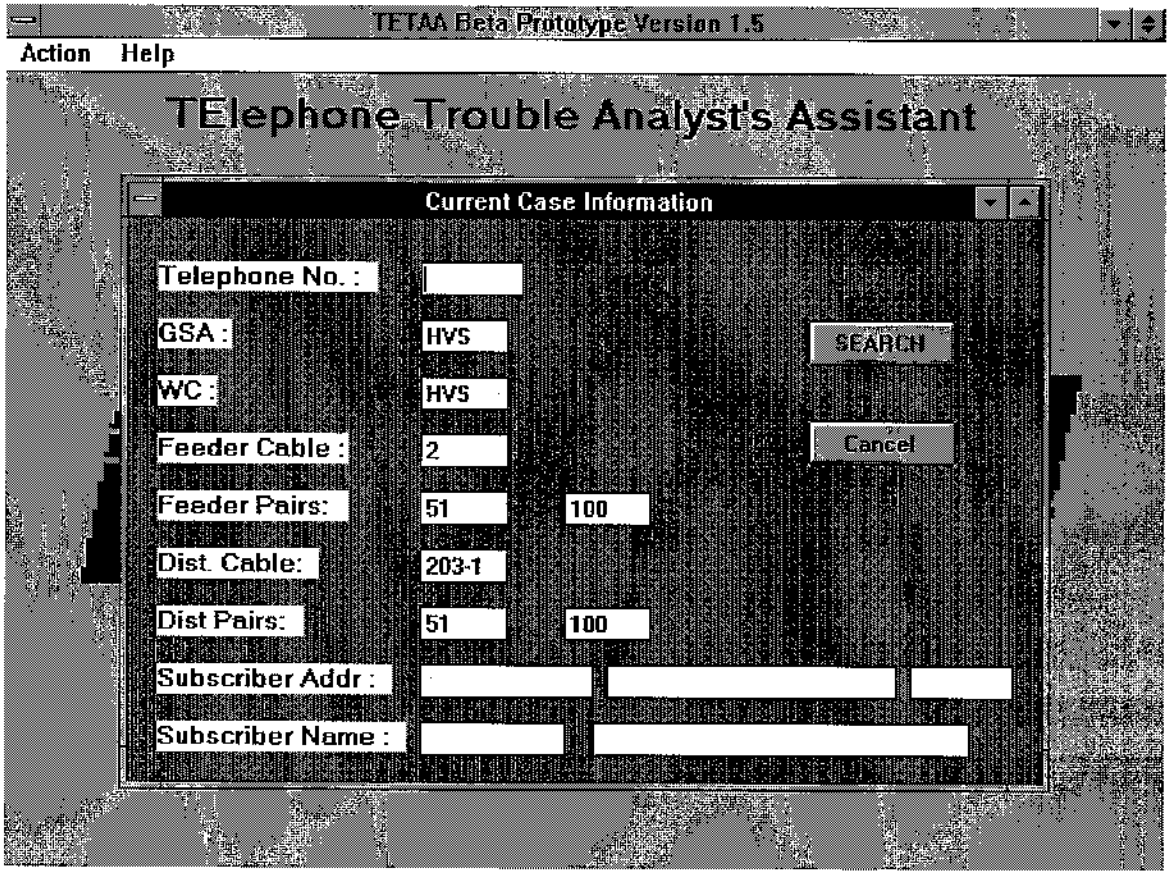


Figure 4.5 TETAA Presented Case Screen

4.4.1.3 Output Summary

The Output Summary screen, shown in Figure 4.6, provides the primary tool for diagnosis in the TETAA system. After the search, initiated from the Presented Case screen, has been completed, the Output Summary screen is displayed containing the 15 cases that most closely match the parameters entered by the user their corresponding diagnostic information. Each matching case listed has a relative score indicating the degree of matching between it and the presented case (see section 3.2.2.5). A screen containing complete details for each of the retrieved cases is made available upon selection of a specific case. The user is also provided with the option of returning to the Presented Case screen to enter new values for the diagnostic case features of the presented case.

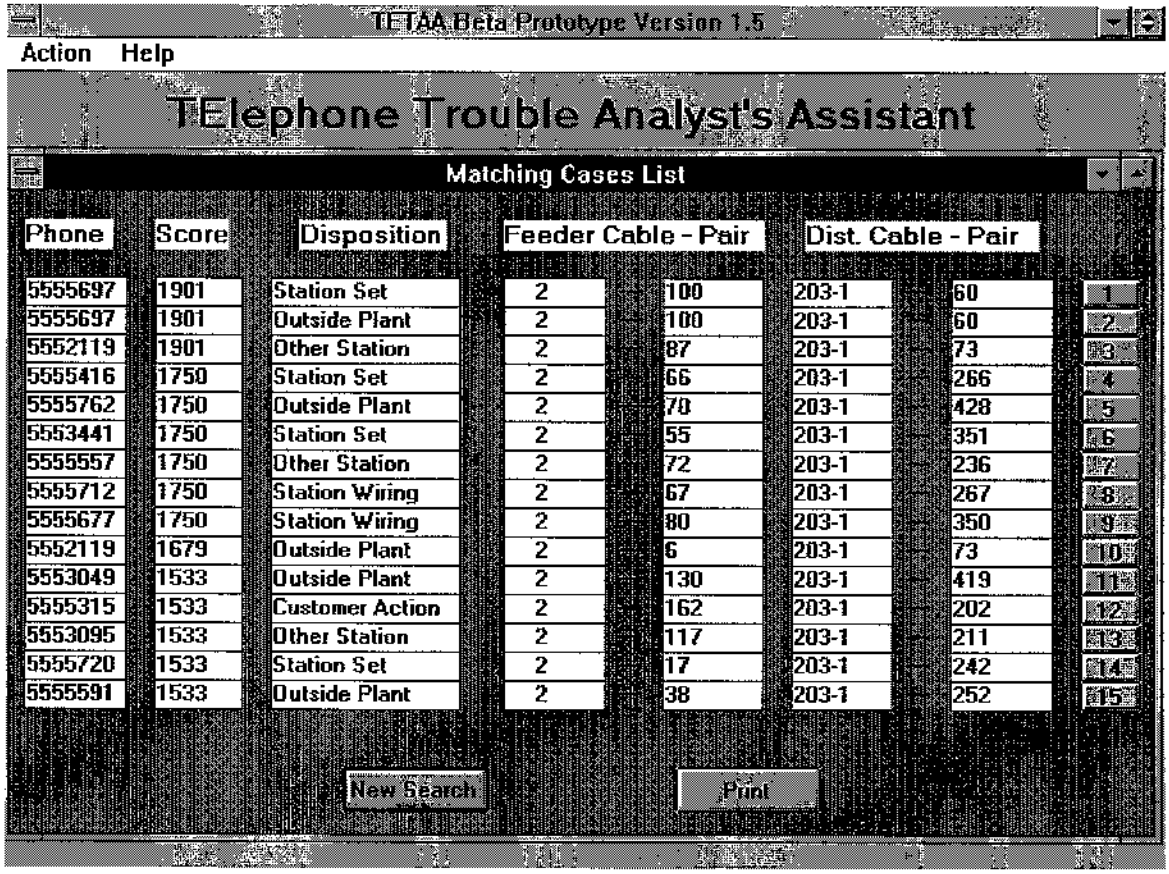


Figure 4.6 TETAA Output Summary Screen

4.4.1.4 Output Detail

The Output Detail screen, shown in Figure 4.7, displays the detailed information describing a specific case. The Output Detail screen is accessed by selecting one of the cases displayed in the matching cases list on the Output Summary screen.

An important field used in diagnosing the current trouble is the "Trouble Found Remarks" field, which is entered by the repair personnel, and describes the diagnosis of the trouble that was found to closely match the current trouble. This field may provide the diagnosis for the current trouble.

The Output Detail screen gives the user the option of returning to the matching cases list to select another case on the list or begin a new search.

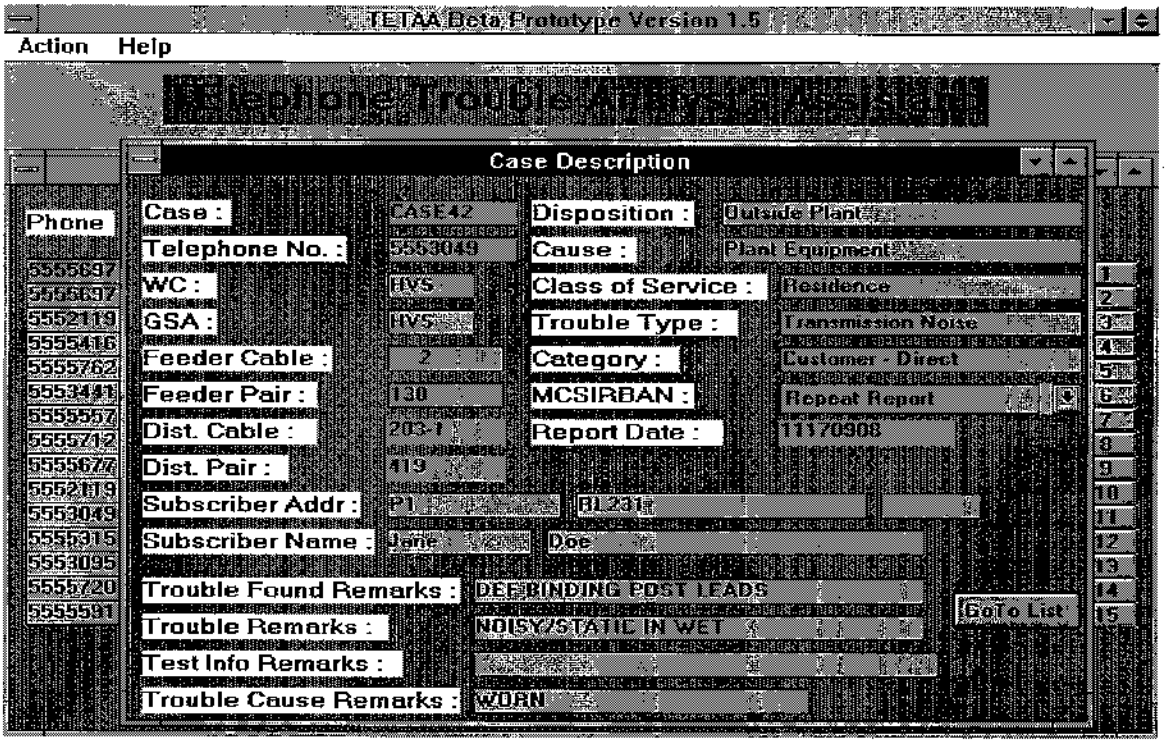


Figure 4.7 TETAA Output Detail Screen

4.4.1.5 Cable Trouble Search

The Cable Trouble Search screen, shown in Figure 4.8, allows the user to initiate a search that will return results describing the number of troubles on specific cable pair ranges. This functionality is used to identify faulty plant by examining recent trends in cable troubles. To accomplish this, the user identifies a specific feeder or distribution cable, the number of pairs to be reported on, and the range of the cable pair groupings.

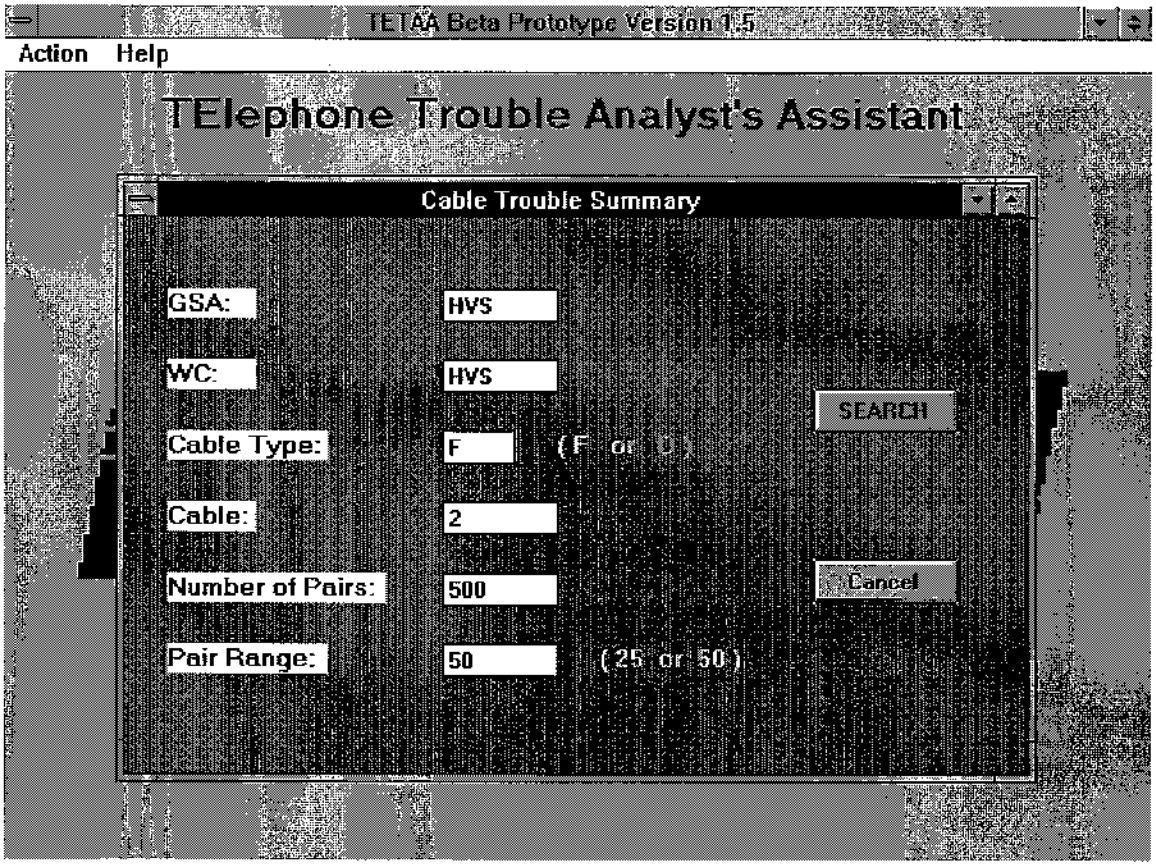


Figure 4.8 TETAA Cable Trouble Search Screen

4.4.1.6 Cable Trouble Search Results

The Cable Trouble Search Results screen, shown in Figure 4.9, provides the user with the results of the cable trouble search that was defined on the Cable Trouble Search screen. The results inform the user of the number of troubles that have recently been

reported and resolved on pair ranges for the specified cable. This information allows the user to identify trends in cable faults, and helps to pinpoint faulty cables that may need to be repaired or replaced. This feature of the system helps to eliminate troubles that may have occurred if the faulty cable has not been found and repaired.

This screen gives the user the option of printing the trouble report, or returning to the Cable Trouble Search screen.

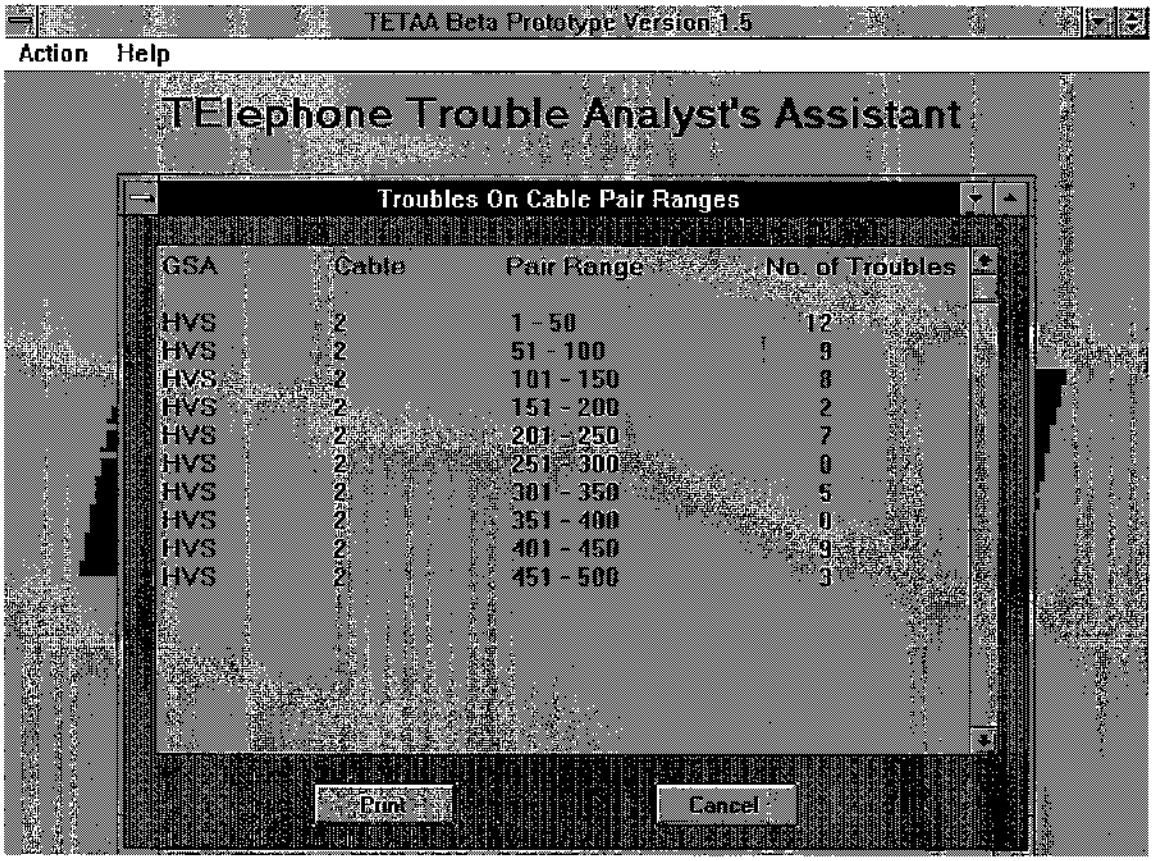


Figure 4.9 TETAA Cable Trouble Summary Screen

4.4.1.7 Save Search Results

The Save Search Results screen, shown in Figure 4.10, allows the user to save search results for future reference, as well as record any comments about the results that

they may want to save. In testing of the prototype, the users were encouraged to record any comments on the usefulness of the system in resolving the current trouble.

The user is presented with this screen after each search, and is given the option of saving the results, initiating a new search, or returning to the initial TETAA screen.

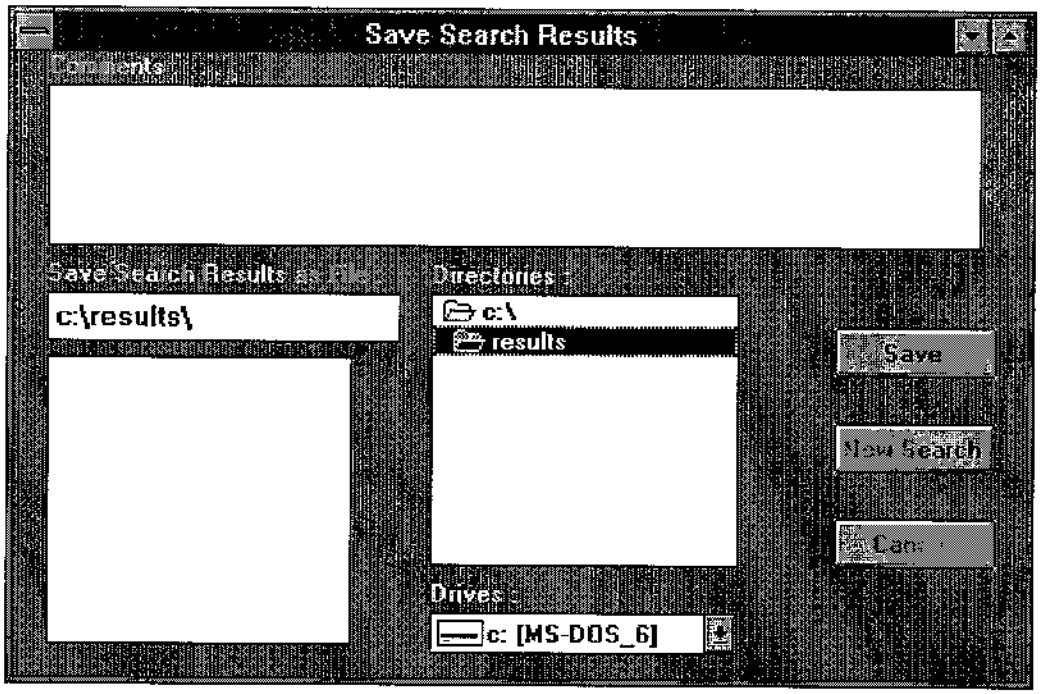


Figure 4.10 TETAA Save Search Results and Comments Screen

4.4.2 The Diagnosis API

The diagnosis API provides the mechanism to allow the GUI to communicate with ART-IM. The API takes the input received from the user through the GUI, and triggers knowledge manipulation in ART-IM in the form of rules or user defined functions (see section 4.4.3). This allows matching on presented case features, and the update of feature match weights. The API then makes the results of the knowledge manipulation available to the GUI to complete the diagnostic process.

The diagnosis API is implemented as a Dynamic Link Library (DLL) using Microsoft C++ and contains a number of functions that can be called from the Visual

Basic GUI code. A list of the API functions and their description is presented in Table 4.1.

The C code that is used to generate the DLL is contained in the CBRADD.C file while the DLL itself is contained in a file called CBRCALL.DLL. Under Microsoft Windows, the DLL file should be placed in the Windows System directory, where it will automatically be found by the TETAA application.

Table 4.1 Diagnosis API Functions and Descriptions

Function	Description
ArtUpdate	Updates the match and mismatch weights of the diagnostic case features by using the ART-IM function *a_modify_schema_value* and asserting the ART-IM fact *start-dwr.*
Artwtinfo	Retrieves the current match weights of the diagnostic case features for display from the match weight storage schema using the ART-IM function *a_get_schema_value.*
TCUpdate	Initiates a trouble search by specifying search parameters using the ART-IM function *a_modify _schema_value* and asserting the ART-IM fact *start-cbr.*
GetCase	Retrieves summary case information for a specific case using the ART-IM function *a_get_schema_value.*
GetCInfo	Retrieves detailed case information for a specific case using the ART-IM function *a_get_schema_value.*
GetMnum	Retrieves the case number key for a specific matched case using the ART-IM function *a_get_schema_value.*
GetMCNum	Retrieves individual values contained in the multi-purpose MCSIRBAN field for a specific case by calling the ART-IM user defined function *split-up-mc.*
GetMC	Retrieves description information for the MCSIRBAN field values for a specific case by using the ART-IM function *a_get_schema_value.*
SUMUpdate	Specifies the values of the regular diagnostic case features for a cable trouble summary search by using the ART-IM function *a_modify_schema_value.*
RNGSrch	Initiates a cable pair range search by specifying the pair range and calling the ART-IM user defined function *range-search.*

4.4.3 Development in ART-IM

In the ART-IM environment, there are three files that are essential for the initial startup and operation of the TETAA prototype. These files are FDCBR.ART, MWEIGHTS.ART, and EXPSCH.ART. The primary ART-IM file is FDCBR.ART which contains the rule base and user defined functions which provide the knowledge manipulation within ART-IM. The MWEIGHTS.ART file contains a schema which is used to store the current match-weight values of the diagnostic case features, and some state information about the values, such as whether they have been changed in the GUI since the last reset. The EXPSCH.ART file contains the schemas which provide the description knowledge for the explanation facility.

Once these files have been loaded on initial startup of the system, a file called SCHEMA.ART contains all of the necessary schema information, including all case schemas and explanation schemas, so that only this file and FDCBR.ART need to be loaded in the future. A sample trouble schema from the SCHEMA.ART file is shown in Figure 4.11.

```
(DEFSCHEMA case202
 (INSTANCE-OF TROUBLE)
 (ACC-INFO " ")
 (ADDRESS "P202  RL290")
 (APP-DATE "01251200")
 (APPNTMT "Y Y   ")
 (AUX-LINE "0000000")
 (BATCH-DATE "930125")
 (CABCMT-REM " ")
 (CABLE-LOC " ")
 (CABLE-NUM "  ")
 (CAI "Y")
 (CARRIER "Y ")
 (CATEGORY 0)
 (CAUSE "5")
 (CAUSE-SUB " ")
 (CLASS-OF-SERVICE "03")
 (CLOSE-DATE "01251126")
 (CLOSED-BY "1")
 (CO-EQUIP 101319)
 (CO-GROUP "P202  RL290 ")
 (COS-SUB " ")
 (DIS-DATE "0125")
 (DIS-TIME "0823")
 (DISP-SUB " ")
 (DISPOSITION "09")
 (DIST "   C1")
 (DPAIR "106")
 (DPAIR25 "101")
 (DPAIR50 "101")
 (EQUIP-TYP "4")
 (EST-DATE "0000")
 (EST-TIME "0000")
 (EXTEN " ")
 (FCI " ")
 (FEEDER " DM11")
 (GSA "HVS")
 (MCSIRBAN "NNN YYYNNY  ")
 (NET-NUM "   ")
 (NTR-IND "N")
 (NUMBER-KEY 202)
 (OP-INIT "ABC")
 (PAIR "17")
 (PAIR25 "1")
 (PAIR50 "1")
 (PHONE-REM "  ")
 (PRITY-IND "2")
 (REC-BY "9")
 (REC-DATE "01250616")
 (REF-DATE "0000")
 (REF-IND "N")
 (REF-ORG-IND "     ")
 (REF-TIME "0000")
 (RPM-INIT "AJD")
 (SR-CNT "000")
 (SR-IND "Y")
 (SRI "N")
 (STUDY "00")
 (SUB-NAME "KEIRSTEAD F")
 (TC-REM "UNKNOWN       ")
 (TEL-NUMBER "5553069")
 (TESTED-BY "1")
 (TF-REM "FOK ")
 (TI-REM "3RD REPORT ON CARR ")
 (TRBL-REM "NOISY CUTS OUT")
 (TROUBLE-TYPE "1")
 (TSTR-INIT "BCL")
 (TT-CODE "R")
 (TT-DATE "0125")
 (TT-IND " ")
 (TT-TIME "0709")
 (WC "HVS"))
```

Figure 4.11 Trouble Schema from SCHEMA.ART

The file and process interaction described above is illustrated in Figure 4.12. In the diagram, the boxes represent physical ASCII files that store logic or knowledge, the ellipses represent processes, and the arrows represent the flow of data between to and from processes.

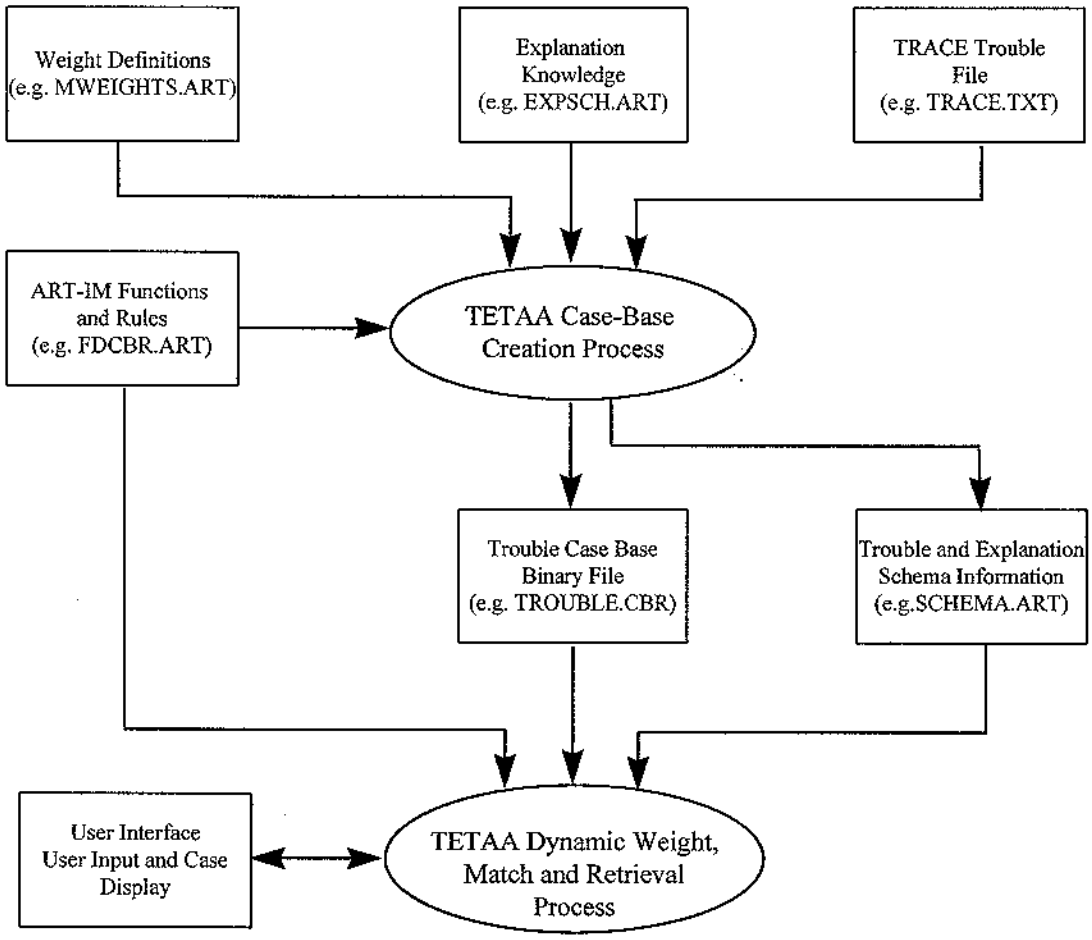


Figure 4.12 TETAA File and Process Interaction

4.4.3.1 The Rule Base

The ART-IM rule base contains 8 rules that manage the three basic processes of the TETAA system. For a description of these rules, see Table 4.2.

The rules BEGIN-CBR and WRITE-TBASE are used to create the initial case base by reading the TRACE.TXT input file, computing derived features, assigning default

match weights, and adding the cases to the case base. Once this is complete, the case base is saved in a binary file called TROUBLE.CBR and all schema information that is currently in the knowledge base is written to the SCHEMA.ART file mentioned in the previous section (see Figure 4.12). These rules are only fired when creating a new case base.

The rules BEGIN-DWR, CHANGE-WEIGHTS-DWR, and BASE-TERM-DWR perform the function of managing the dynamic weight process. These rules are fired if the user has changed the value hierarchy of the diagnostic case features, and updates the attributes of the case base accordingly.

```
(defrule CASE-MATCH-CBR
   (start-cbr t)
=>
   (init-case-base-cbr)
   (send read trouble-case-base)
   (send set trouble-case-base :threshold 0)
   (send set trouble-case-base :max-matches 15)
   (send set trouble-case-base :default-match-type :string)
   (bind ?*match-num* (send match trouble-case-base trouble-case))
   (modify-schema-value match-num num-match (sprintf "%d" ?*match-num*))
   (bind ?*ret-num* 0)
   (assert (get-cases t)))
```

Figure 4.13 CASE-MATCH-CBR Rule

The rules CASE-MATCH-CBR (shown in Figure 4.13), CASE-FIND, and BASE-TERM-CBR provide the match and retrieval functionality of the TETAA system. These rules are fired if a user has initiated a search on a current case.

Table 4.2 ART-IM Rules and Descriptions

Rule	Description
BEGIN-CBR	Creates a new case base from the TRACE input file. Reads the file, determines derived features and adds the cases to the case base. This rule calls the ART-IM user defined functions *init-case-base-cbr, add-slots-dwr, and find-range* and also triggers the *WRITE-TBASE* rule.
WRITE-TBASE	Saves the just created case base , terminates the case base, and dumps all of the schemas currently in the knowledge base to an output file to be used at next start-up.
BEGIN-DWR	Initiates the dynamic weight process. This rule calls the ART-IM user defined function *init-case-base-dwr* and triggers the *CHANGE-WEIGHTS-DWR* rule.
CHANGE-WEIGHTS-DWR	Updates the match and mismatch weights of the diagnostic case features. This rule calls the ART-IM user defined function *set-sum-weights* and triggers the *BASE-TERM-DWR* rule.
BASE-TERM-DWR	Completes the dynamic weight process by terminating the case base.
CASE-MATCH-CBR	Initiates the case-based reasoning match process and performs the match on the current diagnostic case features. This rule calls the ART-IM user defined function *init-case-base-cbr,* and triggers the *CASE-FIND* rule.
CASE-FIND	Retrieves the individual cases specified by the match and their associated features. This rule calls the ART-IM user defined function *get-values,* and triggers the *BASE-TERM-CBR* rule.
BASE-TERM-CBR	Completes the case-based reasoning process by terminating the case base

4.4.3.2 User Defined Functions

The ART-IM user defined functions implemented in the TETAA system are described in Table 4.3. These functions are convenient for implementing knowledge manipulation as they can be called from either the Diagnosis API, individual rules, or another function, thus making them very flexible. These functions are also useful as they

can be used to modularize the sometimes cryptic ART-IM logic and create functions that can be reused across ART-IM implementations.

Table 4.3 ART-IM User Defined Functions and Descriptions

Function	Description
add-slots-dwr	Initializes the diagnostic case feature slot match types and default match and mismatch weights.
cal-mmwt	Calculates the mismatch weight bases on the specified match weight.
init-case-base-dwr	Initializes the case base in the dynamic weight process.
init-case-base-cbr	Initializes the case base in the case-based reasoning process.
split-up-mc	Splits up the MCSIRBAN fields for a specified case into individual values.
look-up-desc	Facilitates explanation by matching on the description schema for a specified case feature.
get-values	Manages all of the description look-ups for a specified case. This function call the look-up-desc function.
print-schema	Prints a specified schema to a specified print stream
find-range	Calculates the derived value to store in the range features when a case is being added to the case base.
set-sum-wghts	Updates the match and mismatch weights of diagnostic case features based on user specification. This function calls the cal-mmwt function.
Range-search	Performs the cable pair range matching on the specified pair range.

4.5 The Experiment

To evaluate the feasibility of the cased-based approach to fault diagnosis, a trial in a real-world situation was proposed. NBTel agreed, and a beta version of the TETAA prototype was installed in the Fredericton test and repair center at 11:00 am, on January 12, 1993. The Fredericton center is responsible for isolating, testing, diagnosing and dispatching all trouble reports for approximately one quarter of the NBTel telecommunications network in New Brunswick.

The tool was evaluated over a two month period by repair personnel experienced in isolating and diagnosing telecommunication network troubles. Experiments were conducted on a number of Geographical Serving Areas (GSA's), which define specific physical network locations, within New Brunswick. The GSA's were selected based on a recent increase in the number of trouble reports in these areas that were proving difficult to diagnose. The selected GSA's included Harvey Station, Boiestown, Doaktown, Chipman and Hoyt, which combined, average approximately 4000 trouble reports per year. This represents about three percent of the total annual trouble rate for the province, which is approximately 140,000 troubles.

The case bases were created using 90 days of accumulated trouble data, taken from the NBTel TRACE system. The tool was evaluated on two Intel 486DX based computers. For performance reasons, separate case bases were created for each GSA, with an average of 200 cases for each test area.

The prototype was used in a number of ways by the evaluators, including (1) to aid in the isolation of the disposition of troubles in the test areas so that line testing could be done, (2) to analyze trouble report trends to aid in fault detection and reduction, specifically in the outside plant component of the network, and (3) to retrieve specific diagnosis information for similar trouble reports to aid in dispatch and repair.

Two mechanisms were implemented to receive feedback and results from the evaluators. A feed back summary form was developed and distributed to the test center personnel (see Appendix II), who were encouraged to provide any comments on the prototype that they felt would be useful. As well, the Save Search Results screen (see section 4.4.1.7) was implemented as part of the tool. This gave the evaluators a chance to save results for future use, but was also useful in collecting comments and results for enhancement and evaluation of the tool.

Three repair centre personnel filled out the feedback questionnaires and Save Search Results comments as they thought necessary. The comments field was used on

most occasions to describe problems encountered during a search, while the feedback forms were mainly used to document suggested enhancements to the system.. In all, 7 feedback forms and 11 documented comments were collected over the two month period.

During the course of the evaluation, six versions of the prototype were installed. Various upgrades were used to fix application bugs, as well as provide new functionality as requested by the evaluators. The case bases were also updated regularly as new trouble history became available for the selected GSA's from the TRACE system.

4.6 Results

Over the two month evaluation period the TETAA tool was used to aid in the isolation and diagnosis of over 150 troubles that were reported in the areas under consideration. It was also used to analyze over 1000 recently resolved troubles. The trouble diagnosis personnel were asked to provide evaluation information in three main areas. The following sections summarize their feedback.

4.6.1 Usefulness

At the beginning of the evaluation, the repair personnel were courteous but skeptical as they were trained in the operation of the tool. They thought the tool was a novel idea, and were open to its use, but were somewhat doubtful as to if it would provide them with any useful information over what they currently had. As they began to use the tool, limited success in aiding in the isolation and diagnosis of the reported troubles was observed. The tool provided useful isolation or diagnosis information for approximately one out of every eight troubles. This level of success was observed throughout the remainder of the evaluation.

As the evaluation continued, the repair personnel began to see potential for the tool in an area that had not been expected or addressed by the developer. The tool had the capability to match on specific diagnostic case features, some of which described specific outside plant facilities, and it could also be used to provide information on the recent

trouble history of these outside plant facilities. This information would allow the repair personnel to identify faulty plant (e.g. cables and pair ranges), thus allowing this plant to be repaired or replaced and reduce trouble report rates through preventative maintenance.

The Cable Trouble Search and results (see sections 4.4.1.5 and 4.4.1.6) were implemented based on this feed back. Although the tool continued to be used for trouble isolation and diagnosis, throughout the second half of the evaluation period the major use of the application was to indicate potentially faulty plant.

4.6.2 Problems

Many technical problems were reported during the evaluation of the TETAA system. Most of these were due to programming errors that were not detected during testing, or problems with the users PC's. The one major recurring problem was the slow performance of the application during matching. This problem was also encountered during testing, and was partially solved by dividing the cases into separate case bases for individual GSA's. Even with relatively small case bases (the average was 200 cases), the matching times still ranged from 30 to 45 seconds. Although not totally unacceptable by the users, this does seems to be a long time for such a small number of cases.

The slow performance of ART-IM for large case bases has also been reported in other work [Taylor, 1994], and is not specific to this implementation. ART-IM uses a memory intensive matching algorithm which maintains a tree-like structure in memory for the case base, which can result in slow performance, especially for machines with limited or slow memory. The resolution of this issue is important to the acceptance and success of the TETAA tool in a commercial environment. Some suggestions for advancement in this area are included in a later section that discusses future work (see section 5.2).

4.6.3 Suggested Improvements

Throughout the course of the evaluation many improvements were suggested for the TETAA system. Many of these were cosmetic in nature, or required minor

development to implement such as additional fields on certain screens, the ability to print and save results, or sort matched cases by a certain feature on the Matching Cases List if they had the same score.

Other suggested improvements were more major in nature and have much more effect on the usefulness of the tool. One of these improvements, whose implementation was described in previous sections, was the addition of the Cable Trouble Search. This addition added an entirely new dimension to the tool, and proved to be the most useful feature in the final evaluation.

The repair personnel also suggested further development to increase the analytical and diagnostic potential of the tool. They observed that the addition of diagnostic case features, such as line test results, would be very beneficial in indicating trouble cases with the same diagnosis. This information is currently not available as part of the trouble profile, but should be added in the future (see section 5.2).

5.0 CONCLUSIONS

5.1 Summary

Research in the field of case-based reasoning has progressed rapidly in recent years. This growth has fueled the adaptation of case-based reasoning for practical applications in commercial domains. This thesis provides justification and a framework for implementing case-based reasoning in commercial fault diagnosis domains.

The telecommunication fault diagnosis categorization model provides an important analytical tool for fault reduction. Its structure is applicable to many other fault diagnosis situations, and can be adapted to categorize faults in any diagnosis domain that utilizes a distributed system and provides a maintenance service (e.g. cable television networks, power distribution networks, computer hardware service).

The TETAA diagnostic aid tool was implemented using Microsoft Visual Basic for the GUI, Microsoft C++ for the diagnosis API, the expert system shell ART-IM, and an IBM Intel 486/DX based PC. This system provides the basis for a diagnostic case-based reasoner but can be evolved to provide improved analytical and diagnostic performance. For example, enhanced explanation, adaptation, and additionally, more applicable diagnostic case features.

The generic design of the TETAA functional and technical architecture makes the software and knowledge base structure readily adaptable to many other case-based reasoning applications, and therefore useful as a template when beginning development of other case-based reasoning systems.

Case-based reasoning is applicable to the fault diagnosis problem, as is indicated by the demonstrated potential of the TETAA prototype during the NBTel evaluation. TETAA was successfully used to analyze over 1000 fault cases in an existing telecommunication network. The feedback on TETAA has been very positive and further

development of the tool will allow for diagnosis results based on more in-depth network information.

The TETAA prototype is a functioning diagnostic case-based reasoning tool that has benefited from feedback collected from a practical evaluation, and a conference presentation [Cookson and Nickerson, 1993]. The refinement of this system based on this feedback has allowed the TETAA tool to evolve to a point where it can function as a useful commercial application, and also serve as a basis for further research in addressing the fault diagnosis problem using case-based reasoning.

5.2 Future Work

Throughout the course of this research, the functionality and performance of the TETAA diagnostic aid tool has consistently been improved based on system testing, evaluation feedback, and speed optimization. This work has allowed the TETAA system to evolve to its current state, but there are still many improvements to be made to allow it to reach its full potential. Following are some suggestions for future work:

1. Continued testing and optimization of the TETAA code.
2. The addition of the trouble found field to the TRS extract and the case base structure. This will allow for complete categorization of the troubles using the trouble diagnosis model.
3. Investigate the design and implementation of adaptation rules to improve diagnostic performance.
4. Add more detailed network information that will contribute diagnostic value to the case base (e.g. the line test results obtained during the trouble diagnosis process).

5. Implement learning by adapting the logic that is used to create a case base to allow the addition of an individual current cases if a satisfactory match is not found.
6. Design and implement an enhanced explanation facility that will determine why a match score was generated, and communicate how each diagnostic case feature contributed to the score.
7. Implement retrieval of case information from a database rather than schemas stored in memory. This will reduce the memory requirements of the TETAA application and provide considerable speed improvements.
8. Further address the speed problem by taking advantage of technical advancements, both in case-based reasoning technology as well as improvements in microprocessor and memory technology.
9. Investigate a distributed implementation for the TETAA system (see section 4.3.1). This will make it more attractive in commercial applications, as well as improve the speed, as this will allow for more powerful UNIX processors to be used.

REFERENCES

Aamodt, A., "A knowledge-Intensive Approach to Problem Solving and Sustained Learning", Ph.D. dissertation, University of Trondheim, Norwegian Institute of Technology, 1991.

Agassi, D.S., "Evaluating Case-Based Reasoning for Heart Failure Diagnosis", Technical Report, Dept. of EECS, MIT. Cambridge, MA, 1990.

Althoff, K.D., "Knowledge Acquisition in the Domain of CNC Machine Centers; the MOLTKE Approach", In *John Boose, Brian Gaines, Jean-Gabriel Ganascia (eds.): EKAW-89; Third European Workshop on Knowledge-Based Systems*, Paris, pp 180-195, 1989.

Ashley, K., *Modeling Legal Arguments: Reasoning With Cases and Hypotheticals.* MIT Press, Bradford Books, Cambridge, 1990.

Bareiss, R., "PROTOS; a Unified Approach to Concept Representation, Classification and Learning", Ph.D. Dissertation,University of Texas at Austin, Dep. of Computer Sciences Technical Report AI88-83, 1988.

Branting, K., "Exploiting the Complementarity of Rules and Precedents with Reciprocity and Fairness", In: *Proceedings from the Case-Based Reasoning Workshop 1991*, Washington DC, Sponsored by DARPA. Morgan Kaufmann, pp 39-50, 1991.

Buchanan, B.G. and Shortliffe, E., *Rule-Based Expert Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project,* Addison-Wesley, Reading MA, 1984.

Console. L.,Portinale, L.,Dupre, D.T. and Torasso, P., "Combining Heuristic and Causal Reasoning in Diagnostic Problem Solving", in *Second Generation Expert Systems,* Springer Verlag, pp 46-68, 1993.

Cookson, R.L. and Nickerson, B.G., "Fault Diagnosis Using Case-Based Reasoning", 5th UNB AI Symposium, Fredericton, Aug. 12-14,, pp.171-181, 1993.

Daube, F., Hayes-Roth, B., "A Case-Based Mechanical Redesign System", *IJCAI-89*, pp.1402-1407, 1989.

De Kleer, J., Mackworth, A.K. and Reiter, R. "Characterizing Diagnosis and Systems", *Artificial Intelligence 56*, pp. 197-222, 1992.

Golding, A.R.and Rosenbloom, P.S., "Improving Accuracy by Combining Rule-Based and Case-Based Reasoning", Mitsubishi Electric Research Labratories, Cambridge MA, 1995.

Goodman, M., "CBR in battle planning", In Proceedings of the Second Workshop on CaseBased Reasoning, Pensacola Beach, FL, US., 1989.

Hammond, K.J., *Case-Based Planning*, Academic Press, SanDiego, 1989.

Hamscher, W., Console, L. and de Kleer, J., *Readings in Model-Based Diagnosis*, Morgan Kauffman, 1992.

Hennessy, D. and Hinkle D., "Applying CaseBased Reasoning to Autoclave Loading", *IEEE Expert*, 7(v), pp.216, 1992.

Hinrichs, T.R., *Problem Solving in Open Worlds*, Lawrence Erlbaum Associates, 1992.

Ignizio, J.P., *Introduction to Expert Systems - The Development and implementation of Rule-Based Expert Systems*, McGraw-Hill, Inc. New Tork, 1991.

Inference Corporation, "Case-Based Reasoning in ART-IM",Version 2.5, Inference Corporation, 550 N Continental Blvd., El Segundo, California, 1991.

Inference Corporation, "ART-IM Function Library Reference",Version 2.5, Inference Corporation, 550 N Continental Blvd., El Segundo, California, 1991.

Inference Corporation, "ART Programming Tutorial: Advanced Topics in ART",Version 3.0, Inference Corporation, 550 N Continental Blvd., El Segundo, California, 1984.

Keane, M., "Where's the Beef? The Absence of Pragmatic Factors in Pragmatic Theories of Analogy" in: *Proc. ECAI-88*, pp. 327-332, 1988.

Kolodner, J., "Maintaining Organization in a Dynamic Long-Term Memory", *Cognitive Science*, Vol.7, pp.243-280, 1983.

Koton, P., "Using Experience in Learning and Problem Solving", Massachusetts Institute of Technology, Laboratory of Computer Science (Ph.D. diss, October 1988). MIT/LCS/TR-441, 1989.

Leibowitz, J.(editor), *Expert System Applications to Telecommunications*, John Wiley & Sons, New York, 1988.

López, B., "Case-Based Learning of Strategic Knowledge", Centre d'Estudis Avançats de Blanes, CSIC, Machine Learning-EWSML-91, pp. 398-411, 1991.

Luger, G.F. and Stubblefield, W.A., *Artificial Intelligence and the Design of Expert Systems*, Benjamin/Cummings Publishing, Redwood City, California, 1989.

Newell, A. and Simon, H, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.,1972.

Oehlmann, R., "Learning Causal Models by Self-Questioning and Experimentation", AAAI-92 Workshop on Communicating Sientific and Technical Knowledge. American Association of Artificial Intelligence, 1992.

Pews, G. and Wess, S., "Combining Case-Based and Model-Based Approaches for Diagnostic Applications in Technical Domains", in *Proc. European Workshop on Case-Based Reasoning 93*, pp.325-328, Kaiserslautern, 1993.

Plaza, E. and López de Mántaras, R., "A Case-Based Apprentice that Learns From Fuzzy Examples", In *Z Ras, M. Zemankova, M. L. Emrich (Eds.) Methodologies for Intelligent System 5*, North Holland, pp 420-427, 1990.

Portinale, P., Torasso, P., Ortalda and C. Giardino, A., "Using Case-Based Reasoning to Focus Model-Based Diagnostic Problem Solving", In *Proc. EWCBR 93*, Kaiserslautern, 1993.

Porter, B. and Bareiss, R., "PROTOS: An Experiment in Knowledge Acquisition for Heuristic Classification Tasks", In:*Proceedings of the First International Meeting on Advances in Learning (IMAL)*, Les Arcs, France, pp. 159-174,1986.

Rabinowitz, H., Flamholz, J., Wolin, E., and Euchner,J., "NYNEX MAX: a telephone trouble screening expert", in *Innovative Applications of Artificial Intelligence 3*, edited by R. Smith and C. Scott, AAAI Press, Menlo Park, CA, 1991, pp. 213-230.

Richter, A.M. and Weiss, S., "Similarity, Uncertainty and Case-Based Reasoning in PATDEX", in *R.S. Boyer (ed.): Automated reasoning, essays in honour of Woody Bledsoe,* Kluwer, pp. 249-265, 1991.

Riesbeck, C.K. and Schank, R.C., *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, New Jersey, 1989.

Rissland, E., "Examples in Legal Reasoning: Legal Hypotheticals",. in: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, Karlsruhe, 1983.

Schank, R.C., *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, Cambridge University Press, New York, 1982.

Schank, R.C., Kass, A. and Riesbeck, C.K., *Inside Case-Based Explanation*, Lawrence Erlbaum Associates, New Jersey, 1994.

Sharma, S. and Sleeman, D., "REFINER; a Case-Based Differential Diagnosis Aide for Knowledge Acquisition and Knowledge Refinement", in: *EWSL 88; Proceedings of the Third European Working Session on Learning*, Pitman, pp 201-210, 1988.

Simoudis, E., Mendall, A. and Miller, P., "Automated Support for Developing Retrieve-and-Propose Systems", *In Proceedings of Artificial Intelligence XI Conference*, Orlando, Florida, 1993

Simpson, R.L., "A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation", Technical Report GIT-ICS-85/18, Georgia Institute of Technology, 1985.

Skalak, C.B and Rissland, E., "Arguments and Cases: An Inevitable Twining", *Artificial Intelligence and Law, An International Journal*, 1(1), pp.3-48, 1992.

Smith, E.E., Adams, N., and Schorr, D.,"Fact Retrieval and the Paradox of Interference", *Cognitive Psychology*, pp. 438-464, 1978.

Stefik, M., Aikins, J.., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F. and Sacerdoti, E. "The Organization of Expert Systems, a Tutorial", Artificial Intelligence, 1982.

Strube, G., "The Role of Cognititve Science in Knowledge Engineering", In: *F. Schmalhofer, G. Strube (eds.), Contemporary knowledge engineering and cognition: First joint workshop, proceedings*, pp. 161-174, 1990.

Sycara, K.P., "Using case-based reasoning for plan adaptation and repair", Proceedings Case-Based Reasoning Workshop, DARPA. Clearwater Beach, Florida. Morgan Kaufmann, pp. 425-434, 1988

Sycara, K.P. and Navinchandra, D., "Integrating Case-Based Reasoning and Qualitative Reasoning in Engineering Design", in *Artificial Intelligence in Design*, Gero, J. (editor), Springer-Verlag, New York, 1989.

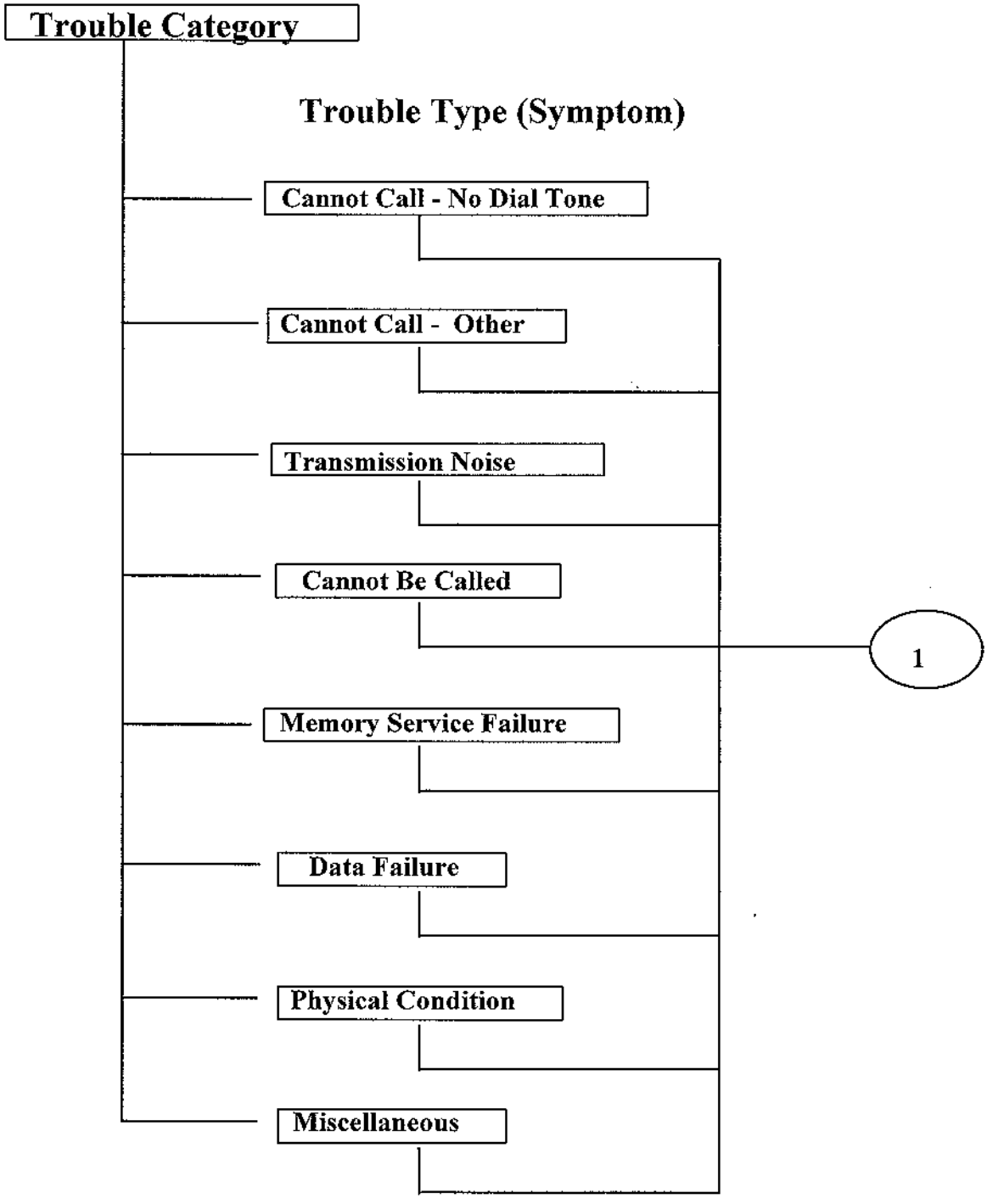
Taylor, K.B., "The Use of a Knowledge-Based System in the Management of Intensive Care Patients", University of New Brunswick, Electrical Engineering Masters Thesis, 1994.

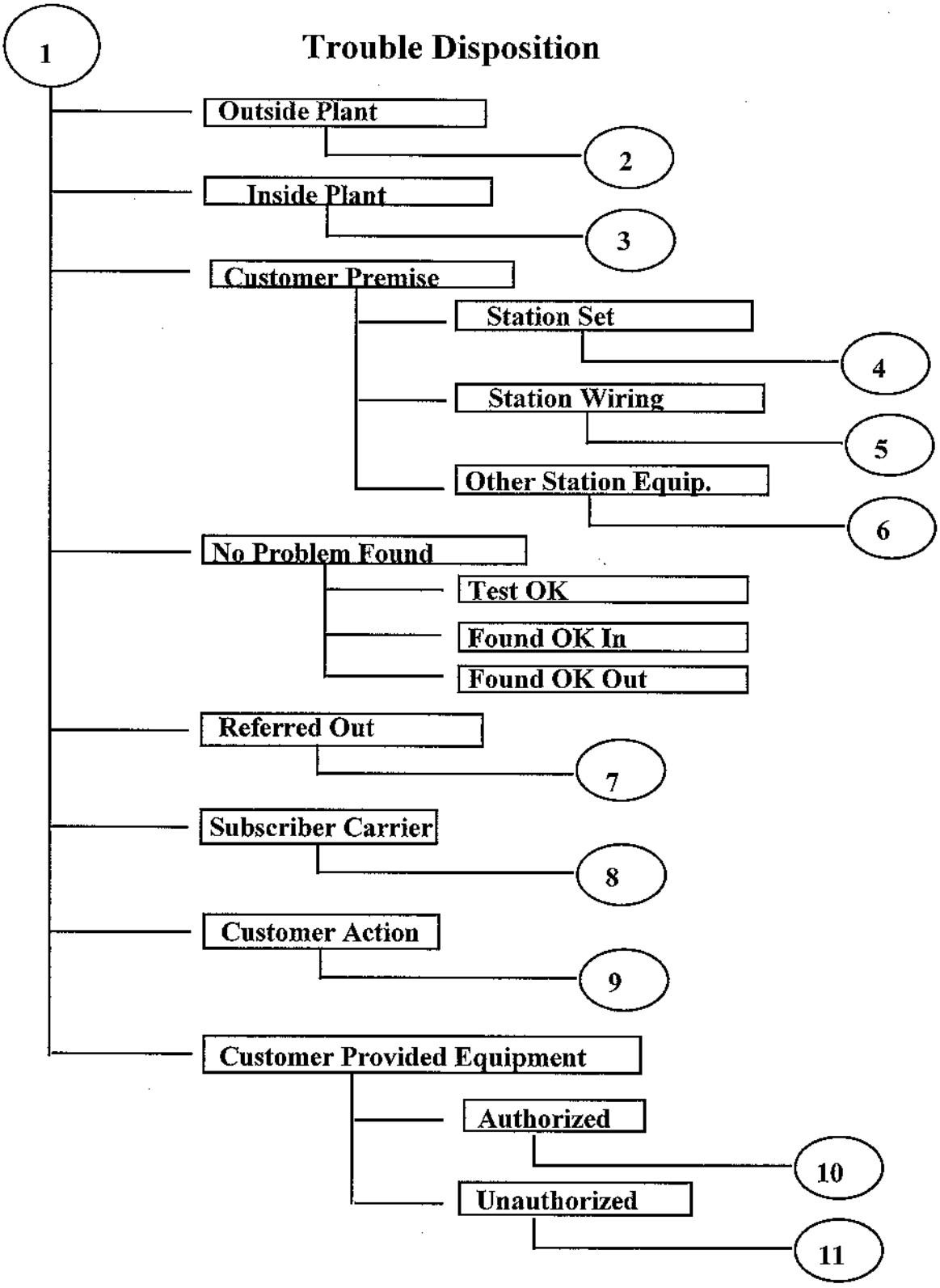
Tierney, P., "Developing a Strategic Platform for Searching and Retrieving Corporate Knowledge", Inference Corporation, Novato, CA, 1995.

Torasso, P. and Console, L., *Diagnostic Problem Solving*, Van Nostrand Reinhold, New York, 1989.

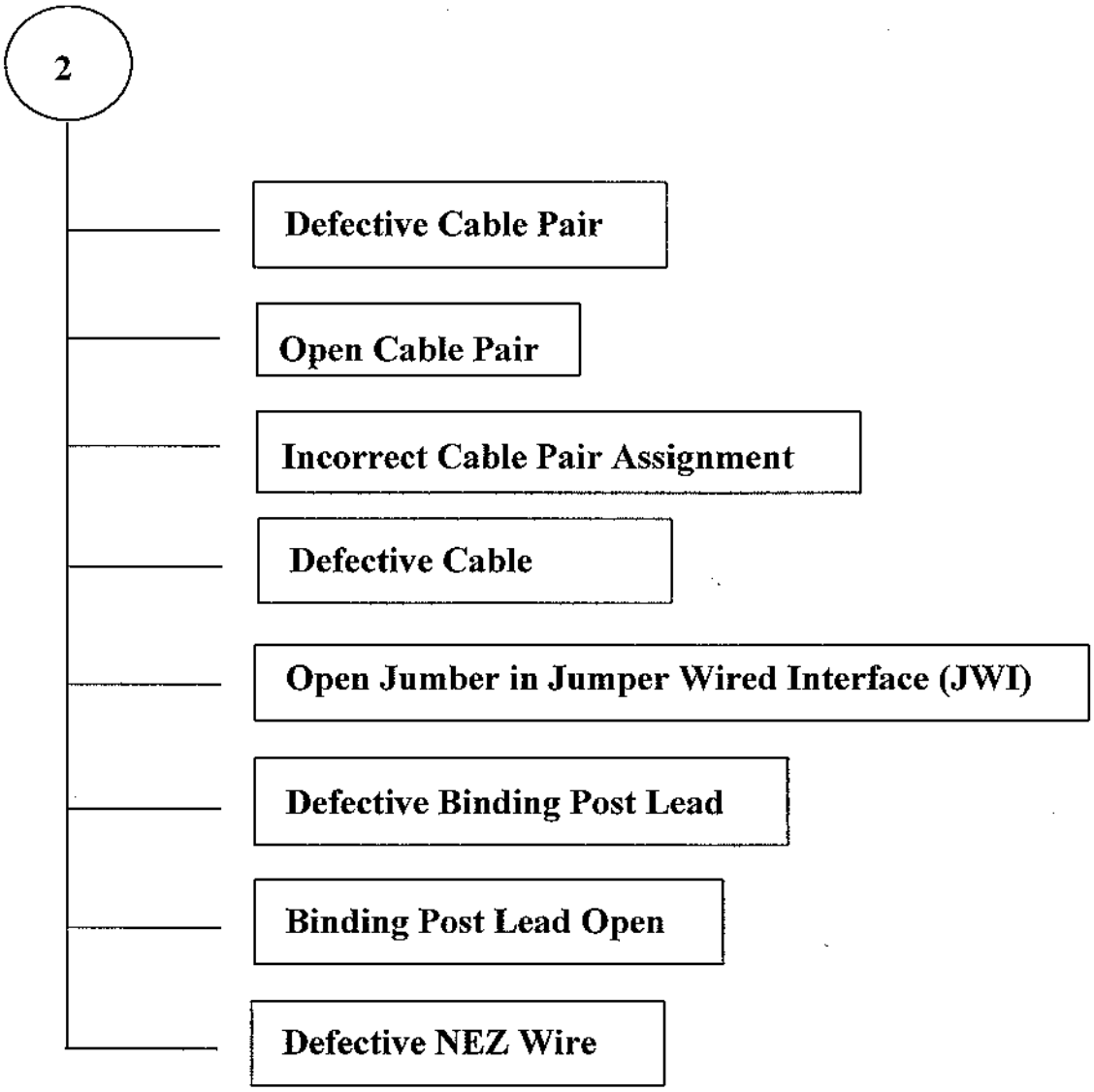
Torasso, P. and Portinale, L., "Combining Experiential Knowledge and Model-Based Reasoning for Diagnostic Problem Solving", University of Torino, Torino Italy, 1995.

APPENDIX I A Trouble Diagnosis Model

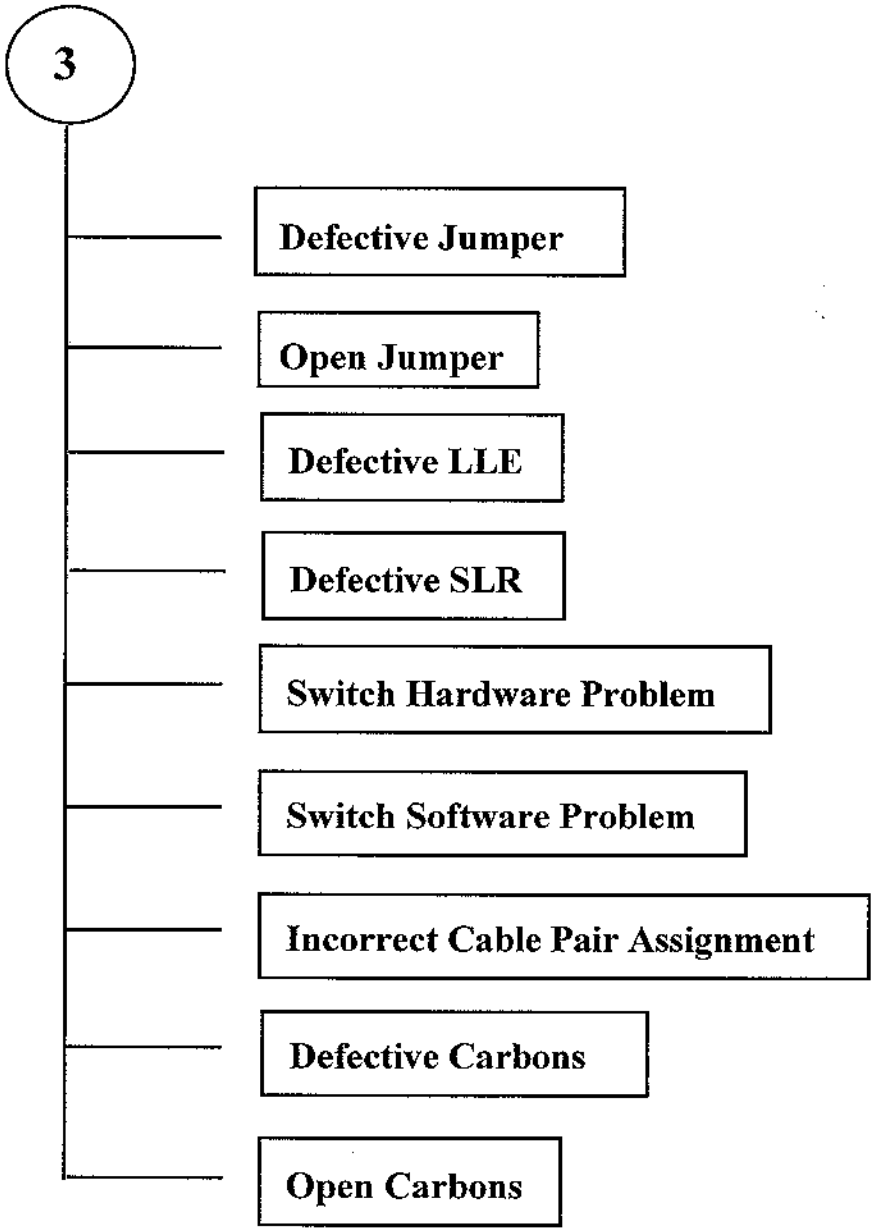



1
Trouble Disposition
Outside Plant
2
Inside Plant
3
Customer Premise
Station Set
4
Station Wiring
5
Other Station Equip.
6
No Problem Found
Test OK
Found OK In
Found OK Out
Referred Out
7
Subscriber Carrier
8
Customer Action
9
Customer Provided Equipment
Authorized
10
Unauthorized
11

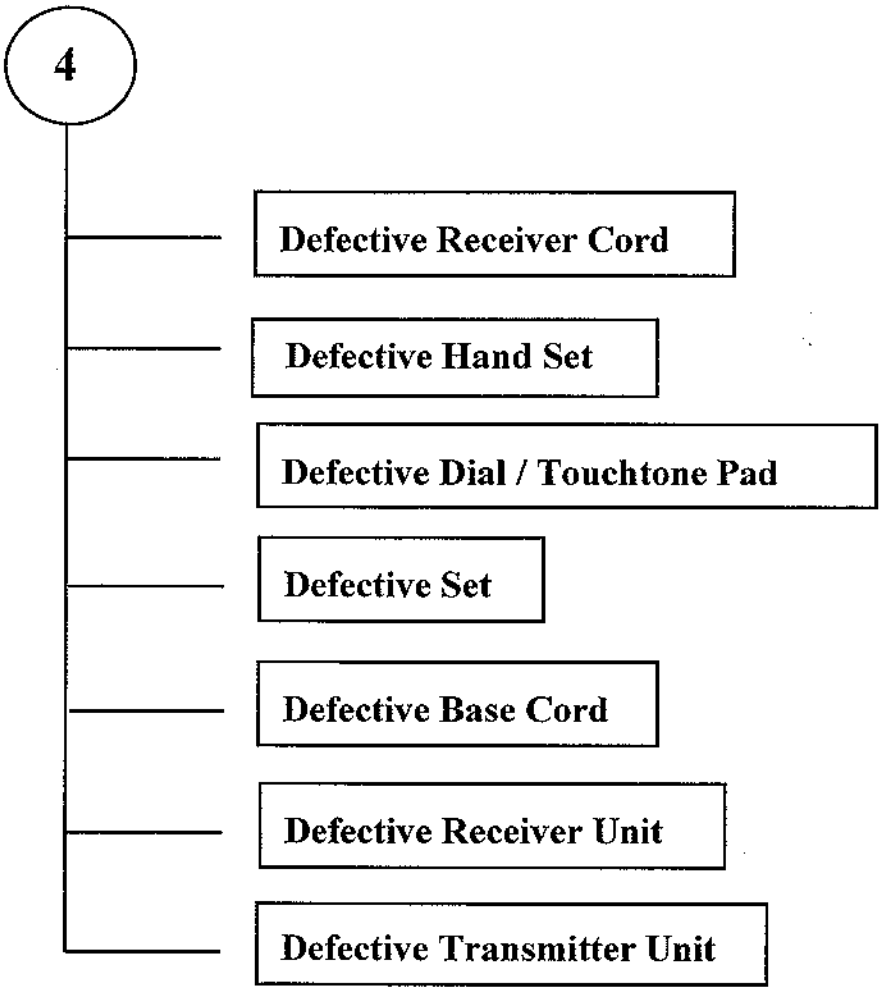
Outside Plant Trouble Found Categories



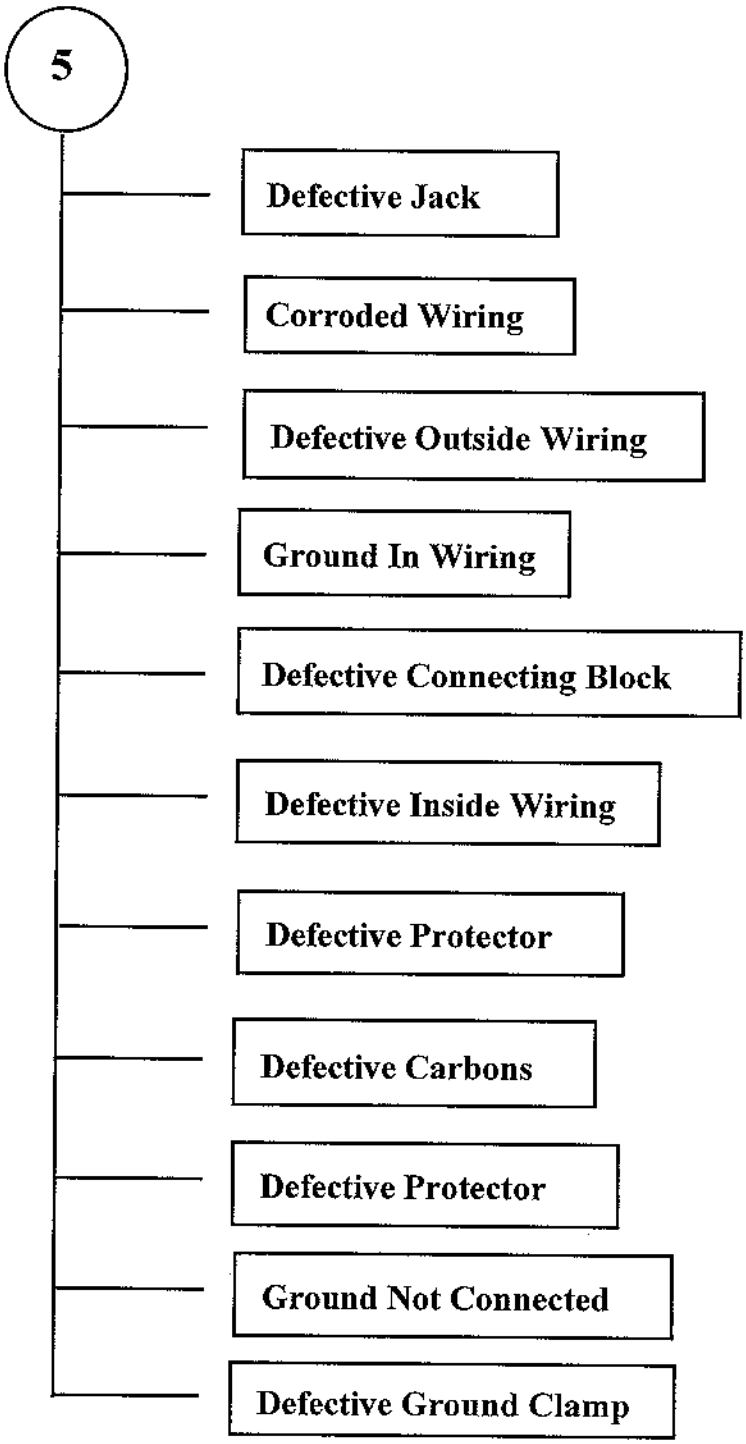
Inside Plant Trouble Found Categories



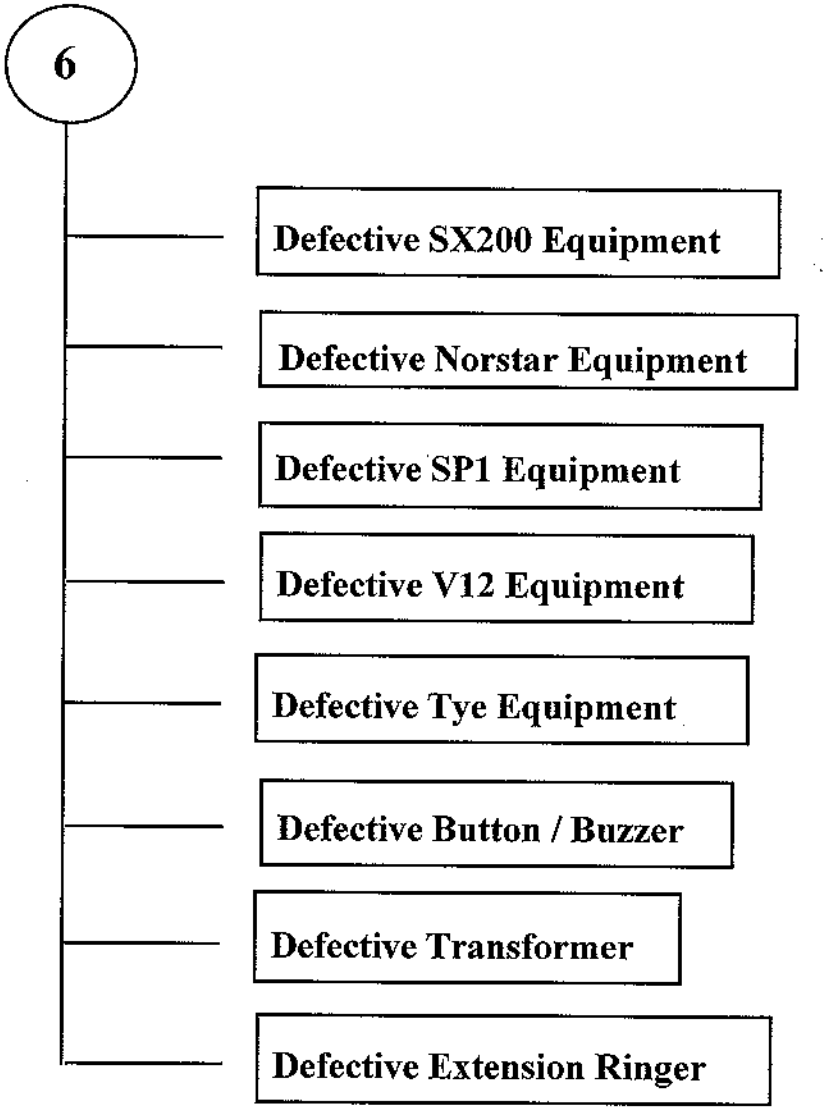
Station Set Trouble Found Categories



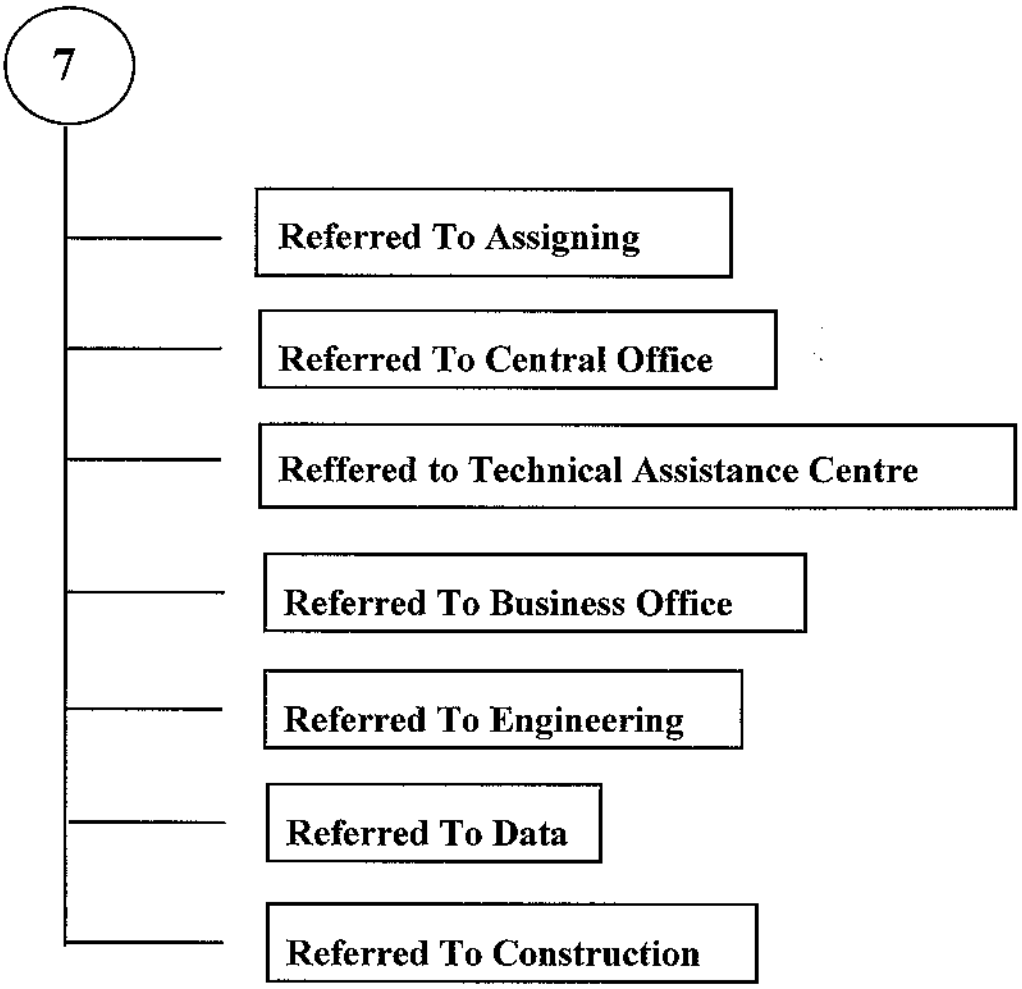
Station Wiring Trouble Found Categories



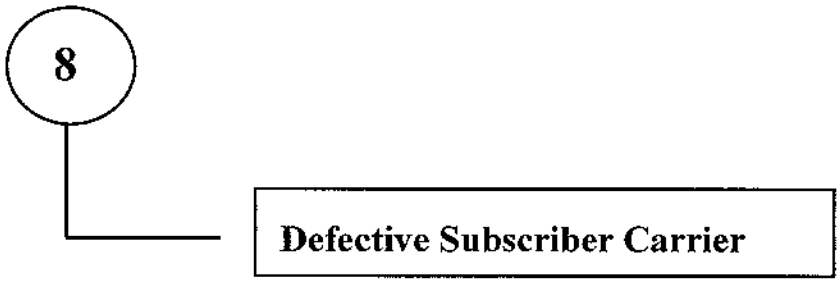
Other Station Equipment Trouble Found Categories



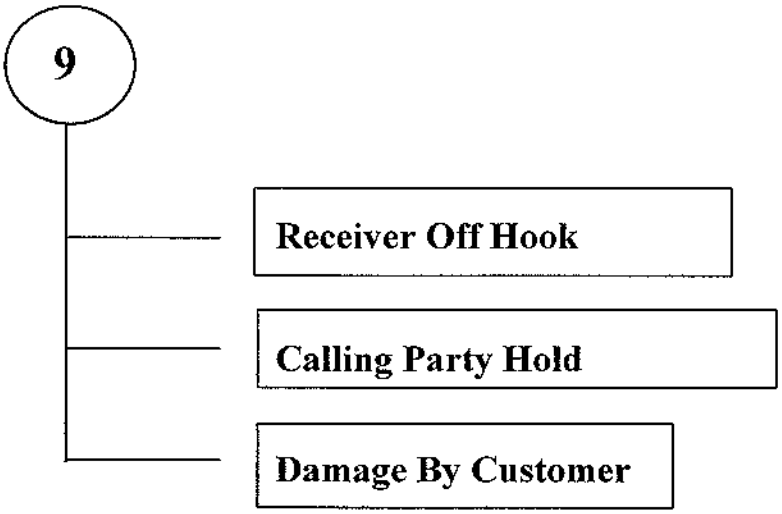
Referred Out Trouble Found Categories



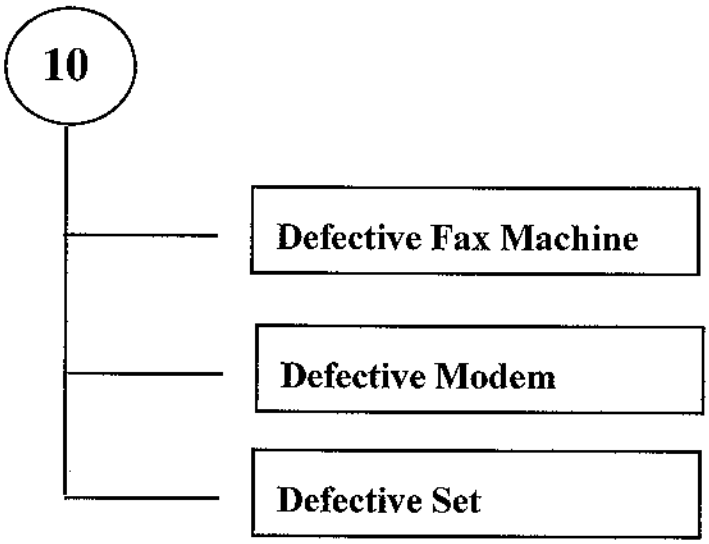
Subscriber Carrier Trouble Found Categories



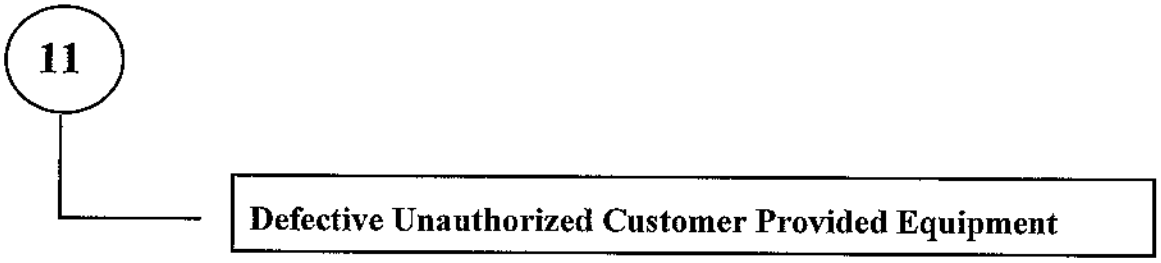
Customer Action Trouble Found Categories



Authorized Customer Provided Equipment Trouble Found Categories



Unauthorized Customer Provided Equipment Trouble Found Categories



APPENDIX II TETAA Feedback Form

Summary of Feedback on TETAA Version 1.x

Performance\Usefulness :

Problems:

Suggested Improvements:

Further Comments:

APPENDIX III TETAA Explanation Facility Look-Up Schemas

```
(DEFSCHEMA look-up
 (field-name)
 (field-code)
 (desc))

(DEFSCHEMA psc-fld1
 (instance-of look-up)
 (field-name "psc")
 (field-code "1")
 (desc "Newcastle"))

(DEFSCHEMA psc-fld2
 (instance-of look-up)
 (field-name "psc")
 (field-code "2")
 (desc "Moncton"))

(DEFSCHEMA psc-fld3
 (instance-of look-up)
 (field-name "psc")
 (field-code "3")
 (desc "Saint John"))

(DEFSCHEMA psc-fld4
 (instance-of look-up)
 (field-name "psc")
 (field-code "4")
 (desc "Fredericton"))

(DEFSCHEMA cat-fld0
 (instance-of look-up)
 (field-name "cat")
 (field-code 0)
 (desc "Customer - Direct"))

(DEFSCHEMA cat-fld1
 (instance-of look-up)
 (field-name "cat")
 (field-code 1)
 (desc "Customer - Relayed"))
```

```
(DEFSCHEMA cat-fld2
 (instance-of look-up)
 (field-name "cat")
 (field-code 2)
 (desc "Customer - RAC"))

(DEFSCHEMA cat-fld3
 (instance-of look-up)
 (field-name "cat")
 (field-code 3)
 (desc "Employee"))

(DEFSCHEMA cat-fld4
 (instance-of look-up)
 (field-name "cat")
 (field-code 4)
 (desc "Referred In"))

(DEFSCHEMA cat-fld5
 (instance-of look-up)
 (field-name "cat")
 (field-code 5)
 (desc "Customer - Excluded"))

(DEFSCHEMA cat-fld6
 (instance-of look-up)
 (field-name "cat")
 (field-code 6)
 (desc "LIT"))

(DEFSCHEMA cat-fld7
 (instance-of look-up)
 (field-name "cat")
 (field-code 7)
 (desc "LIT Cancel"))

(DEFSCHEMA mcsirban-fld1
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "1")
 (desc "Work Comment"))
```

```
(DEFSCHEMA mcsirban-fld2
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "2")
 (desc "Appointment Comment"))

(DEFSCHEMA mcsirban-fld3
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "3")
 (desc "Subsequent Report"))

(DEFSCHEMA mcsirban-fld4
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "4")
 (desc "I Report"))

(DEFSCHEMA mcsirban-fld5
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "5")
 (desc "Repeat Report"))

(DEFSCHEMA mcsirban-fld6
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "6")
 (desc "Displayed Outside"))

(DEFSCHEMA mcsirban-fld7
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "7")
 (desc "Out of Service"))

(DEFSCHEMA mcsirban-fld8
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "8")
 (desc "Repaired Phone Centre"))
```

```
(DEFSCHEMA mcsirban-fld9
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "9")
 (desc "No Access"))

(DEFSCHEMA mcsirban-fld0
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "0")
 (desc "Missed Appointment"))

(DEFSCHEMA mcsirban-flda
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "A")
 (desc "Received After 5 PM"))

(DEFSCHEMA mcsirban-fldc
 (instance-of look-up)
 (field-name "mcsirban")
 (field-code "C")
 (desc "Carried Over"))

(DEFSCHEMA class-fld0
 (instance-of look-up)
 (field-name "class")
 (field-code "00")
 (desc "Spare"))

(DEFSCHEMA class-fld1
 (instance-of look-up)
 (field-name "class")
 (field-code "01")
 (desc "Spare"))

(DEFSCHEMA class-fld2
 (instance-of look-up)
 (field-name "class")
 (field-code "02")
 (desc "WATS"))
```

```
(DEFSCHEMA class-fld3
 (instance-of look-up)
 (field-name "class")
 (field-code "03")
 (desc "Residence"))

(DEFSCHEMA class-fld4
 (instance-of look-up)
 (field-name "class")
 (field-code "04")
 (desc "Business"))

(DEFSCHEMA class-fld5
 (instance-of look-up)
 (field-name "class")
 (field-code "05")
 (desc "PBX"))

(DEFSCHEMA class-fld6
 (instance-of look-up)
 (field-name "class")
 (field-code "06")
 (desc "Centrex"))

(DEFSCHEMA class-fld7
 (instance-of look-up)
 (field-name "class")
 (field-code "07")
 (desc "Coin"))

(DEFSCHEMA class-fld8
 (instance-of look-up)
 (field-name "class")
 (field-code "08")
 (desc "Spare"))

(DEFSCHEMA class-fld9
 (instance-of look-up)
 (field-name "class")
 (field-code "09")
 (desc "Spare"))
```

```
(DEFSCHEMA class-fld10
 (instance-of look-up)
 (field-name "class")
 (field-code "10")
 (desc "Spare"))

(DEFSCHEMA class-fld11
 (instance-of look-up)
 (field-name "class")
 (field-code "11")
 (desc "Rural"))

(DEFSCHEMA class-fld12
 (instance-of look-up)
 (field-name "class")
 (field-code "12")
 (desc "Joint Service"))

(DEFSCHEMA class-fld13
 (instance-of look-up)
 (field-name "class")
 (field-code "13")
 (desc "JET"))

(DEFSCHEMA class-fld14
 (instance-of look-up)
 (field-name "class")
 (field-code "14")
 (desc "Unclassified"))

(DEFSCHEMA class-fld15
 (instance-of look-up)
 (field-name "class")
 (field-code "15")
 (desc "Private Line - Telegraph"))

(DEFSCHEMA class-fld16
 (instance-of look-up)
 (field-name "class")
 (field-code "16")
 (desc "Private Line - Telephone"))
```

```
(DEFSCHEMA class-fld17
 (instance-of look-up)
 (field-name "class")
 (field-code "17")
 (desc "4 Party"))

(DEFSCHEMA class-fld18
 (instance-of look-up)
 (field-name "class")
 (field-code "18")
 (desc "Spare"))

(DEFSCHEMA class-fld19
 (instance-of look-up)
 (field-name "class")
 (field-code "19")
 (desc "Spare"))

(DEFSCHEMA type-fld0
 (instance-of look-up)
 (field-name "type")
 (field-code "0")
 (desc "Cannot Call - No Dial Tone"))

(DEFSCHEMA type-fld1
 (instance-of look-up)
 (field-name "type")
 (field-code "1")
 (desc "Cannot Call - Other"))

(DEFSCHEMA type-fld2
 (instance-of look-up)
 (field-name "type")
 (field-code "2")
 (desc "Transmission Noise"))

(DEFSCHEMA type-fld3
 (instance-of look-up)
 (field-name "type")
 (field-code "3")
 (desc "Cannot be Called"))
```

```
(DEFSCHEMA type-fld4
 (instance-of look-up)
 (field-name "type")
 (field-code "4")
 (desc "Memory Service Failure"))

(DEFSCHEMA type-fld5
 (instance-of look-up)
 (field-name "type")
 (field-code "5")
 (desc "Data Failure"))

(DEFSCHEMA type-fld6
 (instance-of look-up)
 (field-name "type")
 (field-code "6")
 (desc "Physical Condition"))

(DEFSCHEMA type-fld7
 (instance-of look-up)
 (field-name "type")
 (field-code "7")
 (desc "Miscellaneous"))

(DEFSCHEMA disp-fld0
 (instance-of look-up)
 (field-name "disp")
 (field-code "00")
 (desc "Referred Out"))

(DEFSCHEMA disp-fld1
 (instance-of look-up)
 (field-name "disp")
 (field-code "01")
 (desc "Station Set"))

(DEFSCHEMA disp-fld2
 (instance-of look-up)
 (field-name "disp")
 (field-code "02")
 (desc "Other Station Equipment"))
```

```
(DEFSCHEMA disp-fld3
  (instance-of look-up)
  (field-name "disp")
  (field-code "03")
  (desc "Station Wiring"))

(DEFSCHEMA disp-fld4
  (instance-of look-up)
  (field-name "disp")
  (field-code "04")
  (desc "Outside Plant"))

(DEFSCHEMA disp-fld5
  (instance-of look-up)
  (field-name "disp")
  (field-code "05")
  (desc "Central Office"))

(DEFSCHEMA disp-fld6
  (instance-of look-up)
  (field-name "disp")
  (field-code "06")
  (desc "Customer Action"))

(DEFSCHEMA disp-fld7
  (instance-of look-up)
  (field-name "disp")
  (field-code "07")
  (desc "Test OK"))

(DEFSCHEMA disp-fld8
  (instance-of look-up)
  (field-name "disp")
  (field-code "08")
  (desc "Found OK - In"))

(DEFSCHEMA disp-fld9
  (instance-of look-up)
  (field-name "disp")
  (field-code "09")
  (desc "Found OK - Out"))
```

```
(DEFSCHEMA disp-fld10
 (instance-of look-up)
 (field-name "disp")
 (field-code "10")
 (desc "Subscriber Carrier"))

(DEFSCHEMA disp-fld11
 (instance-of look-up)
 (field-name "disp")
 (field-code "11")
 (desc "Customer Provided Equipment - Authorized"))

(DEFSCHEMA disp-fld12
 (instance-of look-up)
 (field-name "disp")
 (field-code "12")
 (desc "Customer Provided Equipment - Unauthorized"))


(DEFSCHEMA cause-fld0
 (instance-of look-up)
 (field-name "cause")
 (field-code "0")
 (desc "Man Made - Telephone Company"))

(DEFSCHEMA cause-fld1
 (instance-of look-up)
 (field-name "cause")
 (field-code "1")
 (desc "Man Made Other"))

(DEFSCHEMA cause-fld2
 (instance-of look-up)
 (field-name "cause")
 (field-code "2")
 (desc "Plant Equipment"))

(DEFSCHEMA cause-fld3
 (instance-of look-up)
 (field-name "cause")
 (field-code "3")
 (desc "Weather"))
```

```
(DEFSCHEMA cause-fld4
 (instance-of look-up)
 (field-name "cause")
 (field-code "4")
 (desc "Other"))

(DEFSCHEMA cause-fld5
 (instance-of look-up)
 (field-name "cause")
 (field-code "5")
 (desc "Unknown"))

(DEFSCHEMA time-fld0
 (instance-of look-up)
 (field-name "time")
 (field-code "0")
 (desc "Under 1/2 Hour"))

(DEFSCHEMA time-fld1
 (instance-of look-up)
 (field-name "time")
 (field-code "1")
 (desc "1/2 - 1 Hour"))

(DEFSCHEMA time-fld2
 (instance-of look-up)
 (field-name "time")
 (field-code "2")
 (desc "1 - 2 Hours"))

(DEFSCHEMA time-fld3
 (instance-of look-up)
 (field-name "time")
 (field-code "3")
 (desc "2 - 4 Hours"))

(DEFSCHEMA time-fld4
 (instance-of look-up)
 (field-name "time")
 (field-code "4")
 (desc "4 - 8 Hours"))
```

```
(DEFSCHEMA time-fld5
  (instance-of look-up)
  (field-name "time")
  (field-code "5")
  (desc "8 - 12 Hours"))

(DEFSCHEMA time-fld6
  (instance-of look-up)
  (field-name "time")
  (field-code "6")
  (desc "12 - 24 Hours"))

(DEFSCHEMA time-fld7
  (instance-of look-up)
  (field-name "time")
  (field-code "7")
  (desc "Over 24 Hours"))

(DEFSCHEMA time-fld8
  (instance-of look-up)
  (field-name "time")
  (field-code "8")
  (desc "Over 48 Hours"))

(DEFSCHEMA time-fld9
  (instance-of look-up)
  (field-name "time")
  (field-code "9")
  (desc "Over 48 Hours No Access"))
```

VITA

Candidates full name:	Robert Laughlin Cookson
Place and date of birth:	Fredericton, New Brunswick September 1, 1966
Permanent Address:	5 Edwards Drive Gondola Point, New Brunswick E2E 4X9
Schools Attended:	Stanley High School Stanley, New Brunswick, 1979-1984
Universities Attended:	University of New Brunswick 1984-1989, BSc.(C.S.)

Publications:

National and International Conference Proceedings

Cookson, R.L. and Nickerson, B.G., "Fault Diagnosis Using Case-Based Reasoning", 5th UNB AI Symposium, Fredericton, Aug. 12-14,, pp.171-181, 1993.