1990

An expert system approach to network performance control

Peter F. Carroll Jr.
Lehigh University

AN EXPERT SYSTEM APPROACH TO NETWORK PERFORMANCE CONTROL

by Peter F. Carroll Jr.

A Dissertation

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

1989

This paper meets the CSEE department's requirements for a thesis as applied toward a Master of Science in Computer Science.

Approved:

K T Denton — 12/15/89
Advising Professor — Date

T. Hareen — 12/15/89
Department Head — Date

ACKNOWLEDGEMENTS

I would like to thank the two people who were of great assistance to me during the course of my thesis development. I am indebted to Dr. Richard Denton, who took many hours of his time to review and advise me concerning the technical content of this paper. I also want to thank my wife Sally Jo, who persevered through the many long hours I took writing this paper, and provided vital editorial support when it was most needed.

CONTENTS

LIST OF FIGURES

List of Examples

ABSTRACT

In the past ten years, the telecommunication world has seen a great push towards the automation of operations that were previously performed by a network operator. This trend is exemplified by ongoing work in the field of network management.

This paper addresses the current state of automation capability for one of the components of a network management system - performance management. By comparing the functions of a desired performance management system to current day implementations, the paper indicates that the performance management system component which necessitates further automation is performance control.

Subsequently, the use of expert system technology is proposed to automate the functions involved with performance control. Through descriptions of expert system theory and present day expert system implementations in the field of data communications, the feasibility of this approach is substantiated. Further elaborations present suggested guidelines for implementing the expert system application, with specific emphasis given to a hybrid knowledge representation approach.

1.0 INTRODUCTION

The increasing use of local and wide area networks has highlighted a greater need for maintaining optimal network operations. As the speed and complexity of these networks increases, so does the need for providing automated facilities to assist in maintaining maximum performance levels. The system that provides these maintenance capabilities is known as a network management system.

Network management systems are responsible for "gathering information on the usage of the network media by network devices, ensuring the correct operation of the network, and providing reports"[1]. Most network management systems are functionally comprised of several distinct components, including the accounting, security, fault, configuration, and performance management elements. The accounting and security components are responsible for maintaining data regarding each user's account, and for ensuring that access to the network is given only to authorized network users, respectively. Because they do not affect the run-time operation of the network, these components could be considered secondary elements in the network management hierarchy.

The other three components, however, are directly related to the run-time control of the system. The configuration management system provides facilities that allow for monitoring and adjusting system state variables, which effectively dictate the physical and logical configuration of the network. Fault management involves the detection and correction of faulty devices in the network. Finally, performance management provides facilities that monitor and control the run-time performance of the system. Of these three primary functions, performance management could be considered the most important because it is the driving force behind the other two. In other words, it dictates the direction of the network configuration for configuration management, and also aids in the detection of faults for fault management.

1.0 INTRODUCTION

The increasing use of local and wide area networks has highlighted a greater need for maintaining optimal network operations. As the speed and complexity of these networks increases, so does the need for providing automated facilities to assist in maintaining maximum performance levels. The system that provides these maintenance capabilities is known as a network management system.

Network management systems are responsible for "gathering information on the usage of the network media by network devices, ensuring the correct operation of the network, and providing reports"[1]. Most network management systems are functionally comprised of several distinct components, including the accounting, security, fault, configuration, and performance management elements. The accounting and security components are responsible for maintaining data regarding each user's account, and for ensuring that access to the network is given only to authorized network users, respectively. Because they do not affect the run-time operation of the network, these components could be considered secondary elements in the network management hierarchy .

The other three components, however, are directly related to the run-time control of the system. The configuration management system provides facilities that allow for monitoring and adjusting system state variables, which effectively dictate the physical and logical configuration of the network. Fault management involves the detection and correction of faulty devices in the network. Finally, performance management provides facilities that monitor and control the run-time performance of the system. Of these three primary functions, performance management could be considered the most important because it is the driving force behind the other two. In other words, it dictates the direction of the network configuration for configuration management, and also aids in the detection of faults for fault management.

In general, present day, first generation performance management systems provide very adequate monitoring and reporting capabilities. However, they are weak in the automatic detection and correction of performance bottlenecks. Because of this weakness, there is a great reliance on the human operator to perform some of the functions required for network control. If the human operator could be replaced by, or augmented by, an intelligent computer interface, many of the problems associated with current day performance management systems could be minimized. The following paper will discuss this issue and suggest an approach, utilizing expert system technology, that could be implemented to replace or complement the tasks required of the network operator.

This paper will be organized into four sections. Section one will provide a detailed overview of the performance management function, providing background information that will be used in examples and discussions throughout the remainder of the paper. Concluding section one will be several actual implementation examples that will illustrate how well the current state of the art reflects the theoretical requirements of performance management systems.

Section two will give an introduction into expert system designs that will provide valuable insight into an approach that could be used to enhance the capabilities of current performance management system implementations. Section three will correlate the information from sections one and two to illustrate the applicability of expert systems in the further automation of the performance management function. Finally, section four will provide a summation of the findings of the thesis paper and make recommendations regarding the future direction of performance management systems, specifically as related to expert system technology.

2.0 PERFORMANCE MANAGEMENT

2.1 OVERVIEW

As with many topics in data communications, performance management is defined differently by various people and committees. Muraldihar's article on performance management of MAP/TOP networks defines it as "dealing with the long term evaluation of a network to collect statistical data for analyzing the trend in behavior and tuning of parameters to improve network performance" [2]. The MAP/TOP committee on Network Management defines it as "the set of activities that collect and analyze network measurements and adjust performance by reconfiguring the network" [3]. IEEE 802.1 Committee on Internetworking and Network Management briefly describes it as supporting the evaluation of a LAN by "collection of performance and traffic statistics, retry counts and error counts which are associated with layer management entities" [4].

While performance management definitions may vary, the main goal is the same; to improve the productivity of the network users. There are two types of network users - network clients and network managers. Network clients are those personnel that utilize a network for various purposes such as file sharing, file transfers, remote logins, remote mail, and distributive processing. For this group, a performance management system is important because it provides information necessary to make intelligent decisions regarding application requirements and design. For example, a time critical distributive application requires that the application developer have knowledge of the throughput and response time values of a network. In this case, a performance management system would improve the productivity of the application designer by removing the need to perform independent tests to determine network throughput and response time.

Network managers are responsible for the overall operation of the network. Their job includes adding nodes to local networks, adding new subnetworks to wide area networks, and ensuring that approved users can get access to network resources. In

addition, they are responsible for maintaining optimal network performance - (ie. ensuring the network is performing up to the requirements of the users).

These jobs require that the network manager has a significant amount of information regarding the current operational state of the network. For example, the network manager needs to know which subnetworks are congested in a large network before he decides where a new client's node will be placed. He may even need to know whether it would be more cost effective to add a new subnetwork to the main network, rather than use existing resources, to satisfy the needs of new clients. Performance measures such as link utilization will affect his decision in such cases. All of these decisions require the availability of both current network performance values and long term performance reports - information that should be provided by a performance management system.

In general, then, while the main goal of a performance management system is to increase the productivity of its users, it is apparent that the definition of "productivity" is user dependent. In order to meet the varied demands of both the network client and the network manager, a minimum set of features is required of all performance management systems. These features are listed as follows:

1. To generate performance statistics.
2. To provide report data necessary for long term network analysis, planning and design.
3. To pinpoint performance problem areas/bottlenecks in the network.
4. To assist in tuning network parameters in order to maintain adequate network performance.

These features are provided through a variety of performance management components.

2.2 PERFORMANCE MANAGEMENT COMPONENTS

Effective implementation of a performance management system, to meet the objectives of system users, involves the development of four distinct functional components. These components, illustrated in figure 1 are: data collection, network analysis, external interface, and performance control. The **data collection** element provides all of the necessary front-end information about individual network variables required during the network analysis phase. This information is stored in a network database until it is needed. The **network analysis** element utilizes the information in the network database in order to determine the current values for performance measures of interest in the network. Current values for these performance measures are stored in a performance database until required by the performance control application. The **performance control** component accesses this information via the **external interface** component and utilizes this information in an effort to maintain optimal network performance.

The following sections will provide detailed descriptions of each of the components of a performance management system.

2.2.1 Data Collection

The data collection component includes all of the functions necessary to collect data at the various remote nodes in the network, and pool this data into a centrally located database. This definition implies that there are two stages to the data collection phase; data accumulation and data centralization.

The collection of data during the data accumulation phase is accomplished through the use of software utilities. These utilities monitor certain events as determined by a specific application, and generate statistics based upon the results of these events. enerally, these utilities consist of two types of data tracking mechanisms; counters and timers.

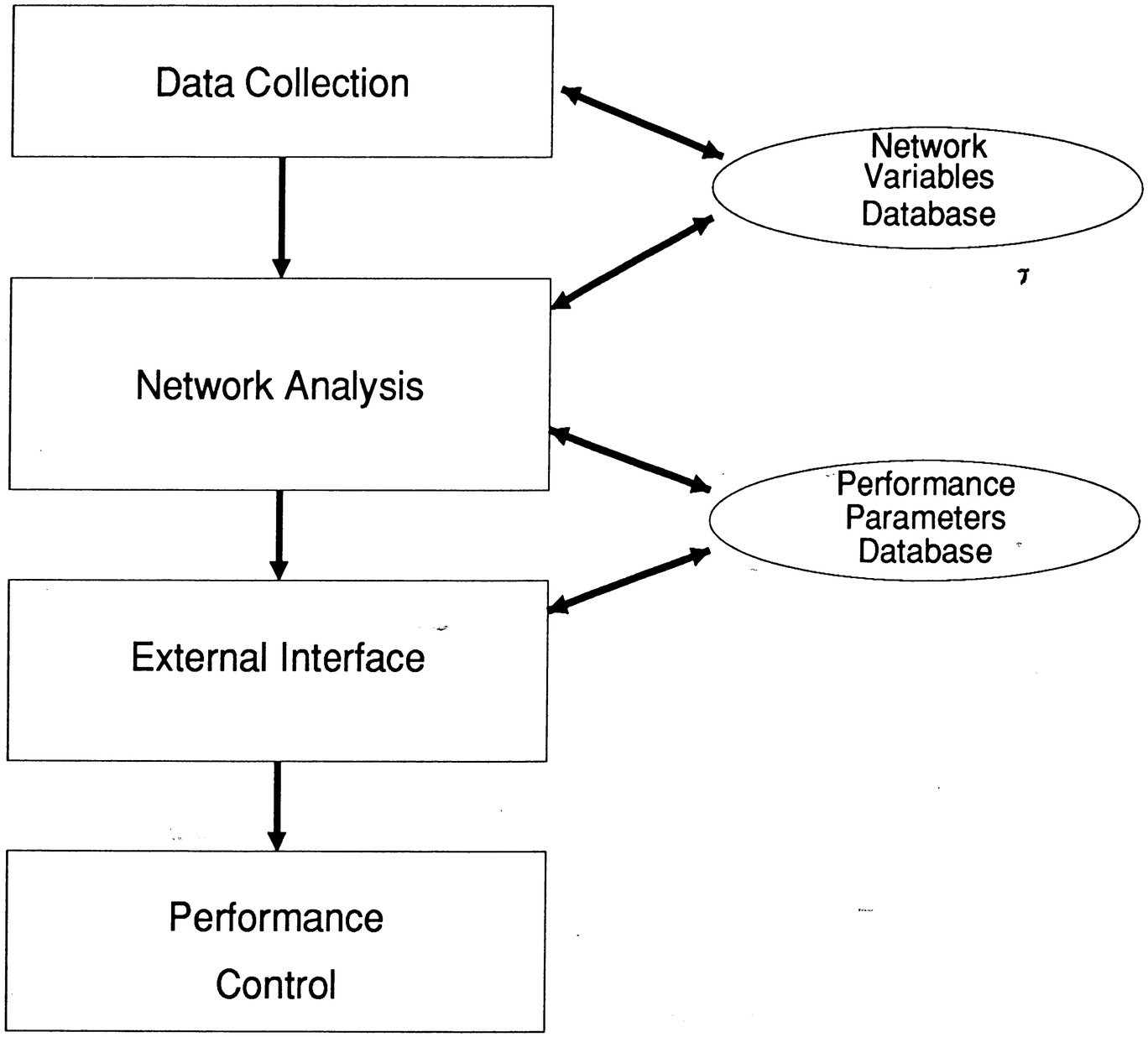


Figure 1 - Performance Management System Components

Counters are used to track the frequency of occurrence of certain events. For example, the network layer could contain counter variables that tally the number of packets arriving from, or destined for, a subnetwork. This information would be pertinent when determining values for network congestion. Another potential use of counters would be in tracking the number of packet retransmissions. This information could be used in calculating throughput and other transport layer dependent functions.

While counters track the number of times a particular event occurs, timers measure the duration of certain events. For example, a timer mechanism could be used to track the time interval between the transmission of data from a certain node and the reception of an acknowledgment at that node. This information would be valuable in computing values for response time and other performance parameters.

Upon completion of the data accumulation phase, the data located at the individual remote nodes must be pooled together at a centrally located database. This pooling capability is provided through the use of network management message passing. A particular example of this is the MAP/TOP 3.0 implementation, in which individual network nodes called agents are polled by a central site called a manager, to collect specific information about the node. Messages such as "M-confirmed_get" request that an agent send specific data element values to a manager's node 5. Once all data is located in the central network database, it is ready to be used by the next component of a performance management system, **network analysis.**

2.2.2 Network Analysis

In the network analysis phase, various analytical techniques are applied to the information contained in the network database to generate actual network performance values. Among the performance measures calculated are the following: 1) throughput; 2) effective throughput; 3) congestion; 4) error rate; 5) response time; 6) link utilization; and 7) fairness. The subsequent sections will describe these measurements and illustrate various techniques employed to calculate them.

Throughput

Throughput is a measure of the amount of information that can be transmitted over a network per unit time. It is based on the buffering capacity of nodes and subnetwork bridges, and on the physical bandwidth of the network media. Network throughput can be determined in a variety of ways.

Initial estimates of the throughput of a network can be determined utilizing various probabilistic models from queuing theory which calculate the throughput of a network as a function of the arrival rate, departure rate, and buffer size. Numerous books and articles on data communications and queuing theory[6] provide details of this approach.

Performance management systems can also utilize information from counter and timer variables in the network database to calculate the throughput of the entire network or subnetwork. For example, counters that are utilized at each network node to track the amount of data arriving at the node can be applied to a numerical technique for determining throughput. This approach involves summing the values of the counters for all of the nodes in a network over a specified interval of time, and then normalizing the resulting value per unit time. Selection of a sufficiently large time interval over which this value is measured will result in a fairly accurate throughput measurement.

These techniques illustrate the variety of methods used to measure throughput in a network. Selection of an appropriate method for a performance management system involves a study which compares the accuracy of the throughput values determined using various mathematical techniques, to known throughput values that were predetermined via testing independent of the performance management system. The method which generates values closest to these predetermined values, over the long term operation of the network, would be the most desirable application.

Effective Throughput

Effective throughput is in some ways a more important gauge of network efficiency than throughput, because it indicates the amount of data that reaches its destination, error free, per unit time. Successful determination of effective throughput involves a procedure that uses values from throughput and message retransmission count calculations.

A value for the retransmission count is obtained through the use of counters that track the number of times a particular message packet is sent before an acknowledgment is received. The retransmission value obtained is used in conjunction with the throughput value to calculate effective throughput. This is a straightforward calculation illustrated as follows:

Description - a performance management system collects the following information for a 10 second interval:

a.) total # of bytes transmitted
(sum of all nodes) = 10 mbytes
b.) total # of retransmissions(bytes) = 100 kbytes
(sum of all nodes)

Effective throughput is calculated using the values from (a) and (b).

$$\text{Effective throughput} = \frac{\text{Bytes transmitted} - \text{bytes retransmitted}}{\text{Time interval}}$$

$$\text{Effective throughput} = \frac{10{,}000{,}000 - 100{,}000 \text{ (bytes)}}{10 \text{ sec}}$$

Effective throughput = 990,000 bytes/sec.

Example 1 - Sample Effective Throughput Calculation

By using approaches such as the one in the above example,the effective throughput of the system can be continuously monitored.

Congestion

According to the MAP/TOP specification, "congestion is the condition during network activity where additional network load will decrease throughput"[7]. This definition implies that if it is possible to measure throughput in a network, and it is also possible to calculate the load of the network at any time, then it is possible to determine the point in which congestion conditions will exist.

Network load values can be calculated using information in the network database such as counts of the amount of data entering and exiting a particular subnetwork per unit time. By constantly monitoring this network load value, as well as the value for throughput, a determination can be made as to when the increase in the network load will adversely affect throughput,(ie. cause congestion).

The specific throughput and network load values at which congestion conditions exist can be compared against the values calculated through theoretical analysis of the network (eg. M/M/N/K queuing models) to aid in the prediction of future congestion conditions. Network planners and router applications utilize this information to determine optimum paths for data and placement of new network nodes.

Error Rate

The error rate is a measure of the number of received packets containing errors at each node, per unit time. Analysis of the bit error rate for an entire network involves tracking the number of packets received with bit errors at all of the nodes in the network, and then combining these figures into a single value over a specified time interval.

The process of tracking the bit errors at each node is a fairly straightforward procedure, because the mechanisms necessary to track these errors are already in place. The following steps outline one approach which utilizes these mechanisms to measure the bit error rates:

1. A counter variable is placed at each node. This variable increments each time an error event is triggered.
2. The counter values at each node are summed over a specified period of time, resulting in a total error count.
3. The total error count for the time period specified is normalized per unit time, resulting in a value for the error rate for the system.

This technique is an effective method to continuously monitor the error rate in the system.

Response Time

Response time is a measure of the length of time between the transmission of a packet at node A, to the servicing of the packet at node B. Typically, this value is measured between transport layers of the two nodes. There are a variety of things that affect response time including; nodes service time, queuing delays at each node, queuing delays at intermediate networks, routing delays, and processing delays.

Calculation of response time involves analysis of data collected at various layers in the network interface. In terms of the logical link and network layers, timers are used to determine the average amount of queuing delay per packet. Service time at a node is

also determined through the use of timers which measure the interval from receipt of a packet at the physical layer to the receipt of an acknowledge back from the transport layer. All of these timer values are combined into a single value which represents the total amount of time it takes for a packet to traverse the network. This total value represents the response time for a particular link in the network. Averaging values for response time for all of the links in the network results in a value for the average response time of the entire network.

Link Utilization

Link utilization measures the average number of bytes transmitted over a particular link per unit time as a percentage of the maximum effective throughput of this link per unit time. The resulting value indicates the percent utilization of a link on a network and can be used by network planners for long term analysis and planning.

Calculation of link utilization involves tracking the load of a network over a long period of time and determining an average load value. This value is matched against the maximum effective throughput of the network to generate the desired percentage of link utilization for a network.

Fairness

The term fairness as applied to network management can take on different meanings depending upon the particular user or application. Thus, in order to effectively measure the "fairness" of a network, "fairness" must be defined within the context of the network requirements. For example, a system which defines fairness with respect to optimizing mean response time would determine this value by monitoring user access time allocations. Conversely, if maximizing throughput in a network is desired, then priority should be given to nodes with maximum demand [8].

These examples illustrate that in order to determine a quantitative value for fairness in a network, a performance management system must have some qualitative description of fairness. Once this description is provided, nodes can be monitored to determine if these fairness goals are being realized.

2.2.3 External Interface

The previous section described the components of a performance management system, as well as some of the techniques employed to calculate actual network performance values. These values are stored in a performance database and made available to external applications through interface modules, which perform format conversions and other necessary functions. Collectively, these interface modules comprise the **external interface** component of a performance management system. This interface provides the capabilities necessary to distribute data stored in the performance database to the performance control component. Specifically, it provides mechanisms for report generation and presentation, and utilities for manipulating the performance database according to user requirements.

The exact utilities provided by the external interface component are specific to particular performance management system implementations. However, there are two particular capabilities that should be provided by all performance management systems; 1) performance display monitor and reporting, and, 2.) software access to the performance database.

The performance display monitor and reporting capability is necessary because it provides a fast and convenient method to track performance values of interest. An example of an external interface that provides this capability is depicted in figure 2. In this example, a man machine interface responds to operator inputs and graphically displays selected performance reports on a display terminal. This exemplifies the ease of accessing performance data using the monitor.

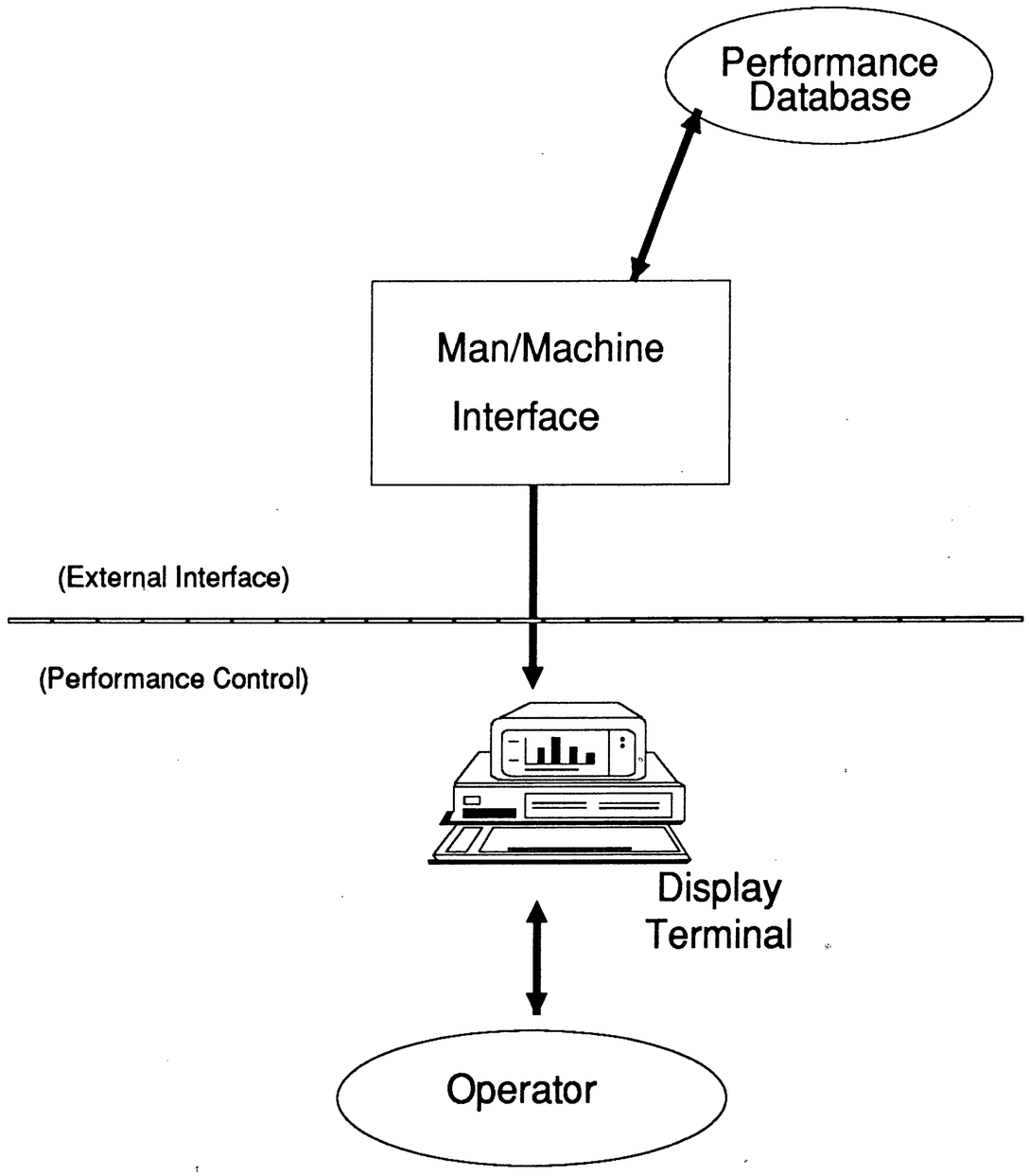


Figure 2 - Sample External Interface Using a Man/Machine Interface

The second feature, as illustrated in figure 3, is to provide the capability to access performance data via software procedures. Depicted in figure 3 is an external interface that includes a software library of functions which allows access to the performance database through subroutine calls. This feature provides the application designer an easy mechanism by which to access the performance database, which makes the task of automating performance management functions less difficult. Since the current direction of performance management systems is toward increased automation, a capability such as this would be highly desirable.

In summary, an effective external interface would be one which provides, at a minimum, both of the features described above. The performance display monitor and reporting capability is a quick and easy method for accessing performance data, while the software library feature is a good utility when developing automated functions. These capabilities would provide data in the formats required by various performance control implementations.

2.2.4 PERFORMANCE CONTROL

The true measure of the effectiveness of a performance management system is its capability to maintain optimum network performance. The purpose of the **performance control** component, which is the fourth and final component of a performance management system, is to provide the procedures necessary to ensure that this desired result, optimization of network performance, is realized. Most current day performance management systems do not specifically provide utilities that perform the tasks required of the performance control component. Some do provide tools that aid the network manager in the development of these utilities. However, in these cases, it is left up to the network manager to design and implement the optimization procedures.

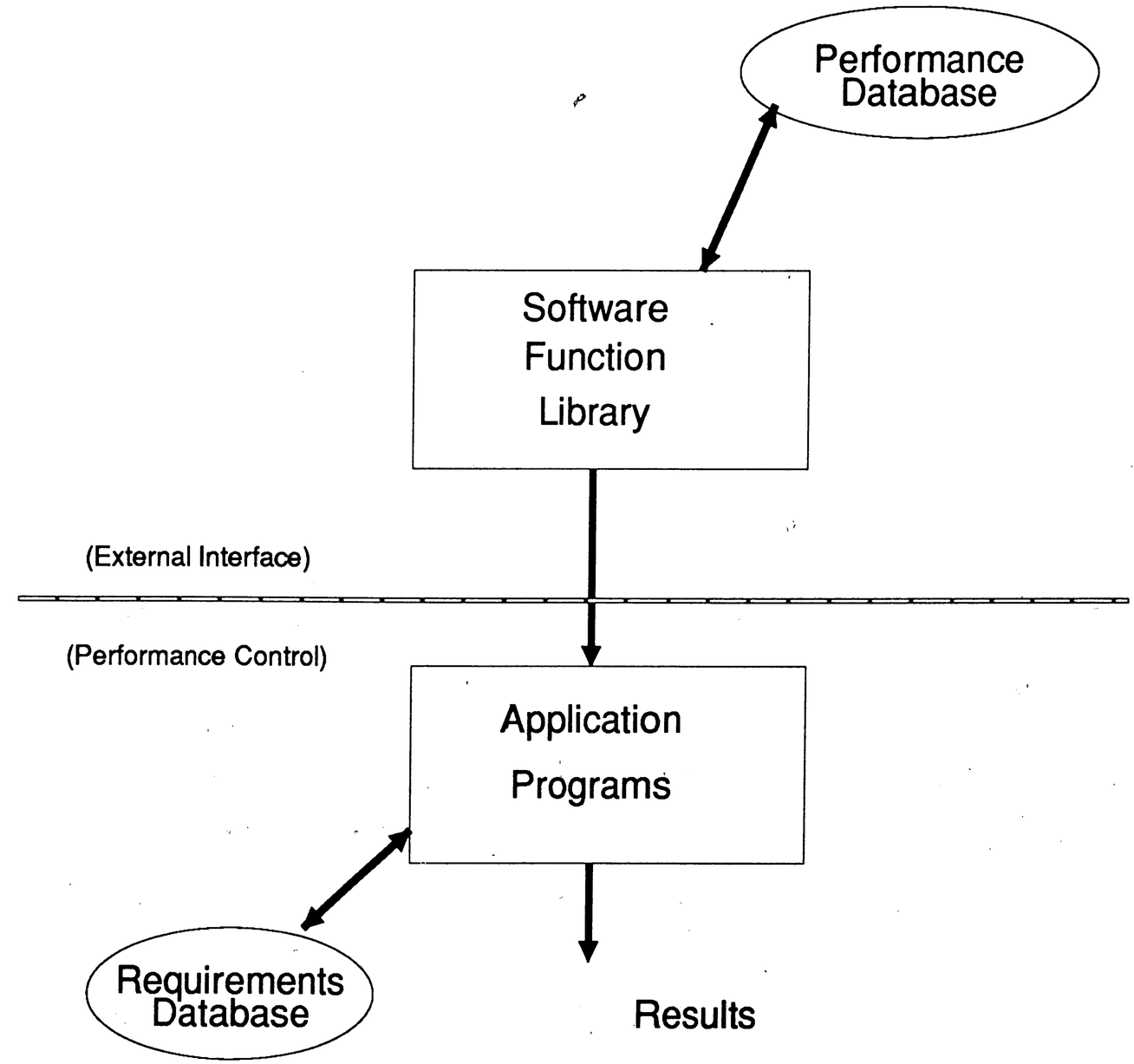


Figure 3 - Sample External Interface Using Software Function Libraries

Figure 4 illustrates a high level approach to network performance optimization. Typically, in present day applications, this function is performed by the network operator. The initial process, accessing the network database, is performed using external interface utilities, as discussed in section 2.3.3. The performance values accessed are then compared against previously established performance parameters determined through the joint participation of both the network managers and network clients . If discrepancies between actual and desired performance are noted, the optimization process advances to the final stages. These final two stages provide **identification** and **correction** of the problems that caused the undesirable performance measures.

2.2.4.1 Identification of Performance Problems

Various algorithms and mathematical functions are available that model the behavior of communication networks. The mathematical techniques applied to accurately model the network fall into a wide range of categories including:

> "*analytical techniques*, that use the methods of mathematical analysis to derive explicit formulae that can be used in practical application; *probabilistic techniques*, that are in someways similar to analytical techniques, but use probability theory and are needed because networks have random elements in their behavior; *numerical techniques*, that calculate numerical estimates and predictions of network behavior and performance; and *simulation*, that in effect ..... computes the course of simplified representations of some events in a network in order to obtain rough estimates and prediction of its behavior and performance."[9]

Numerous mathematics and data communication books that discuss queuing theory and various other network analysis techniques detail these modeling approaches[10]. These methods are utilized by the performance control process in an effort to identify performance problems by localizing network values that fall outside of an acceptable range.

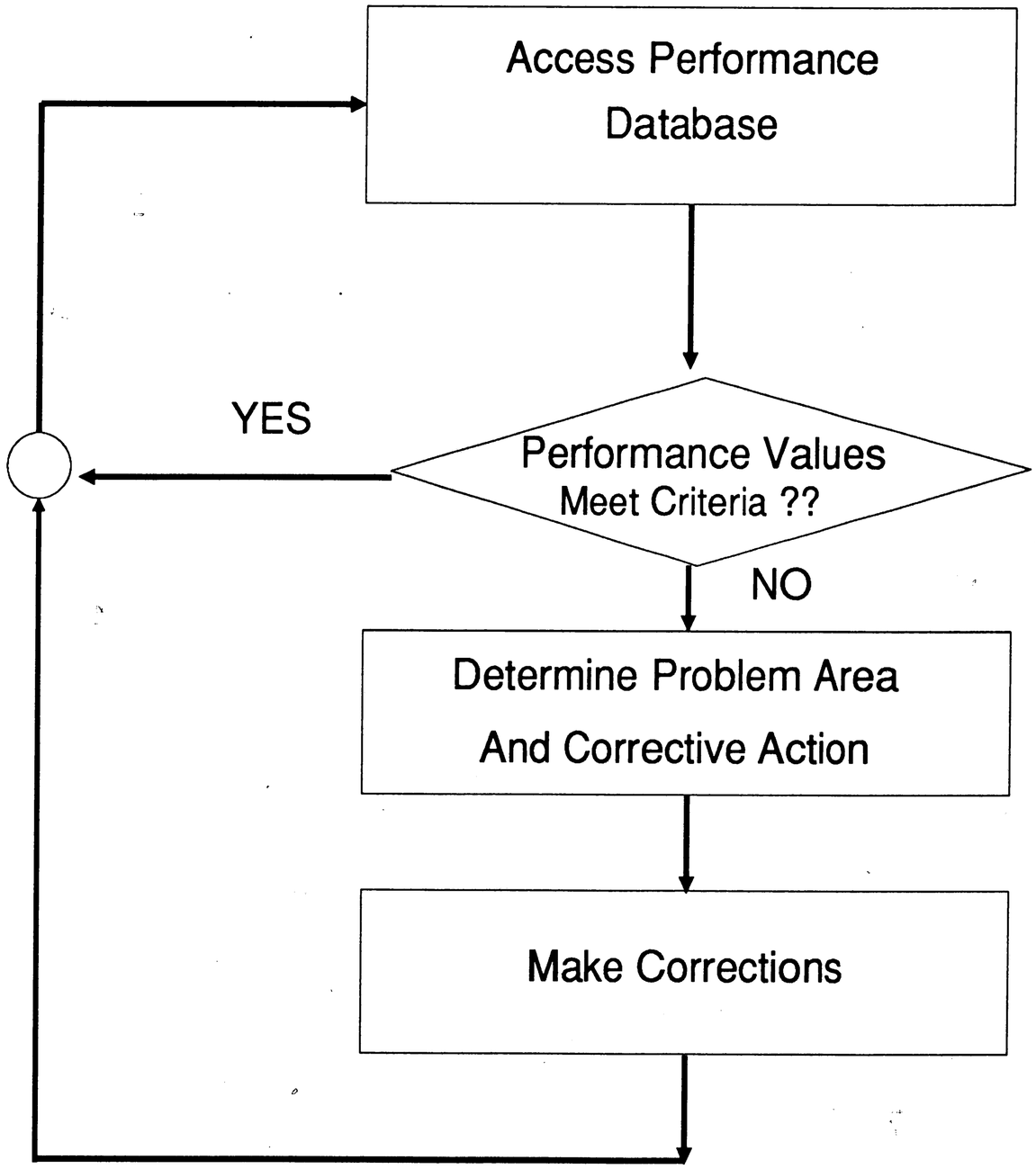


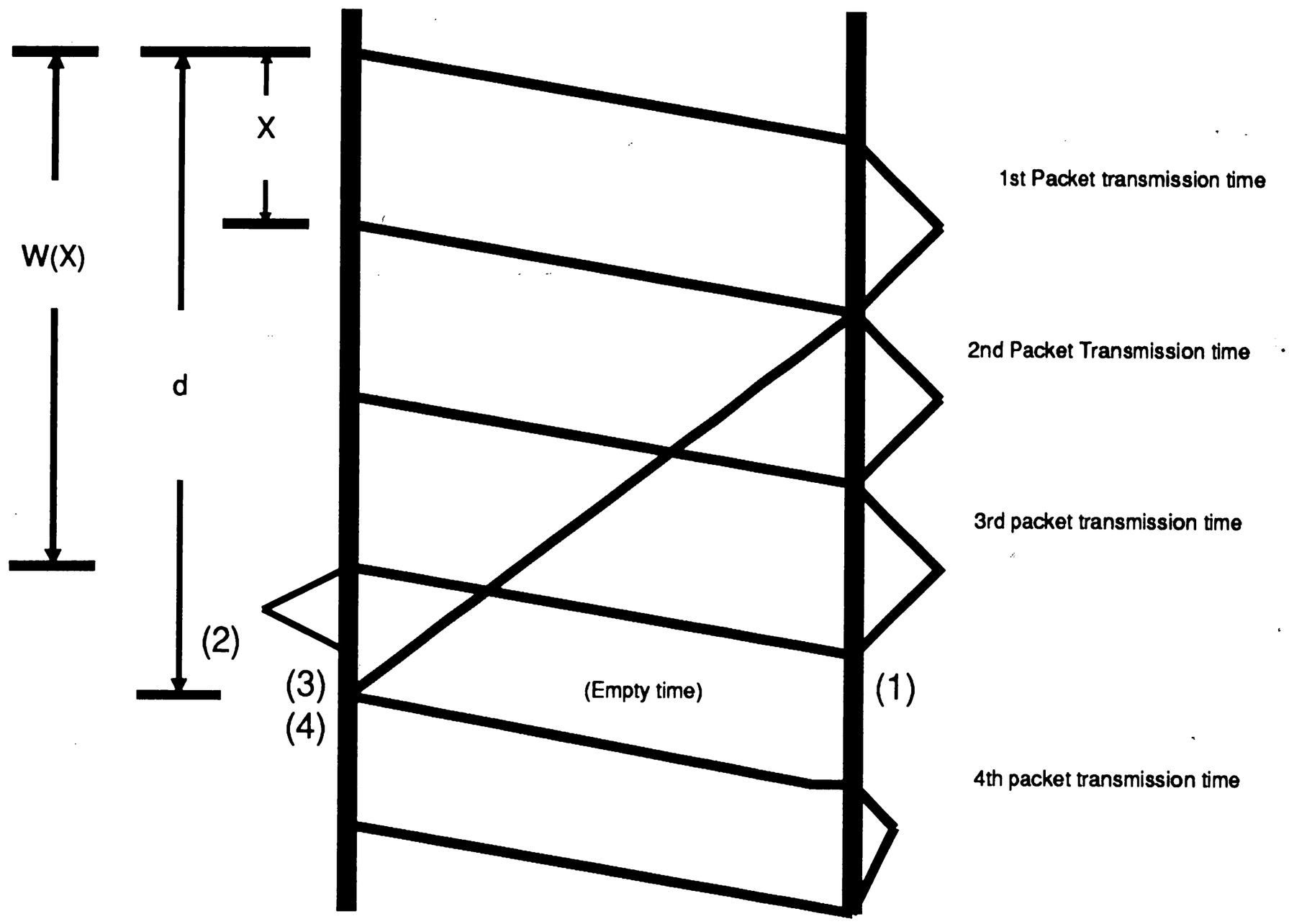
Figure 4 - High Level Network Optimization Approach

While these methods are useful for many applications, there are certain instances where they are unable to identify performance bottlenecks. In such cases, operators rely on "rule of thumb" or "experiential" knowledge to identify the problem. In either case, the goal of this analysis is to identify those network parameters that require "tuning" in order to effect a favorable change in network performance. Examples of "tunable" network parameters include the following: window size, buffer size and acknowledge timers.

Window Size

Various window flow control mechanisms are used in network architectures in order to control the average delay per message packet, fairness to network users, and to prevent buffer overflow. The main goal of window flow control is to maintain high effective throughput in the network. This is accomplished by making the transmitting node wait to transmit certain packets until acknowledges from packets previously sent are received. This results in limiting the quantity of retransmitted packets (caused by an error condition or timeout) since only a small "window" of data can be sent at any time. The size of the window can also affect the response time of a packet, since a small window size implies that a transmitted packet will be serviced in a more expedient manner.

Figure 5 illustrates a sample window flow control system. This particular figure depicts a window flow control mechanism that will allow the transmission of up to three packets of information before it requires an acknowledgment back for the first packet. Once it receives the first acknowledge back, it "slides" the window up one packet and allows the transmission of another packet. This process repeats itself continuously, effectively resulting in controlling the flow of data in the network.



W(X) = window size = 3
d = round trip delay
X = packet transmission time

Sequence of Events

(1) - End of third packet transmission
(2) - Wait period (flow control active)
(3) - First packet acknowledge received
(4) - window slides up; fourth packet sent

11

Figure 5 - Window Flow control system example

(from Denton - Lehigh University
Class notes - ECE 413, page 106)

If window flow control is not being implemented properly, and effective throughput and/or response time measurements are not meeting desired levels, analysis techniques are performed to determine corrective action. Subsequent determination of an optimum window size is driven by the current state of the network. For example, if the system is in a state where high load conditions exist, and effective throughput and response time are not meeting desired levels, a small window size would be desirable. However, at times where low load conditions exist, and the occurrence of errors is not prevalent, a larger window size (which would impose less flow restrictions on the network) would be desirable.

Thus in order to determine the optimum window size of a network, the network control process will analyze the current state of the network with regards to congestion, error rates, acknowledge timeouts, effective throughput and response time in order to determine a window size that would be most desirable. Subsequent "tuning" of the window size to this newly calculated value should improve the performance of the network by bringing effective throughput and/or response time up to an acceptable level.

Buffer Size

At each layer in a network interface, buffer areas exist that temporarily store arriving packets until they can be serviced. Typically, a certain amount of memory is allocated to provide this buffering capability. The amount of memory allocated for this purpose can often be dynamically adjusted. If this is the case, the buffer size network parameter is considered a "tunable" parameter, which can be altered to improve the operational performance of a network.

For example, the effective throughput of a system may be below desired levels, because the packet arrival rate is high relative to the packet service time, and the receiving buffer is unable to accommodate all of the arriving packets. This could be corrected by adjusting the buffer size at the receiving node. In order to determine a new

value for buffer size, a combination of analytical and probabilistic techniques are employed. Using queuing theory and a Poisson model of a network, a minimum buffer size can be calculated which would ensure, within a certain probability range, that the buffer would not be full and thus lose incoming packets. The following formula provides the algorithm necessary to calculate the desired buffer size:

P_n = probability that the buffer is full.
lambda = arrival rate
mu = service rate
rho = lambda/mu
n = buffer size

$P_n = ((1- rho) * rho^n)/(1-rho^{n+1})$

Example 2 - Calculation of Optimum Buffer Size
For a network node 12

The following are the steps involved with implementing the above algorithm:

1. Determine an acceptable probability value (P_n).
2. Measure the arrival rate (lambda) and service rate (mu) for the node.
3. Activate an iterative procedure that increment the value of "n", and recalculates the formula until the desired probability value is attained.
4. At this point, the current value of "n" is the desired buffer size.

Using techniques similar to the one illustrated above, an optimum buffer size for all of the nodes on a network can be maintained.

Acknowledge Timers

Upon transmission of a data packet, the transmitting node activates a timer that is used to track the amount of time between transmission of the packet and reception of an acknowledgment packet back from the receiving node. The purpose of the acknowledge timer, is to insure that the transmitting node does not wait an unreasonable amount of time for an acknowledge response. After a defined interval, the timer will "time-out" and an event will be triggered. This event is an indication to the transmitting node that the packet has been lost and must be retransmitted. Packet retransmission results in lower effective throughput.

Often, delays in intermediate network nodes and routers, as well as propagation delays in the physical network media, pose significant time constraints on transmitted packets. In these situations, acknowledge timeouts can occur simply because of the unrealistic time interval specified for the reception of an acknowledge. Thus even packets that arrive at their destination error free may be retransmitted. In such cases, "tuning" of the acknowledge timer by increasing the time interval for acknowledge timeouts will provide an effective solution that will increase network efficiency by increasing effective throughput. In effect, a simple adjustment in timeout ranges for acknowledgment timers may be all that is required to bring about positive changes in network performance.

2.2.4.2 Correction of Performance Problems

Once network performance bottlenecks are isolated by the network operator, and solutions to these problems have been determined, corrective action must be taken to effect the required change in network performance. It is during this activity that the actual adjustment or "tuning" of network parameters is executed. Utilities required to carry out these functions are provided in the configuration management function of a network management system.

The network operator must activate the configuration management system and indicate the new values for these parameters. The actual adjustment processes performed by configuration management are described in various books and articles, but are beyond the scope of this document.

2.3 CURRENT PERFORMANCE MANAGEMENT IMPLEMENTATIONS

Since the goals and components of performance management systems have been detailed in the previous sections, the next logical step is to discuss some actual performance management system implementations in order to illustrate how closely the actual implementations meet the specified goals and descriptions.

The following section will describe two current performance management system implementation approaches. The first section describes the MAP/TOP 3.0 network management requirements specification. The second section describes NETVIEW, which is a network management system available from IBM.

MAP/TOP 3.0

The MAP/TOP 3.0 Network Management Requirement specification was developed by a committee organized to define the activities required for a network management system. The MAP/TOP group endorses a distributive approach to network management systems. This approach, which is outlined in terms of functional components in figure 6, utilizes the concept of SMAPs (System Management Application process) to collect network information and store the information in a management information database. Remote network nodes, known as "agents" collect and store the network data specific to their node in a unique management information database. This information is then combined with similar information from other "agents" in a system wide database, at the manager's SMAP. Transfer of this data from the agent SMAP to the manager SMAP is performed using "manager-agent" protocol. Once this information is available in the

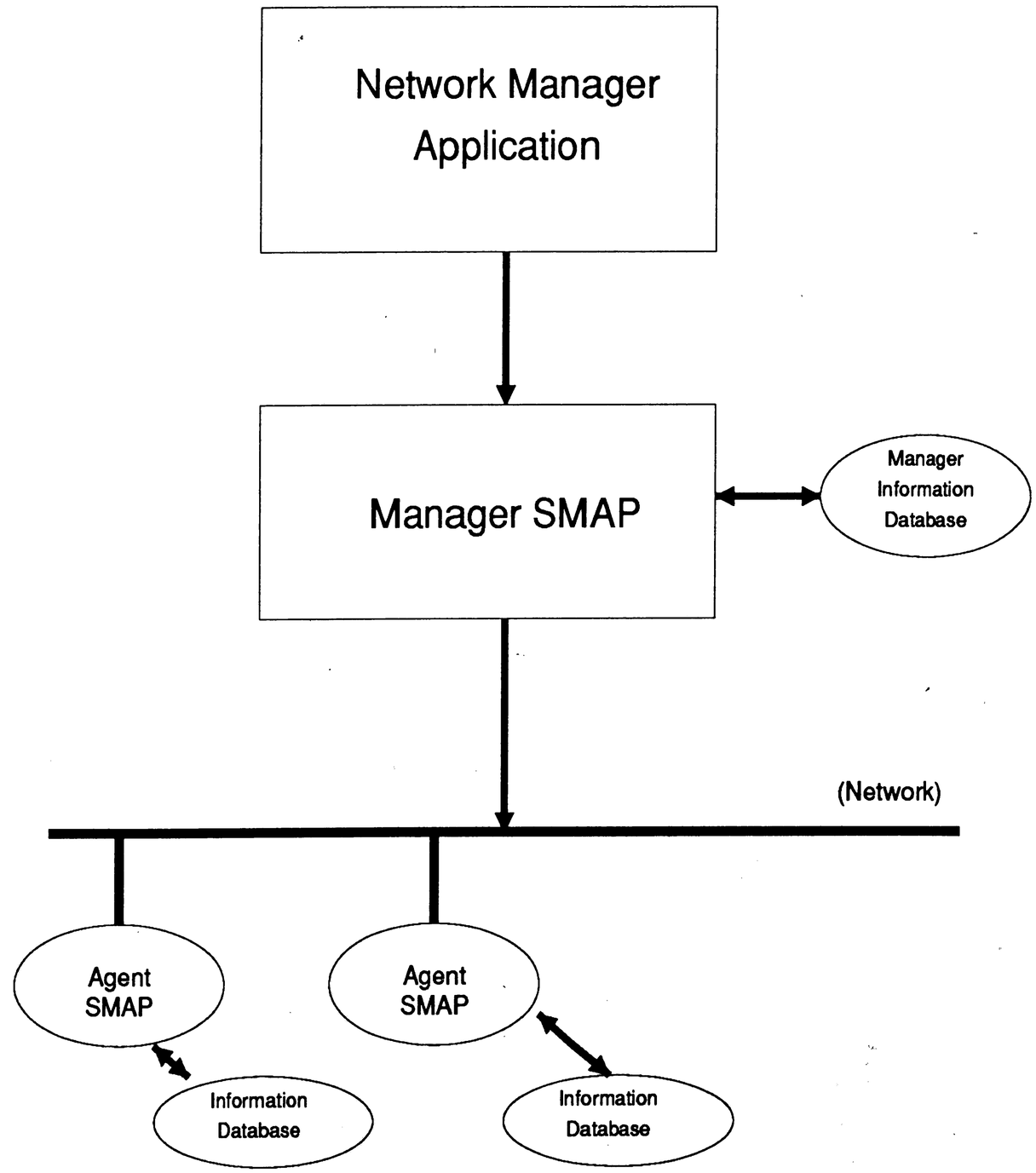


Figure 6 - MAP/TOP Functional Components

system wide database at the Manager SMAP, procedures resident in the network manager application will perform the various activities required of performance management systems. MAP/TOP defines the activities required for performance management as being those tasks that perform data collection and network analysis.

The **data collection** function is performed through SMAP procedures resident on individual agent nodes in the network that collect information from various network layers. The actual utilities that are used to collect this data are called "mechanisms" by the MAP/TOP specification. Mechanisms are defined as "the means by which functions required to accomplish network management are affected".[13] Examples of mechanisms used in data collection include "transportTimeout", which is a counter that increments every time a transport entity exceeds its persistence time or maximum number of retransmissions[14], and, "numberTPDUsent" which is a counter incremented every time a transport data unit is transmitted [15]. Through the use of mechanisms, the MAP/TOP specification provides the counters and timers necessary to collect information required for the network analysis phase.

In terms of **network analysis**, the MAP/TOP specification dictates that the following capabilities be provided: determination of congestion conditions; determination of message delay; determination of network reliability; determination of routing and bridge throughput; determination of network efficiency; and determination of link utilization and available capacity. While the MAP/TOP specification identifies the functions required to perform network analysis, it does not specify how to implement them. This decision lies with the individual network designer.

The MAP/TOP 3.0 specification does not detail the functions of either the **external interface** or **performance control** components. However, the specification does provide a very general description which states that these are functions to be provided by the network manager application component, which is shown in figure 6. According to the specification, the network manager application "provides the mechanism for the network

administrator, a human, to read or alter data, control the network and access reports. The network manager application can be a simple command line interface to the SMAP or could be an expert system requiring very little interaction with the network administrator." 16

In summary, MAP/TOP details the features required for the data collection and network analysis components of performance management systems. It also goes as far as stating the need for external interface and performance control components, but does not provide any details.

NETVIEW

NETVIEW is IBM's network management solution for its SNA network[17]. The performance management function is provided via NETVIEW's performance monitor. This monitor provides the capabilities of automatic network data collection, network analysis, and report generation[18]. In addition, it has the ability to interface to high level software modules,which is an essential part of the external interface component of a performance management system.

Netview accomplishes **data collection** through the use of various monitors that are provided with the system. The hardware monitor enables the network operator to access problem determination information generated at various resources, including statistics of traffic and recoverable errors. The session monitor collects and correlates information about particular sessions and routes. Included in this information are values from various network timers and counters utilized during a session. The information from these monitors at each node is then automatically collected by the main performance monitor and stored in the database required for the analysis phase.

Netview's **network analysis** feature provides a variety of capabilities. It determines values for response time, volume data (or congestion), throughput, effective throughput, and traffic flow. In addition, it provides "alerts" based on user specified

thresholds. All of this information is made available to the user through external interfaces.

The **external interfaces** provided by Netview vary. It provides a graphical presentation of performance measures upon user request, including volume and response time for various nodes and color graphs of real time and historical data. Netview also provides an interface to high level languages such as "C" and PL/1. This feature is analogous to the software library illustrated in figure 4, in that it allows procedural access to the performance database. This external interface feature also allows the integration of knowledge base tools such as IBM's Knowledge tool, providing the possibility of an expert system implementation of the performance control component.

The functions required for **performance control** are not provided as standard products by Netview. References are made to Netview's ability to aid in tuning the performance of a network, but no specific functions that provide this capability are included.

In summary, NETVIEW provides excellent data collection, network analysis and external interface features. As with the MAP/TOP specifications, however, Netview lacks in the features required of performance control.

2.3.1 Summary of Current Implementations

The current capabilities of performance management systems, as discussed in the previous sections, can be summarized as follows:

1. The data collection and network analysis components are well supported and available in most systems.
2. The external interface component varies widely depending upon the specific implementation. The MAP/TOP specification leaves the implementation of this

component up to the system designer. On the other hand, IBM's NETVIEW supports a more developed external interface, providing both report generation and software procedural access capabilities.

3. The performance control component is usually identified as a necessary element, but standard utilities are not provided by current day performance management system implementations.

It is therefore evident that one of the most important aspects of a performance management system (performance control) is the most neglected. In order to provide this capability, an expert system approach is proposed. The subsequent sections will discuss expert system theory in order to provide background material necessary to substantiate this proposal.

3.0 EXPERT SYSTEMS

As the field of data communication becomes more complex a greater demand is placed on the human operators who control or maintain these systems. As a result, much emphasis has been placed on the development of automated systems that can replace or enhance operations that were previously performed by a human operator. Many of these systems are purely algorithmic in nature, in that they perform simple functions using known iterative approaches.

However, for some applications, the use of algorithmic approaches to automate functions is ineffective due to the empirical nature of the problem solving process involved. In the past, the ability to solve problems requiring empirical knowledge was reserved for the human expert. More recently, there has been a strong emphasis on using computers to solve these types of problems. A result of this has been the development of a

field of research, dedicated to simulating the decision making ability of a human, known as expert systems.

The performance control component of a performance management system is an ideal candidate for an expert system implementation. This is due to the empirical nature of the decision making process involved with maintaining the optimum performance of a network. While many of the steps involved with performance control utilize algorithmic problem solving, the decisions regarding which formulas to use and which procedural paths to take during the overall control phase do not follow known algorithms. As a result, current day performance management systems rely on a human expert to make these decisions.

The following sections will discuss expert system theory in order to illustrate how an expert system can be applied to the performance control function. Section one provides definitions and a brief description of each of the components of an expert system. A general design approach for developing an expert system is discussed in detail in the subsequent section. Finally, the third section presents three examples of expert system implementation in problem domains related to performance management.

3.1 EXPERT SYSTEM OVERVIEW

An expert system is a system that emulates the problem solving expertise of a human being for a specific problem domain. The general concept behind expert systems is to capture the problem solving knowledge typically associated with an expert in a particular field, and represent this knowledge in such a way that a computer can approximate the expert's ability to solve a particular problem. [19]

Figure 7 illustrates the components of an expert system. These components work together to simulate operations that are typically performed by various parts of the human body. As such, an analogy can be made between the anatomy of a human body and the anatomy of an expert system.

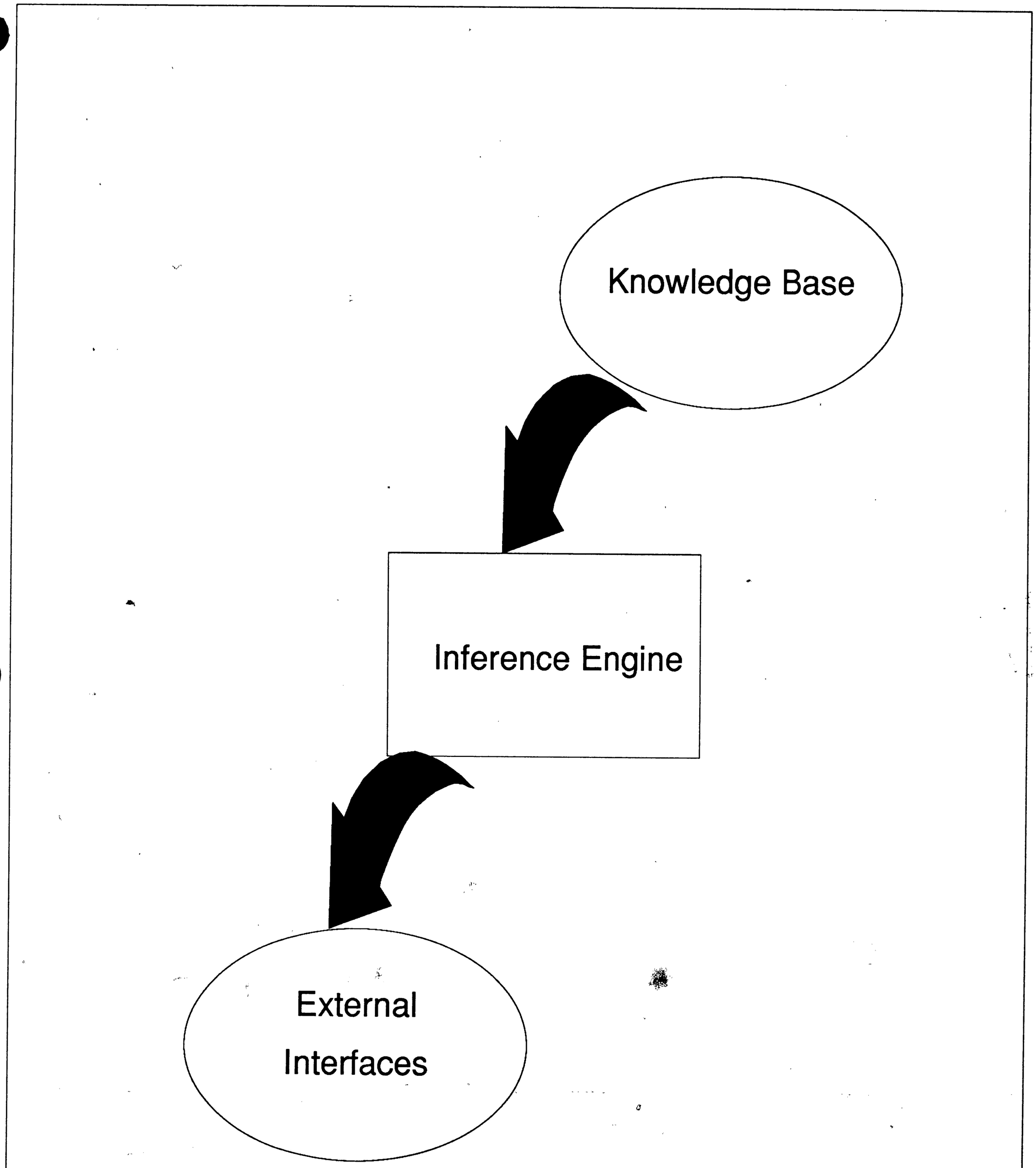


Figure 7 - Components of an Expert System

A human's ability to solve problems is based on the knowledge he has stored in his memory. He obtains knowledge from lessons, books, and experience, stores the knowledge in memory, and uses it to determine a solution which is based on logic. The **knowledge base** of an expert system is comparable to the memory area in a human brain. It contains all of the information that is necessary to solve problems in a specific domain. The various types of information, or knowledge, that would typically be stored in this database include "shallow" or empirical knowledge, and "deep" or public knowledge.

"Shallow" knowledge is typically the information that identifies an individual as an expert. This information cannot be found in any books or readings but is acquired from other expert's in the field, or from first hand experience. It consists of various "rules of thumb" that have been determined by trial and error over a period of time. Typically, this type of information cannot be expressed in terms of algorithms that can be easily implemented in traditional programming languages.

"Deep" or public knowledge is information that is available in text books and articles, and can be expressed in terms of formulas and algorithms. This information is essential in the decision making process because it is through this type of knowledge that experts understand how to initially approach an unfamiliar problem.

As with a human, an expert system is useless unless it has the ability to draw conclusions based on logical thinking. Just as the cerebrum provides this capability for humans, the **inference engine** provides this facility in expert system technology. An inference engine is defined as a "general purpose software machine that draws on information from the knowledge database to deduce a solution or infer the consequences from a new piece of information" [20] Various design approaches can be taken in the development of this component. The design approach selected is dependent upon the way in which information is represented in the knowledge database. Further elaboration of this topic will be provided in the subsequent sections of the paper.

The final component of an expert system is the **external interface**. This component is to an expert system what the eyes, ears and other communication mechanisms are to a human. In other words, the external interface provides the capabilities required for a user to input information into an expert system and to receive information (and solutions) from an expert system. While this component is the least recognized component of an expert system, it is extremely important, due to the inherent need of all systems to be able to communicate effectively to the outside world.

3.2 EXPERT SYSTEM DEVELOPMENT

The development cycle for an expert system design is shown in figure 8. As illustrated, this cycle involves three major tasks; determining the feasibility of an expert system design, defining the requirements, and designing the system.

The **feasibility study** generally involves determining answers to the following questions:

1. Can the cost of an expert system design be justified by the long term benefits it will provide?
2. Does the general problem domain that the expert system will operate on lend itself to an expert system implementation?

If these questions can both be answered positively, the development of an expert system can be justified.

The next stage, **requirements** is the first stage in a system development effort, because it defines the problem domain for the system. Since the effectiveness of an expert system is affected by the level of detail specified in the problem domain, the requirements phase will significantly impact the remainder of the development effort . That is, an expert

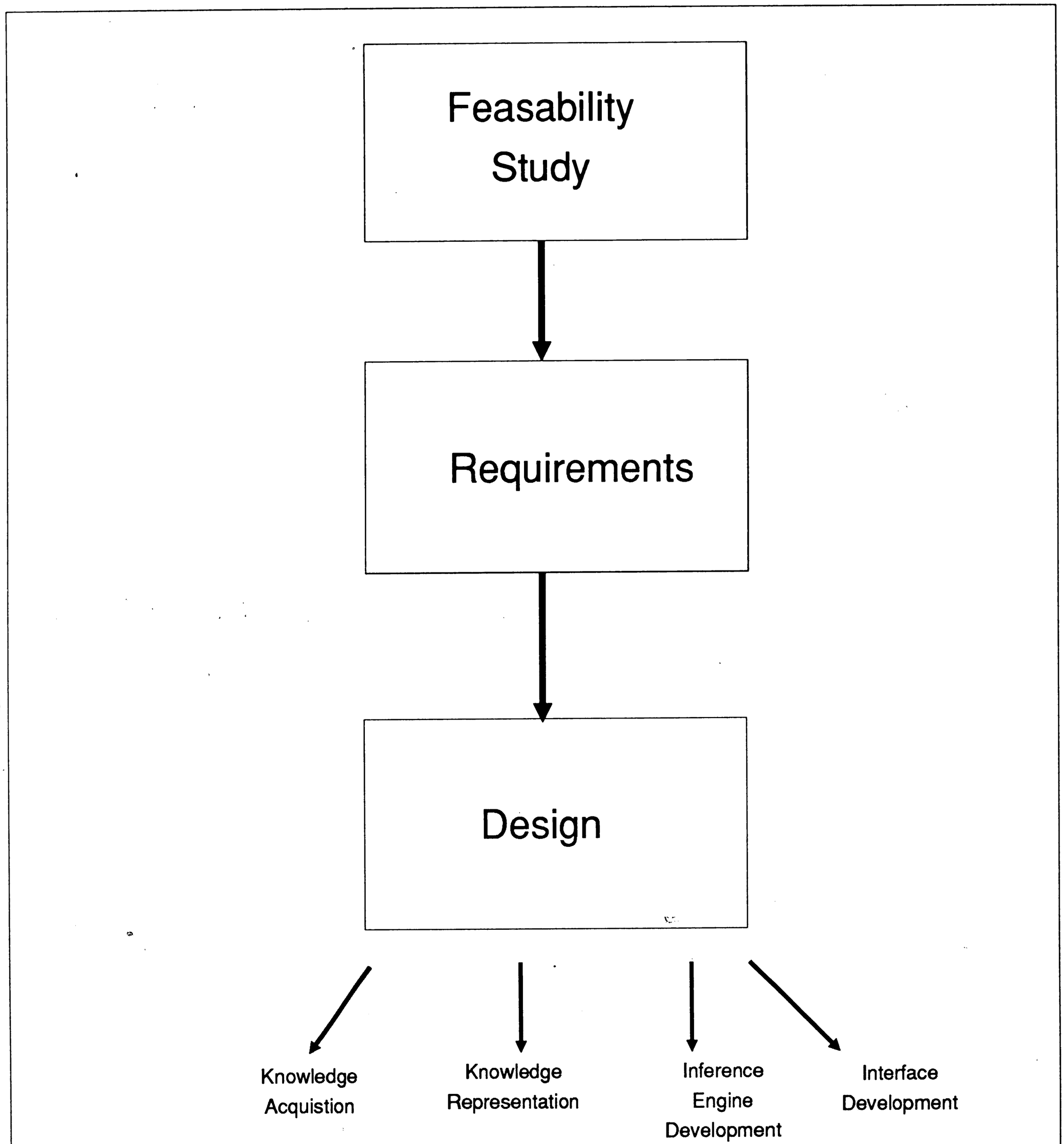


Figure 8 - Expert System Development Stages

system will be able to more effectively simulate the problem solving capability of a human if the problem to be solved is clearly defined.

The third stage is the **design** stage. This stage is divided into the following functions; knowledge acquisition, knowledge representation, inference engine development, and interface development. The first two sections; knowledge acquisition and knowledge representation are front end processes required prior to the design of the actual expert system software. The **knowledge acquisition** activity encompasses all of the procedures involved with the accumulation of knowledge from an expert in a specified problem domain. The **knowledge representation** phase, determines how this acquired knowledge will be converted and store in a computer database. It will be through this phase that the knowledge base for the expert system will be developed.

Next is the development of the **inference engine**. The procedures applied to the knowledge base, in order infer a solution to a particular problem, are developed in this stage. Once the basic engine has been developed, the **external interface** development begins. During this phase, the system input and output handling procedures are designed and implemented. Each of these development phases is explained in more detail in the following sections.

3.2.1 Feasibility Study

As compared to the development of a typical, purely algorithmic software system, development of an expert system is significantly more costly due to the increase in complexity that is inherent in an expert system design. Therefore, in order for an expert system design to be feasible, the benefits of greater automation must outweigh the development costs. The purpose of the **feasibility study** is to determine if this is the case. As a prerequisite for developing an expert system application, the following criteria should be met:

1. No known algorithm or solution exists for the

problem domain.

2. The solutions of the expert are correct (but may suffer from time delays).
3. Decisions made by nonexperts are different from those of the expert and may have significant impact on the organization because they are more costly; more resource consuming; have greater time delays, and are less consistent.[21]

In order to determine if these criteria are met for a proposed system, the feasibility study examines two types of issues: technical and economic feasibility.

Technical Issues

There are several technical issues to address during the feasibility study. These include determination of the suitability of a problem, determination of the characteristics of knowledge, determination of the characteristics / availability of experts and determination of the availability of interfaces to the outside world.

Problem suitability involves identifying whether known algorithms exist, determining the bounds of a problem domain and identifying the development time for a system. A problem domain which has defined bounds, in which solutions cannot be generated using defined algorithms and which can be developed in a reasonable amount of time is a good candidates for an expert system.

Determination of knowledge characteristics involves identifying whether the information available for a particular problem domain can be easily represented in a form that a computer can operate on. Specifically, it involves determining whether the data can be represented using known knowledge representation formats and can be processed using known inferencing techniques. The availability/characteristics of experts is another important issue to investigate before a knowledge based system design can commence.

While it may be possible to prove on paper that an expert system design is feasible given that tangible issues such as "no known algorithms exist", there are some less tangible items which must be taken into consideration. One example is whether there are enough cooperative experts available during the knowledge acquisition phase. As a result, the availability and level of cooperation of experts must be identified during the feasibility study.

The final technical issue is the availability of interfaces to the outside world. Possible design problems that could be faced during the interface development phase of an expert system design should be identified at this time. If the external interfaces are known standards or accepted commercial products, no difficulty exists. However, if the current system interfaces are customized "one of a kind" interfaces, a significant amount of interfacing difficulty may be present during the expert system design. Thus this issue must be taken into account before a final development decision is made.

Economic Issues

The economic issues identified include the cost and economic benefits of the design. The economic benefits are estimated over the long term and provide a measure of how much the proposed expert system solution will save a company. Possible benefits include lower processing costs, improved quality, more consistent results, faster decision making, greater availability, and lower training costs.[22]

The costs associated with a knowledge-based approach are based on a variety of things. Initially, the price of the special hardware and software that needs to be purchased is considered. Secondly, the development time associated with the design of an expert system is taken into account, because of the significant costs incurred for manpower and overhead during the development cycle. Finally, the costs incurred during the period following initial release of the system due to the "growing pains" associated with learning how to use this new system are identified. Proper identification of these economic issues

allow for more intelligent decisions to be made with regards to a cost versus benefits tradeoff study.

If a careful study involving all of the issues presented in the previous two sections is performed, an accurate assessment of the feasibility of an expert system design can be made. Assuming that this study has indicated that an expert system implementation is a viable and desired option, the actual development process begins.

3.2.2 Requirements

During this development phase, a precise definition of the problem domain is determined and a formal problem statement is generated. In order "to fully take advantage of expert system techniques, the area of expertise (or problem domain) should be sufficiently narrow and self contained, involving primarily symbolic (nonnumeric) reasoning and require the use of heuristics"[23] It is the purpose of the requirements phase to provide this level of detail for the problem domain. An approach for developing these requirements for an expert system is outlined in figure 9. The initial stage in requirements development involves the identification of user requirements. This stage necessitates the joint cooperation of four groups, who all have different reasons for their involvement. These four groups are the management staff, system experts, system users and expert system developers. **Management** is most concerned with the tradeoffs of costs versus benefits for the proposed system and with how the expert system will affect the operating environment of the organization. Thus, their requirements are more tailored towards cost efficiency. **System experts**, on the other hand, are more concerned with ensuring that the knowledge base to be developed accurately represents the decision making process. As a result, they will take great care in precisely defining the problem domain.

System users also have different priorities. They are involved with ensuring that the new system will provide the capabilities that the old system provided, and will present the information in an understandable format. Thus, requirements that will ensure

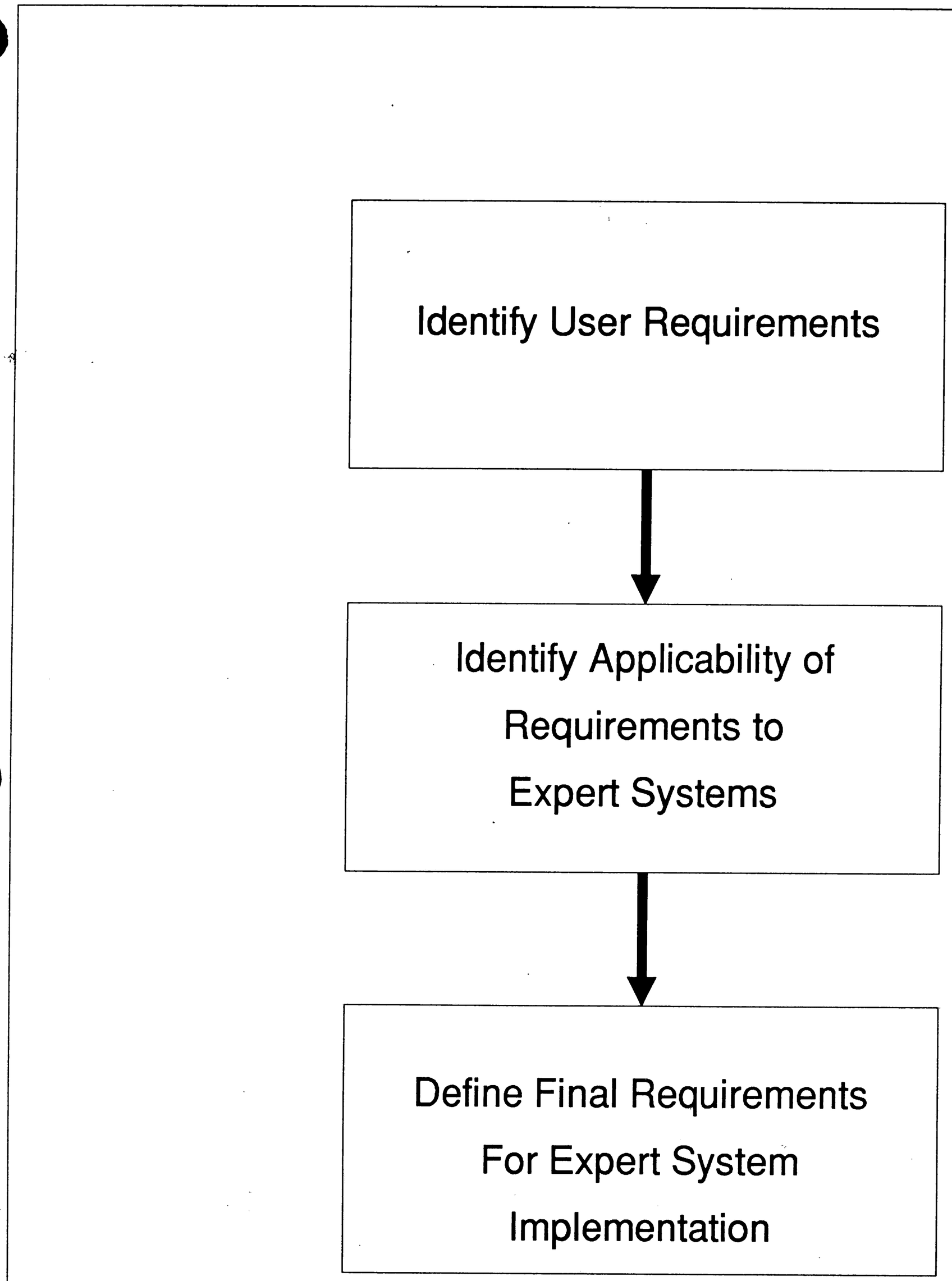


Figure 9 - Development of Expert System Requirements

compatibility to existing systems will be specified by this group. Finally, the **system designers** - those who are aware of the limitations of expert system designs, also contribute in the requirements phase by ensuring that the requirements specified are feasible for an expert system.

The work of these groups results in the generation of answers to the following general requirement questions:

1. Who will be using the system?
2. What is the physical environment? - (ie. is this system going to be local to a single site or distributed throughout many sites).
3. What are the time limitations? - (eg. is there a requirement for real- time problem solving capabilities?).
4. What decision making ability is required? (ie.identifying the purpose of the current system).
5. What are the priorities for each task.

Answers to the above questions will result in the development of a detailed statement of user requirements.

In the next phase in the requirements process, the applicability of these requirements to an expert system implementation is determined. This process involves identifying a subset of the initial requirements that are the best candidates for implementation using a knowledge-based approach. Since the requirements for which expert system approaches are most feasible are those that cannot be solved with known algorithms or formulas, it is these requirements that will be contained within this subset.

The final stage in the requirements cycle is the formal definition of the requirements that will be implemented through an expert system design. This involves combining all of the requirements included in the "phase-two" subset into a single document. Once this formal document is completed, the design stages of the expert system can commence.

3.2.3 Design

3.2.3.1 Knowledge Acquisition

The knowledge acquisition phase involves the procedures used to obtain knowledge from the experts in a specified problem domain. Since this knowledge lies at the root of the expert system, effective acquisition procedures are imperative.

The actual process of acquiring data involves interviewing the experts regarding approaches to be taken to satisfy the problem domain requirements for the system. Various interview techniques have been suggested in several sources [24]. These include two on one interviews, and interviewing multiple experts. Generally though, the expert should be asked the same set of questions with regards to the system requirements. Sample questions that are asked of the expert include the following:

1. What diagnostic procedures can be utilized in order to determine if the specific requirement is being met? (This involves defining the analysis procedures for a specific requirement. Descriptions will contain algorithmic and "rule of thumb" approaches)
2. If a problem exists in the system, what corrective actions are necessary ? (Specifies "rule of thumb" solutions to problems

Detailed and accurate responses to those questions will result in a successful knowledge acquisition process.

3.2.3.2 Knowledge Representation

Once the knowledge is acquired, it must be represented in some format in a computer database. The purpose of the knowledge representation phase is to determine which formats should be used to represent this knowledge. Based on the problem domain and the extent of initial available data, knowledge engineers select the appropriate knowledge representation framework. There are four frameworks for knowledge representation:

1. Rule Based
2. Frame Based
3. Model Based
4. Hybrid

Each of these formats have distinctive characteristics and advantages over the other formats for particular applications. As a result, the following sections will discuss the relative merits of each type.

Rule Based

This format is used to capture knowledge of the type "if A and B then C". It uses knowledge of past experiences to define steadfast rules to follow. This type of representation is useful for processes that are repeatable over a long period of time. A system designed using rule based representation can process rules with related predicates and consequents to infer information about the system. For example, an expert performance management system might have the following rules concerning a network:

1. If it is Friday morning

   Then the network is undergoing maintenance.

2. If the network is undergoing maintenance

   Then the response time is twice as long as usual.

The expert system can utilize these rules to infer that the response time on Friday morning is twice as long as normal is because of maintenance.

A probability factor can also be applied to a rule in the knowledge base. This probability factor is particularly significant in data communications where modeling network applications are often based on probability distributions. For example, assuming the arrival and departure rates are known for a network a rule can be developed to determine the optimum buffer size for a node so that no overflow conditions will occur (see example 2 which further illustrates this point).

The development of a knowledge base using rule based formats involves the formal declaration of a set of machine readable rules. These rules are grouped into logical rule sets in a database, which makes it easier for an inference engine to operate on them. The rule set may be a single large set or may be comprised of a collection of subsets.

There are various problems with rule based representation. The first, is that not all information can be easily represented in terms of rules. Often times this information is forced into a set of rules resulting in inefficient and ineffective reasoning procedures during the inference stage. Secondly, if the designer of a rule base is not careful, rules may be reevaluated during the inferencing process, and could result in endless loop conditions. "Rules often become complex, incorporating representations that when evaluated change the state of the knowledge database."[25] If this occurs, there is a possibility that the rules "IF A then B" and "IF B then C" will result in an expression that will lead the consequent "C" to reevaluate the rule for "A". If this occurs, an endless loop condition results.

While rule based formats are advantageous for representing certain types of knowledge, they are obviously not ideal for every condition. As a result, other representation techniques are employed. The following section will address another technique - **frame based** representation.

Frame Based

In this approach, information is represented as a collection of discrete objects that are defined in terms of a name and associated value. The entire knowledge base is organized as sets of these discrete object representations, known as "frames", which define the structure of an expert's reasoning process.

Frames are used to organize data having some underlying structure. Data that can be related to other data in some fashion belongs to the same "class" of information. Objects in the same "class" inherit the attributes of other members of the class. This inheritance feature is a very powerful concept because it allows for a much more organized and expedient traversal through the database during the inferencing process.

Actual implementation of frame-based representation in the knowledge database involves defining each frame in terms of a data record. The data record is composed of a set of "slots" , each of which has a specific data item and value associated. Frames are related to other frames through the application of set theory, where various frames are subsets of larger frames. Utilizing this approach, the structure of the knowledge database is defined.

While frame-based representations typically allow for a more efficient traversal through the knowledge base, there are several disadvantages of frame based reasoning. The major drawback is that it is difficult to express rules involving reactions to a particular event using frame-based reasoning. For example, where it may have taken two simple rules to define such a reaction using rule-based approaches, it would take a significant effort using frame based approaches. A second disadvantage with frame based

representation is that if this approach is abused, a point of diminishing return can result. That is, while one of the greatest attributes of frame based representation is the inheritance feature, if this feature is used to often, and every object inherits the attributes of every other object, then it may take a significant effort for the inference engine to traverse all of the paths of the frame. Thus, a designer using frame based representations must insure that the database development process is carried out in a judicious manner.

Model Based

Knowledge that can't easily be expressed by a set of distinct rules or as frames in a hierarchy is often represented in terms of models that define the **structure** and **behavior** of the problem domain. Knowledge represented in this fashion utilizes so called "deep" or public knowledge in defining the static and dynamic performance of a system. Static states can be represented in terms of flow charts which define the structure of the system. Dynamic states can be represented using Petri nets, which define the time dependent behavior of a system.

The physical organization of a system is defined through the **structural** part of the model. Once the structure of a system is defined, it can be used to pinpoint problems in a system. For example, figure 10 shows a sample wide area communications network that provides two possible data paths from node X to node Y. If there is a response time problem when transmitting data from node X to Y along path 1, the system will attempt to transmit data along path 2. If no response time problem exists using this route, the system can infer that congestion conditions exist in path 1. This is an example of how the structure of a system can be used to make decisions in a system.

The **behavior** of a system can be defined through system simulations and/or through a detailed collection of related procedures. For a given input, a system process should act in a certain way to generate a certain output.. In order to model this behavior, procedures are developed that simulate the process[26]. This output is compared with actual

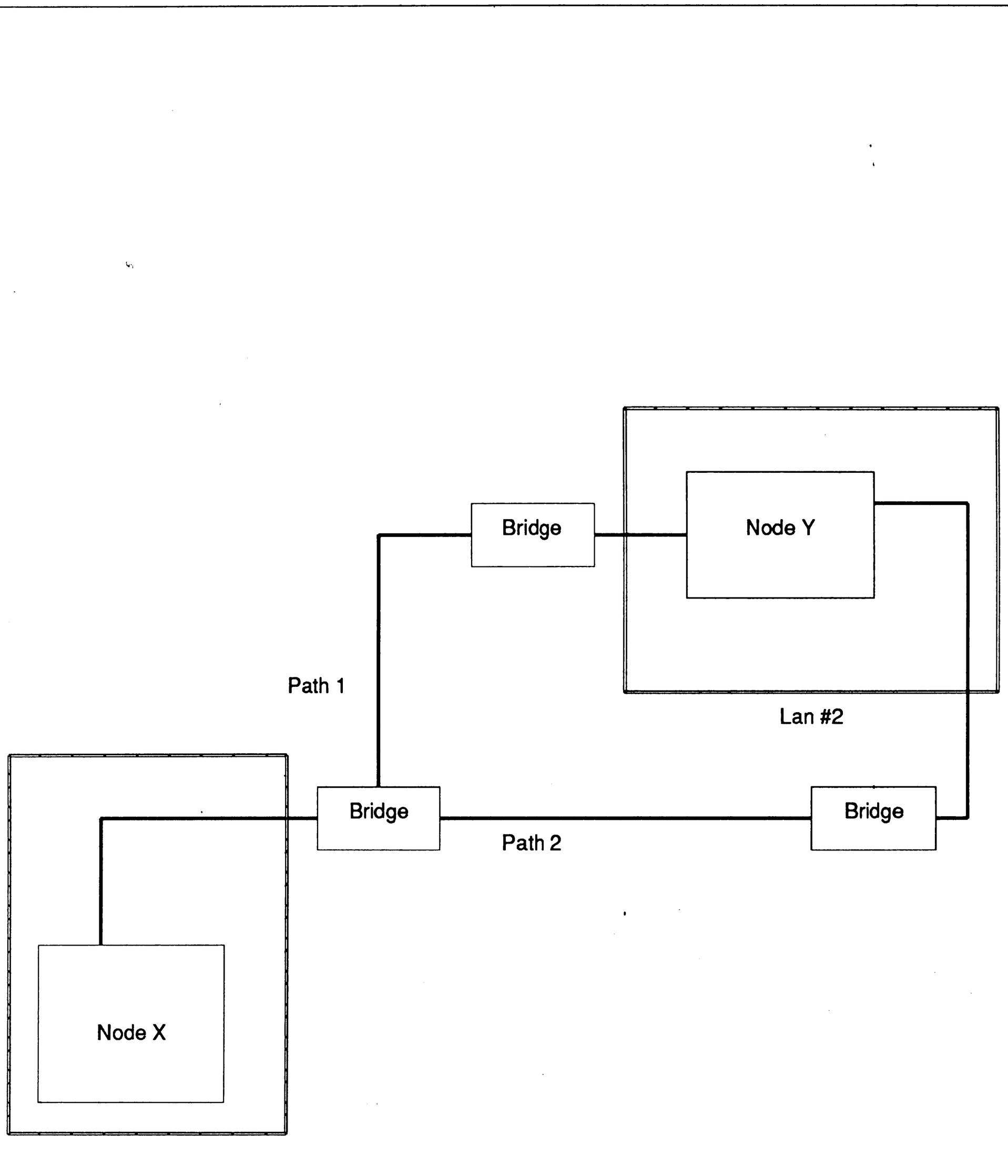


Figure 10 - Example of Structure In Model Based Representation

results. If the two differ, a problem is detected. Thus, the behavior model can be used to make decisions in a network by pinpointing inconsistencies in a system.

The model based representation technique is a very effective method for representing knowledge of complex systems. However, there are some disadvantages with this approach. First, since the model based approach does not take advantage of time saving heuristics, it is typically slower and less capable of solving problems for which there are no defined problem solving procedures or behavioral patterns. Secondly, the interface between current day knowledge based tools and traditional simulation tools is very undefined and difficult to implement. This makes the tasks of integrating both of these tools into a single system difficult. [27]

In conclusion, it appears that there are certain types of implementations where a model based approach would be favorable over either a rule based or frame based approach. Therefore, it can be implied that not one technique is the best one for all possible applications. As a result, certain current day implementations of expert systems have favored the use of a fourth type of implementation that combines the desired features of the other three.

Hybrid

Figure 11 graphically depicts the representation of a knowledge database using a hybrid approach. Hybrid implementations are necessary for more complex expert system designs, because they offer the ability to apply the most desirable features of the other three formats in a single knowledge base.

To illustrate this point, consider the use of a rule based format to represent all of the knowledge in a system. For systems where knowledge is based on a set of simple heuristics, this rule based approach is desirable. However, for complex system designs which have many different knowledge formats, a rule based approach results in "more complicated fragmentation and coding of knowledge, and causes the inference engine to

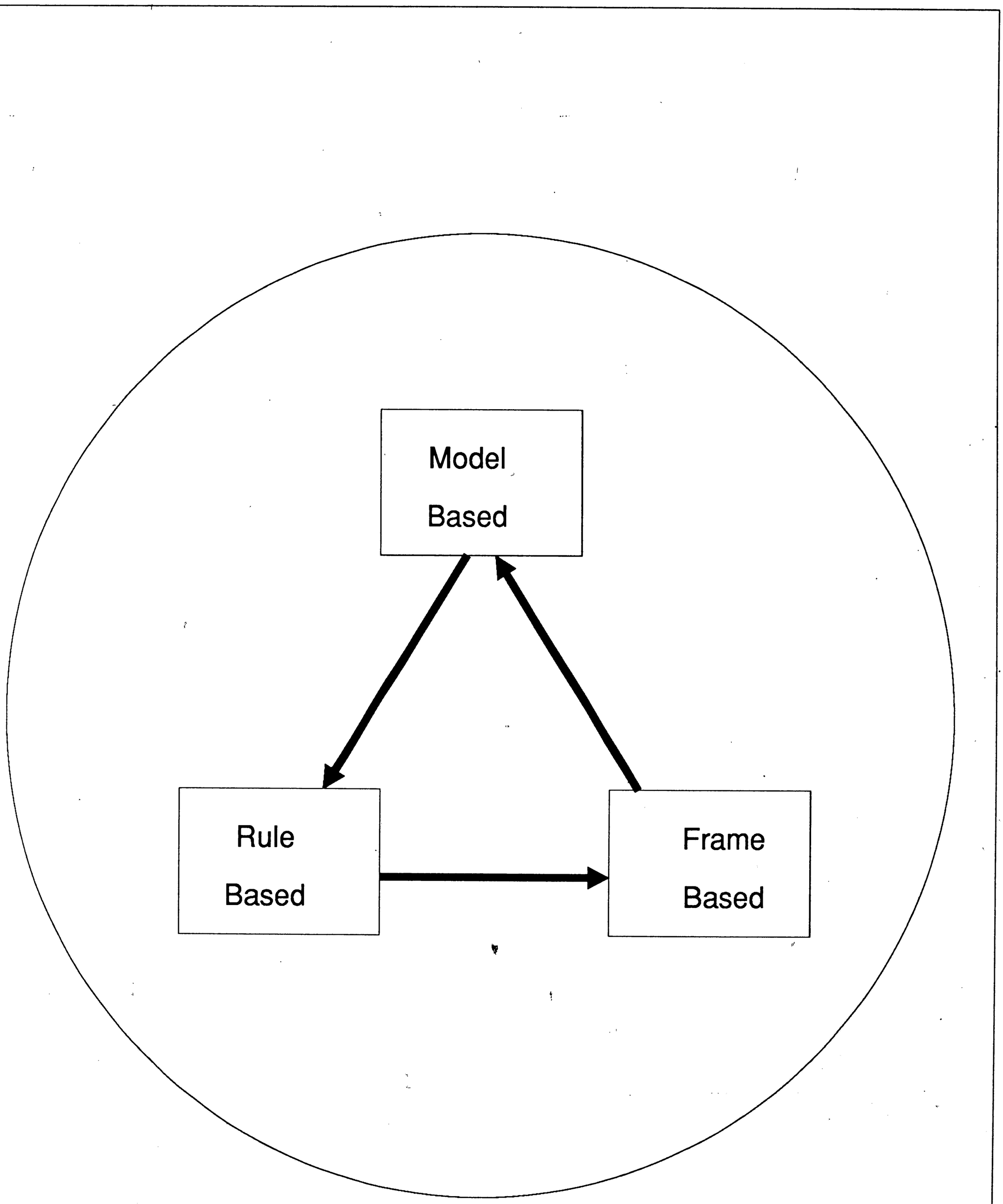


Figure 11 - Hybrid Knowledge Representation

reconstruct procedural knowledge through a heavy nondeterministic search process".[28] In this case, a knowledge base that utilizes different knowledge representation formats based on the nature of the knowledge is desired.

The development of a hybrid knowledge base entails identifying which knowledge format would best represent each piece of expert information, and determining how the different knowledge formats can be implemented in a single system. For example, a sample approach for a complex system might entail utilizing the various knowledge formats in the following manner:

1. Rule Based Representation - Used to define the "rule of thumb" knowledge for the system, and to define the behavior of simple system modules.
2. Frame Based Representation - Used to organize the information in the knowledge base. Utilization of various frame concepts such as inheritance allow for reductions in rule sets and behavioral models, resulting in faster processing of the knowledge base by the inference engine.
3. Model Based Representation - Used to represent the behavior and structure of complex modules that are part of the knowledge base.

The use of hybrid representation techniques to represent the knowledge in a system is the most flexible and appropriate approach to take when implementing complex systems which require multiple types of reasoning competence. Because the field of data communications is complex, the hybrid approach appears to be the most desirable.

3.2.3.3 Inference Engine Development

One the knowledge base development is completed, the software that will perform the decision making activities must be designed. These are developed in "modules" which are collectively called the **inference engine**.

The inference engine is similar to the human expert in the way it makes decisions with respect to a particular problem domain. The human decision making process involves scanning the knowledge stored in the brain, selecting information relevant to the particular problem and correlating the appropriate data to infer a solution. Similarly, the inference engine traverses a knowledge database of information, utilizes various processing techniques to locate and correlate relevant information, and determines a solution. The accuracy of the resulting solution is dependent on the completeness of the data in the knowledge base, and the capability of the inference engine to simulate the decision making ability of the human expert.

The procedures involved with traversing the database and correlating knowledge in an inference engine, (ie. "reasoning techniques") are a function of the way in which knowledge is represented in the system. The following sections describe various "reasoning techniques" that are applied to knowledge represented using rule-based, frame-based, model-based and hybrid formats.

Rule-Based Techniques

The reasoning processes applied to information represented as a set of production rules involves utilizing existing rules to infer new rules about the system that result in a solution to specific problems. In order to accomplish this, the database is traversed, and new rules are inferred using one of two techniques: **forward chaining** or **backward chaining**.

In **forward chaining** the inference engine utilizes existing rules and works forward from known predicates to derive as many consequents as possible.[29] For example, a system which has a production rule set containing the following rules

If A then B

If B then C

would utilize the production rules for known predicates "A" and "B" to infer a new production rule: "If A then C".

Conversely, in **backward chaining**, the inference engine works backward from hypothesized consequents to locate known predicates that would provide support [30] (ie. satisfy the consequent). The reasoning process begins with the inference engine taking the initial hypothesis about a problem and locating all rules having the hypothesis as a consequent. It then examines the predicate of each rule to determine if is true. If any of the predicates are true, the consequent is established. For instance, a particular expert system has the hypothesis "maintenance" for a consequent:

If system crash then maintenance

If Friday then maintenance.

When this system experiences an unexplained response time problem in the network, the system hypothesizes that maintenance is being performed. In order to verify this hypothesis, the two rules presented above would be examined. If either of these rules prove to be true, the system infers that its initial hypothesis was correct. This is an example of the use of backward chaining techniques to infer a solution in a system.

Frame-Based Techniques

The nature of frame-based knowledge representation lends itself nicely to a set of processes that could be developed to take advantage of the structure of the knowledge base. These processes are simply database traversal techniques that utilize the hierarchical

nature of the knowledge base to locate and correlate the relevant facts about a problem domain.

The development of a set of simple, high level procedures provide the traversal mechanisms for the inference engine. For example, figure 12, illustrates the frame-based structure of a sample expert system. This figure represents a frame-based database for a performance control system which distributes the knowledge required for decision making into six classes representing the particular type of problem with which the system is faced. In order to traverse this knowledge base, "case" statements similar to those depicted in figure 13 would be executed. The execution of these "case" statements, which determines to which class a particular problem belongs results in localizing the information required for a solution.

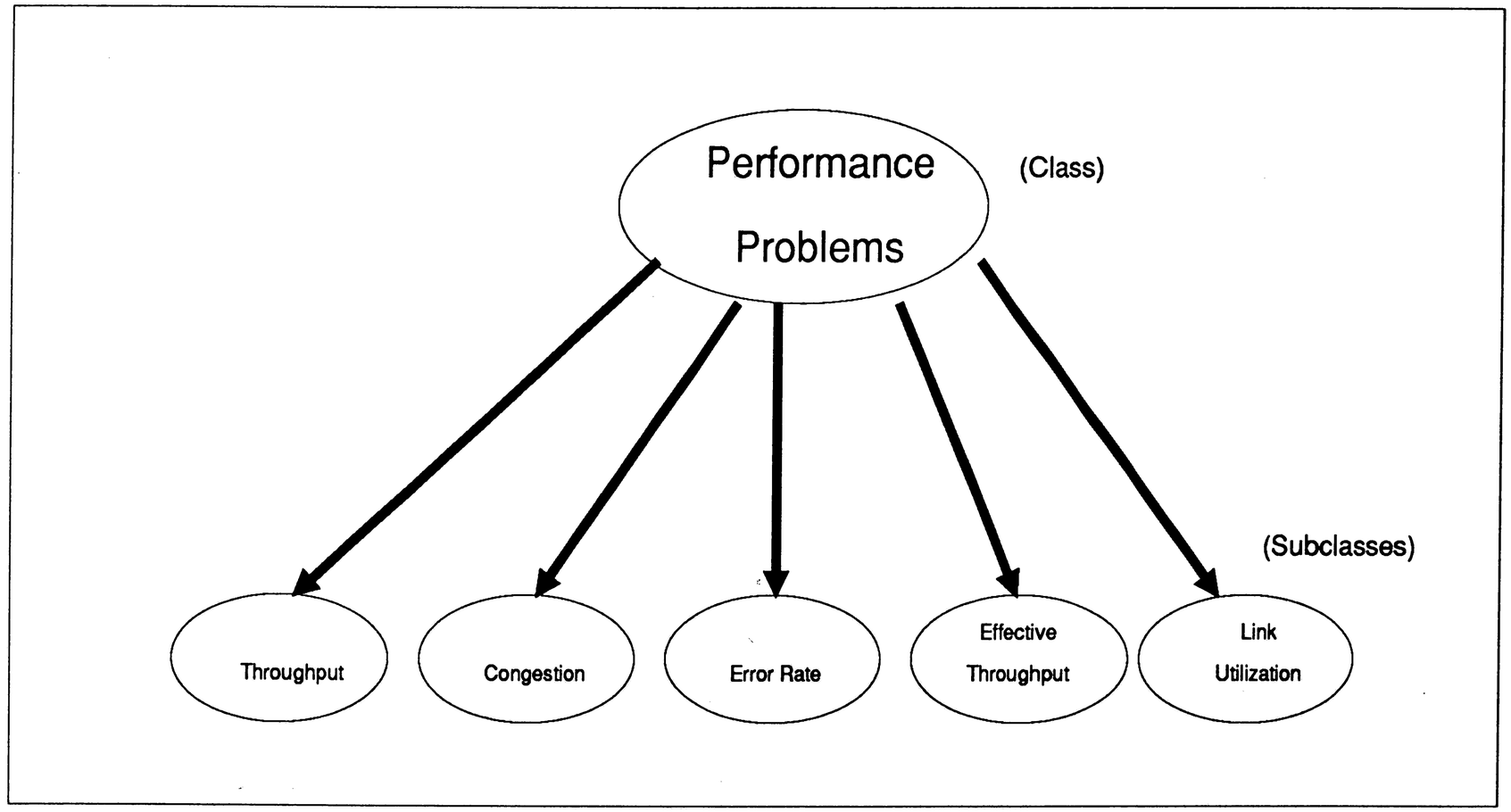
Through this use of high level traversal rules, the inference engine localizes the subset of information required to solve a particular problem. A continuous execution of these traversal procedures will ultimately result in the localization of the problem to an narrow problem domain, from which a solution can be inferred.

Model Based Techniques

The reasoning techniques utilized with model based representations apply the structural and behavioral models of the system to infer solutions. For a given problem, the general structure of a system is scanned and components relevant to the problem are identified. Similarly, the general behavioral model of the system is analyzed and a subset of applicable behavioral models is located. Following these localization procedures, the detailed reasoning process for model based representation begins.

When the inference engine performs detailed processing of the **structure** of the system, the following activities are performed:

1. Each of the components in the subset of localized data is removed from the system, one by one, and



(Frame-Based Knowledge Base)

Figure 12 - Frame Based Representation
For Theoretical Performance Control
Expert System

CASE: PERFORMANCE PROBLEMS

THROUGHPUT: ANALYZE_THROUGHPUT_CLASS;
CONGESTION: ANALYZE_CONGESTION_CLASS;
ERROR_RATE: ANALYZE_ERROR_RATE_CLASS;
EFFEC_THROU: ANALYZE_EFFEC_THROUGHPUT_CLASS;
LINK_UTILIZ: ANALYZE_LINK_UTILIZATION_CLASS;

FIGURE 13 - CASE RULES FOR FRAME-BASED INFERENCING EXAMPLE

effects on the system are noted.

2. Step one continues until the removal of a component results in the alleviation of the problem. At this point the system infers that the removed component is the one that was causing the problem.
3. Further analysis of the sub-structure of the particular component is performed when possible. This results in an even greater localization of the problem.
4. Ultimately, the inference engine localizes the problem to a small enough domain that a solution may be inferred.

The processing of the behavioral model of the system involves the activation of routines that simulate the behavior of the localized problem domain. The results of the simulation models are then compared to required values to determine if discrepancies exist. When inconsistencies between ideal and actual behavior exist, the problem is pinpointed, and solutions can be inferred.

Hybrid Techniques

Since the hybrid knowledge representation combines the formats of the rule based, frame based and model based approaches, it seems natural that the reasoning techniques to be applied to a hybrid knowledge base combine the techniques applied to the other three. However, since the formats for hybrid systems vary widely, specific reasoning techniques are difficult to identify. The decisions regarding the combination of known reasoning techniques for a hybrid system are up to the expert system designer.

In order to illustrate this point, consider an expert system,where knowledge is represented using hybrid techniques. This system might approach its decision making process using the following steps:

1. The problem domain is localized using model-based structural techniques.
2. The "shallow" knowledge of the expert, represented as a set of rules, is analyzed using rule-based techniques in an effort to locate a quick solution to the problem. Frame based organization of the rule base would aid this process, since it allows for faster traversal through the knowledge base.
3. If no problem can be inferred, the "deeper" knowledge of the expert, represented by various behavioral models is analyzed using model-based techniques.
4. If no solution is determined, but some discrepancies are noted, the entire process repeats itself.

The above example illustrates that the reasoning processes involved with hybrid representation are simply combinations of techniques, applied to the other knowledge formats. When combined efficiently, a powerful and effective inference engine mechanism can result.

3.2.3.4 Interface Development

The ability to communicate to the outside world, and to the various systems that comprise the problem domain, is one of the most important requirements of an expert system. Often, this requirement is difficult to satisfy, because of the different data

representation techniques used for each system. That is, while it is appropriate to represent data in a computer as a series of "0"'s and "1"'s, it would be less desirable to provide data in that format to the external user. The processes involved with converting and presenting the data to various users of the expert system are provided in the interface component.

The following describes the stages involved in the development of the expert system interface:

1. The input / output data requirements are determined. This involves identifying every data element that will be input or output from the expert system.
2. The presentation formats are defined. This involves specifying the various ways the data will be presented to the user.
3. Based on the information from steps (1) and (2) a conversion approach is determined that will satisfy user requirements.
4. The conversion and presentation procedures are implemented.

At the completion of these stages, an interface is available that provides the necessary communication facilities for an expert system.

3.2.4 Expert System Development Summary

The development of an expert system is a long term, complex effort. It is apparent that failure to take the time to perform all of the stages in the development process, would be a costly mistake, since each stage feeds off the information obtained in the previous stage. For example, if the knowledge acquisition stage was neglected,it would be very difficult to determine which knowledge representation format was the best

for a specific application. Thus, it is safe to say that successful implementations require taking the time to insure that each stage in the development process is carried out in a diligent manner.

3.3 CURRENT EXPERT SYSTEM IMPLEMENTATIONS

While it is important to discuss the theory and approach behind the development of an expert systems an equally important topic to be addressed is the current state of the art of actual expert system implementations as they relate to data communications. In order to accomplish this objective, three present day expert system implementations used in various sectors of the data communication world will be presented. Each of these examples were chosen for a particular purpose.

The first example, MAPCON[31], is an expert system for the configuration of MAP networks. This system was chosen in order to illustrate an application that involves an expert system development for a network management system component.

The second example, ITEST[32] (Integrating Testing Expert System for Trunks) was chosen because of its use of a hybrid knowledge representation. This example will illustrate the merits of utilizing a hybrid representation format in the development of complex expert systems.

The third example Troubleshooter[33], is a developmental system that is being used in a laboratory at Bell Labs. Troubleshooter was chosen because it illustrates the future direction of expert system technology - machine learning.

MAPCON

MAPCON is a knowledge base tool used in the configuration of MAP networks. It provides MAP network configuration expertise to network operators who are not fluent with MAP protocols, in order to aid the operator in setting up a MAP network. Given a list of data from the network administrator regarding expected traffic types and anticipated

amounts of data, MAPCON provides a list of configuration parameters and their suggested values. This list of parameters includes retransmission limits, and window times.

Actual implementation of MAPCON involves the development of a knowledge base comprised of a set of production rules and empirical formulas that represent how a MAP network expert would determine values for network parameters. The inferencing mechanisms that are applied to this knowledge base utilized **forward chaining** to traverse the rule base and generate values for configuration parameters. "The parameters that did not follow any rules are assigned using an empirical formula or derived from another parameter value"[340].

MAPCON is a good example of an expert system implementation aimed at enhancing the capability of a network management system. Many of the key issues that must be addressed in any expert system implementation of a network management system component were addressed during this implementation. Of these, the following considerations were most valuable:

1. Since the values of certain network layer parameters depended on the value of other layer parameters, dependencies between parameters must be identified.
2. Accurate determination of parameter values utilizes both analytical and synthetic reasoning techniques. As a result, the interaction between these reasoning techniques must be ascertained prior to the development of the expert system. [35]

The MAPCON implementation illustrates that the application of an expert system to a network management function is a viable option and can prove to be quite useful in

providing expertise where none exists. Many of the design considerations and approaches taken can be applied to performance management system implementations.

ITEST

The Integrated Testing Expert System for Trunks (ITEST) is a system aimed at automating the testing of communication line trunks. The designers of ITEST recognized the inherent problems with an expert system based solely on a rule-based representation - (eg. " although a typical rule based system can be used, it makes the knowledge representation unnatural by dispersing the available procedural knowledge into a fragmentary and declarative representation"[36]). Instead, they combined different knowledge representations together in one system.

The decision as to how certain types of knowledge would be represented was based solely on the nature or "makeup" of the data. Heuristic, non-structured knowledge was represented in a conventional rule base. Structured, procedural knowledge that described the behavior of the system was defined in a more algorithmic, "programmed" approach using model-based formats. A frame-based format was applied to represent the various screen formats used in ITEST. Since many of the screens shared the same format, ITEST utilized the frame-based inheritance feature to consolidate some of the knowledge in the database.

The actual reasoning processes applied in ITEST involved a combination of rule-based, frame-based and model-based techniques. Initial control of the system is provided through traversal of a model based control database, which dictates the system behavior and decides when to activate subtasks. Activation of subtasks effectively results in a change of state in the system which causes frame-based and rule-based reasoning processes to commence. Frame-based reasoning is applied to the screen format database in determining which screen should be displayed. Rule-based reasoning processes traverse

the information available on the current screen and filter out the information necessary for decision making.

In summary, ITEST puts the theory of hybrid expert systems into practice. It utilizes a top down approach to hybrid development, in that it depends on the more complex knowledge based format of model-based representation to control the system, and relies on rule-based and frame-based techniques to provide secondary support utilities. This project is an excellent example of the development of an expert system for a complex environment such as data communications. By taking advantage of the various desirable features of rule-based, frame-based and model-based representations, ITEST illustrates that a hybrid implementation can be an effective approach.

TROUBLESHOOTER

Troubleshooter is a system used to manage virtual circuit switches at AT&T Bell Labs. It determines problems in the circuit switches and advises a network administrator regarding corrective action.

Troubleshooter receives information regarding a specific problem from the administrator. It then utilizes model-based reasoning techniques to localize the components involved with the particular problem through analysis of information in the network management database. Troubleshooter then performs statistical analysis on this localized subset of components to isolate the most likely cause of failure, and, then triggers the appropriate diagnostics.

Following the detection and correction of the problem, Troubleshooter proceeds one step further, and provides a feature that sets it apart from most current day expert systems. In this step, Troubleshooter stores the cause of the detected error and the corrective action taken in a historical database, which is used by the system in correcting future problems. Through this technique, the Troubleshooter system effectively "learns"

from past mistakes and makes itself more intelligent. This capability which is known as machine learning appears to be the future direction of expert systems.

4.0 APPLICABILITY OF EXPERT SYSTEM TO PERFORMANCE MANAGEMENT

The discussion of performance management system implementations given in section 2.4.4 concluded that the major weakness in a current day performance management system is in the lack of a standardized, automated, performance control function. This function, which is outlined in figure 14, correlates the performance information received from the performance analysis function, to an optimization criteria, in order to insure that the network is operating efficiently. In so doing, the performance control function provides one of the most important capabilities of a performance management system - maintenance of optimum network efficiency.

The purpose of the following sections is to discuss the reasons why this component is not as well supported as the other performance management system components, and to suggest an approach, based on expert system technology, that can be applied to provide this capability.

The performance control function is a difficult one to automate using standard programming techniques, because of the heuristic nature of the decision making process involved. Section 3.0 noted that while many of the techniques involved in pinpointing performance problems utilize analytical techniques, the decisions regarding which detection procedures to execute when faced with a problem follow no known algorithm. As a result, present day implementations of the performance control function, as shown in figure 15, rely on a network operator, a human, to provide the necessary heuristics. When faced with a particular problem, this operator is responsible for accessing the performance database to determine current values for the performance measures of interest (step 1); and performing the analytical techniques necessary to determine corrective action (step 2).

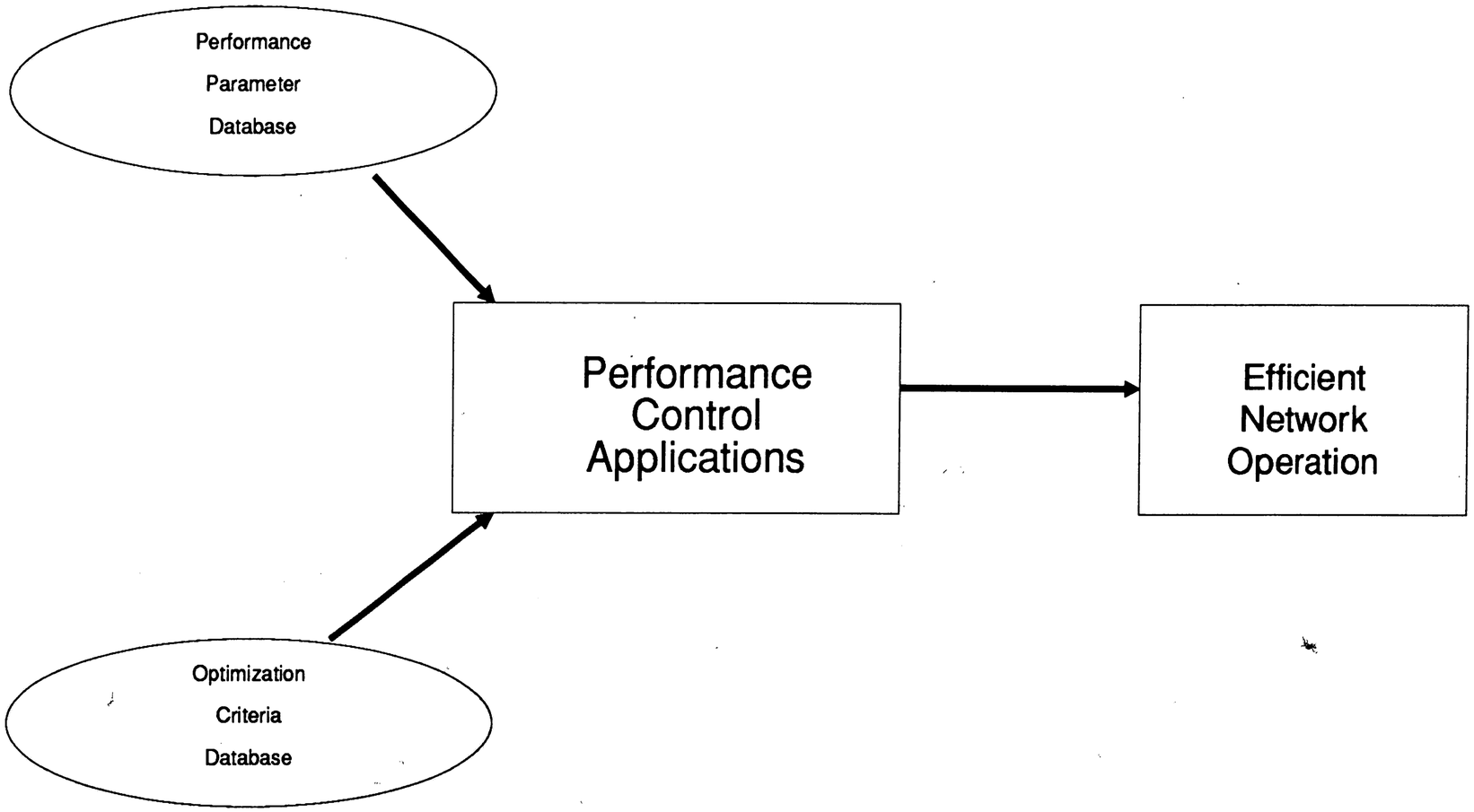
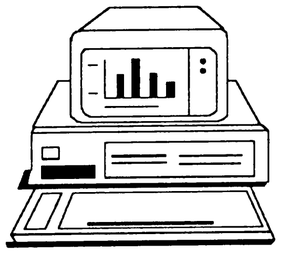


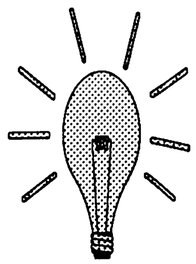
Figure 14 - The Performance Control Function

Presentation of
Performance Data



1. Performance Parameters are presented to the network operator

Operator Analysis
and Correction



2. The network operator determines if performance measures are meeting required levels. If not, the operator determines the problem and makes the required adjustments.

Figure 15 - Traditional Approach
To Performance Control

There are many problems associated with this current method. First, because of the need for operator intervention, this approach is not one that can be standardized and supplied as part of an "off the shelf" performance management system. Secondly, the reliability of this function is questionable, since it is dependent on the availability of network experts, typically scarce resources. Thirdly, even if the network operator is available, he may have an "off day", or make a mistake that a properly programmed computer would not make. Thus it can be seen that an automated approach to the performance control function would provide tremendous benefits to the overall operation of the network.

Successful implementation of an automated performance control capability, requires that the decision making processes typically performed by a network operator be simulated or emulated by a computer. Theoretically, this is a capability that can be provided by an expert system. As a result, the design of such a system would follow the principles of expert system design. This approach proves to be feasible for performance control, for the following reasons:

1.) Mechanisms are already in place, or are allocated for in many current day performance management systems, that allow for the addition of expert applications. For example, the Netview implementation provides a link to Knowledge Tool(see section 2.4.4) and, the MAP/TOP specification mentions the possibility of expert system additions to the management application process (see section 2.4.4)

2.) Expert systems never have an "off day", or make mistakes when programmed properly.

3.) Expert systems are able to comprehend, digest,

retain, and process much more data at a much faster rate than a human expert.

4.) Expert systems can replace a very scarce resource- human expertise.

The proposed development approach is outlined in figure 16. As illustrated, the actual performance control function is provided by a classical expert system. The knowledge base contains all of the information provided in the problem domain, including performance values, which are updated constantly from the performance database; operator expertise, which is represented in various knowledge formats; and optimization values, which are taken from the requirements of network clients and network managers. The inference engine provides the "reasoning" processes for the system. It detects efficiency problems in the system and determines which parameters require "tuning" in order to correct these problems. Subsequently, it modifies the tunable parameter database to cause the necessary change in network performance.

The system presented here goes one step beyond the detection and correction phases by providing a "machine-learning" capability. This is accomplished through the use of a feedback loop, which adds the cause of the latest error and corrective action taken to the knowledge base. Effectively, this results in increasing the intelligence of the knowledge base, because future decision making by the inference engine will make use of this new information.

The actual system development process follows all of the procedures detailed in section 3. Appendix A provides detailed coverage of all of these stages for the particular system design. In so doing, this appendix provides an excellent overview of the performance management function, and the stages in an expert system design, and can be used as a tutorial or quick reference guide for those interested in implementing an automated performance control function.

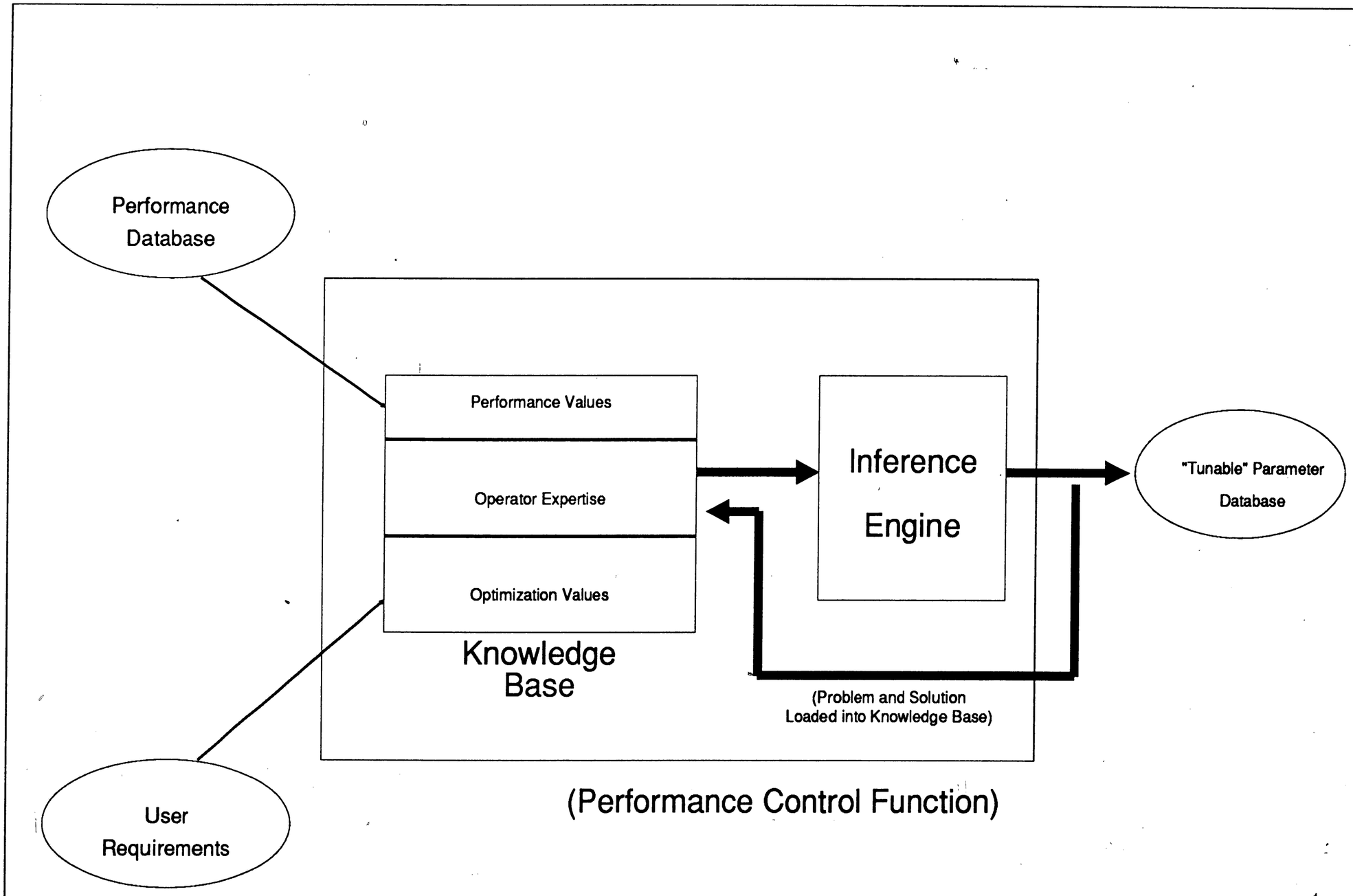


Figure 16 - Expert System Approach To Performance Control

5.0 CONCLUSION

The need for greater automation in data communications has necessitated an emphasis on identifying those areas that would benefit most from the increase in automation. The performance control component of a performance management system is one such area. Current day applications of this function are typically performed by a network operator because of the various heuristic procedures involved. This results in less efficient and consistent operations, because of the inherent limitations of the human operator.

I have shown that it is feasible to replace some of the functions performed by the human operator by an intelligent, automated system. Implementation of such a system requires knowledge of performance management system functions and expert system design theory. Each of these topics was discussed in detail in this document. As a result, this paper will serve as both a justification for the use of expert system design in performance management systems, and as an outline that can be followed in developing such a system.

While this paper can serve as a good starting point for the development of an expert performance control function , much work remains. Prior to any application or design decisions, more detailed research is required, specifically in the relatively new areas of hybrid representations and machine learning.

Expert systems implementations for data communication systems show a great deal of promise for the years ahead. As the price of computer memory drops and CPU speeds of computer systems increase, many of the problems associated with first generation expert system technology will diminish. This trend should continue for many years, up to the point where expert systems will become commonplace applications in the data communication world.

APPENDIX A

DEVELOPMENT APPROACH FOR AN EXPERT

PERFORMANCE MANAGEMENT SYSTEM

A development approach for an expert system that implements the performance control component of a performance management system is discussed in the following section. This approach follows the guidelines set forth in the paper for the development of an expert system. It is organized into sections which correspond to the development stages of an expert system. To review, the stages in the development process are: **feasibility study**, **requirements** and **design**. The feasibility of developing a system has already been discussed in section four. As a result, the following sections discuss the remainder of the development process, **requirements** and **design**.

Requirements

As is illustrated in figure 9 (page 39), the general requirements cycle is organized into three stages: identifying user requirements, identifying applicability of these requirements to expert systems and generating formal requirements.

In the case of performance control, the user requirement identification stage can be further subdivided into general problem statement definition, user group identification, and physical resource identification.

The general problem statement for performance control specifies that a capability of maintaining optimal network performance levels is required. The definition of "optimal network performance" is user specific. That is, one definition of optimal performance may place very low error rates as its top priority, while another might emphasize fast response time. As a result the system to be developed must allow for priorities to be placed on various performance measures. These priority values can be changed over the life of the system, which would effectively change the optimization criteria.

Identification of user groups involves specifying the various people who will be using the system, and identifying the ways in which they will be using this system. In the case of performance management, the users can be segmented into two groups - network clients and managers. Generally speaking the network managers are responsible for the

overall operation of the network, and network clients use the network to satisfy some application requirement. An overview of the performance management capabilities required by these two groups is provided in section 2.2.

Physical resource identification involves determining the resources available that can be used for the task, and subsequently, identifying any other resources that will be required for the system. For a performance control system, this process involves an identification of: the network resources; the performance management system that will provide performance reports; the computing resources available; and the computing resources required.

Once the user requirement stage is completed, a more detailed and careful analysis of the requirements is begun, in order to identify those requirements that are candidates for implementation using expert system design. Section 3.2.2 specified that those requirements that cannot be satisfied using known analytical techniques were the best candidates for expert systems. In the case of performance control, one example of this type of requirement is the overall run-time executive. This executive is responsible for deciding which process to perform next, based on current network conditions, in order to maintain optimum efficiency levels for the network. Since many of the decisions that the executive makes are based on heuristic knowledge, an expert system approach is desirable.

Following the localization of the user requirements to those necessitating an expert system approach, the formalizing of these requirements occurs. This process involves the development of formal requirement specifications that are typically presented to a review committee for approval. For an application as complex as performance control, it is highly recommended that this process be carried out carefully, so that a formal, detailed requirements specification can be available to the system designers.

Design

The stages in the design process, as illustrated in figure 8 (page 34) are as follows: **Knowledge Acquisition**, **Knowledge Representation**, **Inference Engine Development**, and **Interface Development**. The following sections discuss these stages as they apply to the performance control problem.

In the **knowledge acquisition** stage, the information necessary to simulate the decision making capability of system experts is obtained. In the case of performance control, network experts, who are knowledgeable of the operation of the network, and the operation of the current performance management system are interviewed by several knowledge engineers in an effort to obtain this information. The exact questions that are asked of them relate to the ways in which network efficiency is maintained and the ways in which problems are detected and corrected.

All of the information acquired during the knowledge acquisition phase is sorted and determinations are made with regards to how the knowledge will be represented. For performance control, the most intelligent choice for a **knowledge representation** approach is a hybrid approach. For a discussion of the reasoning behind this choice, reference sections 3.2.3.2 (Hybrid representation) and 3.3 (ITEST). A hybrid representation combines information specified in other knowledge formats, including rule-based, frame-based and model-based formats. In the case of performance control, the following is a possible hybrid representation approach:

1. A **rule-based** format is utilized to represent
   the general decision making process for the network
   operator. These rules are used to determine
   the next course of action at a particular state in
   the system. Once in a particular state, further
   rules are utilized to determine subsequent

actions.

2. **Model-based** formats are utilized to represent the more detailed network performance models (eg. queuing models) and also to define the structure of the network (eg. defining all of the possible paths for data to travel).
3. Organization of the various rule sets and models is provided using **frame-based** techniques. The information provided is organized according to current operational states in the system. For example, frames are used to separate the rule base into sections corresponding to the current problem detected in the system.

The **inference engine** stage involves the development of the processes involved with traversing the knowledge base to make decisions regarding a specific problem. For performance control, these processes include the high level executive tasks responsible for the control of the system. The executive program flow, which is illustrated in figure 17, involves operating in one of two states: "normal maintenance" and "problem detection/ correction". Normal maintenance procedures are responsible for monitoring the values of performance parameters and comparing these parameters against optimization values. If problems are detected, the executive activates its problem detection and correction procedures.

During "normal maintenance", the program flow for the executive follows a **rule based** approach based on the state of the system. Upon detection of a discrepancy, more detailed rule traversal procedures located in "problem detection / correction" modules are

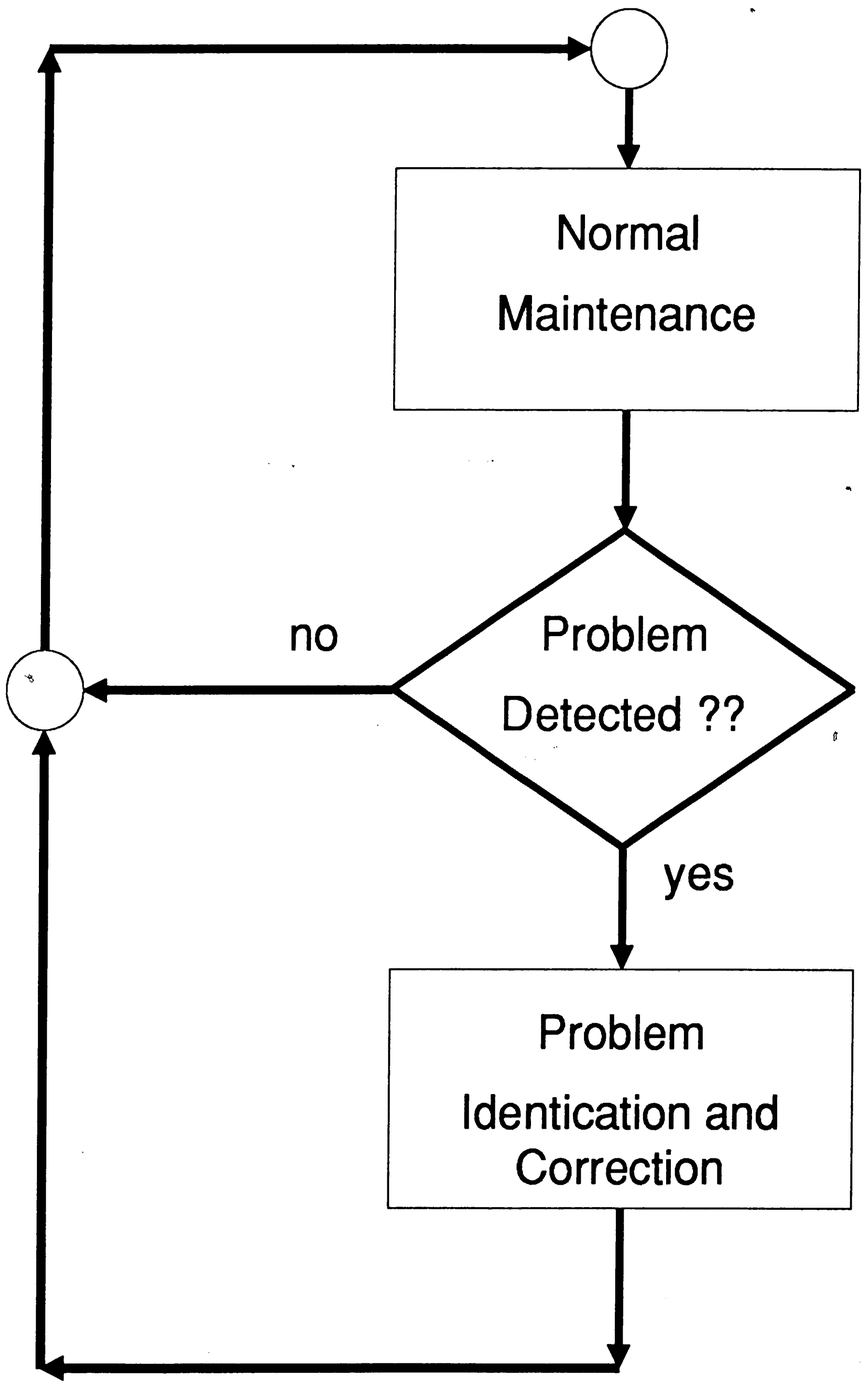


Figure 17 - Expert System Program
Executive Flow

applied. When the rule base information has been exhausted, more detailed analysis is performed using various behavioral and structural models.

Behavioral analysis, involves procedures that apply formulas and empirical algorithms to simulate the behavior of parts of the system. As applied to performance control, this would include various queuing theory algorithms, such as the one described in example 2. **Structural analysis,** as it applies to performance control, involves localizing potential problems by analyzing the various pieces of equipment in the network that can be the cause of a detected problem. Details of the model- based approach is described in section 3.2.3.

In both of the main executive states, **frame based reasoning** techniques are employed to allow for easier traversal through the various databases. Frames organize the rule base with respect to possible problems in a network. For example, when a problem with throughput is detected, only those rules which have some relation to throughput (ie. they are in the same "class") will be searched. See figure 12 and the subsequent explanation on page 53 for more information on this topic.

The development of expert system interfaces completes the design cycle. These interfaces are required in order for the expert system to access data from external systems. Figure 18 illustrates that there are two required interfaces for an expert performance control system.

The first interface allows communication between the expert system and the performance management system. This link is necessary to permit access to the performance database by the expert system. The second interface provides a communications path between the expert system and a configuration management system, which gives the expert system access to the "tunable" network variables database.

The complexity of design for this component is dependent on the characteristics of the two external system interfaces. If these systems provide utilities that allow procedural access to their databases, the design effort is fairly simple. However, if no

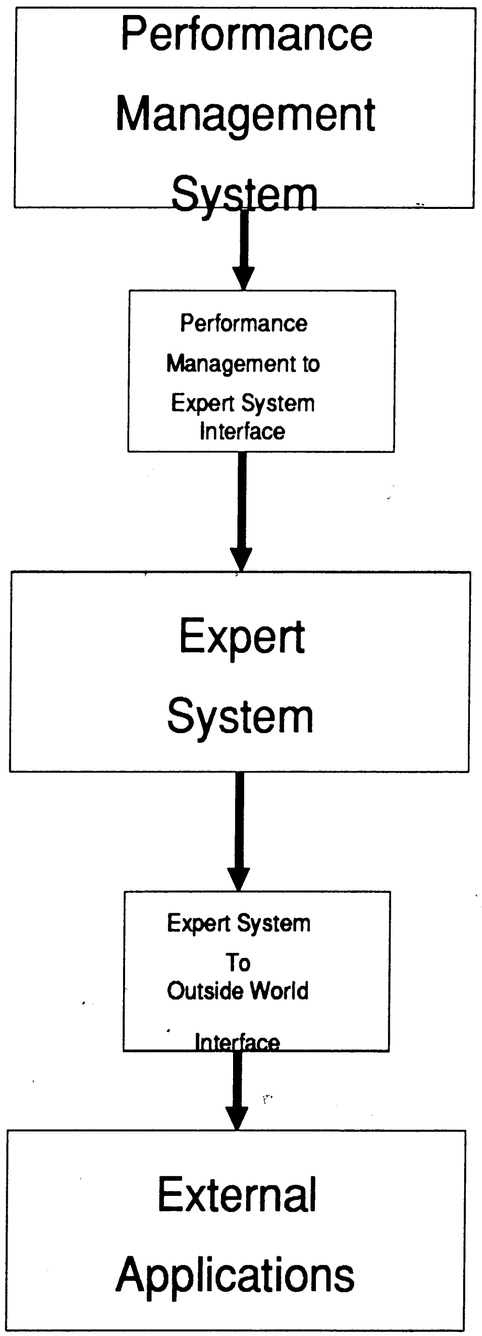


Figure 18 - Required Expert System Interfaces

such features are available, then interface modules would have to be developed to provide the database access capability. This task is usually difficult, because of incompatibilities that exist between computer systems.

REFERENCES

1. Chao, Hui-Chu. "Network Management"
Internal Memorandum from MAP/TOP Network Management Committee, (November 1986) p. 1.

2. Muralidhar,K.H. "Performance Management - Measures,Analysis,Control and Optimization" IEEE 11th Conference of Local Computer Networks (1986) p.20.

3. MAP/TOP Network Management Task Group. MAP/TOP 3.0 Network Management Requirements Specification (March 1987) p.49.

4. IEEE Project 802. Draft IEEE Standard 802.1 - Overview, Internetworking and Network Management. Supplement B: Network Management (December 1987) p. 13.

5. MAP/TOP Network Management Task Group. MAP/TOP 3.0 Network Management Requirements Specification (March 1987) p.183.

6. Bertsekas,D and Gallagher,R. Data Networks. New Jersey:Prentice Hall, 1987.

7. MAP/TOP Network Management Task Group. MAP/TOP 3.0 Network Management Requirements Specification (March 1987) p.51.

8. Muralidhar,K.H. "Performance Management- Measures,Analysis, Control and Optimization" IEEE 11th Conference of Local Computer Networks (1986) p.22.

9. Ahuja,V. Design and Analysis of Computer Communication Networks. McGraw Hill:New York, (1982) p 194.

10. Bertsekas,D and Gallagher,R. Data Networks. New Jersey:Prentice Hall, 1987.

11. Denton, R. Data Communication Notes - for ECE 413. Lehigh University, (1989) p 106.

12. Denton, R. Data Communication Notes - for ECE 413. Lehigh University, (1989) p 25.

13. MAP/TOP Network Management Task Group. MAP/TOP 3.0 Network Management Requirements Specification (March 1987) p.18.

14. MAP/TOP Network Management Task Group. MAP/TOP 3.0 Network Management Requirements Specification (March 1987) p.106.

15. MAP/TOP Network Management Task Group. MAP/TOP 3.0 Network Management Requirements Specification (March 1987) p.111.

16. MAP/TOP Network Management Task Group. MAP/TOP 3.0 Network Management Requirements Specification (March 1987) p.18.

17. IBM Corporation. NETVIEW Release 3. Internal IBM release. September 1988.

18. IBM Corporation. NETVIEW Performance Monitor. White Plains, NY . March 1989

19. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.4.

20. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.6.

21. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.7.

22. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.72.

23. Macleisch, K. "Mapping the Integration of Artificial Intelligence into Telecommunications" IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p892.

24. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988).

Badiru, A. "Expert Systems and Industrial Engineers: A Practical Guide to a Successful Partnership" Computers Ind. Engng, Vol 14, No 1. pp 1-13 (1988)

25. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.204.

26. Khan, N., Callahan, P., Dube, R., Tsay, J. and Van Dusen, W. "An Engineering Approach to Model-Based Troubleshooting in Communication Networks" IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p793.

27. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.300.

28. Liu, D.D. and Pelz, Dennis. "I-Test: Integrated Expert System for Trunks". IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p801.

29. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.196.

30. Walters, J. and Nielsen, N. Crafting Knowledge Based Systems. New York. John Wiley and Sons., (1988) p.196.

31. Muralidhar, K.H. and Irish, B. "MAPCON: An Expert System for Configuration of MAP Networks". IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p869ff.

32. Liu, D.D. and Pelz, Dennis. "I-Test: Integrated Expert System for Trunks". IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p800ff.

33. Bernstein, L. "Expert Systems in Network Management - The Second Revolution" IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p786.

34. Muralidhar, K.H. and Irish, B. "MAPCON: An Expert System for Configuration of MAP Networks". IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p872.

35. Muralidhar, K.H. and Irish, B. "MAPCON: An Expert System for Configuration of MAP Networks". IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p870-871.

36. Liu, D.D. and Pelz, Dennis. "I-Test: Integrated Expert System for Trunks". IEEE Journal on Selected Areas in Data Communications, Vol 6, No. 5 (June 1988) p801.

VITA

Peter F. Carroll Jr. received his B.S. degree in Electrical Engineering form Lafayette College in 1984 and his M.S. degree in Computer Science from Lehigh University in 1990.

He is a member of the Systems Integration Division at the Naval Air Development Center, where he is involved with avionics simulation. Mr. Carroll's technical interests include system development, network communications and expert systems.

Mr. Carroll is the son of Mr. and Mrs Peter Carroll of Staten Island, NY. He resides in Doylestown, PA with his wife Sally Jo.