DESIGN OF AN INQUIRY SUPPORT SYSTEM FOR MANAGERIAL PROBLEM DIAGNOSIS

by

HWALSIK CHANG, B.B.A., M.B.A.

A DISSERTATION

IN

BUSINESS ADMINISTRATION

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY

Approved

_____
Chairperson of the Committee

_____

_____

_____

_____

Accepted

_____
Dean of the Graduate School

December, 1993

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my committee chairperson, Dr. Jim Burns, for his encouragement, patience, and guidance. Without his guidance, this dissertation would not have been possible. I would like to thank my committee members, Dr. Larry Austin, Dr. David Hale, Dr. George Kasper, and Dr. Henry Wright, for their constructive criticisms and support. I would also like to thank Keith Blair for helping me in data collection.

I have always been indebted to my parents for their continuous support, confidence, and inspiration. My most special appreciation goes to my wife, Whasook, for her love, understanding, and sacrifice. I also thank my sons, Yoon-Soo and Yoon-Sung, for keeping me motivated.

This research was supported by the State of Texas Higher Education Coordinating Board Advanced Technology Program Grant No. ATP 1989-003644018.

CONTENTS

ABSTRACT

Problem diagnosis, the process of discovering and constructing the causal structure of a problem, is one of the most critical aspects of decision making. Management information system (MIS) research, as a discipline concerned with improving managerial decision making, must provide adequate support for this critical and difficult decision process.

The purpose of this research was to formulate a conceptual framework for developing problem diagnosis support systems (PDSS), design PDSS based on the framework, and validate the design based on prototype system implementation. The conceptual framework identified essential elements of PDSS, delineated logical steps in PDSS development, specified the individuals' roles, and discussed major issues to be addressed for PDSS development. The framework indicated that PDSS must combine user-driven and system-driven approaches to support semi-structured or unstructured problem diagnosis process. Thus, PDSS must have both cognitive and normative elements. Furthermore, PDSS must reduce, or at least attempt to reduce, the errors, biases, and uncertainties in conceptualizing a problem. The framework indicated that PDSS must structure complex causal relationships, support the entire problem diagnosis process, and maintain extensive interactions with the users.

This research identified various activities inherent in, and essential for, successful problem diagnosis. Also identified are a variety of views necessary to support the problem diagnosis activities. This research integrated structural, statistical, and rule-based modeling for problem diagnosis support. The support functions are designed from both cognitive and normative perspective. This research incorporated the inquiry guaranteeing concepts into PDSS design.

The framework and the conceptual design of PDSS provided a sound basis for developing more powerful, comprehensive, and cooperative PDSS. The prototype system, tested with two different diagnostic problems, demonstrated that it can overcome the limitations of previous PDSS by integrating structural, statistical, and rule-based modeling approaches, supporting the entire problem diagnosis process, and achieving a more cooperative inquiry. The key concepts synthesized in this research include the cooperative system concept, multiple causal modeling approaches, the process-oriented decision support approach, and the inquiry "guaranteeing" concepts.

The framework and the conceptual design of PDSS provided a sound basis for developing more powerful, comprehensive, and cooperative PDSS. The prototype system, tested with two different diagnostic problems, demonstrated that it can overcome the limitations of previous PDSS by integrating structural, statistical, and rule-based modeling approaches, supporting the entire problem diagnosis process, and achieving a more cooperative inquiry. The key concepts synthesized in this research include the cooperative system concept, multiple causal modeling approaches, the process-oriented decision support approach, and the inquiry "guaranteeing" concepts.

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Decision making constitutes the core of managerial activities (Simon, 1960). This process, in general, consists of problem recognition and diagnosis, alternative design and evaluation, and solution selection and implementation (Mintzberg et al., 1976). In a logical sense, problem diagnosis follows the problem recognition phase in which a decision maker perceives the existence of problem symptoms (Pounds, 1969; Schoennauer, 1981). Once symptoms are recognized, the decision maker, based on the available information and his or her knowledge and experiences, needs to understand the underlying causal structure that has produced the symptoms (Lyles, 1982; Dutton et al., 1983).

Problem diagnosis, the process of discovering and constructing the causal structure of a problem, is one of the most critical aspects of decision making (Ackoff, 1974; Mintzberg et al., 1976; Said, 1978; Einhorn and Hogarth, 1982; Dery, 1983; Schwenk and Thomas, 1983; Smith, 1988; Adams et al., 1990). Because problem diagnosis frames the problem in the early stages of decision making, it can potentially affect the direction of all subsequent courses of the decision process (Mintzberg et al., 1976; Volkema, 1983; Smith, 1989). Addressing a problem based on symptoms, or without an adequate understanding of the problem structure, often results in solving the wrong problem (Pounds, 1969; Kiesler and Sproull, 1982; Yadav and Korukonda, 1985). This phenomenon is known as committing "an error of the third kind" (Mitroff and Featheringham, 1974), i.e., framing a wrong hypothesis. Because the way a problem is defined limits the way the problem can be solved (Getzels, 1979; Lyles, 1987), inadequate problem diagnosis usually leads to ineffective decision making and damaging

actions with significant organizational consequences (Schoennauer, 1981; Kiesler and Sproull, 1982; Volkema, 1983).

Problem diagnosis is a complex decision process requiring various activities, including problem boundary establishment, key element identification, model formulation, and model validation (Mitroff et al., 1979; Mason and Mitroff, 1981; Lyles, 1982). The process of inquiring into the problem situation in search of the causal structure is a difficult task, because most significant managerial problems are embedded in highly interdependent systems; and problems themselves are often too complex to be comprehended as a whole (Ackoff and Emery, 1972; Lyles and Mitroff, 1980; Ackoff, 1981; Checkland, 1981). In other words, managerial problems tend to have unclear problem boundaries (Reitman, 1964; Mitroff et al., 1979), multiple symptoms and multiple causes mixed with noisy data (Anderson and Janson, 1979), ambiguous, uncertain, and complicated causal relationships (Einhorn and Hogarth, 1986), and no obvious technique to test the causal relationships (Lyles, 1987; Paradice, 1986).

For such complex and ill-structured problems, successful problem diagnosis is largely dependent on the individual's cognitive processing, knowledge, and experiences (Lyles, 1982; Volkema, 1983). Unfortunately, individuals have critical cognitive limitations in dealing with complex problems (Miller, 1956; Simon, 1957) and are subject to various cognitive biases (Tversky and Kahneman, 1974; Hogarth, 1980; Sage, 1981; Hogarth and Makridakis, 1981). Furthermore, their ability to learn from experience deteriorates rapidly as the complexity of the problem increases (Argyris, 1976). The decision makers' unsatisfactory performances during problem diagnosis have been reported by many decision theorists and cognitive psychologists (Kiesler and Sproull, 1982; Dery, 1983; Schwenk and Thomas, 1983; Barnes, 1984; Meredith, 1984; Smith, 1989).

Given the importance of problem diagnosis in decision making and the difficulties that decision makers have in that process, there is a need to design support systems that extend the decision maker's cognitive processing, reduce biases, and facilitate learning. In this spirit, decision theorists have proposed a number of problem diagnosis aids that range from theoretical methodologies (Checkland, 1976; Mitroff and Betz, 1972; Ackoff, 1974; Mason and Mitroff, 1981) to empirical heuristics (Kepner and Tregoe, 1981; Meredith, 1984; Ramakrishna and Brightman, 1986).

Surprisingly, problem diagnosis has received relatively little support from management information systems (MIS) and decision support systems (DSS) research. MIS has made major contributions to the problem recognition phase of decision making by providing clues about the existence of the problem through periodical and exceptional performance reports. However, MIS provides little information for problem diagnosis, e.g., the information about the factors responsible for the performance deviation (Said, 1978; Blanning, 1984).

On the other hand, DSS research has emphasized providing solutions, or the end-products, rather than supporting the decision process that leads to the solutions (Landry et al., 1985; Parker and Al-Utaibi, 1986; Cats-Baril and Huber, 1987; Weber and Konsynski, 1987). Most DSS assume that the decision problems have already been recognized and diagnosed (Ackoff, 1974; Naylor, 1982; Courtney et al., 1987). In other words, DSS have been "oriented toward products, or decisions, rather than toward the entire range of processes that produce those decisions" (Weber and Konsynski, 1987, p. 64).

In recent years, however, a line of research has emerged with a special emphasis on computer-based support for managerial problem diagnosis. These systems have a variety of capabilities: discovering causal relationships from

empirical data (Billman, 1989), providing a tool to represent and examine causal relationships (Pracht, 1984; Ramaprasad and Poon, 1985; Khazanchi, 1991), evaluating user-asserted causal relationships with empirical data (Paradice, 1986; Paradice and Courtney, 1986), searching for most probable causes for a given set of observed symptoms based on predetermined causal models (Ata Mohamed, 1985; Courtney et al., 1987; Jung, 1990), and providing multiple views of a problem situation (Baldwin, 1989). These systems are built upon the premises that DSS should support the entire range of the decision process, and that problem diagnosis is one of those unsupported decision process components (Pracht, 1984; Ata Mohamed, 1985; Paradice, 1986; Courtney et al., 1987; Baldwin, 1989).

1. 1. Problem Statement

MIS/DSS are disciplines concerned with improving managerial decision making (Keen and Scott Morton, 1978; Sprague and Carlson, 1982) and have unique qualifications to provide such improvement with computer-based decision support. One major problem in DSS research, however, is that the managerial decision process that DSS is intended to support has been largely ignored in DSS design (Klein and Hirschheim, 1985; Parker and Al-Utaibi, 1986; Weber, 1986; Elam and Konsynski, 1987; Weber and Konsynski, 1987; Nunamaker et al., 1988). Problem diagnosis support systems constitute an attempt to alleviate this problem by supporting one of the most neglected phases of the decision making process. Although these systems, limiting their scope to problem diagnosis or one specific decision phase, need to be integrated into a more comprehensive DSS environment, they are significant as the first step toward supporting the overall decision making process.

The previous problem diagnosis support systems (PDSS) have a number of limitations, however. First, the majority of these systems (e.g., Pracht, 1984; Ramaprasad and Poon, 1985; Khazanchi, 1991) employ a simple modeling construct,[1] the structural model, as the sole basis of approaching problem diagnosis. Structural models, or cognitive maps (Axelrod, 1976) are designed to represent whether the causal relationships between pairs of variables exist. Because of its simplicity, the structural modeling construct has critical limitations in representing the complex relationships observed in the management domain (Paradice, 1986; Baldwin, 1989). It is unlikely that managerial problem diagnosis can be successfully carried out based on this simple modeling construct.

Second, the previous PDSS have focused on different phases of the problem diagnosis process much in the same fashion that MIS and DSS have focused on different phases of the overall decision making process. For example, Billman (1989) and Paradice (1986) emphasize causal model formulation; Pracht (1984) concentrates on causal model representation; Ramaprasad and Poon (1985) and Khazanchi (1991) focus on causal model comparison and integration; Jung (1990) underscores the importance of causal model refinement; and Ata Mohamed (1985) highlights the importance of causal model applications. Thus, individual systems cannot support the entire problem diagnosis process.

Third, the previous PDSS tend to emphasize either a user-driven approach or a system-driven approach to problem diagnosis. For example, the systems developed by Pracht (1984), Ramaprasad and Poon (1985), and Khazanchi (1991) are essentially user-driven. These systems are memory-aid systems with minimum

---

[1] A modeling construct refers to the grammatic view of a modeling technique. For example, the modeling construct of the linear programming technique states that (parts of) the world can be modeled with a linear objective function and several linear constraints.

inferencing capabilities. On the other hand, those systems developed by Ata Mohamed (1985), Billman (1989), and Jung (1990) are mainly system-driven, requiring almost no interaction with users. An effective PDSS, however, must combine both user-driven and system-driven problem diagnosis (Keen, 1987; Hale and Kasper, 1989; Sundstrom, 1991).

In summary, the computer-based managerial problem diagnosis support research, to date, exhibits at least three major limitations. First, it relies on a very simple modeling construct. Second, it focuses on different phases of the problem diagnosis process. Third, it emphasizes either a user-driven approach or a system-driven approach.

Problem diagnosis is essentially an inquiry process, i.e., the process of creating knowledge about problem situations. A fundamental premise of DSS research is that decision making can be enhanced by forming a cooperative inquiry community between a decision maker and a computer-based support system (Licklider, 1960; Hale and Kasper, 1989). To form a cooperative inquiring community, the systems must structure complex causal relationships, support the entire problem diagnosis process, and maintain extensive interactions with users. The previous PDSS are not satisfactory in these respects.

1.2. Objective of the Research

The primary objectives of this research are: (1) to formulate a conceptual framework for developing managerial PDSS; (2) to design a PDSS that overcomes the limitations of the previous PDSS; and (3) to validate the conceptual design through implementing a prototype system and evaluating its capabilities. The conceptual framework must provide a theoretical basis to organize the fragmented

previous PDSS research, identify key design issues for the current research, and furnish valuable guidelines for future research in this area.

From a pragmatic viewpoint, the PDSS developed in this research should be more powerful, comprehensive, and cooperative than the previous PDSS. First, a powerful PDSS means that the system should not be bounded by a simple modeling construct, but utilize a variety of modeling constructs robust enough to represent complex relationships. Second, a comprehensive PDSS means that the system should not limit its support to a particular problem diagnosis activity, but support the entire problem diagnosis process. Finally, a cooperative PDSS indicates that the system should combine both user-driven and system-driven approaches instead of choosing one approach over the other.

The major objectives of this research were:

1. To formulate a conceptual framework for developing managerial PDSS:
   - 1.1. To clarify and adopt the "cooperative inquiry system" concept as a basis of PDSS design;
   - 1.2. To establish requirements for PDSS
     - 1.2.1. To establish the knowledge structure requirements of PDSS;
     - 1.2.2. To establish the problem processing requirements of PDSS;
     - 1.2.3. To establish the interaction requirements of PDSS.
2. To design, develop, and implement a PDSS that overcomes the limitations of the previous PDSS:
   - 2.1. To develop a knowledge base representing various causal models;
   - 2.2. To develop a set of problem diagnosis support functions;
   - 2.3. To develop a PDSS that combines both user-driven and system-driven problem diagnoses.

3. To validate the design as to its feasibility, soundness, and generalizability:
   3.1. To implement a prototype system and apply it to different problem diagnosis situation, i.e., testing feasibility and generalizability;
   3.2. To validate the design through evaluating the system's capabilities.

1.3. Overview of the Research

This research can be best characterized by its five key concepts. The key concepts are: (1) problem diagnosis; (2) multiple modeling constructs; (3) process-orientation; (4) cooperative inquiry; and (5) a support system.

First, problem diagnosis indicates that the scope of this research is limited to the early stages of the problem solving process. The reason for limiting the scope is to develop a deep understanding of the problem diagnosis process and explore in depth the PDSS design issues. Evidently, a DSS requires a variety of resources, such as a data base, a model base, and a repository of analysis techniques. By limiting the scope to problem diagnosis, this research can develop a detailed design of the support system: for example, to the level of designing and implementing various support functions. This in-depth analysis contrasts with the "generalized DSS" approach (Bonczek et al., 1981), which remains a theoretical concept by not being able to demonstrate its design details. The first major task of this research was to formulate a conceptual framework for developing PDSS.

Second, multiple modeling constructs imply that a PDSS should not impose onto the decision makers a modeling construct that is too simple to represent problem situations. It appears that managerial problem diagnosis may utilize at least three modeling constructs: structural models, statistical causal models, and rule-based models. Structural models are useful to represent whether the causal relationships between pairs of variables exist (Axelrod, 1976; Sage, 1977).

Statistical causal models are valuable to describe the causal relationships among quantitative variables. Rule-based causal models are necessary to describe the relationships among qualitative variables or events (Newell and Simon, 1972). The second major task of this research was to establish the knowledge structure requirements of the PDSS and design a knowledge base capable of representing these causal models.

Third, process-oriented decision support indicates that problem diagnosis is a process consisting of various activities, and that PDSS should provide support for each of these activities. Managerial problem diagnosis involves various activities such as problem perception, goal formulation, symptom identification, model formulation and validation, and diagnosis generation and evaluation. The third major task of this research was to establish the processing requirements of the PDSS and develop a set of problem diagnosis support functions.

Fourth, inquiry means that managerial problem diagnosis is a process of creating knowledge about problem situations, and that the process is surrounded by uncertainties and subject to various errors. One way to reduce the uncertainty and errors is through combining user-driven inquiries with computer-based inquiries. Another way to reduce the uncertainties and errors is to incorporate the "guaranteeing" concepts (Churchman, 1971) into the PDSS. For example, a PDSS may: (1) invoke several induction algorithms to see whether they form a Lockean consensus; (2) verify models to maintain a Leibnitzian consistency; (3) apply Kantian multi-models and evaluate their performances; (4) present conflicting Hegelian arguments; and (5) eventually, support Singerian model refinement. The fourth major task of this research was to determine the interaction requirements, incorporate the inquiry "guaranteeing" concepts into a PDSS design, and develop a PDSS that allows interactive problem diagnosis.

Finally, a support system indicates that the final evaluation of this design research should be performed in terms of the software system implemented. The final major task of this research was to develop a prototype system based on the conceptual design, apply the system to several diagnosis problems, and evaluate the validity of the design based on the system's performance.

1.4. Significance of the Research

The primary significance of this research is that it formulates a conceptual framework for developing effective support systems for managerial problem diagnosis. Decision scientists have long considered problem diagnosis as the most critical phase of the decision making process (Mintzberg et al., 1976; Schwenk and Thomas, 1983). In a recent study, Adams et al. (1990) reaffirm that problem identification (diagnosis) is the most critical and the most difficult decision process that requires the most thought of DSS users. Without a doubt, supporting such a critical and difficult decision process is of significant importance.

First, the framework is significant in terms of its paradigm orientation. Underlying the conceptual framework is a "cooperative inquiry system" concept. The cooperative inquiry concept encompasses both cognitive and normative support. Traditionally, PDSS research has emphasized either cognitive or normative support. This research contributes to PDSS research by establishing a sound theoretical foundation for PDSS research. In addition, the conceptual framework identifies essential elements of PDSS, delineates logical steps in developing and using a PDSS, specifies the roles played by individuals, and discusses major issues to be addressed for PDSS development and use. By organizing these important issues, the conceptual framework facilitates a systematic development of PDSS research.

Second, this research is significant in that it relates and combines structural, statistical, and rule-based modeling techniques for the purpose of problem diagnosis support. The majority of the previous PDSS support only structural modeling. Structural models have serious limitations in representing complex causal relationships. In general, statistical models can overcome many limitations of structural models. Statistical models, however, are useful only for examining quantitative relationships. The necessity of rule-based causal modeling is evident from the fact that many variables in the business domain are qualitative in nature. Besides, decision makers tend to use qualitative reasoning even when they are faced with a quantitative diagnostic problem (Bouwman, 1983).

Prior research has certainly utilized these modeling constructs for problem diagnosis support. The focus of this research, however, is not limited to the individual techniques but extends to the interrelationships between the techniques. It should be noted that DSS researchers have long proposed the integration of the rule-based modeling with the traditional quantitative modeling (Turban and Watkins, 1986; Finlay and Martin, 1989; White, 1990). In fact, Ata Mohamed (1985) and Paradice (1986) have emphasized the importance of adopting such an approach, especially for the purpose of problem diagnosis support. However, there has been no such PDSS research.

Third, this research is significant in that it develops a process-oriented support system for managerial problem diagnosis. The importance of process-oriented decision support has been emphasized by almost all DSS researchers (Keen and Scott Morton, 1978; Sprague and Carlson, 1982; Weber, 1986; Elam and Konsynski, 1987; Nunamaker et al., 1988; Adams et al., 1990). The general consensus reached by the DSS research community is that "DSS of the future are to serve the full concert of activities that occur during the problem-solving

process" (Weber and Konsynski, 1987, p. 79). Moreover, the DSS "must be explicitly integrated into" the decision process (Adams et al., 1990, p. 224). In spite of the widely held belief, there has been no process-oriented PDSS. The previous PDSS research tended to identify or develop a technique and then adopt the technique for problem diagnosis support. In contrast, this research analyzes the managerial problem diagnosis process first, derives the system requirements based on the analysis, and then identifies and develops appropriate support functions. This research provides a significant first step toward developing a comprehensive, process-oriented PDSS.

Fourth, this research is significant in that it reemphasizes the fundamental premise of DSS. A cooperative problem solving system must combine the normative capabilities of the computer with the descriptive abilities of the human (Hale and Kasper, 1989). That is, an effective PDSS must combine both user-driven and system-driven problem diagnosis (Keen and Scott Morton, 1978; Sage, 1981; White, 1990; Sundstrom, 1991). Most of the previous PDSS, however, have emphasized either user-driven or system-driven problem diagnosis. In addition, this research incorporates the inquiry "guaranteeing" concepts into PDSS design.

Finally, the implemented prototype system provides significant advantages over previous PDSS. Specifically, the prototype system provides a more powerful representational capability, a more comprehensive set of problem diagnosis support functions, and a more cooperative user-system interaction than the previous PDSS do. The prototype system's capabilities can be ultimately attributed to the following design concepts synthetically utilized in this research. They include multiple modeling constructs, process-oriented decision support (Sprague and Carlson, 1982), cooperative systems (see Hale and Kasper, 1989), and inquiry "guaranteeing" concepts (Churchman, 1971).

1.5. Structure of the Dissertation

The structure of this dissertation mirrors the stages of the research methodology. This chapter identified the problems of the previous PDSS research, established the objectives of the research, discussed the research scope, and emphasized the importance of the research. Chapter II reviews relevant literature. The review consists of two major parts. The first part examines the managerial problem diagnosis process. The second part examines the inquiry system concept (Churchman, 1971) and reviews previous PDSS research. Chapter III addresses the methodological issues pertinent to the current research.

Chapter IV presents a conceptual framework for developing managerial PDSS. The framework develops a cooperative inquiry system concept, proposes a PDSS development strategy, delineates steps in PDSS development and use, and defines the roles of participants. In addition, the framework examines the requirements of PDSS from three interrelated aspects: knowledge representation, knowledge processing, and knowledge exchange. Chapter V performs the conceptual design of PDSS. This chapter identifies views necessary for problem diagnosis, formulates support functions for various problem diagnosis activities, and incorporates the inquiry "guaranteeing" concepts into PDSS design.

Chapter VI performs the symbol-level design of PDSS. This chapter operationalizes the conceptual design in terms of abstract, but well-defined, symbols. Chapter VII describes the prototype system, demonstrates its capabilities, and evaluates its performances. Chapter VIII evaluates the validity of the research. Chapter IX concludes this dissertation by discussing the contributions and limitations of the research.

CHAPTER II

LITERATURE REVIEW

The search for causality is an intrinsic tendency of humans and an essential ingredient to learning (Piaget, 1974). It is also a critical element for the advancement of science in which the primary objective is the formulation of causal principles that explain physical and social phenomena (Bunge, 1979). Every great philosopher has given some thought to causality, even though epistemological orientations and ontological foundations among philosophers vary (Wallace, 1974; Fales, 1990).

Literature in a variety of disciplines illustrates the importance of problem diagnosis, which is a process of understanding the causal structure of a problem. For example, in the area of medicine, a physician's understanding of the causal relationships between diseases and symptoms is considered to be critical for successful clinical diagnoses (Rogers et al., 1979; Patel and Groen, 1986). In fact, finding causal structure is a significant theme of research in any discipline in which progression is measured by theory construction and evidence generation (Bunge, 1973; Glymour, 1980).

Without a doubt, problem diagnosis is an important aspect of managerial decision making. Mintzberg et al. (1976) observe that it is difficult to imagine strategic decision making without some form of diagnosis. The necessity for problem diagnosis, however, is not limited to the strategic level but applies to every level of organizational decision making (Lyles, 1982; Kepner and Tregoe, 1981; Dutton et al., 1983). Moreover, problem diagnosis is important for decision making in every functional area of management, including marketing (Fogg, 1985), finance (Bouwman, 1983), accounting (Brown, 1985), personnel

management (Cherrington, 1983), and information systems (Markus and Robey, 1988). Lyles (1982) states that there is a fundamental causal structure underlying every outcome or consequence an organization produces, and that understanding the causal structure is essential for designing effective solutions.

The purposes of this chapter are twofold: (1) to develop a better understanding of the problem diagnosis process; and (2) to assess the current state of computer-based systems that support managerial problem diagnosis. The first section discusses the core concepts underlying the terms "problem" and "problem diagnosis." The second section examines the major factors that affect the problem diagnosis process. The third section reviews the literature related to the problem diagnosis process. The fourth section discusses the decision strategies that may facilitate the problem diagnosis process.

The focus then shifts towards the computer-based systems that support managerial problem diagnosis. The fifth section examines the inquiring system concept (Churchman, 1971). The sixth section reviews computer-based systems that are designed to support managerial problem diagnosis. The last section summarizes the literature review.

2.1. Problem and Problem Diagnosis

Previous research indicates that the term "problem" has multiple definitions (Nadler, 1983; Volkema, 1983). Prior research also shows that the terms denoting certain decision making aspects, such as problem diagnosis, problem definition, and problem formulation, encompass different activities in the literature (Lang et al., 1978; Baldwin, 1989; Smith, 1989). Ackoff et al. (1962) recommend that researchers define such terms clearly and use them consistently to avoid confusion. The purpose of this section is to develop a clear understanding of the

core concepts underlying the terms "problem" and "problem diagnosis," so that the scope and role of problem diagnosis support systems can be clarified.

2.1.1. Theoretical Perspectives of the Nature of a Problem

Although there exist a number of different definitions for the term "problem," most definitions discuss the existence of a difference between the way things are and the way one wants them to be (Reitman, 1964; Pounds, 1969; MacCrimmon and Taylor, 1976; Smith, 1989). Individuals perceive a problem when the current state falls significantly below a certain aspiration level (Kiesler and Sproull, 1982). Individuals also perceive a potential problem when the current situation is far better than expected or planned (Schoennauer, 1981). The aspiration level is not static, but shifts continuously according to the decision maker's cognitive processes (Dutton et al., 1983), workload (Volkema, 1983), and experiences (Kiesler and Sproull, 1982).

A problem is essentially a conceptual or cognitive entity that individuals create to cope with their environments. That is, in dealing with a problem situation, decision makers must develop mental models of the situation (Simon, 1960; Pounds, 1969; Ackoff, 1978). There are two major theoretical or philosophical perspectives within which decision makers view the nature of the problem situation in relation to their conceptual models: objective reality and subjective reality (Klein and Hirschheim, 1985, 1987). The perspectives on the nature of the problem are important for DSS research, because each of these perspectives suggests not only different approaches to problem solving (Dery, 1983) but also different ways of providing decision support (Landry et al., 1985).

The objectivist perspective maintains an ontology of "realism," which contends that the world is made up of hard, tangible objects that exist

independently from one's perception of it (Burrell and Morgan, 1979). Therefore, individuals should uncover a problem, as it really exists in the world, by observation and analysis. Problem recognition research focusing on problem detection errors maintains that problems exist in objective reality, and the quality of "problemness" is inherent in environmental stimuli (Kiesler and Sproull, 1982).

The subjectivist perspective, on the other hand, embraces an ontology of "nominalism" that sees the world in terms of labels, names, or other artificial creations that impose some form of structure on reality (Burrell and Morgan, 1979). From this perspective, individuals should construct a problem by cognitive and affective activities. The problem recognition research focusing on aspiration-level maintains that the "problemness" depends more on the state of the perceiver than on anything inherent in the stimuli (Kiesler and Sproull, 1982).

Landry et al. (1985) draw different DSS design guidelines from these alternative views of the nature of a problem. From the objectivist view, DSS should provide normative support that emphasizes how decisions should be made rather than how they are actually made (Keen and Scott Morton, 1978; Keen, 1987). Therefore, DSS should: (1) stress the analysis of a problem; (2) eliminate manager's subjective value judgment; and (3) emphasize analysis over perception. In short, DSS should provide "strong support" (Moore and Chang, 1983) to change the decision makers' suboptimal decision processes. Underlying this recommendation, however, is the disturbing presumption that decision analysts or DSS designers have a privileged ability to understand reality. It is naive to believe that observation and analysis always lead to a correct identification of the problem (Bunge, 1973).

From the subjectivist view, DSS should provide cognitive support that focuses on the decision makers' mental processes of formulating problems.

Landry et al. (1985) suggest that a DSS must: (1) emphasize the design of support over the analysis of a problem; (2) assume user's value judgment as an integral part of the decision process; and (3) assign a supporting role to experts. That is, DSS should provide "weak support" (Moore and Chang, 1983) to facilitate the decision maker's natural decision processes. One potential problem with this approach, however, is that decision makers may use DSS as a powerful tool for reinforcing their idiosyncratic predispositions (Huber, 1983).

Current DSS designs tend to have a strong predisposition toward the objectivist view of a problem (Dery, 1983; Landry et al., 1985; Klien and Hirschheim, 1985). MIS/DSS, like MS/OR, have largely ignored descriptive models of the decision process (Parker and Al-Utaibi, 1986; Nunamaker et al., 1988; Weber and Konsynski, 1987). One reason for this predisposition is that the rational conception of decision making, i.e., how the decision should be made, has precision and logic that the descriptive decision models lack. On the other hand, some researchers (Landry et al., 1985; Klein and Hirschheim, 1987) suggest that the subjectivist view better guarantees the effectiveness of DSS, because it is the decision makers who are ultimately responsible for the decisions after all. However, it is essential that any system builder be as concerned with descriptive realism as with normative idealism (Keen and Scott Morton, 1978; Sage, 1981; Hale and Kasper, 1989; Weber and Coskunoglu, 1990; White, 1990).

In general, the term "problem situation" refers to the objective reality from which individuals formulate problems; and "problems" refer to the subjective realities conceptualized by individuals. This research views decision makers as inquiring systems that attempt to generate knowledge about problem situations, i.e., systems that are trying to reflect the objective reality as closely as possible in their subjective minds. PDSS are viewed as yet another inquiring system, but with

quite different capabilities. The objective of this research is to develop a PDSS that can form a cooperative inquiring community with decision makers.

2.1.2. Definition of Problem Diagnosis

There are a number of definitions for the term "problem diagnosis" (see Figure 2.1). One way to clarify the meaning of problem diagnosis is to compare it with other similar terms, such as problem identification, formulation, recognition, conceptualization, and structuring. Even though it is difficult to precisely differentiate these terms, they all are related to the intelligence stage of Simon's model (1960) or the problem identification phase of Mintzberg's model (Mintzberg et al., 1976). The problem identification phase of Mintzberg's model (1976) consists of problem recognition (identification of symptoms) and problem diagnosis (identification of causal structure).

While problem recognition logically precedes problem diagnosis (Schoennauer, 1981), the logical precedence does not imply that the problem diagnosis routine begins at the end of the problem recognition routine. Decision processes are recursive and cyclic (Mintzberg et al., 1976; Lang et al., 1978; Bravoco and Yadav, 1985). An intermediate result of diagnosis often calls for the detection of additional symptoms; and newly discovered symptoms affect the diagnosis process (Dutton et al., 1983; Kolodner and Kolodner, 1987). Thus, there is no logical inconsistency between Bouwman (1983), who includes problem recognition as a part of problem diagnosis, and Cowan (1986), who includes problem diagnosis as a part of problem recognition. In this research, the term "problem diagnosis" is broadly defined to include problem recognition activities.

Problem diagnosis is conceptually similar to problem structuring, which is rather precisely defined in management science as the process of identifying the

Definition of Problem Diagnosis	Reference
A process to comprehend the evoking stimuli and determine cause-effect relationships for the decision situation.	Mintzberg et al. (1976, p. 253)
Hypothesizing about or searching for causal relationships among variables believed to be associated with the problem at hand.	Courtney et al. (1987, p. 373)
A step in which representation of the problem is defined, described, and understood in terms of major problem components and boundary conditions.	Bartee (1973, p. 442)
The identification of the state of the underlying system on the basis of a set of observed symptoms.	Bouwman (1983, p. 653)
The process of formulating the present set of conditions, symptoms, causes, and triggering events into a problem or sets of problems sufficiently well specified.	Schwenk & Thomas (1983, p. 240)
Those activities and processes by which data and stimuli are translated into focused issues (i.e., attention organizing acts) and the issues explored (i.e., acts of interpretation).	Dutton, et al. (1983, p. 307)
An attempt to achieve greater certainty about a problem description.	Cowan (1986, p. 766)

Figure 2.1. Definitions of Problem Diagnosis.

elements of the problem and their relationships (Woolley and Pidd, 1981). In fact, the framing of hypotheses among variables corresponds to the derivation of model structure (Harary et al., 1965; McLean and Shepherd, 1976). Problem diagnosis is also conceptually similar to problem formulation (Schwenk and Thomas, 1983). Problem formulation, however, tends to include additional aspects, such as stakeholder identification, value and objective specification, and problem prioritization (Volkema, 1986; Ireland et al., 1987).

Another way to clarify the meaning of problem diagnosis is to examine its inputs and outputs. The inputs to the process include the problem stimuli (Mintzberg et al., 1976; Cowan, 1986), the concepts and beliefs held by the decision maker (Kiesler and Sproull, 1982; Bouwman, 1983; Dutton et al., 1983), and the assumptions, theories, and information to frame the situation (Taylor, 1975; Cowan, 1988).

The most significant output of the process is an understanding of the cause-effect relationships (Mintzberg et al., 1976; Dutton et al., 1983; Courtney et al., 1987). Other outputs include identification of the specific factors responsible for the problem symptoms (Kepner and Tregoe, 1981; Ata Mohammed, 1985; Smith, 1988), determination of a category into which the problem can be classified (Kolodner and Kolodner, 1987; Chandrasekaran and Goel, 1988), and development of predictive judgment (Einhorn and Hogarth, 1982). Predictive judgment means that the causal structure identified during the problem diagnosis process provides a basis of designing different effects on the problem situation. Problem diagnosis, thus, provides a critical input into the alternative generation and evaluation phases of decision making.

Finally, the meaning of problem diagnosis can be further clarified by relating various definitions listed in Figure 2.1. Problem diagnosis is a process to

comprehend the evoking stimuli (Mintzberg et al., 1976). Individuals sense the stimuli by observing triggering events and scan the environment to identify a set of conditions and symptoms (Schwenk and Thomas, 1983). During this process, decision makers identify the problem boundaries (Bartee, 1973) and focus on important aspects of the problem (Dutton et al., 1983). To comprehend the stimuli, decision makers need to determine the causal structure of the problem (Mintzberg et al., 1976; Bouwman, 1983) by searching, hypothesizing, and testing causal relationships (Courtney et al., 1987) between the observed symptoms and their underlying causes (Bouwman, 1983). This process improves the certainty about a problem description (Cowan, 1986).

2.2. Contingency Factors for Problem Diagnosis Effectiveness

Volkema (1983) identified a list of factors that affect the effectiveness of problem formulation. They are: (1) the complexity of the problem; (2) the capabilities and experiences of the decision maker; (3) the environment in which the decision making takes place; and (4) the formulation process used by decision maker. These factors seem to determine not just the effectiveness of the problem formulation but also the effectiveness of the overall decision process. In fact, Mason and Mitroff's MIS research framework (1973) identified a similar set of variables as the determinants of decision effectiveness: (1) problem types; (2) psychological types; (3) organizational contexts; (4) methods of evidence generation; and an additional variable emphasizing the information system's role, i.e., (5) the information system's modes of presentation. In management literature, information systems are often discussed as elements of the decision environment factor (e.g., Volkema, 1983).

Because these factors affect the effectiveness of the overall decision making process in general, and the problem formulation process in particular, it is reasonable to consider that they also affect the effectiveness of the problem diagnosis process. This section examines the first three variables that affect the problem diagnosis process: (1) the problem characteristics; (2) the decision maker; and (3) the decision environment. The remaining sections of this chapter examine the problem diagnosis process (Section 2.3), problem diagnosis strategies (Section 2.4), and problem diagnosis support systems (Sections 2.5 and 2.6).

2.2.1. Problem Characteristics

There are various classification schemes to categorize different problems. Two commonly used taxonomies are: (1) well-structured versus ill-structured problems; and (2) routine versus non-routine problems (Simon, 1960; Gorry and Scott Morton, 1971; Mason and Mitroff, 1973). These characteristics greatly affect the required inquiring modes for successful problem diagnosis.

Structured problems have well-defined models for the goal state, problem state, and transformation process (Newell, 1969; Klein and Weitzenfeld, 1978). Unstructured problems have indeterminate goals (Smith, 1988), unfamiliar problem space (MacCrimmon and Taylor, 1976), or weak solution procedures (Newell, 1969). Dery (1983) argues that structured problems may well be unstructured problems, i.e., structured problems are structured because we choose to treat them as such. Structured problems allow much of the problem diagnosis process to be automated, while unstructured problems require appreciation, evaluation, judgment, and insights from decision makers.

Routine problems tend to provide more data about the problem state, the attempted transformation process, and the decision results than non-routine

problems do (Mason and Mitroff, 1981). The frequency of problem occurrence often correlates with, but does not determine, the degree of problem structure (Mintzberg et al., 1976; Taylor, 1975; Smith, 1988).

First, fault diagnosis systems (e.g., Bennett and Hollander, 1981) deal with structured and repetitive problems. These systems assume a pre-formulated logical model that represents the causal connections between system components under a normal system state (de Kleer and Williams, 1987; Reiter, 1987). When the object system breaks down, the diagnosis system automatically identifies the causes of breakdown, i.e., the components responsible for the malfunction. For this type of system, problem diagnosis is often reduced to abductive reasoning[2] (Peng and Reggia, 1990).

Second, Kepner and Tregoe (1981) suggest a method to deal with structured and non-routine problems. Their method presumes the existence of the decision maker's objective knowledge about the causal structure and suggests that decision makers should logically deduce the causes of the problem by examining the problem in terms of physical location, time, symptom occurrence frequencies and durations, etc. This method, however, is plausible only if the causal relationships are simple and obvious, e.g., problems of technical nature (Anderson and Janson, 1979; Ramakrishna and Brightman, 1986).

Third, diagnostic expert systems solve unstructured and routine problems based on the codified rules of thumb, statistical intuition, and past experience of

---

[2] Abductive reasoning can be best explained in contrast with deductive reasoning and inductive reasoning. Deductive reasoning consists of a general rule (major premise) and a specific case (minor premise) from which a specific result can be deduced. Inductive reasoning consists of specific cases and specific results from which a general rule can be hypothesized. Abductive reasoning consists of a general rule and a specific result from which a specific case can be hypothesized.

the domain experts. Research shows that domain experts often utilize both abductive reasoning with backward chaining and deductive reasoning with forward chaining (Patel and Groen, 1986). When a large amount of data are readily available, as seen in some medical areas, inductive reasoning may also yield diagnoses with significant accuracy (Rogers et al., 1979).

Finally, when a problem is ill-structured and non-routine, problem diagnosis is a difficult process surrounded with ambiguity and uncertainty (Lyles, 1987). In strategic diagnosis, managers often have to rely largely on their subjective perceptions and judgment (Barnes, 1984; Ireland et al., 1987). They usually begin with little comprehension of the situation, and their understanding deepens as they work on the problem (Thomas, 1984). For this type of problem, a DSS needs to enhance the decision maker's creativity (Elam and Mead, 1987; Evans, 1989), simplify the problem (Schwenk, 1984), promote learning (Weber, 1986), and process qualitative ideas (Young, 1983). Unfortunately, computer-based systems with these capabilities are in their early infant stages. The main focus of this research is to develop a support system for semi-structured or unstructured problems that are repetitive.

2.2.2. Decision Maker

Obviously, the problem diagnosis process is also dependent upon the knowledge and experience of the decision maker. Knowledge development and learning from experience are, in general, interdependent with the cognitive capacity of the decision maker. Human information systems have several inherent cognitive limitations, however. These limitations include serial processing, small short-term memory, and long-term memory with slow storage speed (Miller, 1956; Simon and Barenfeld, 1969; Simon and Newell, 1971). Because of these

limitations, decision makers construct simplified mental models when dealing with complex problem situations (Simon, 1957; Hogarth, 1980; Kiesler and Sproull, 1982; Mason and Mitroff, 1981).

During the simplification process, decision makers are subject to various cognitive biases (Tversky and Kahneman, 1974; Hogarth, 1980; Sage, 1981; Hogarth and Makridakis, 1981). Cognitive biases relevant to problem diagnosis include, but are not limited to, illusory causation, illusory correlation, adjustment and anchoring, reference effect, and inconsistent statistical interpretation (Barnes, 1984; Thomas, 1984; Lyles and Thomas, 1988; Schwenk, 1986, 1988). While some of these biases are unavoidable characteristics of decision makers with limited cognitive capacities, they typically lead to systematic errors in conceptualizing problems (Tversky and Kahneman, 1974; Sage, 1981; Barnes, 1984). Moreover, they can interact and reinforce each other (Schwenk, 1986).

Research in cognitive psychology has attempted to postulate the mental constructs that serve as a basis of structuring the knowledge of decision makers. Two models utilized most often in the decision making literature are cognitive maps (Axelrod, 1976; Weick, 1979) and cognitive schemata (Kelley, 1973; Fiske and Linville, 1980). Cognitive schemata are a set of conceptual constructs that decision makers use to classify objects based on a set of attributes and attribute relationships (Taylor and Crocker, 1983). A "cognitive map" is a particular type of schemata or a part of a broader schema (Weick, 1979; Schwenk, 1988).

The concepts, beliefs, assumptions, and cause-effect relationships in the cognitive schemata determine how a problem will be diagnosed (Lyles, 1981; Dutton et al., 1983; Lyles and Thomas, 1988). Cognitive schemata are not infallible, and their construction and application are subject to incompleteness, inaccuracy, and inconsistency (Ireland et al., 1987). Individuals differ in the

construction and use of their cognitive schemata, and this often manifests itself in terms of cognitive style dimensions (Ramaprasad and Mitroff, 1984; Cowan, 1986).

MIS and DSS research have focused considerable attention upon cognitive style as a basis for information system design (see Zmud, 1979; Benbasat and Taylor, 1982). However, Huber (1983) argues that cognitive style should play a very limited role, if any, in MIS/DSS design. According to Huber, neither the current literature on cognitive styles nor any further research is likely to provide a satisfactory basis for deriving operational guidelines for MIS/DSS design. Ramaprasad (1987), therefore, suggests that the MIS/DSS designers focus on cognitive process rather than cognitive style. As contrasted to cognitive style, which emphasizes the general traits of decision makers, cognitive process emphasizes the cognitive activities performed by decision makers for specific problem situations. Tsai (1991) suggests that cognitive processes are contingent upon task demands but rather independent of decision makers' cognitive styles.

2.2.3. Decision Environment

Many factors in the decision maker's environment can affect the problem diagnosis process, including individual, interpersonal, and organizational decision making contexts (Mock, 1973). Individual decision making contexts are such factors as time constraints and work load (Wright, 1974; Smart and Vertinsky, 1977; Billings et al., 1980). High time pressure and heavy work load tend to impose cognitive burdens on decision makers and discourage formal problem diagnosis (Janis and Mann, 1977; Smart and Vertinsky, 1977; Lyles and Thomas, 1988). Stress, as a function of one's ability to cope with the complexities and uncertainties of the problem environment, can affect the time and effort devoted to the problem diagnosis (Volkema, 1983).

From a behavioral perspective, decision making is a political process involving conflict between people with varying degrees of power (Rowe, 1989). During problem diagnosis, individuals with different beliefs and political interests may interact to challenge assumptions, clarify cause-effect relationships, and alter the judgments of others (Volkema, 1986; Lucas, 1987). Thus, the interpersonal context of decision making may influence control of data and models, creating a particular focus and direction in the diagnosis (Naraynan and Fahey, 1982; Dutton et al., 1983).

In a similar fashion, organizational factors affect individuals' problem recognition and diagnosis (Kiesler and Sproull, 1982). These factors include communication channels (Lyles and Mitroff, 1980), organizational policies (Sage, 1981), and information system structure (Kiesler and Sproull, 1982).

2.3. Problem Diagnosis Processes

Decision theorists have noted that little is known about the problem identification phase of the decision making process (Leavitt, 1975; Mintzberg et al., 1976; Getzels, 1979; Dery, 1983). In recent years, however, a growing number of articles on this subject have appeared in managerial decision making literature, especially in the area of strategic planning (Lyles and Mitroff, 1980; Dutton et al., 1983; Ramaprasad and Mitroff, 1984; Volkema, 1986; Ireland et al., 1987; Lyles and Thomas, 1988; Smith, 1989). Expert system development has also produced a significant amount of knowledge about medical and financial problem diagnosis processes (Shortliffe et al., 1979; Szolovits and Pauker, 1978; Bouwman, 1983; Patel and Groen, 1986).

The purpose of this section is to review the literature on the problem diagnosis process. First, the general characteristics of the managerial problem

diagnosis process are discussed. Second, Piaget's cognitive developmental model is examined to understand how individuals develop causal thinking. Finally, the managerial problem diagnosis literature is reviewed to understand how managers perform problem diagnosis.

2.3.1. Process Characteristics of Managerial Problem Diagnosis

Dutton et al. (1983) suggest that the dynamic nature of managerial problem diagnosis can be captured by three process characteristics: (1) recursiveness, (2) retroductivity, and (3) heterarchy. The first two characteristics are relevant to the diagnosis process on the individual decision maker's level. The third characteristic, heterarchy, is related to group decision making.

First, the recursiveness of the diagnosis process indicates that the mental model of the problem situation needs to be defined and redefined several times, as the decision maker discovers additional symptoms or restructures the causal relationships (Taylor, 1975; Cowan, 1986). The successive revisions of judgment imply that stimuli incorporation influences further stimuli detection, and that the sequence of detecting stimuli may change the whole problem diagnosis process and results (Bouwman, 1983).

Second, the "retroductivity" of the diagnosis process signifies the coexistence and interplay of both deductive and inductive modes of thinking (Einhorn and Hogarth, 1986). The mental models of the problem situation developed by individuals depend on both the cognitive maps of the individuals and the data on hand (Cowan, 1988). The deductive mode of reasoning has traditionally been highlighted, because it is the cognitive map that frames, interprets, and incorporates the data on hand. However, when the cognitive maps do not ensure a sufficient basis for deductive reasoning, decision makers need to

invoke inductive modes of thinking. The inductive inferences that individuals make will influence their cognitive maps (Lyles and Thomas, 1988).

Finally, the "heterarchy" of the diagnosis process describes the interactions and collisions among various organizational actors. Because individuals have different cognitive maps and political interests, the problem diagnosis process is not driven solely by data or logic, but is susceptible to the actions of individual participants (Lyles and Mitroff, 1980).

2.3.2. Piaget's Developmental Model

Cognitive process models can be classified broadly into developmental models and performance models. Developmental models are useful to explain how individuals develop an understanding of their environments over time, while performance models are useful to understand the cognitive processes of decision makers under a particular circumstance. Among various developmental models including that of Sigmund Freud, Piaget's model (1974) is most widely used in cognitive psychology (Inhelder et al., 1987), artificial intelligence (Flanagan, 1991), and management literature (Ramaprasad and Mitroff, 1984).

Piaget (1952) posits that a human starts life with a set of reflexes and inherent ways of interacting with the environment. These inherited ways of interaction refer to the tendency of thought to be organized and adapted to the environment. Although infants know <u>almost</u> nothing about the world, they have the potential to know <u>almost</u> everything. A child develops intellectually through four discrete stages: sensorimotor (age 0-2), preoperational (age 2-7), concrete operational (age 7-11), and formal operational stages (age 11+). Underlying the intellectual development is the logico-mathematical structure (LMS). LMS is the

information processing structure, i.e., cognitive schemes, which can be very simple cause-effect models or very complex well-developed theories.

LMS is utilized and updated by three cyclic operations: (1) simple abstraction; (2) application; and (3) reflexive abstraction (Piaget, 1974). Simple abstraction is the process of extracting important qualities from observations. Simple abstraction is always based on a previously developed LMS. Application is the process of making deductions about a situation based on LMS and testing the validity of the deductions using data from observations. Reflexive abstraction is the process of modifying LMS when the data do not fit the deductions.

Reflexive abstraction entails a mental leap, from the perception of data to the induction of patterns and derivation of meaning from the data. As LMS develop, simple abstraction and reflexive abstraction also develop. Better LMS enable more refined observations. Better data from such observations also facilitate testing and development of the LMS. Simple abstraction, application, and reflexive abstraction reinforce each other.

Ultimately, this evolving process leads to the development of abstract and refined LMS, indirect observation, and complex rules of inference. When LMS are sufficiently refined, the process of application is replaced by the process of attribution. At this stage, the LMS becomes the model of reality instead of a hypothesis. If a problem is similar to a previously encountered one, LMS from the decision maker's repertoire may be attributed to the problem. The choice of LMS is determined by cues in the problem situation.

2.3.3. Managerial Problem Diagnosis Process

Kiesler and Sproull (1982) identify three distinct phases of problem identification: (1) stimuli detection; (2) stimuli interpretation; and (3) stimuli

incorporation. Not surprisingly, these phases correspond to the three cognitive operations of Piaget's model.

First, stimuli detection is a process of monitoring the environment and noting important qualities. Individuals monitor their environment through deliberate and automatic scanning (Hasher and Zacks, 1979). Deliberate scanning is an intentional information gathering activity in which individuals have objectives, exert effort, and follow learned patterns (Feldman, 1981). It is limited by the individuals' attentions (March and Feldman, 1981) and restricted by organizational context (Cowan, 1986). Automatic scanning is a direct perceptual process that is environmentally activated in response to a specific array of stimulus energies (Sage, 1981; Kiesler and Sproull, 1982). Problems also may be sensed without a direct observation of the situation. Lyles and Mitroff (1980), based on an empirical study, indicate that managers usually become aware of significant problems through informal communication and intuition.

Decision makers have a tendency of using only easily available data and information and ignoring not easily available sources of significant information (Hogarth and Makridakis, 1981; Sage, 1981). In other words, search effort of problem diagnosis is allocated (Freeland and Stabell, 1978). Likewise, decision makers do not attend to every monitored stimulus. Rather, they process the stimuli selectively by filtering out insignificant ones (March and Simon, 1958; Mason and Mitroff, 1981). Selective perception is a way of avoiding information overload (Ackoff, 1967) and maintaining cognitive economies (Mischel, 1979).

Individuals tend to concentrate on salient material, i.e., events that are unpleasant, deviant, extreme, intense, unusual, sudden, etc. (Kiesler and Sproull, 1982). Mackie (1965) states that individuals regard something abnormal or wrong as more causal than what is normal or right. Salience of stimuli depends upon

the strengths of the stimuli, one's aspiration level, and one's tolerance for discrepancy (Billings et al., 1980). Consequently, stimuli being attended to vary with decision task (Ireland et al., 1987), decision maker's cognitive style and experience (Feldman, 1981), and stress level (Janis and Mann, 1977).

Second, stimuli interpretation is a process to construct meaning for, or assign meaning to, the sensed stimuli. One important aspect of interpretation is to classify the perceived situation as a problem or non-problem (Cowan, 1986). Decision makers may interpret the situation along other dimensions, such as human/technical problems and strategic/operating problems (Gorry and Scott Morton, 1971).

Interpretation is affected by certain assumptions that are the basic elements of individuals' frames of reference or world views (Taylor, 1975; Mitroff et al., 1979; Mason and Mitroff, 1981). To recognize a problem, decision makers often use multiple stimuli (Ramakrishna and Brightman, 1986). Cowan (1988) posits that decision makers search for additional symptoms when they are not able to interpret the perceived situation clearly based on the information on hand. Consistent stimuli facilitate problem clarification (Schwenk and Thomas, 1983), while inconsistent, weak, or discontinuous stimuli often lead to multiple views about the situation and debates (Lyles and Thomas, 1988). Individuals show a tendency to prefer consistent information and discount conflicting evidence (Hogarth and Makridakis, 1981).

Inherent in stimuli interpretation is the need for pattern recognition (Sage, 1981; Schwenk and Thomas, 1983). Pattern recognition generally consists of two phases: (1) extracting a pattern from the stimuli; and (2) identifying its best matching schema from memory (Taylor, 1975; Sage, 1981; Bouwman, 1983). If the pattern of the stimuli can be successfully framed within a selected schema,

individuals may easily identify the structure of the given problem (Kiesler and Sproull, 1982; Schwenk, 1988). However, if an individual is unfamiliar with the problem pattern, more time and effort are taken to structure the situation, and more information is usually scanned (Feldman, 1981; Cowan, 1986).

Some problem symptoms are so clear that they may force themselves on the individual involved (McKeeney and Keen, 1974), i.e., many managerial problems literally make themselves known (Smith, 1989). However, such stimuli occur only across a narrow domain of managerial decision making (Volkema, 1986). Stimuli interpretation becomes an important cognitive activity when a problem needs to be discovered, constructed (Volkema, 1986), created (Getzels, 1979), or designed (Yadav and Korukonda, 1985) rather than being presented.

Individuals interpret the same situation differently (Volkema, 1983; Cowan, 1986). The interpretation of the problem discrepancy is dependent on the individual's background and experience (Taylor, 1975; Lyles and Thomas, 1988) as well as the organizational goals, policies, and procedures (Weick, 1979; Kiesler and Sproull, 1982). Individual's cognitive biases in stimuli interpretation may result from stereotyping, anchoring and adjustment, expectations, reference effect, selective perceptions, and spurious cues (Hogarth and Makridakis, 1981; Sage, 1981). The potential for too easily misreading situations also increases as one becomes accustomed to a particular organizational context (Cowan, 1986).

Finally, stimuli association is a process to retain stimuli and relate them with other relevant concepts (Einhorn and Hogarth, 1982). Associative thinking implies that stimuli incorporation is dependent upon the knowledge and experience of individuals (Piaget, 1974) as well as the modeling constructs into which the knowledge is embedded (Lyles and Thomas, 1988). Individuals often employ causal association to structure a situation (Axelrod, 1976; Weick, 1979).

Although how individuals acquire their cognitive maps is largely unknown (Ford and Hegarty, 1984), individuals, in searching for causes, utilize various cues, such as temporal order, constant conjunction, contiguity in time and space, and similarity (Einhorn and Hogarth, 1982, 1986).

Taylor (1975) notes that stimuli association is constrained by the set of schemata invoked for the problem. Sage (1981) interprets two later stages of Piaget's cognitive developmental model (1952) in terms of schema usages. The concrete operational thought process utilizes a constrained set of schemata that has been established by previously learned behavior patterns. In novel situations, where learning is required, individuals typically apply a formal operational thought process where a set of schemata is developed to structure the situation.

When decision makers use similarity or analogy to define a problem, they may not recognize that there are critical differences between their models and the decision situation they face. The interpretive lens of a cognitive map selects certain aspects of an issue as important, ignores other, and links them to certain actions (Dutton et al., 1983). Likewise, use of schemata has focusing and organizing effects (Kiesler and Sproull, 1982; Ireland et al., 1987). Framing a situation within one particular view usually results in an incomplete understanding of the situation (Baldwin, 1989). However, when managers diagnose a problem, they tend to develop only one world view or perspective (Lyles and Mitroff, 1980).

2.4. Strategies for Problem Diagnosis

In order to improve problem diagnosis a number of support methodologies and techniques have been developed. These strategies include: (1) the convergent thinking approach; (2) the divergent thinking approach; (3) the dialectical method; (4) the creativity stimulant approach; and (5) the modeling approach.

These strategies make different assumptions about the deficiencies of the human cognitive process, and thus recommend different ways of improving the cognitive process. For example, the convergent thinking approach is mainly concerned with reducing cognitive processing requirements. On the other hand, the divergent thinking approach attempts to improve the quality of cognitive processing, and this objective usually calls for heavier cognitive processing.

First, the convergent thinking approach intends to produce a diagnosis as efficiently as possible. This approach emphasizes the cognitive limitations of individuals and attempts to reduce the cognitive complexities (Schwenk and Thomas, 1983; Schwenk, 1984). For example, MacCrimmon and Taylor (1976) provide the following simplification steps: (1) determining the boundaries of a problem; (2) focusing on changes; (3) factoring the problem into subproblems; and (4) focusing on the controllable components. Kepner and Tregoe (1981) propose a diagnosis method that reduces the problem space systematically with a series of structured questions similar to "20 questions." Other methods related to this approach include the Alpha Omega method, means-ends analysis, input-output analysis, selective focusing, and problem classification (see Anderson and Janson, 1979; Volkema, 1983, 1986; Cowan, 1988). In general, these methods are useful to handle complex, but well-structured problems (Schwenk and Thomas, 1983). The convergent approach, however, involves a high risk of committing the error of the third kind (Leavitt, 1975; Dery, 1983).

Second, the divergent thinking approach has an opposite premise from the convergent thinking approach. Thus, it suggests expanding the decision maker's cognitive sets that constrain the development of problem understanding (Taylor, 1975). Advocates of this approach believe that decision makers frequently over-constrain their conception of the problem, and that opening of the constraints will

improve the chances of finding a correct problem (Volkema, 1983). The fact-net model (Ramakrishna and Brightman, 1986), for example, proposes to expand the psychological constraints by intentionally creating as many theories about a problem situation as the situation permits. Volkema (1983) proposes defining the problem boundaries implicitly rather than explicitly, because an expansion of the problem boundaries usually results in an better understanding of the situation (Mitroff et al., 1979). Many of the group-oriented problem formulation techniques are designed to promote divergent thinking (Taylor, 1975; Schwenk and Thomas, 1983). The disadvantage of this approach is that the information produced may exceed the decision maker's intellectual capacity.

Third, the dialectical approach is based upon Hegelian philosophy (Churchman, 1971). This approach intentionally creates cognitive conflicts between group members. The initial conflict is usually attributed to the different assumptions underlying the participants' mental models. The assumptions are challenged to form a better view of the problem situation. Two variations of this approach involve the dialectical inquiry and the devil's advocate (Cosier, 1981). One disadvantage of these methods is that they may create conflicts even when conflicts do not seem necessary.

Fourth, the creativity stimulant approach emphasizes the cognitive processes that combine and recombine existing thoughts, ideas, etc. into more refined cognitive sets and apply them to new problems in more refined ways (Elam and Mead, 1987; Evans, 1989). Creativity, in general, can be promoted by reorganizing one's knowledge, altering one's cognitive process, and eliminating the external factors that block the creative thinking (Smith, 1988). There are several techniques designed to promote creative thinking and group interaction, e.g., brainstorming, the Delphi method, the nominal group technique, and

morphological analysis (see Taylor, 1975; Schwenk and Thomas, 1983; Weber, 1986).

Finally, the modeling approach focuses on the point that cognitive process is not independent of the structure of mental models. This approach is very specific about the desired representation of problems as well as the desired cognitive operations. The structural modeling approach deserves attention, because almost all the previous PDSS are extensions of this approach. Structural models (see Figures 2.2 and 2.3) have a very simple modeling construct: nodes represent conceptual variables; and arcs represent the causal relationships between the variables. An arc in Figure 2.2, or the value of 1 in Figure 2.3, signifies the connection between two variables; and no arc, or an entry of zero in the matrix, means there is no causal connection.

Considering that problem diagnosis, or human intelligence in this matter, is concerned with understanding objects and object relationships, the structural model is the simplest form of representing reality. All that the models can represent are the unitary variables and two truth values for a single type of relationship, such as (1) "A" affects "B" or (2) "A" does not affect "B." Surprisingly, however, this form of inquiry abounds. For example, the research hypothesis, "Does the usage of DSS affect the decision makers' performance?" (e.g., Sharda et al., 1988), is in this form. One reason for the popularity of structural modeling as a basis of computer-based problem diagnosis support is that the structural models are very simple but useful. However, because each modeling construct has its own specific world view and, thus, has limited ways of representing and making inferences about objects and object relationships, it is inappropriate to base problem diagnosis solely on this simple modeling construct.

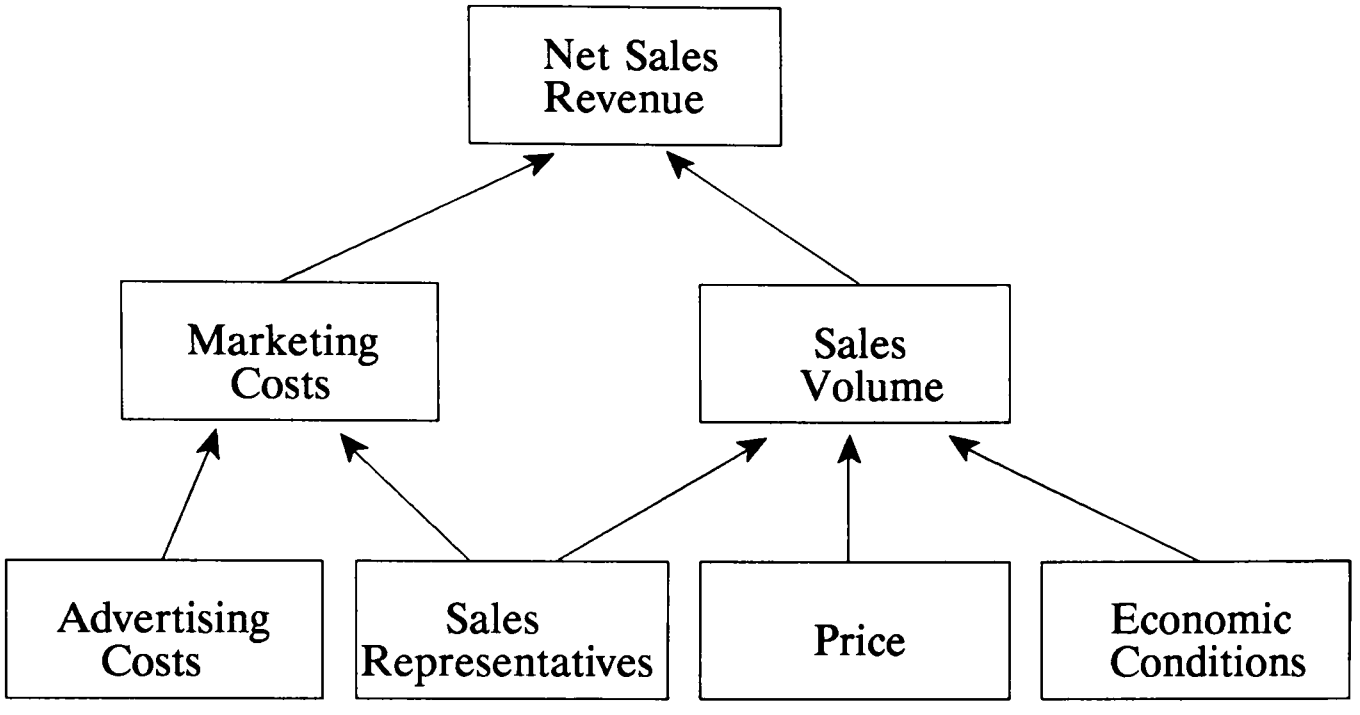


Figure 2.2. A Hypothetical Structural Model (Ata Mohamed, 1985, p. 57)

	Sales Rev.	Market. Costs	Sales Volume	Ad. Costs	Sales Reps.	Price	Econ. Cond.
Sales Rev.	0	0	0	0	0	0	0
Market. Costs	1	0	0	0	0	0	0
Sales Volume	1	0	0	0	0	0	0
Ad. Costs	0	1	0	0	0	0	0
Sales Reps.	0	1	1	0	0	0	0
Price	0	0	1	0	0	0	0
Econ. Cond.	0	0	1	0	0	0	0

Figure 2.3. A Matrix Representation of the Structural Model

2.5. Inquiring Systems

A PDSS, as it intends to generate knowledge about problem situations, is an inquiring system (Mason and Mitroff, 1973). Therefore, it is subject to various epistemological questions. An early effort to establish an epistemological foundation for information system design is Churchman's work. Churchman proposes that there exist at least five well-established types of inquiring systems that differ in their epistemological orientations. The following discussion is based on Churchman's book, The Design of Inquiring Systems (1971).

2.5.1. Lockean Inquiring Systems

Lockean inquiring systems are the archetype of empirical, inductive, consensual systems. Lockeans assume that there is no a priori axiomatic structure or theory of the world that one can possibly conceive prior to one's experience of the world. For any problem, Lockean systems start with a set of premises derived from their experience. Reasoning from these empirical premises, Lockeans build up fact nets, or networks of increasingly more universal inductive generalizations.

The guarantor that Lockean systems obtain true knowledge is consensus. An empirical generalization is judged to be objective, true, or factual if there is sufficient widespread agreement on it by a group of experts. Given any problem, Lockean systems tend to present a single model for which group consensus is the highest. The final information content of Lockean systems is purely empirical.

The strength of Lockean inquiring systems lies in their ability to utilize the rich sources of experiential data. The weaknesses are: (1) they fail to realize the importance of a priori knowledge; (2) while experience is undoubtedly rich, it can also be extremely fallible and misleading; and (3) the cost of producing consensus can become too prohibitive.

2.5.2. Leibnitzian Inquiring Systems

Leibnitzian inquiring systems are the archetype of formal, deductive, logical systems. The initial state of Leibnitzian systems assumes to have a set of elementary premises of the world, i.e., innate ideas. Given any problem, Leibnitzians start with these premises and deduce a network of increasingly more general propositional truths using formal and logical rules of inference.

The guarantors of Leibnitzian systems are logical consistency, coherence, and precision. A proposition of contingent truth becomes a likely truth, as the proposition becomes an integral part of a large fact net. The final information content of Leibnitzian systems is purely analytical. Leibnitzians view that data are always particulars, and there is no logical guarantee that particulars will repeat themselves.

The strengths of Leibnitzian inquiring systems are the strengths that characterize all formal systems: consistency, rigor, precision, coherence, etc. Their drawbacks are: (1) they are rich in analytic or formal content but extremely low in empirical content; and (2) they have difficulty defending why they chose to represent and solve the particular problem the way they did.

2.5.3. Kantian Inquiring Systems

Kantian inquiring systems are the archetype of multi-model, synthetic system. They believe that in order to observe the world, one must make certain assumptions a priori about the nature of the world. On any problem, Kantian systems first construct at least two different, but not necessarily conflicting, Leibnitzian theories. Next, they collect different sets of empirical data to support each theory. A Lockean strategy, then, may be used to determine which theory fits the data better and how the theory should be modified.

The guarantor of Kantian systems is the degree of fit between the theory and the empirical data collected under the presumption of that theory. The information content of Kantian inquiring systems is neither purely analytical nor purely experiential. Kantian inquirers consider that neither theory nor data is more fundamental than the other.

The strength of Kantian inquiring systems is that they may counter the weaknesses of both Lockean and Leibnitzian systems. Their weaknesses are: (1) the multiple models may not cover the right model; and (2) the decision maker can be overwhelmed than aided by the multitude of models.

2.5.4. Hegelian Inquiring Systems

Hegelian inquiring systems are the archetype of conflictual, synthetic systems. Hegelian systems, like Kantian systems, recognize the importance of a priori knowledge and the use of multiple models. Hegelian systems, however, always build antithetical models, i.e., opposing Leibnitzian theories.

In contrast to the Lockean systems, Hegelian systems consider conflict as the best principle for guaranteeing the goodness of a model. Hegelian inquirers hope that out of a dialectical confrontation the assumptions will be brought up to the surface. Once the assumptions are exposed, it is believed that witnesses to the debate will develop a better view of the situation by synthesizing the conflicting views. Like Kantian systems, the information content of Hegelian inquiring systems cannot be localized in any single component of the inquiry systems and is a property of the inquiring process as a whole.

The main strength of Hegelian inquiring systems is that they actively involve decision makers in the information creation process. Hegelian systems, however, can create conflicts when such may not be appropriate.

2.5.5. Singerian-Churchmanian Inquiring Systems

Singerian inquiring systems are the archetype of interdisciplinary, meta-inquiring systems. As meta-inquirers, they study other inquirers and incorporate other inquiring modes in an attempt to produce the most comprehensive view. Essential to successful formulation of a comprehensive view are continual learning and adaptation. Whenever a consensual view emerges, Singerian systems partition the view into several parts to create a disagreement. Then, they convert the disagreement into an agreement on a higher level.

Singerian systems consider the guarantor notions of other inquiring systems as objects for infinite inquiry. Inquiry is an endless process in which any factual statement is simply a self-imposed imperative of the inquiring community. The information content of Singerian systems can be formal, experiential, synthetic, conflictual, ethical, or of almost any other nature.

The strength of Singerian inquiring systems is that they give the most comprehensive view of any problem, utilizing various inquiring modes together with science, ethics, philosophy, etc. However, Singerian systems are very expensive to design, operate, and maintain.

2.6. Computer-Based Problem Diagnosis Support Systems

The focus of this section is to review the computer-based support systems specifically designed to support managerial problem diagnosis. These PDSS include structural modeling systems, causal relationship discovery systems, causal model development systems, and causal model application systems. In addition, this section examines Bouwman's financial diagnosis system (1983), Baldwin's multiple viewpoint system (1989), and MYCIN (Shortliffe, 1976).

The main utilities of structural modeling systems are: (1) to allow users to represent variables and their causal relationships in a computer system; and (2) to provide the users with an analysis of the causal connections. For example, Pracht's system (1984) accepts a user's structural model in a form similar to Figure 2.2 and analyzes the causal connections. Pracht's system allows the users to denote the arcs with either +1 (positive relationship) or -1 (negative relationship). This empirical research found that the structural modeling system is an effective tool for high analytic subjects but not for low analytic subjects.

Ramaprasad and Poon (1985) proposed a problem structuring technique, called mapping influence diagrams (MIND). MIND consists of five steps: (1) identifying classes of problem elements; (2) specifying major elements; (3) specifying relationships between elements; (4) specifying strengths of relationships; and (5) mapping the influence diagrams. The users carry out the first four steps. During the last step, the computer system analyzes the connectivity between elements. MIND can compare and integrate several related influence diagrams.

Khazanchi (1991) also developed a similar structural modeling system called cognitive lens support system (CLSS). CLSS allows users to represent, analyze, compare, and integrate several structural models. Khazanchi's system displays structural models graphically (i.e., Figure 2.2), whereas MIND displays them digitally (i.e., Figure 2.3). Burns (1985, 1989) has been involved in the development of structuring tools for causation.

In contrast to the above systems, which rely on users for hypothesis formulation, Billman's system (1989) formulates hypotheses of causal relationships between pairs of variables without interacting with users. The discovery system is based on three causal reasoning methods: (1) Piaget's model of the hypothetico-deductive operations underlying causal thinking development (Piaget, 1974); (2)

John Staurt Mill's methods of experimental inquiry; and (3) Einhorn and Hogarth's model of causal thinking (1982, 1986). Essentially, the role of the discovery system is to automatically fill in the matrix of Figure 2.3. Although Billman (1989) recognizes that a Singerian inquiring system is most appropriate for managerial problem diagnosis, the discovery system is Lockean in nature.

All the above-discussed systems assume either user-driven modeling or system-driven modeling. Paradice's emphasis (1986), however, was to develop an interactive modeling support system. Like structural modeling systems, Paradice's system allows users to assert causal relationships based on their a priori knowledge. The role of the system is to evaluate the causal assertions by formulating appropriate statistical models. That is, users hypothesize causal relationships; the system builds statistical models; and, then, the users reevaluate the initially asserted relationships based on the statistical models.

The statistical models used in Paradice's system rely on correlation. Although correlation does not necessarily imply causation (Einhorn and Hogarth, 1982), Paradice and Courtney argue that "very convincing arguments for causality can be constructed from statistical inference, postulated relationships developed from knowledge of the subject matter, and common sense" (Johnson and Wichern, 1982, p. 346). In fact, this remark reflects the difficulties in formulating and testing causal relationships in the management domain. In general, causality is inferred from covariation (Brown, 1985). It is possible for certain managerial problem diagnosis situations, e.g., quality control and scientific market research, to provide experimental data. However, most managerial problem situations provide only experiential data, if any. With experiential data, as contrasted with experimental data, it is extremely difficult, if not impossible, to formulate and test the causal relationships solely based on the data. Consequently, decision makers'

subjective judgments should play an integral part in the cooperative human-machine problem diagnosis system.

Jung (1990) adopted the neural network concept for managerial problem diagnosis. Unlike a typical neural net that derives a model from the empirical data, this system derives the network from a user-provided structural model. The system restructures the user's model to achieve a certain network equilibrium.

The above-discussed systems are essentially concerned with model construction. The emphasis of Ata Mohamed's system (1985), however, is on model application. The system, called PRADS (problem recognition and diagnosis system), identifies symptoms by comparing the current values of key variables against their historical values. When the variables show significant deviations, PRADS groups the variables into a causation tree based on a preestablished knowledge base. The leaves of the tree are the base symptoms that explain all other deviations on the tree. Problem diagnosis in PRADS endeavors to determine a minimum set of base symptoms. PRADS does not support user modeling. Thus, the effectiveness of PRADS is highly dependent upon the accuracy of the preestablished causal models (Ata Mohamed, 1985).

Bouwman (1983) explored the possibility of developing a diagnostic computer system that could emulate the cognitive processes of decision makers during financial diagnosis. The main purposes of the research was to understand the cognitive processes of financial analysts and encode the analysts' cognitive activities in a computer program. The research reported that financial diagnosis "consists of a number of phases, generating successively more detailed representations" of the firm being evaluated (Bouwman, 1983, p. 656). This research indicates that it is possible, in certain managerial domains, to build a computer system that can mimic human problem diagnosis processes.

Baldwin (1989) observed that multiple views are essential for successful managerial problem formulation. A frame-based multiple viewpoint system exhibits several interesting features, such as presenting a problem from many different viewpoints, translating certain well-defined models into some other well-defined models, and maintaining alternative hypotheses. The system can be best characterized as a model management system, because its primary function is to structure a large amount of a priori declarative knowledge. The system, however, provides a more elegant way of structuring, representing, accessing, and integrating models than many other model management systems. From the perspective of problem diagnosis, however, this system is weak in utilizing data and provides little support for user modeling. That is, the system is assumed to have a relatively sufficient amount of knowledge; and the system helps users to study the problem by selecting and composing different sets of pre-formulated views of the problem. Thus, this system is very useful for structured problems.

Finally, MYCIN (Shortliffe, 1976) is a medical expert system that generates diagnoses and selects therapies for patients with bacteremia or meningitis infection. Its knowledge base contains several hundred production rules. It carries on an interactive dialogue with a physician and uses the rules to reason backward from its goal of finding significant disease-causing organisms to the clinical data available. The following is a typical MYCIN rule:

IF	the stain of the organism is gramneg, and the morphology of the organism is rod, and the aerobicity of the organism is aerobic
THEN	there is strongly suggestive evidence (.8) that the class of the organism is enterobacteriaceae.

2.7. Summary

A problem is essentially a conceptual or cognitive entity that individuals create to cope with their environment. In dealing with a problem situation, decision makers develop mental models of the situation. The problem situation is believed to exist in an objective reality. Without accepting this proposition, it is difficult to differentiate our perception of reality from our dreams. Our perception, however, deals with the images of reality, rather than reality itself, e.g., our brains store the image of a mountain, not the mountain. For human inquirers, the perception, memory, and processing of these images are not perfect, but subject to various errors and biases. One purpose of problem diagnosis is to develop a better picture of the situation by reducing the errors and biases.

Problem diagnosis, the process of developing an understanding of the problem structure, is a critical aspect of managerial decision making. The input to the problem diagnosis process include: (1) the problem stimuli; (2) the concepts and beliefs held by the decision makers; and (3) the assumptions, theories and information to frame the situation. The outputs of the process include: (1) an understanding of the cause-effect relationship; (2) the identification of the specific factors responsible for the problem symptoms; (3) a determination of a category into which the problem can be classified; and (4) the development of predictive judgment. The dynamic nature of managerial problem diagnosis process can be captured by three process characteristics: (1) recursiveness; (2) retroductivity; and (3) heterarchy.

The effectiveness of problem diagnosis depends upon a number of contingency factors, including: (1) the decision maker; (2) the problem characteristics; (3) the decision environment; (4) the inquiry process or strategies; and (5) the inquiry support systems. The inquiry process is also dependent upon

the first three above-mentioned factors. The problem diagnosis process has three distinct operations: (1) stimuli detection; (2) interpretation; and (3) association. Problem diagnosis strategies suggest that there are many different ways of executing these cognitive operations.

The role of a PDSS is to help users to better understand and interpret the problem situation. A computer-based system can help to achieve a better inquiry by facilitating the user's cognitive decision process and/or altering the decision process. An ideal support system may require both cognitive and normative elements. The formation of a cooperative inquiring community between a decision maker and the support system may improve the quality of inquiries in a number of different ways. For example, the support system can provide users a vehicle to arrive at a Lockean consensus, a mechanism to utilize Leibnitzian formalism, a basis to form multiple Kantian models, a source of Hegelian antitheses, and, eventually, a foundation for Singerian evolution.

The previous PDSS offer a variety of problem diagnosis supporting capabilities. However, they also exhibit a number of limitations. First, they rely on a very simple modeling construct that can state only whether a causal relationship exists between two concepts. Second, they focus on specific problem diagnosis activities. Third, with an exception of Paradice's system (1986), most of the previous PDSS support either user-driven or system-driven problem diagnosis. Other related systems include the financial diagnosis system (Bouwman, 1983), the multiple viewpoint system (Baldwin, 1989), and MYCIN (Shortliffe, 1976).

CHAPTER III

RESEARCH METHODOLOGY

In the history of science there seems to have been an agreement that scientific research should have at least three parts: theory, evidence, and method (Bunge, 1973). The key element of theory (model) is that it abstracts a few characteristics of reality in an attempt to isolate and describe the central features of the reality. The role of evidence is to confirm or disconfirm the theory by applying it to the real world. Method is concerned with the process of creating knowledge, i.e., the process of theory building and evidence generation.

Theory building in MIS/DSS design research includes the "development of new ideas and concepts, and construction of a conceptual framework, new methods, or a model" (Nunamaker et al., p. 94). Theory development, in general, should be well-grounded. Weber (1987) contends that information system design should be based upon some theory of information systems rather than upon a particular research framework:

> If IS researchers continue to focus on research frameworks and to pursue research in the contexts of these frameworks, the field will be an anomaly ... Furthermore, young researchers in the field are unlikely to make significant contributions if they train with a research framework orientation ... (Weber, 1987, p. 7)

One difficulty, however, is that MIS/DSS, as principles of "fragmented adhocracy" (Banville and Landry, 1989, p. 56), exhibit "no clear theoretical base and no match between theory and method" (Keen, 1980, p. 10). In fact, "DSS research seem to be based more on intuitive, atheoretic exploratory research rather than in strong theoretical referent disciplines" (Cooper, 1988, p. 92). This is more true for DSS design research than for the empirical DSS research.

Banville and Landry (1989) believe that this tendency is not much of a problem at the current state of DSS development. They present an argument that MIS/DSS design research may elect to establish its own new and creative research framework and pursue research within the contexts of the framework. Their argument, in part, is based on Herbert Simon's view of scientific progress:

> Confusion, by another name, is progress to which we have not yet become accustomed. ... Science, like all creative activity, is exploration, gambling, and adventure. It does not lend itself very well to neat blueprints, detailed road maps, and central planning. (in Banville and Landry, 1989, p. 50)

Kohen (1986) suggests that what really matters is whether the framework is sound or arbitrary. A sound research framework, in the absence of the theory, helps a system designer open new lines of inquiry. On the other hand, an arbitrary framework can be misleading.

An ultimate test for the soundness of a design framework is to develop and implement a system based on the framework, and then validate the concepts of the framework by evaluating the performance of the system (Cooprider and Henderson, 1991; Nunamaker et al., 1991). In fact, the development of information systems to substantiate a conceptual model is a standard research method in software engineering, artificial intelligence, expert system development, and DSS design (Newell and Simon, 1976). O'Leary states that the development of actual information systems demonstrates a "proof of concept" (O'Leary, 1988, p. 32). Nunamaker et al. (1991) also advocate the system development approach to evidence generation for MIS/DSS design research:

> The pivotal role of system development ... is ... that the developed system serves both as a proof-of-concept for the fundamental research and provides an artifact that becomes the focus of expanded and continuing research. (Nunamaker et al., 1991, p. 92)

In design or engineering research, the prototype system development, together with simulation, is perhaps the most well-established research method.

Two research methodologies that discuss both the theory building and evidence generation aspects of information system design were examined to guide this research. They were a "unified research methodology" (Yadav and Baldwin, 1990) and a "system development research process" model (Nunamaker et al., 1991). They both combine the traditional social science research method with information system design methods. This research utilizes insights gained from the research methodologies and proceeds in the following steps:

1. Formulate the Problem;
2. Construct a Conceptual Framework;
3. Design the System on the Conceptual Level;
4. Construct and Implement the System;
5. Test and Evaluate a Prototype System; and
6. Validate the Design.

The first step is akin to observation and problem recognition (Chapters I and II). The second and third steps are essentially model building or hypothesis formulation activities (Chapters IV and V). The fourth step (Chapters VI) approximately corresponds to the experimental design stage of the traditional social science research. The fifth step (Chapter VII) is equivalent to an experimentation. The final step (Chapter VIII) evaluates the experimental results and draws conclusions.

3.1. Problem Formulation

A logical first step in any research is to identify a research problem and establish the research objectives. Like managerial problem recognition, research

problem identification requires an understanding of the current state and the conception of a goal state.

A review of problem diagnosis literature revealed that problem diagnosis requires a variety of activities. PDSS must interactively support these activities to improve the problem diagnosis process. The system must allow users to represent their mental models and, most often, extend the mental models.

Section 1.1, however, indicated that the previous PDSS reviewed in Section 2.6 are not satisfactory in three major aspects. They: (1) employ a simple modeling construct; (2) limit their focus to certain problem diagnosis activities; and (3) rarely combine user-driven and system-driven problem diagnosis.

One major objective of this research was to establish a conceptual framework for developing inquiring support systems for managerial problem diagnosis. The basic premise of this research is that the shortcomings of the previous PDSS can be overcome by: (1) formulating a sound conceptual framework for PDSS development; (2) carefully deriving the requirements of the PDSS; and (3) designing PDSS that satisfy the requirements.

3.2. Conceptual Framework Development

A scientific inquiry requires theoretical foundations. Scientific research builds upon existing concepts, refines the concepts, creates new concepts, and generates evidence to affirm or disaffirm the validity of the concepts. To refine existing concepts and formulate new concepts useful for the design of PDSS, this research took the following steps.

The first step formulated a cooperative inquiry system concept by relating and synthesizing problem diagnosis literature, MIS frameworks, DSS design frameworks, a cooperative system concept, and inquiry "guaranteeing" concepts.

The second step developed a conceptual framework for PDSS development based on the cooperative inquiry system concept. The framework delineates essential elements of PDSS, logical steps in developing and using a PDSS, and the roles played by the individuals involved in the system development and use. The third step presented the assumptions of this research.

The next three steps detailed the framework with respect to the knowledge requirements of PDSS. The fourth step established the knowledge structure requirements of PDSS, i.e., the system's capability to acknowledge and manage different types of causal models. Logical arguments were used to derive the knowledge structure requirements. The fifth step determined the problem processing requirements of PDSS. The problem processing requirements were established by identifying those activities essential for successful problem diagnosis and inherent in most problem diagnosis processes. The sixth step established the interaction requirements, i.e., the system's capabilities to communicate and exchange knowledge with users.

3.3. Conceptual Design

One design approach that has been repetitively advocated in computer science and MIS is the multiple-level design approach. In general, an information system design can be carried out on three levels, i.e., system requirements definition level, a functional specification level, and a programming level (Langefors, 1973; Ross and Schoman, 1977).

Based on the requirement definitions established in the previous step, the design of PDSS was carried out on two levels: conceptual design and symbol-level design. Conceptual design approximately corresponds to functional specification or knowledge-level design (Newell, 1982). Conceptual design establishes the

structures and functionalities of the system. This research performed the conceptual design of PDSS in the following three steps. The first step identified major types of views that constitute the system's knowledge base. The second step formulated various support functions that may facilitate decision makers' problem diagnosis processes. The third step designed the ways that the users and the system may interact to reduce the uncertainties and errors.

3.4. Symbol-Level Design and System Implementation

Symbol-level design is concerned with the realization of knowledge-level design in terms of well-defined symbols. It operationalizes the elements, functions, and behaviors of PDSS delineated on the conceptual design level in terms of abstract symbols. This research performed symbol-level design in the following steps.

The first step developed knowledge representation schemata. The schemata outlined the attributes and attribute structures of various views using a set of precisely defined symbols. The second step defined critical system-driven problem processing functions in terms of the symbols defined in the previous step. Eventually, the symbol structures and symbol manipulation procedures must be encoded into a computer program. This research adopted C++ as the base development language. C++ offers great flexibility and efficiency. As an object-oriented programming language, C++ supports data encapsulation, inheritance, and polymorphism. These features promise simplicity of programming, ease of program maintenance, and reusability of the codes.

3.5. Prototype System Evaluation

The soundness of the design was examined through construction of a prototype system and evaluation of the system's performances. Prototype systems

are often developed as preliminary versions of the entire system, so that the benefits of the total information system being envisioned can be assessed without committing too many resources. Thus, prototype systems are often considered to be disposable in practice. From a research viewpoint, however, prototyping is similar to an experimentation design. Thus, prototype system development in a research setting must be carefully planned and executed.

A critical step in the prototype evaluation stage is to determine the application domain. The most adequate situations where the prototype system exhibits its maximum spectrum of capabilities are semi-structured to ill-structured problems with some empirical data. One such case is the bond rating problem. This problem has been studied by many financial and statistical researchers. To demonstrate the system's generalizability, the same prototype system is used to solve yet another problem. The second problem is the managerial diagnosis of blackline burning feasibility (Wright et al., 1992). These problems are relatively ill-structured.

3.6. Research Validation

The objectives of this research were to formulate a conceptual framework for developing PDSS, to establish the PDSS requirements within the conceptual framework, to design a PDSS that overcomes the limitations of the previous PDSS, to implement a prototype system based on the design, and to evaluate the design based on the prototype system. This research was evaluated with respect to four types of validity.

The first step examined the face validity of this research. Two major issues addressed at this step were: (1) whether the research objectives were worth pursuing; and (2) whether the conceptual framework was intuitively valid. The

second step examined the internal validity of this research; i.e., whether the design process had been carried out in a consistent manner. The third step examined the construct validity. Construct validity was indirectly assessed by comparing the prototype system's capabilities with those of previous PDSS. The final step discussed the external validity of the research. Perhaps the best way to examine the effectiveness of the design is to employ the system as a tool in a laboratory setting and examine its effect. Unfortunately, the limited research resources did not allow such an empirical test.

3.7. Summary

This chapter explained the research method employed in this study. The research method consists of two major parts: model construction and evidence generation. In this research, model construction included: (1) formulation of a conceptual framework; and (2) a conceptual design of PDSS. The framework was derived from synthesizing problem diagnosis literature, MIS/DSS frameworks, and other sound design concepts such as process-oriented decision support, a cooperative system concept, and the inquiry "guaranteeing" concepts. Conceptual design establishes the structures and functionalities of the system. The conceptual design of PDSS included: (1) the identification of critical knowledge base elements; (2) the formulation of problem diagnosis support functions; and (3) the incorporation of inquiry "guaranteeing" concepts into PDSS design.

Symbol-level design and prototype system implementation provide a proof that the conceptual design is viable. Symbol-level design is concerned with the realization of knowledge-level design in terms of well-defined symbols. It operationalizes the elements, functions, and behaviors of PDSS delineated on the conceptual design level in terms of abstract symbols. Eventually, the symbol

structures and symbol manipulation procedures must be encoded into a computer program. This research used C++ as the base development language. Prototype implementation should be consistent with symbol-level design. Furthermore, the prototype system must achieve the structures, functions, and behaviors specified on the conceptual design level. This research tested the prototype system using two different problems. The final phase of this research evaluates the conceptual framework, the conceptual design, the symbol-level design, and the prototype system.

CHAPTER IV

A CONCEPTUAL FRAMEWORK FOR MANAGERIAL PROBLEM DIAGNOSIS SUPPORT SYSTEMS

One of the earliest and most widely used DSS design frameworks of Sprague (1980) identifies three major structural components of a DSS: the data base management system (DBMS), the model management system (MMS), and the dialogue generation and management system (DGMS). Although Sprague believes the DGMS "clearly the most important subsystem" (Sprague and Carlson, 1982, p. 29), it has been the DBMS and the MMS that receives the most attention in DSS design. Data and models are two major components that determine the knowledge content of the system (Alter, 1977; Bonczek et al., 1980).

The data-oriented DSS design approach maintains that a key to effective decision making is to provide the right information at the right time to the right decision maker. On the other hand, the model-oriented DSS design approach views the identification, representation, integration, and management of models as the key requirements for effective DSS development. The "duality between data and models" (Dolk, 1986, p. 73) has been a critical issue not only in DSS but also in scientific research and philosophy. A general agreement achieved in all of these areas is that data and models are interdependent of each other (Bahm, 1953; Ackoff et al., 1962; Hogarth, 1986).

Sprague and Watson (1976) state that a DSS is "to give access to data and models at a speed which matches the thought processes of the manager" (p. 659). That is, DSS is a service system for which utilities cannot be determined independent of the users' decision processes (Parker and Al-Utaibi, 1986). Thus,

one critical issue in DSS design is how a DSS could be, or should be, combined with the users' decision processes (Rangaswamy and Federowicz, 1983).

One school of thought advocates the user-oriented modeling approach. Here, users are assumed to have the ability to correctly construct and apply specific decision aids by combining several data and model sets in some sequence of their choice. For this purpose, a DSS should provide high-level data manipulation and model specification languages (e.g., Liang, 1985; Ghiaseddin, 1986; Geoffrion, 1987; Burns, 1990). That is, the role of DSS is to help users structure a semi-structured or unstructured problem.

Another school of thought recommends the system-oriented modeling approach (Bonczek et al., 1981; Blanning, 1982; Hwang, 1985; Klein et al., 1985; Klein, 1986; Liang and Jones, 1988). Here, the computer systems determine a suitable decision aid by selecting appropriate data and models. The role of DSS is to help users solve a semi-structured or unstructured problem by taking over the structured portion of the problem.

From the teleological perspective of DSS, i.e., a cooperative man-machine problem solving system, it is necessary to combine both user-driven and system-driven approaches. The need of coordination between the two processes highlights the importance of DGMS.

With this brief overview of DSS design approaches, this chapter presents a conceptual framework for PDSS development. The first section provides an overview of the framework. The second section examines the processes of PDSS development and use. The third section discusses the assumptions of this research. The remaining three sections detail the framework in terms of PDSS knowledge requirements: knowledge structure requirements, problem processing requirements, and user-system interaction requirements.

4.1. An Overview of the PDSS Development Framework

This section provides an overview of the conceptual framework for PDSS development. The framework for PDSS development: (1) identifies contingency factors for problem diagnosis effectiveness; (2) establishes three essential elements of PDSS; (3) formulates a cooperative inquiry system concept; (4) delineates three logical steps in PDSS development and use; (5) discusses individuals' roles in the PDSS development and use processes; and (6) formulates the requirements for PDSS. The purposes of this section are: (1) to identify those factors that affect problem diagnosis effectiveness; (2) to identify essential elements of PDSS; (3) to frame the outcomes of the first two steps with the cooperative system concept and inquiry "guaranteeing" concepts; (4) to establish a PDSS development strategy and identify logical steps in PDSS development and use; and (5) to combine the outcomes of step 3 and step 4.

4.1.1. Contingency Factors for Problem Diagnosis Effectiveness

As examined in Chapter II, the effectiveness of problem diagnosis is affected by those factors shown in Figure 4.1. The decision process refers to a series of activities that a decision maker performs to deal with the problem situation. Underlying the decision process are a set of cognitive activities, functions, or operations. The mental models refer to the conceptual models formulated for the problem situation. The decision process and its subsequent mental models are affected by the problem situation, decision environment, and PDSS. The mental models and the decision process are tightly coupled. The decision process affects the formulation of the mental models. The mental models being developed affect the directions of decision process.

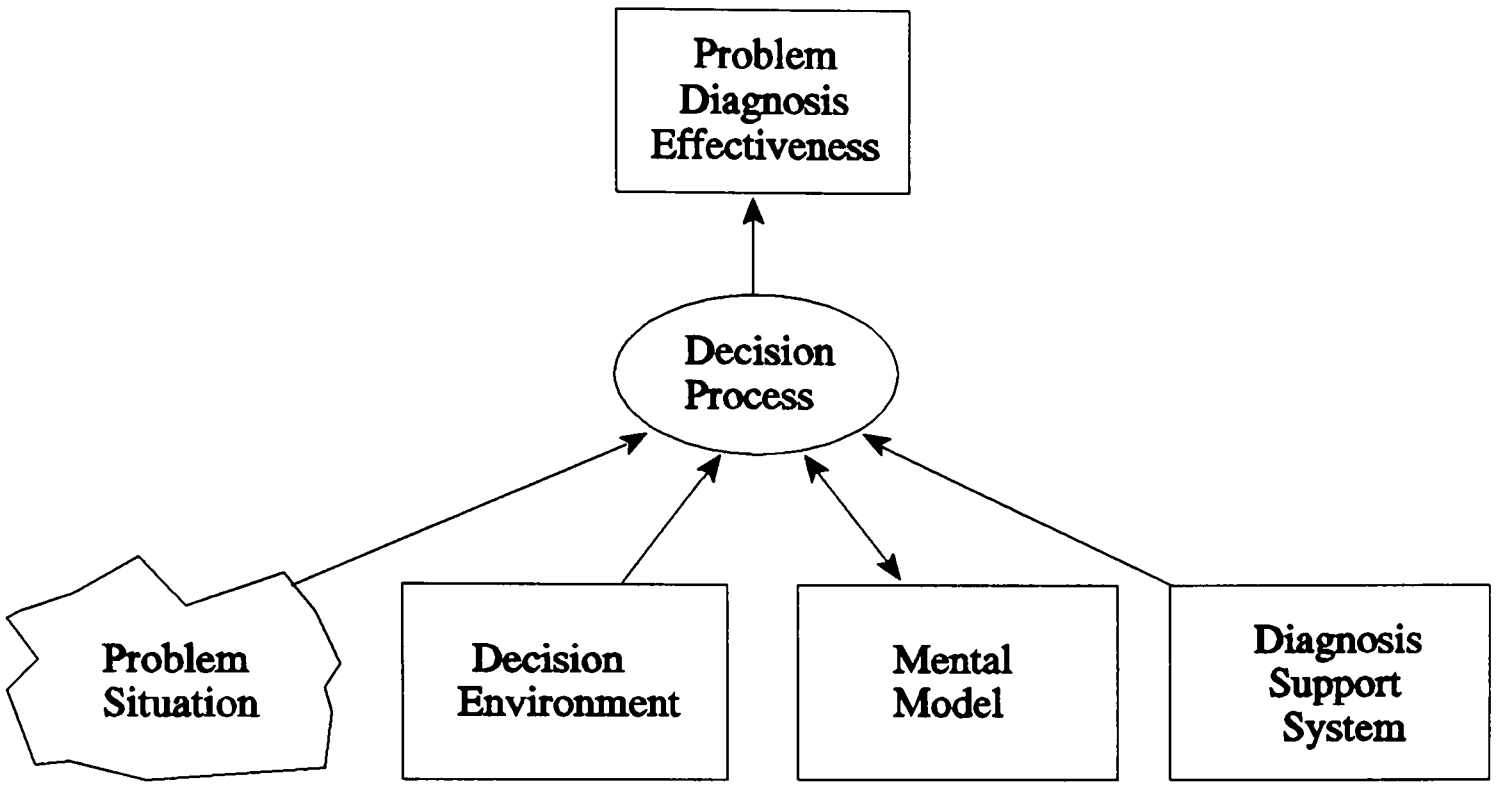


Figure 4.1. Factors Affecting the Effectiveness of Problem Diagnosis

Figure 4.2 maps the factors in Figure 4.1 with the factors discussed in six widely-used MIS research frameworks. Figure 4.2 indicates that Figure 4.1 is fairly consistent with extant MIS research frameworks.

4.1.2. Three Essential Elements of PDSS

In general, knowledge can be largely classified into declarative and procedural knowledge. Declarative knowledge refers to the mental models, logico-mathematical structures, cognitive schemes, or cognitive maps. Procedural knowledge refers to the basic cognitive activities, functions, or operations. In fact, studying a system in terms of its declarative (structural) and procedural (functional) components is, by no means, new in information system research. In object-oriented programming, an object has two major elements, data structure and methods (Howard, 1988). A system analysis methodology, IDEF (see Bravoco and Yadav, 1985), employs three diagrams: (1) $IDEF_0$ models the system's functions; (2) $IDEF_1$ models the information structure; and (3) $IDEF_2$

Factors \ Frameworks	Mason & Mitroff (1973)	Chervany et al., (1971)	Lucas (1973)
Problem Diagnosis Effectiveness		Decision Effectiveness	Performance
Decision Environment	Organizational Context	Decision Environment	Situational & Personal Factors
Problem Situation	Class of Problem		
Decision Maker's Mental Models	Psychological Type	Directly & Indirectly Acquired Attributes	Decision Style Attitude/Perception
Decision Maker's Decision Process	Evidence Generation Method		Analysis & Actions
Problem Diagnosis Support System	Mode of Presentation	Information System	Use of System Quality of System

Factors \ Frameworks	Mock (1973)	Gorry & Scott Morton (1971)	Ives et al., (1980)
Problem Diagnosis Effectiveness	Performance Variables		
Decision Environment	Sociological, Environmental, Organizational, Interpersonal Variables	Levels of Managerial Activity	Environmental Variables
Problem Situation		Problem Structure	
Decision Maker's Mental Models	Individual/ Psychological Variables		Use, Development, Operation Processes
Decision Maker's Decision Process			
Problem Diagnosis Support System	Information Structure Variables	Information System Type	Information Subsystem

Figure 4.2. Factors Identified by MIS Research Frameworks (see Ives et al., 1980)

models their dynamic interactions. In AI, a knowledge-based system must have a knowledge base and an inference engine. Newell (1982) states that:

Knowledge --> Representation x Inference.

The symbol, -->, should read "implies that the left-hand-side can be decomposed into the elements on the right-hand-side." The above principle can be restated, using DSS terminology, as follows:

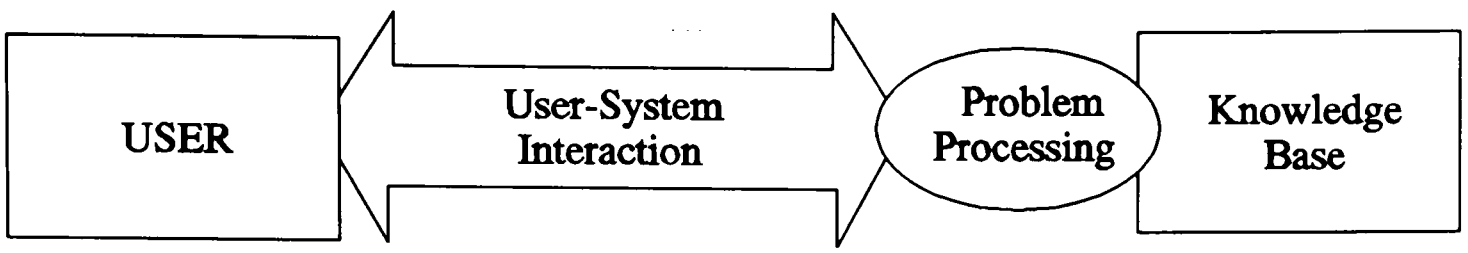
Knowledge --> Knowledge Base x Problem Processing .... (4.1).

However, unlike most knowledge-based systems in AI, a PDSS may not solve problems by itself. The system must interact with the users. Thus,

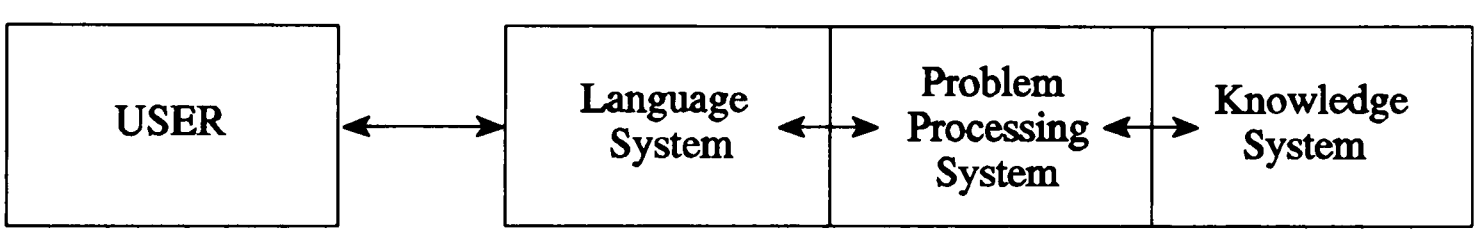
PDSS --> Knowledge Base x Problem Processing x User-System Interactions .... (4.2).

In fact, this conceptualization is not very different from extant DSS models (Bonczek et al., 1980; Sprague, 1980). The knowledge system in Figure 4.3.b refers to a knowledge base. A knowledge base stores both data and models; and problem processing encompasses data and model manipulation (see Figure 4.3.c).

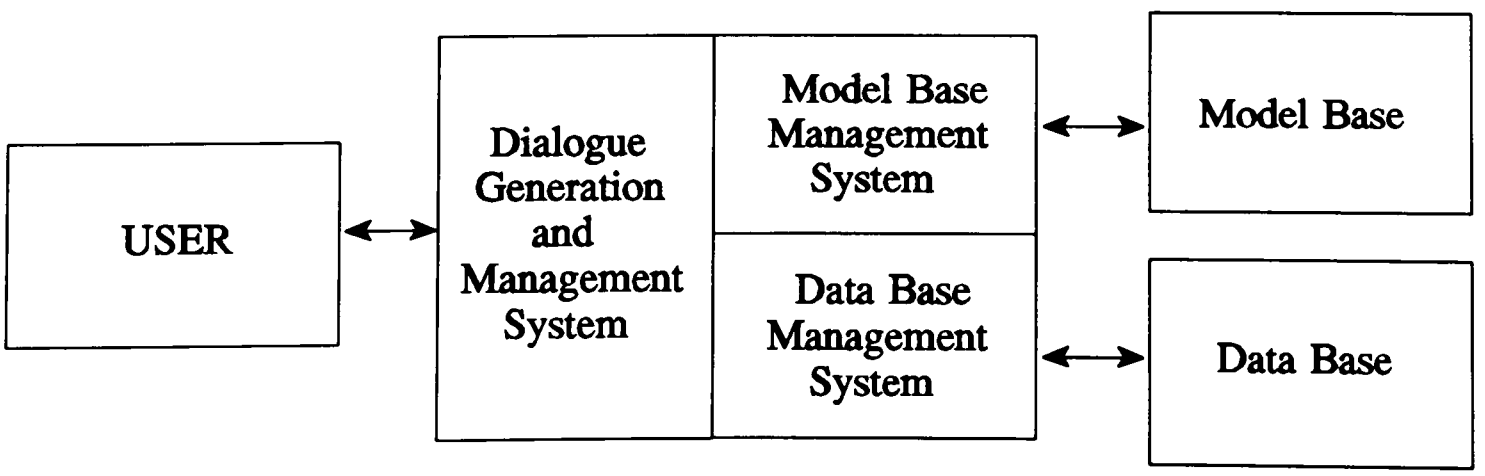
One subtle difference, however, is that user-system interaction in Figure 4.3.a has a broader meaning than user-system interface (i.e., language system in Figure 4.3.b or DGMS in Figure 4.3.c). User-system interface design is essentially concerned with developing a user-friendly communication language (Bonczek et al., 1980, p. 617; Sprague, 1980, p. 17). User-system interaction design (Woods, 1986; Hale and Kasper, 1989), however, addresses broader, or more fundamental, issues such as how to allocate problem solving efforts between the user and the system. From a broad perspective, user-system interaction design, thus, determines the overall functional boundaries of the support system. From a narrower viewpoint, interaction design refers to user-system interface design.



(a) Three Essential Elements of PDSS

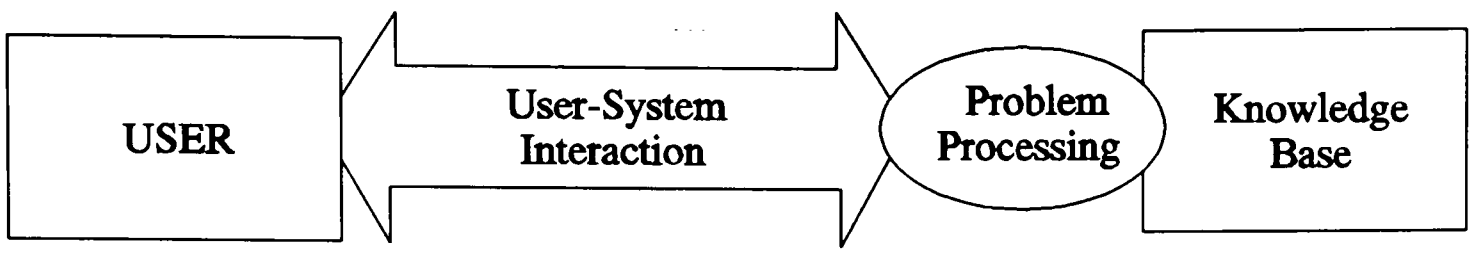


(b) Generic DSS (Bonczek et al., 1980, p. 619)

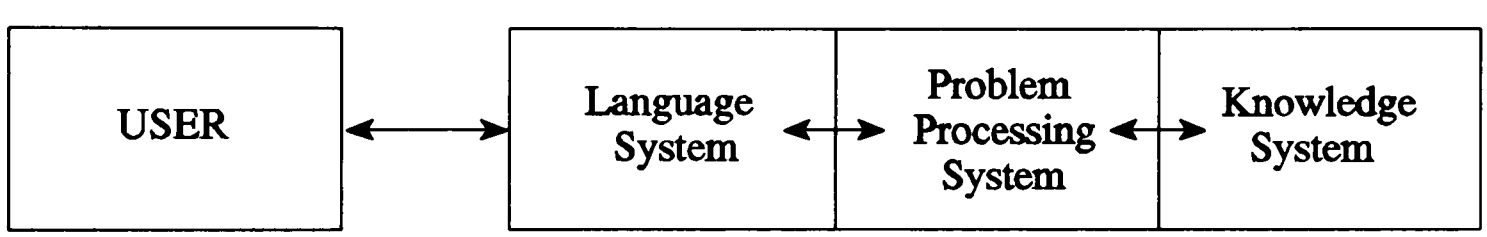


(c) Components of the DSS (Sprague, 1980, p. 15)

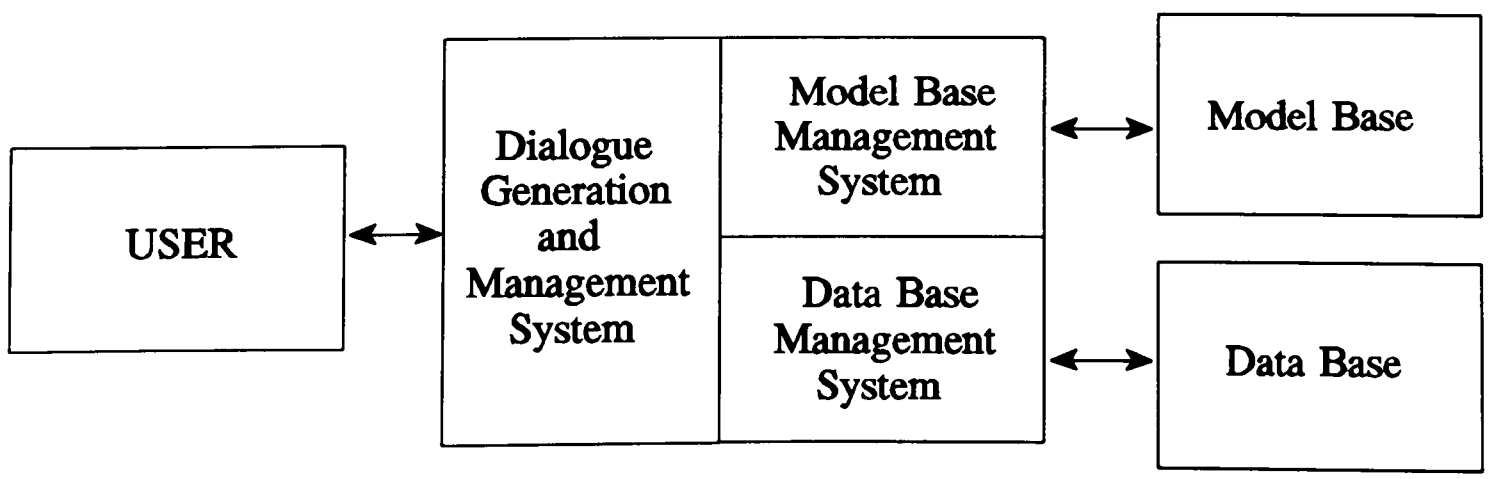
Figure 4.3. Three Elements of Decision Support Systems



(a) Three Essential Elements of PDSS



(b) Generic DSS (Bonczek et al., 1980, p. 619)



(c) Components of the DSS (Sprague, 1980, p. 15)

Figure 4.3. Three Elements of Decision Support Systems

4.1.3. A Cooperative Inquiry System

Figure 4.4 shows the essence of a cooperative inquiry system. Figure 4.4 is based on both Figure 4.1 and Figure 4.3.a. Figure 4.1 examined PDSS from a user's (performance) perspective. Figure 4.3.a examined PDSS from a developer's perspective. Figure 4.4 provides a more comprehensive view by relating and synthesizing problem diagnosis literature, MIS/DSS frameworks, the cooperative system concept (Hale and Kasper, 1989), and the inquiry "guaranteeing" concepts (Churchman, 1971). The cooperative inquiry system concept emphasizes the following points.

First, decision environment is a factor in which the problem situation is embedded, by which the decision maker is constrained, and of which the information system is a part. Thus, the decision environment factor includes sociological, organizational, interpersonal (Mock, 1973), personal (Lucas, 1973), and information system development and operation variables (Ives et al., 1980).

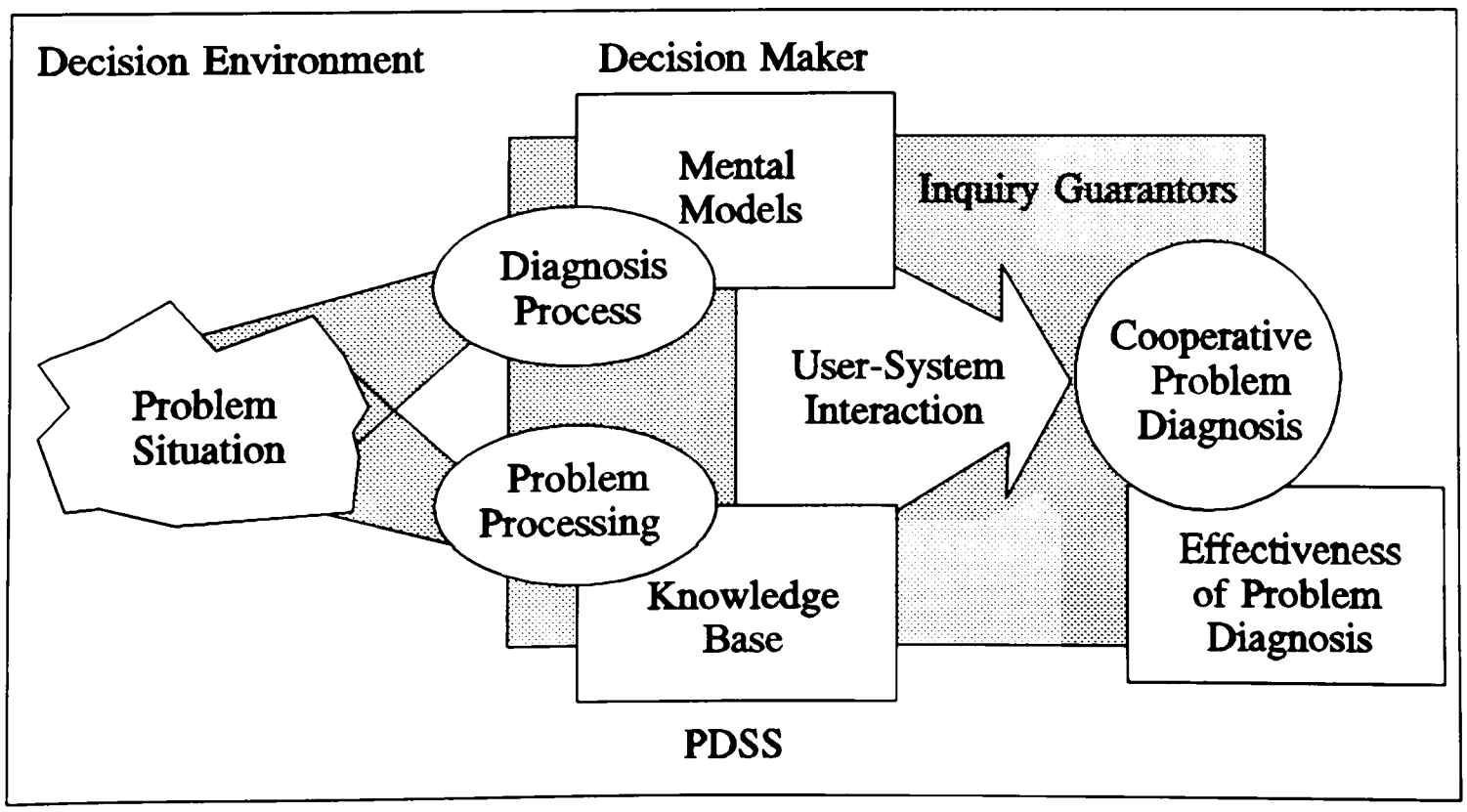


Figure 4.4. A Cooperative Inquiry System

Second, mental models and diagnosis process are the elements related to the decision maker. Likewise, knowledge base and problem processing are two major information sources of PDSS. The decision maker and the support system must cooperatively interact to understand the problem situation. Problem diagnosis effectiveness is a direct function of the cooperation achieved by the user and the PDSS for the given problem situation and decision environment.

Third, a cooperative system must have both cognitive and normative elements (Keen and Scott Morton, 1978; Sage, 1981; Hale and Kasper, 1989; White, 1990). From a cognitive perspective, a PDSS must provide users with: (1) a knowledge base that can easily accommodate users' mental models; and (2) processing functions that can facilitate the users' natural problem diagnosis processes. From a normative perspective, a PDSS must provide users with: (1) a knowledge base that can extend the users' mental models; and (2) processing functions that can alter the users' suboptimal problem diagnosis processes.

It is relevant to examine the assumptions underlying these approaches. Cognitive PDSS assume that users may better conceptualize a problem if the PDSS can facilitate the users' natural processes to formulate their mental models. Normative PDSS assume that users may develop more accurate pictures of the situation, if the PDSS can eliminate the users' biases and errors.

Finally, a cooperative PDSS must have mechanisms that guarantee, or at least attempt to guarantee, the quality of the inquiring process. A system, having both cognitive and normative elements, cannot avoid accepting both of the above assumptions. That is, a user may better understand the problem, if the PDSS helps to: (1) represent and analyze the mental models; and/or (2) augment and alter the mental models. From an opposite side, however, this also implies that both assumptions can be false. That is, the user may employ the PDSS as a

powerful tool for reinforcing his or her idiosyncratic predispositions, and/or the system's recommendations can be false. Then, how do the user and/or (less likely) the system determine whose knowledge is more "truthful?" Although a definite answer is hard to find (for unstructured problems), perhaps the best way to reduce the errors is to utilize the inquiry "guaranteeing" concepts (Churchman, 1971). In short, the cooperative inquiry system concept incorporates the inquiry "guaranteeing" concepts into the cooperative system concept.

4.1.4. Three Processes of PDSS Development and Use

One critical issue is designing information systems is the generalization level that the system seeks. If a DSS is designed for a specific problem situation, a specific decision maker, and a specific decision environment, the information requirements for the specific DSS may be derived from a careful observation of the situational factors. These systems are called decision-oriented DSS (Arinze, 1991). They can "exploit unique aspects of a particular decision process and may be tailored to the decision style of a particular user, thus making them efficient in handling particular problem areas" (Rangaswamy and Federowicz, 1983, p. 13). They, however, they suffer two major limitations: the high cost of development and inflexibility (Ghiaseddin, 1986).

Non-specific DSS are often called generalized DSS (Bonczek et al., 1981) or DSS generators (Sprague, 1980). Roughly, generalized DSS refer to the non-specific DSS directly used by decision makers, whereas DSS generators assume their users to be DSS developers. In any case, the role of these systems is to support a variety of decision processes. The recent calls for process-oriented DSS (Parker and Al-Utaibi, 1986; Weber, 1986; Elam and Konsynski, 1987; Nunamaker et al., 1988; Adams et al., 1990) seem to be under this context.

The purpose of this research is to develop a non-specific PDSS. There are two obvious reasons for studying non-specific PDSS. First, non-specific PDSS are more flexible and can be easily adapted to the changing problem situations, decision processes, and decision environments. Second, from a research viewpoint, specific DSS do not provide the desired level of generalizability.

A weakness of this approach, however, is that the non-specific PDSS may not be able to reflect the unique aspects of particular problem situations as closely as decision-oriented PDSS do. The tradeoffs between the level of generalization and the richness of description hold not only in empirical DSS research (Todd and Benbasat, 1987) but also in DSS design research.

Once a non-specific PDSS is developed, the system must be tailored to the situational factors before being used by the decision makers. Thus, there are three logical steps in developing and using the PDSS. They are non-specific PDSS development, specific PDSS development, and specific PDSS use processes. As Figure 4.5 shows, these processes are carried out by DSS designers, decision analysts, and decision makers, respectively.

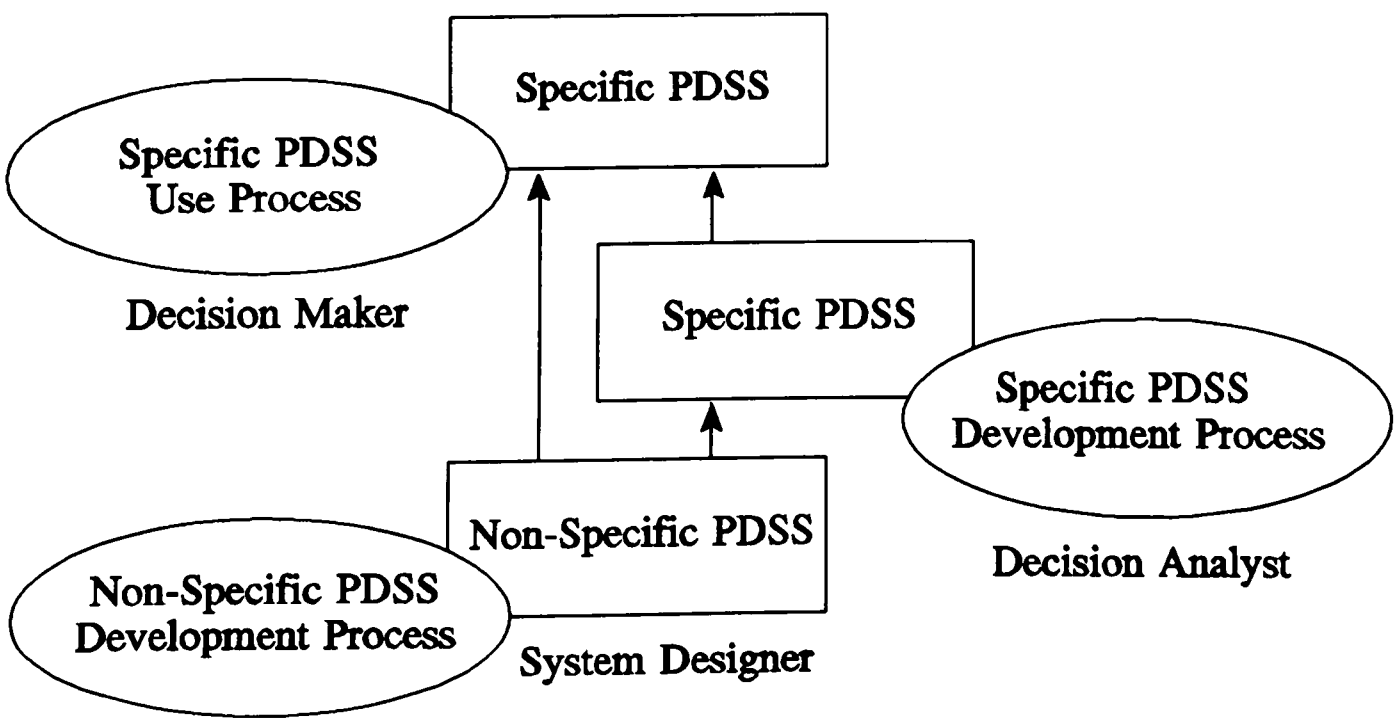


Figure 4.5. PDSS Development and Use Processes

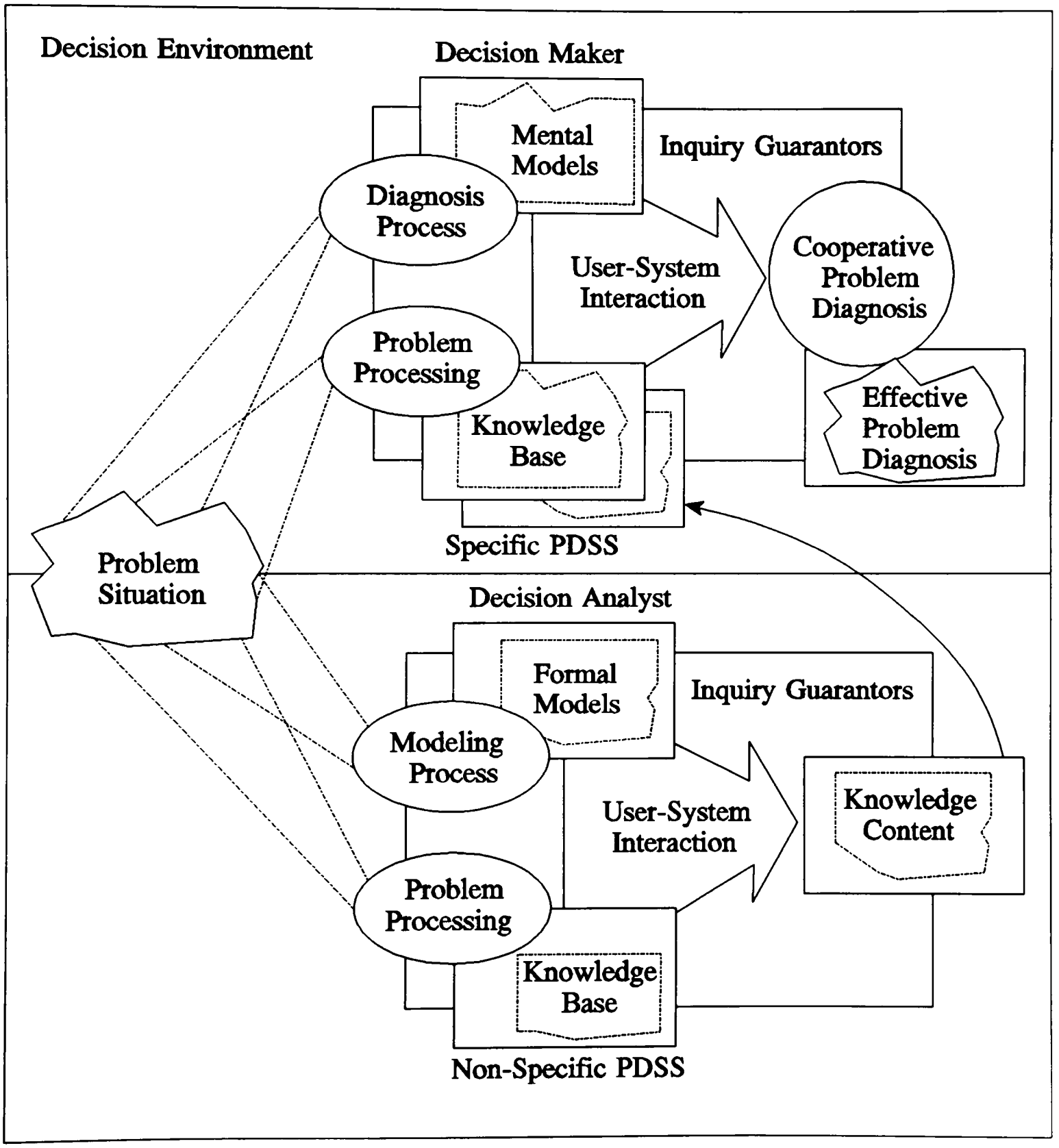


Figure 4.6. An Overview of PDSS Development Framework

decision makers should be able to accept, reject, or refine the models provided by the decision analysts. Because specific PDSS build upon a non-specific PDSS, the capabilities of the non-specific PDSS are utilized throughout the specific PDSS development and use processes.

The next section examines the PDSS development/use processes first. Then, the assumptions of this research will be presented. The remaining three sections will detail the framework in terms of PDSS requirements.

4.2. PDSS Development/Use Processes

The purpose of this section is to examine the PDSS development/use processes. The non-specific PDSS development process is the main focus of this research. In designing the non-specific PDSS, however, the designer should look forward into the PDSS development/use processes and build the necessary capabilities into the non-specific PDSS.

4.2.1. Non-Specific PDSS Development Process

Given that the goal of a PDSS is to generate knowledge about the problem situations and achieve a cooperative problem diagnosis, non-specific PDSS development process should focus on designing system's general capabilities to acquire, represent, validate, apply, and communicate knowledge.

A PDSS requires a knowledge base to represent the acquired knowledge. A knowledge base has two major elements: knowledge structure and knowledge content. Baldwin (1989) suggests that a view is composed of two elements:

View --> Grammar x Notion,

where a view refers to a piece of knowledge; a grammar refers to a certain form used to represent the view; and a notion refers to the content of the view consistent with the grammar. Assuming that the knowledge base consists of a set

of distinctive but interrelated views, the principle essentially states that:

View --> View Form x View Content,[3] or equivalently

Knowledge Base --> Knowledge Structure x Knowledge Content ..... (4.3).

Based on (4.2) and (4.3), it can be concluded that:

$$\text{PDSS} \dashrightarrow \begin{matrix}\text{Knowledge}\\ \text{Structure}\end{matrix} \times \begin{matrix}\text{Knowledge}\\ \text{Content}\end{matrix} \times \begin{matrix}\text{Problem}\\ \text{Processing}\end{matrix} \times \begin{matrix}\text{User-System}\\ \text{Interactions}\end{matrix} \quad \text{..... (4.4).}$$

A PDSS, then, must have four elements: (1) knowledge structure; (2) knowledge contents; (3) problem processing; and (4) user-system interactions. During the non-specific PDSS development process, however, it is very difficult to establish the contents of the knowledge base. It is simply not possible to state something specific about the problem situation without first knowing the particular problem situation. Thus, the system designer focuses on developing the knowledge base structure, problem processing functions, and user-system interaction capabilities. That is,

$$\begin{matrix}\text{Non-Specific}\\ \text{PDSS}\end{matrix} \dashrightarrow \begin{matrix}\text{Knowledge}\\ \text{Structure}\end{matrix} \times \begin{matrix}\text{Problem}\\ \text{Processing}\end{matrix} \times \begin{matrix}\text{User-System}\\ \text{Interactions}\end{matrix} \quad \text{..... (4.5).}$$

First, knowledge structure requirements refer to what types of views a PDSS should contain, or what kinds of schemes should be used to hold the views. A difficult question that must be answered is how to determine knowledge structure without first knowing the knowledge content. This research approaches the problem from the causal modeling perspective.

---

[3] The form and content of knowledge (Fischler and Firschein, 1987) is an important issue in ontology. Ontological inquiry requires "the basic logic of categoreal scheme" (Browning, 1990, p. 45). This means that a certain form is required to say something about objects and their relationships (Campbell, 1976, p. 108; Haak, 1978, p. 1-134). The form approximately corresponds to mental constructs in psychology, representational schemes in AI, grammars in language, modeling constructs in MS/OR, or data model and metadata in data base management. The content, then, could be memory contents, facts, vocabularies, model instances, and data occurrences in respective areas.

Second, problem processing requirements refer to what the system should do, or how the system generates knowledge. Figure 4.4 showed that problem processing should complement a decision maker's problem diagnosis process. Thus, a logical way to derive the problem processing requirements is to: (1) identify those activities that are inherent in most problem diagnosis and essential for successful problem diagnosis; and (2) support each of the identified activities. Combinations of these functions may support various problem diagnosis routines and, ultimately, the overall problem diagnosis process.

Finally, the interaction requirements refer to how the system should cooperate with the users. Interaction design addresses perhaps the most fundamental issue in DSS design, i.e., how to effectively integrate the knowledge and capabilities of the human and the computer. That is, interaction design should achieve an effective division of labor between the users and the system.

4.2.2. Specific PDSS Development Process

A non-specific PDSS, once developed, can be refined and utilized through two processes: (1) the specific PDSS development process of decision analysts; and (2) the specific PDSS use process of decision makers. There has been a debate whether a DSS should be used directly by a decision maker or through a staff assistant (Elam and Konsynski, 1987). Although the use of DSS by a non-decision maker in place of the decision maker should be discouraged (Alter, 1977), studies have indicated that the active participation of intermediaries is a prerequisite to successful DSS implementation (Andriole, 1982).

One responsibility of decision analysts is to incorporate the problem-specific knowledge into the system. This task could be as simple as connecting the non-specific PDSS with an organizational data base and as complex as

developing detailed causal models for the use of decision makers. As pointed out in Section 2.2, a structured problem for a decision analyst can be an unstructured problem for a decision maker and vice versa. If a domain expert plays the role of the decision analyst, the specific PDSS may have elements of an expert system.

In general, the amount of problem-specific knowledge that the decision analysts must incorporate into the system depends on the decision environment, mental model, and cognitive process of the decision maker who uses the specific PDSS. Thus, the decision analysts should interact with the potential decision makers to identify the specific PDSS the decision makers need and/or want.

In essence, the specific PDSS development process can be characterized by:

$$\text{Specific PDSS} \dashrightarrow \text{Non-Specific PDSS} \times \text{Knowledge Content} \quad \text{..... (4.6).}$$

There are at least three major approaches to knowledge acquisition, (1) the expert-driven approach, (2) the engineer-driven approach, and (3) machine-driven approaches (Kim and Courtney, 1988). The first two approaches suggest that a non-specific PDSS should interact with users to acquire knowledge, i.e., learning by being told. Implicit in this statement is that the problem-specific knowledge should be accepted and stored by the non-specific PDSS rather than hard-coded by the DSS designers. In this sense, a non-specific PDSS serves as a modeling environment for developing specific PDSS. The interactive modeling capability helps the users to adapt the system to changing problem spaces.

Machine-driven knowledge acquisition indicates that the system's problem processing should include learning and discovery of new knowledge. Specifically, machine-driven knowledge acquisition refer to the inductive, deductive, and abductive reasoning. Inductive reasoning generates models from data. Inductive reasoning is often called learning-from-examples or learning-from-observations,

inclusive of parameter learning and rule discovery. Deductive reasoning generates specific conclusions based on the models. Abductive reasoning is a special method used to identify causes and generate diagnostic explanation/justification.

A cooperative PDSS must utilize both analyst-driven and machine-driven knowledge acquisition methods. Thus, the system's knowledge base is likely to contain not only analyst-provided views but also system-generated views. For the same reason, the non-specific PDSS requires two categories of functions: system-driven and user-driven. The system-driven functions refer to the various inferencing methods. The user-driven functions allow users to state and examine their beliefs in a computer system.

One strategy to combine the user-driven and machine-driven approaches is to divide problem diagnosis tasks into two mutually exclusive sets in a way that maximally utilizes each individual system's capabilities. Another strategy, suggested by Woods (1986), is to introduce redundancies between the allocated tasks. The redundant design approach appears to be more appropriate for supporting semi-structured or unstructured problem diagnosis.

4.2.3. Specific PDSS Use Process

An ultimate purpose of developing and using a PDSS is to improve decision makers' problem diagnosis performances. The theoretical basis of the expected performance improvement is a holistic human-computer partnership that achieves information processing not possible by either the human or the computer alone (Licklider, 1960). An effective problem diagnosis requires a symbiotic combination of: (1) decision maker's mental models and cognitive processing; and (2) the system's knowledge and problem processing capabilities (Hale and Kasper, 1989). The human information processing system and computer-based systems

have some contrasting generic traits, which can be used as a general basis for task allocation. Computer systems can store/retrieve large amounts of information and perform massive computation accurately in a short time. Decision makers, on the other hand, tend to have highly organized knowledge and perform more abstract and qualitative computations better. In fact, the problem processing functions of the non-specific PDSS are designed with such a consideration.

During the specific PDSS use process, however, decision makers may utilize not only the system's generic processing functions but the knowledge formalized by a decision analyst as well. The contents of the system's knowledge base and its qualities depend on the decision analyst's knowledge as well as the time/efforts devoted to the development of the system. Thus, once the system accumulates significant amount of problem-specific knowledge by interacting with the decision analyst, it may not be clear, from the system's perspective, whether the decision analyst's models or the decision maker's models are more truthful.

Consequently, it is difficult to statically allocate the tasks between the decision maker and the system for the specific PDSS use process. The decision maker and the decision analyst must interact with each other to define their roles. For example, if the decision environment dictates an immediate decision, the decision maker may want to simply apply the analyst-provided models and reflect the model results in the decision. Some decision makers may want to retrieve the analyst's models, modify them, and compare the modified models with the analyst's models. Other decision makers may want to develop their own models.

In summary, the role of decision maker is to utilize and/or refine the analyst-provided specific PDSS. The development of the specific PDSS continues in the PDSS use processes. The decision maker, together with the decision analyst, must define the role of specific PDSS according to the situational factors.

4.3. Research Scope and Assumptions

When the domain of a PDSS encompasses a number of different problem situations, the information requirements of the PDSS cannot be determined based on simple observation. Rather, the requirements should be established based upon an abstraction of problem situations. Abstraction and assumption-making are necessary ingredients for the design of a non-specific PDSS. In this research, the following key assumptions are made.

First, the problems to be supported by the PDSS are semi-structured/ unstructured, but repetitive. There is no need to justify the first part of the assumption, because the goal of DSS is to support semi-structured/unstructured problems (Gorry and Scott Morton, 1971; Sprague and Carlson, 1982). A semi-structured or unstructured problem means that the problem structure is ambiguous or unknown. Without the assumption of problem repetitiveness, the support system should then begin with little information resource; i.e., unknown models and no data. Under such a situation, there is simply not much a computer system can do. Repetitive problems are assumed to ensure that there exists a certain amount of experiential data pertinent to the problem situation.

Second, the information stored in the data base is made up of critically relevant attributes of a problem situation. This assumption is reasonable in that most business organizations maintain massive data resources, and that the data are collected for critical variables. In addition, it is assumed that the data reflect the problem situation in a unbiased fashion, more or less. One essential role of PDSS is to develop causal models, together with the users, based on the available data. If the data are totally unreliable, it is difficult to expect that the computer system, or even a human, can develop credible models. In fact, this assumption is implicitly and explicitly held by all inquiring systems that conduct inductive

explorations and/or model refinement through observation and testing, e.g., Lockean, Kantian, Hegelian, and Singerian inquiry systems.[4]

Third, the cause-effect relationships are assumed to be the most important information for successful problem diagnosis (Wallace, 1974; Fales, 1990). Aristotle wrote that "what is called Wisdom is concerned with the primary causes and principles" (in Bunge, 1979, p. 27). Spinoza's remark, "everything in nature is a cause from which there flows some effect," denotes that no scientific explanation is possible without hinging upon the causation category. However, it should be noted that the cause-effect relationship, in spite of its central role, is not the sole category of relationships that individuals formulate during the problem diagnosis process. In fact, David Hume points out that causality is only a special type of relation that individuals formulate to associate experiences rather than facts in general. Mackie (1965), a philosopher, also points out that causal thinking can be hardly separated from the "causal field" or the background knowledge. Nonetheless, the importance of causal relationship for learning (Piaget, 1974), understanding (Mackie, 1965; Crocker, 1981), and judgment (Nisbett and Ross, 1980) cannot be overemphasized.

Fourth, it is assumed that individual decision makers, not a group of decision makers, use the PDSS. If a group decision making situation is assumed instead, the most essential role of PDSS is perhaps to promote the interactions among the group participants, (Bui and Jarke, 1986), i.e., "level-I group decision support" (Gallupe et al., 1988). This research focuses on the interactions between

---

[4] Leibnitzian systems are also susceptible to false information, because false information can be developed into alternative fact nets. The basic assumption of Leibnitzian systems, however, is that "the nets containing false reports will eventually 'shrink' in comparison with the realistic fact net" (Churchman, 1971, p. 98). This assumption, in turn, builds upon the belief that "God is on our side."

a decision maker and the system rather than the active interactions among the decision makers.

Finally, the focus of this research will be limited mainly to the non-specific PDSS development process. There are various issues to be addressed for the specific development and use processes: (1) how the decision analysts and the decision makers should cooperate; (2) how the decision environment will affect their cooperation; (3) how the specific PDSS should be adjusted for changing decision environment; and so on. The main question answered in this research is what attributes a non-specific PDSS must have to promise, on its own part, its effectiveness.

4.4. Knowledge Structure Requirements of PDSS

Idealistically, a knowledge base should store all the "correct" views of a problem situation. The knowledge structure requirements of a non-specific PDSS, however, is more concerned with the identification of critical view types than the specific views themselves. A question that must be answered is, "How can the critical view types be identified without first knowing the specific views themselves," i.e., "How do we know something without experiencing it?" The answer must be found in rationalism; and this research answers the question by adopting a causal modeling approach.

The purpose of this section is to identify an essential set of modeling constructs to represent the cause-effect relationships (other necessary views are identified in the next chapter). Perhaps, the most generic way to model a problem situation is to conceptualize it in terms of objects and object relationships. This approach, categorizing reality in terms of objects and their relationships, is most natural in our minds (Piaget, 1974) and is widely used in

almost all inquiring communities, including philosophy (Bahm, 1953), metaphysics (Campbell, 1976), logic (Haak, 1978), psychology (Piaget, 1974), science (Bunge, 1973), data base (Chen, 1976), management science (Geoffrion, 1987), etc.

Concentrating on the causal relationship, a simple structural model, e.g., the one shown in Figure 2.2, should be able to represent objects and their relationships. It is possible for the objects and object relationships to have values. For instance, in the standard logic, a statement about object or object relationship, i.e., a proposition, can have one of two possible values, either true or false.[5] The type of inference that a simple structural model can produce is precisely this: whether an object exists in the model, and whether a relationship exists between objects.

The objects, i.e., nodes, in the structural model can represent either: (1) variables, such as "market growth" and "financial performance"; or (2) events, such as "slowly growing market" and "unsatisfactory financial performance." A variable may assume either a quantitative domain or a qualitative domain. A qualitative domain consists of a set of distinctive labels (inclusive of the truth values), while a quantitative domain consists of a set of measured numbers (often on a continuous scale). An event is a variable associated with its domain value(s): e.g., market-growth > 0 percent, or financial-performance = unsatisfactory (see Figure 4.8).

Figure 4.7 presents a structural model representing the relationships between quantitative variables. In studying the relationships, inquirers often want to know how variables are related (Figure 4.7) in addition to whether they are related (Figure 2.2). That is, the inquirers are often interested in measuring the causal impacts between variables. In a simplest case, the inquirer may want to

---

[5] Multi-valued logic formalizes other truth values. For example, fuzzy logic considers truth values lying on a scale between true and false.

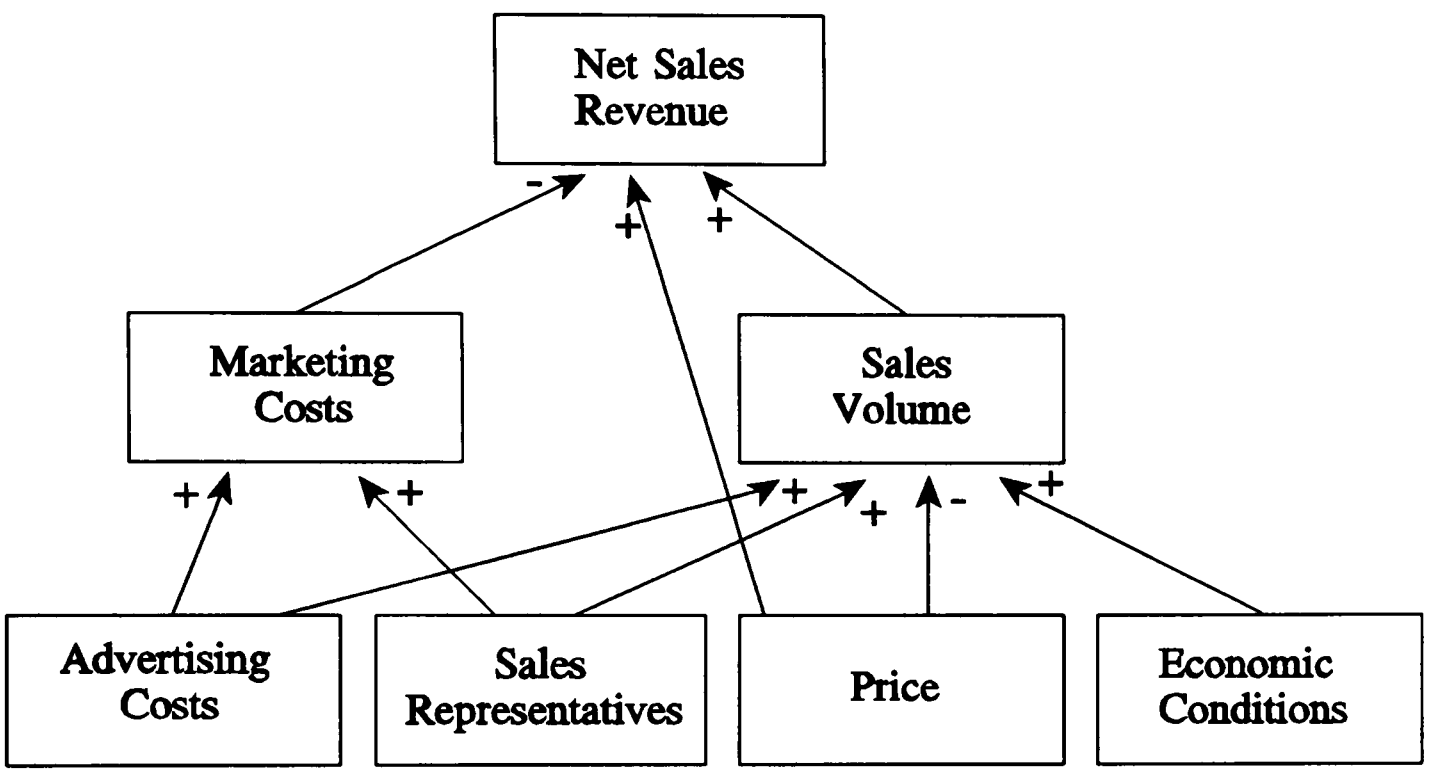


Figure 4.7. A Structural Model Representing Variables and Their Relationships

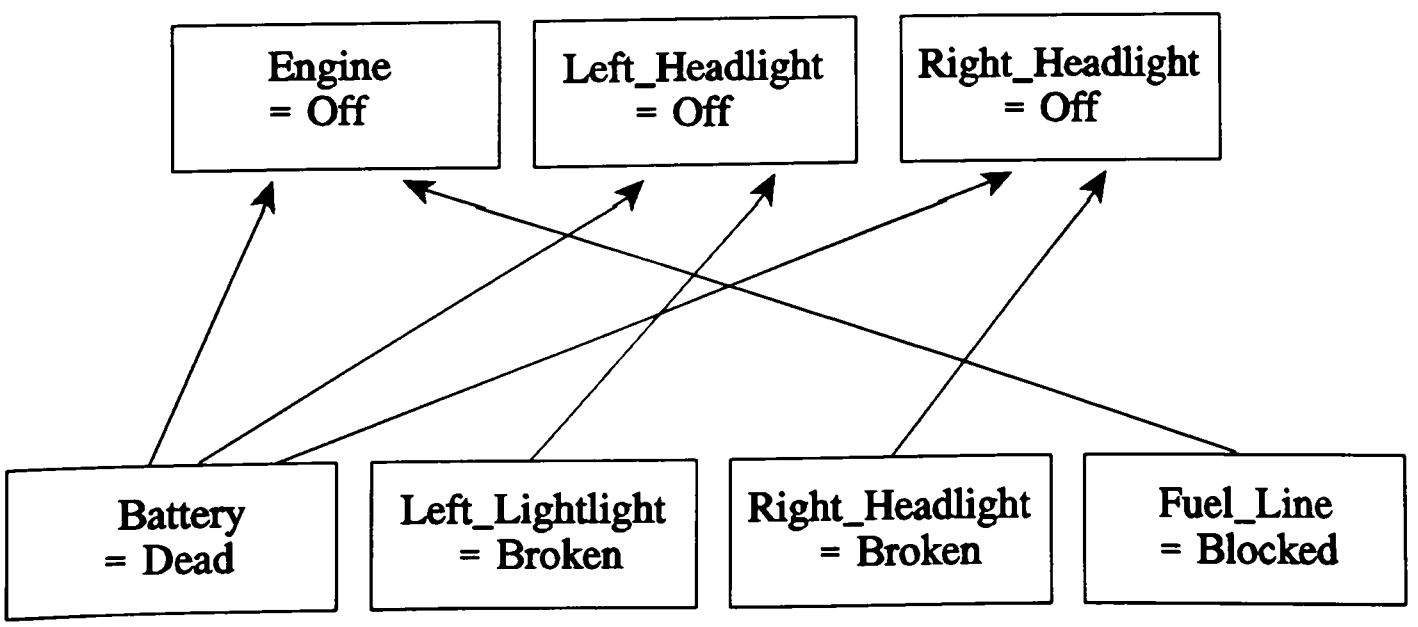


Figure 4.8. A Structural Model Representing Events and Their Relationships

study the sign of the causal impacts; i.e., categorizing causal relationships as either positive or negative relationships. If such is the case, a structural model shown in Figure 4.7 will suffice. To be noted is that Figure 2.2 corresponds to two-way hypothesis testing, while Figure 4.7 corresponds to one-way hypothesis testing.

Structural modeling construct, however, has several inherent limitations. For example, Figure 4.7 cannot explain how net sales revenue might change, when both marketing costs and sales volume increase. In addition, there are relationships that cannot be easily categorized as either positive or negative. For instance, it is not clear whether price has a positive or a negative direct effect on net sales revenue. Obviously, it depends on other variables. However, structural models are typically drawn based on pairwise relationships (Ford and Hegarty, 1984). That is, if all other variables are held constant, then price should have a positive direct impact on net sales revenue. Figure 4.7, however, again fails to explain what is the total impact of price on net sales revenue.

Decision makers may want to know the coefficients of causal impacts, not just the signs. Thus, statistical analysis techniques are necessary for causal modeling. Statistical causal modeling[6] provides not only a systematic way to derive the impact coefficients but also a formal way to represent the interactions among quantitative variables. In general, structural models are based on the inquirer's mental models (or theories), while statistical models are based on the empirical data as well as the mental models. Thus, the statistical causal models build upon the structural models.

---

[6] Although the term "causal modeling" is used here, like in numerous other research, the statistical methodology, by itself, is not capable of proving causal relationships. Cause-effect relationships are derived from theory, and theory comes from outside of statistics. A statistical causal model provides a quantitative interpretation of the qualitative theory (Johnson and Wichern, 1982).

Statistical models capture the relationships among quantitative variables. Although statistical models can handle one or two qualitative variables by using indexes, they have limitations to handle the relationships among qualitative variables. That is, unless the domains of a variable is at least ordinal, it is difficult to describe the relationships in an orderly fashion with a single term, i.e., coefficient. Such relationships must be labeled instead.

Associated with a qualitative variable is a set of labels or events. Figure 4.8 showed a structural model representing several events and their relationships. A typical way to extend this type of structural models is to denote the nodes and arcs with probabilities or confidence factors. Another limitation of Figure 4.8 is that it cannot tell whether events are related in conjunction or in separation. A modeling construct that overcomes the limitations is rule-based modeling.

In summary, the structural, statistical, and rule-based modeling constructs constitute a necessary set of grammatical terms to express the causal relationships. Rule-based models are useful to represent specific or complex relationships among objects, while structural models and statistical causal models are useful to represent simpler or more general relationships. For example, consider the relationships between price and sales volume in Figure 4.7. A structural model may be used to state that there is a causal relationship between them. A statistical causal model may be used to describe the relationship with an equation. When the relationship is too complex to be modeled with an equation (e.g., price differentiation), then rule-based models must be used.

One point to be made is that some philosophers define causality as a concept incompatible with statistical determinism. On the other hand, the quantum theory states that the world is essentially stochastic. In general, causal relationships are rather probable than provable. They are probable in that the

causal relationships cannot be observed in perfect isolation with their surrounding conditions, and that, consequently, people must make inference beyond the information given (Einhorn and Hogarth, 1986). In this research, causal relationships among variables are considered to be logical connections and thus deterministic, while the relationships among events are probabilistic.

4.5. Problem Processing Requirements of PDSS

A DSS should support all phases of the decision-making process being considered (Sprague, 1980; Weber, 1986; Elam and Konsynski, 1987; Nunamaker et al., 1988; Adams et al., 1990). The determination of these composite decision phases or activities is not a simple task, especially for non-specific DSS. Because a non-specific DSS intends to support multiple problems, these phases should be identified from decision models that are general enough to be applicable to almost any problem situation: e.g., Simon's (1960) and Mintzberg's (1973) decision models. Sprague made a note on adopting a particular decision process model.

> Simon's model, although widely accepted, is only one model of how decisions are actually made. In fact, there is no universally accepted model of the decision-making process, and there is no promise of such a general theory in the foreseeable future. (Sprague, 1980, p. 13)

In scientific research communities, a universal generalization is a fantasy rather than a reality (Mitroff and Kilmann, 1978; Bunge, 1979). Then, how can we identify the decision phases in the absence of the theory of decision making? One practical solution is to synthesize all relevant decision process models, e.g., the model of Lang et al. (1978), which is an abstraction of fifteen other decision process models. This synthetic approach expands conceptual constraints and reduces the chances of committing the error of the third kind.

Taking this approach, Figure 4.9 presents the problem diagnosis routines and activities abstracted from problem diagnosis literature. The problem diagnosis process requires at least three conceptually distinctive routines: (1) problem detection routine; (2) causal model development routine; and (3) causal model application routine (Bouwman, 1983; Cowan, 1986). As noted by problem solving process models (Mintzberg et al., 1976; Lang et al., 1978; Bravoco and Yadav, 1985), these routines do not necessarily occur in a sequential manner. For example, decision makers may skip or reiterate certain routines or activities.

Problem detection is a routine that identifies symptoms indicative for the current state of the problem situation. Detected symptoms are interpreted according to the mental model, which is a subset of decision maker's cognitive schema, world views, logico-mathematical structures, Weltanschauung, or belief systems. If the mental model does not satisfactorily explain the symptoms, the decision maker will either develop a new model or refine the model. Causal model development is a routine that identifies key variables and hypothesizes/ tests cause-effect relationships among the variables. Causal model development becomes a critical routine especially for semi-structured or unstructured problems. Causal model application is a routine that formulates alternative diagnoses based on the detected symptoms and causal models, evaluates the alternatives, and produces diagnostic conclusions and justifications.

To be noted is that the model development routine and the model application routine both require hypotheses formulation and testing activities. In fact, alternative diagnoses generation is called "problem-hypotheses" formulation (Bouwman, 1983, p. 664). The difference is that the model development is concerned with finding the cause-effect relationships in general, while model application is concerned with finding the cause-effect relationships for the given

Problem Detection	Perceive stimuli from a problem situation and identify the parts of situation that require cognitive processing
Sensing	Monitor/scan a problem situation and notice stimuli
Filtering	Select significant stimuli and discard insignificant and/or irrelevant stimuli
Labeling	Assign symbols to the selected stimuli

Model Development	Develop and refine an understanding of the causal structure of a problem situation
Causal Variable Identification	Identify variables that are important for understanding the problem situation
Model Construction	Hypothesize relationships among variables and build causal models
Model Validation.	Examine the hypotheses and evaluate the validity of the causal models

Model Application	Produce a diagnosis and explain the causal structure of the problem situation
Alternative Diagnosis Generation	Create alternative diagnoses based on the causal models
Alternative Diagnosis Evaluation	Evaluate the alternatives as to their strengths in explaining the problem situation
Final Diagnosis Selection	Select a diagnosis and provide justifications for the selected diagnosis

Figure 4.9. Problem Diagnosis Routines and Activities

problem instance. That is, model development requires inductive and deductive thinking, while model application requires deductive and abductive thinking. The following paragraphs examine the problem diagnosis activities in Figure 4.9.

1. Sensing is an activity which involves scanning or monitoring a problem situation and noticing stimuli (Hasher and Zacks, 1979; Billings et al., 1980; Lyles and Mitroff, 1980; March and Feldman, 1981; Schwenk and Thomas, 1983). Monitoring and scanning are two different sensing modes. Decision makers, with limited time and cognitive resources, are likely to select certain key performance variables and monitor those variables, if possible. Monitoring partly embeds filtering activity and tends to result in divergent, inside-out, or top-down problem diagnosis process. On the other hand, decision makers, when faced with an unfamiliar or critical problem situation, may want to scan and examine all aspects of the problem situation before making any judgment. Scanning usually results in convergent, outside-in, or bottom-up problem diagnosis.

2. Filtering is an activity to select significant and relevant stimuli (March and Simon, 1958; Ackoff, 1967; Mason and Mitroff, 1980; Hogarth and Makridakis, 1981; Kiesler and Sproull, 1982; Dutton et al., 1983; Volkema, 1983). Filtering depends upon individuals' aspiration levels. Managers' aspiration levels are based upon a variety of models and change continuously. Decision environment factors also affect the amount of information being filtered out. For instance, high time pressure and heavy work load tend to reduce the amount of information that individuals process.

3. Labeling is an activity to assign symbols to the selected stimuli (Taylor, 1975; Sage, 1981; Bouwman, 1983; Nadler, 1983; Lyles and Thomas, 1988). The well-known research finding of Miller (1956) indicates that humans can handle only about five to nine "chunks" of information at any given time. Decision

makers, in order to overcome the cognitive limitation, tend to use qualitative labels. If qualitative labeling is, in fact, a ubiquitous activity, rule-based causal modeling is an indispensable ingredient for problem diagnosis support.

4. Causal variable identification is an activity involving identifying problem-pertinent variables and specifying whether causal relationships between variables exist (Anderson and Janson, 1979; Schwenk, 1984). It includes the simplest type of causal hypothesis formulation, i.e., H_o: there is no causal relationship between two variables, and H_1: there is a causal relationship. This rather arbitrary boundary between the causal variable identification and hypothesis formulation is established, because identification of problem-pertinent variables is difficult, if not impossible, without relying on causal thinking.

5. Hypothesis formulation and model construction is an activity to hypothesize the nature of causal relationships among variables (Dutton et al., 1983; Taylor and Crocker, 1983; Tversky and Kahneman, 1974). This is an effort to describe the causal relationships or measure the causal impacts. The simplest way of describing the causal relationship is to classify the relationships as either positive or negative. Managerial problem diagnosis, however, often requires more accurate description of the causal relationship. For example, managers are likely to ask the question, "how many more products will be sold, if we decrease the price by one dollar," given that "the price negatively affects sales volume."

Decision makers also tend to develop more complex hypotheses, as they develop a deeper understanding of the problem. That is, the causal associations among variables become more complex, developing into an intricate causal network model. For example, a more complex hypothesis might be "if the competitors' prices and the market demand do not change, then an increase in sales price will negatively affect sales volume."

6. Hypothesis testing and model validation is an activity involving the evaluation of the validity of the hypothesized relationships and causal models (Crocker, 1981; Schwenk and Thomas, 1983; Ramakrishna and Brightman, 1986; Schwenk, 1986). It is difficult to define a method that could precisely measure the validity of a causal hypothesis, given that the method should work on non-experimental data. As mentioned previously, causation is a concept consistent with covariation. Covariation means that a change of X will result in a change of Y, if all other conditions are held constant. That is, covariation means not only a concomitant covariance but also no spurious correlation. Unfortunately, it is extremely difficult, if not impossible, to examine the non-spuriousness from experiential data. Obviously, it is the experimental design, not a data analysis technique, that eliminates or randomizes the compounding variables. Faced with this difficulty, decision makers should use multiple criteria for causal model evaluation, e.g., face-validity, consensus, consistency, coherence, and cross-validity.

7. Alternative diagnoses generation is an activity to identify a set of causal variables that explain the observed problem symptoms (MacCrimmon and Taylor, 1976; Kepner and Tregoe, 1981; Dutton et al., 1983; Cowan, 1986). This activity requires an abductive or deductive reasoning method, depending on the way the causal models are represented.[7] Very often, this activity also means the determination of categories into which a problem situation can be classified, i.e., a classification problem. For example, a physician's alternative diagnoses could be a simple cold, bronchitis, pneumonia, etc.

---

[7] For example, MYCIN (Shortliffe, 1976) represents cause-effect relationships in the form "if symptoms then causes." In this case, cause identification requires deductive reasoning or forward chaining. In cases where the relationships are set up in the form "if causes then effects," cause identification requires abductive reasoning or backward chaining.

8. Alternative diagnoses evaluation is an activity involving the examination of the candidate diagnoses in terms of their strengths in providing explanations for the problem (Einhorn and Hogarth, 1982, 1986; Brown, 1985). Diagnoses evaluation has two goals that often conflict each other: (1) explaining all the problem symptoms sufficiently; and (2) minimizing the complexity of explanation.

9. Diagnosis selection is an activity involving reaching a diagnostic conclusion and justifying the conclusion. Although this activity is logically considered to be the last step of the problem diagnosis process, it is possible for a decision maker to initiate a new cycle of problem diagnosis process, with any activity as a starting point, when the conclusion is not satisfactory.

These activities are mainly related to the "problem identification" phase (Mintzberg et al., 1976). It should be noted that the alternative diagnosis generation and evaluation activities discussed above may differ from the alternative solution design and evaluation phase mentioned at the very beginning of Chapter I. For instance, a doctor may start to design and evaluate alternative medications (solutions) after reaching a final diagnosis. For a diagnostic problem, the above-discussed problem diagnosis activities may fairly well cover the entire decision process, however.

This section has identified activities essential for successful problem diagnosis and inherent in most problem diagnosis processes. The system should support these activities to facilitate the user's problem diagnosis process.

4.6. Interaction Requirements of PDSS

A DSS can be viewed as a system embedded in a human decision making system. A synergetic problem solving requires interactions between these two systems (Woods, 1986; Hale and Kasper, 1989). From the user's viewpoint, an

interface of the DSS that defines the interactions "is the system" (Dos Santos and Holsapple, 1989, p. 1). That is, the cooperation between the users and system can be discussed in terms of the knowledge exchanged between the systems.

A PDSS requires at least two generic interaction functions for information exchange: TELL and ASK. Originally defined by Levesque (1984), TELL allows a user to provide information to the system; and ASK allows the user to seek information from the system.

TELL: S-Knowledge <-- S-Knowledge x U-Assertion; ..... (4.6)

ASK: S-View <-- S-Knowledge x U-Query ..... (4.7).

the symbol "<--" means that the elements on the right-hand-side are combined to produce the element on the left-hand-side. TELL states that the system updates S-Knowledge (system knowledge) based on U-Assertion (user assertion). S-Knowledge on the right-hand-side is the system's a priori knowledge; the one on the left-hand-side is the system's a posteriori knowledge (which becomes a priori knowledge on the next call).

U-Assertion can be examined from two different aspects: in terms of its effect on S-Knowledge command (U-Action); and its constituent objects and relationships (U-View). S-Knowledge, according to Function (4.1), also consists of two elements: knowledge base (KB) and problem processing (PP). Focusing on the system's a priori knowledge, TELL can be restated as,

TELL: S-Knowledge <-- KB x PP(TELL) x U-Action x U-View.

PP(TELL) should satisfy the desired U-Action; and KB should organize U-View.

Two simplest U-Actions are add and delete. Add allows users to put new information into KB; delete allows users to remove unwarranted views from KB. Modify is a combination of delete and add. Thus, TELL can be simplified as:

TELL: S-Knowledge <-- KB x {add, delete} x U-View ..... (4.8).

The operation set, TELL, simply indicates that the system should accept and store statements from the users. Of note is that the user may want to modify the system's problem processing behavior. To support such operations, KB must store the procedural knowledge, i.e., problem processing functions, declaratively. User-programming, however, is not considered here.

The second interaction type, ASK, states that the system accepts U-Query and produces S-View (system's view or answer) based on S-Knowledge. In the simplest form, ASK will retrieve a view or elements of a view from the KB. A system that claims to be more than a simple memory aid, however, must perform more complex problem processing. The system should create and provide new information to the users. That is, a function category, inference, should be established to include inductive, deductive, and abductive processing. Inference is perhaps the most important function category, because it generates new knowledge; i.e., makes implicit information in the KB more explicit. The results of system's inference may be stored in the KB and/or directly presented to the user. ASK can be restated as:

ASK: S-View <-- KB x {inference, retrieve} x ?S-View ..... (4.9).

The utility of TELL and ASK depends much on the knowledge held by the problem solving agents. For instance, ASK is of no practical use, when KB is empty. On the other hand, TELL has no value, if KB is complete and truthful. In general, S-Knowledge is neither empty nor perfect. It is also safe to assume that the same is true for the user's knowledge about the problem situation.

The system-driven PDSS (e.g., Ata Mohamed, 1985; Billman, 1989; Jung, 1990) tend to focus on the inference category. Because these systems support limited TELL operations (typically designed for data input), the validity of the results they generate are highly dependent upon the correctness of the models

stored in KB and/or the problem processing functions. On the other hand, the user-driven PDSS (e.g., Pracht, 1984; Ramaprasad and Poon, 1985; Khazanchi, 1991) tend to fully support TELL operations but are weak in the inference category. Therefore, their abilities to augment the user's decision process are relatively limited.

A cooperative PDSS must allow users to fully utilize both TELL and ASK. It should be noted that the redundant design concept (Woods, 1986) indicates that a better view may be derived by overlapping U-View in Function (4.8) and S-View in Function (4.9). In general, decision analysts should be responsible for preparing various views for the decision maker's examination and comparison (In this research, the system's generic capability to induce decision rules from the example data is perhaps the most significant redundancy incorporated).

Finally, a cooperative inquiry PDSS must reduce, or at least attempt to reduce, the errors and biases of the users in conceptualizing a problem. PDSS may utilize the following inquiry "guaranteeing" concepts. First, PDSS may employ multiple induction algorithms to achieve a Lockean consensus. Second, PDSS may cross-validate models to maintain a Leibnitzian consistency. Third, PDSS may need to manage multiple models for Kantian, Hegelian, and Singerian inquiries. Finally, PDSS may need to allow users to partition a view into several parts to achieve a refined Singerian agreement.

4.7. Summary and Conclusions

This chapter has developed a conceptual framework for managerial PDSS development. The framework consists of the following elements: (1) identification of contingency factors for problem diagnosis effectiveness; (2) identification of essential elements of PDSS; (3) development of a cooperative inquiry system

concept; (4) delineation of logical steps in PDSS development/use processes; (5) discussions on individuals' roles in those processes; and (6) formulation of PDSS requirements. The following paragraphs summarize the conceptual framework.

1. The effectiveness of problem diagnosis depends on: (1) the problem situation; (2) the decision environment; (3) the decision maker's mental models and diagnosis process; and (4) the PDSS. This conceptualization is consistent with various MIS frameworks. The decision process and mental models are, in turn, affected by the problem situation, the decision environment, and the PDSS.

2. A PDSS has three essential elements: (1) a knowledge base; (2) problem processing; and (3) user-system interaction. This conceptualization is consistent with two widely-used DSS models (Bonczek et al., 1980; Sprague, 1980).

3. By combining the contingency factors (MIS research frameworks) and the elements of PDSS (DSS models), it can be shown that the problem diagnosis effectiveness is determined by the degree of cooperation achieved by the decision maker and the PDSS for the given problem situation and decision environment.

4. A cooperative system integrates the knowledge and capabilities of the human and the PDSS (Hale and Kasper, 1989). A cooperative system must have both cognitive and normative elements (Keen and Scott Morton, 1978). From a cognitive perspective, PDSS must easily accommodate users' mental models and facilitate the users' natural problem diagnosis processes. From a normative perspective, PDSS must extend the users' mental models and alter the users' suboptimal problem diagnosis process.

5. A PDSS, as it intends to generate knowledge about problem situation, is an inquiring system. To generate "truthful" knowledge, PDSS must utilize the inquiry "guaranteeing" concepts (Churchman, 1971).

6. Non-specific PDSS are more significant than specific PDSS from a research viewpoint as well as from a pragmatic viewpoint. Abstraction and assumption-making are necessary ingredients for the design of a non-specific PDSS. Because of this very reason, the non-specific PDSS may not be able to reflect the unique aspects of a problem situation as closely as possible.

7. If a non-specific PDSS development approach is taken, there are three logical steps in PDSS development and use: (1) non-specific PDSS development process; (2) specific PDSS development process; and (3) specific PDSS use process. They are carried out by DSS designers, decision analysts, and decision makers, respectively. The basic capabilities of non-specific PDSS are used throughout the specific PDSS development and use processes. Decision analysts must adjust the PDSS according to the situational factors.

8. The requirements of non-specific PDSS can be examined from three interrelated perspectives. They are knowledge base structure requirements, problem processing requirements, and user-system interaction requirements.

9. The previous PDSS have relied on a simple modeling construct that has inherent limitations. PDSS must utilize, at least, the following causal modeling constructs: (1) structural models delineating the relationships between variables; (2) statistical models describing the relationships among the quantitative variables; and (3) rule-based models describing the relationships among the events.

10. A logical way to derive the processing requirements of PDSS is to study the basic problem diagnosis activities. A process-oriented PDSS should support the following problem diagnosis routines and activities: (1) problem detection routine (sensing, filtering, and labeling); (2) model development routine (variable identification, model formulation, and model testing); (3) model application routine (alternative diagnoses generation, evaluation, and selection).

11. Decision makers and PDSS are two inquiring systems with different capabilities. PDSS must interact with the users throughout the problem diagnosis process. Two basic types of user-system interactions are TELL and ASK. PDSS should: (1) allow the users to add, delete, and retrieve views; and (2) provide reasoning support. To "guarantee" the quality of a cooperative problem diagnosis process, PDSS may employ multiple induction algorithms, check model consistencies, manage multiple models, and refine models.

The conceptual framework provides a theoretical basis to organize the fragmented previous PDSS research, identifies key design issues for the current research, and furnishes valuable guidelines for future PDSS research. In the following, the contributions of the conceptual framework are examined.

1. The framework is significant in terms of its paradigm orientation. The cooperative inquiry system paradigm combines a cooperative system concept with the inquiry "guaranteeing" concepts. Traditionally, PDSS research has emphasized either normative or cognitive support. For instance, PRADS performs system-driven normative problem diagnosis. For PRADS, "the quality of the diagnosis generated is only as good as the model used" (Ata Mohamed, 1985, p. 111). On the other hand, CLSS supports user-driven cognitive problem structuring. With CLSS, a decision maker will look at the problem "through his or her tinted glass; i.e., some type of an 'innate model' or a cognitive lens" (Khazanchi, 1991, p. 58).

This research points out the risks involved in either approach and suggests the cooperative system concept as a basis of PDSS design. Moreover, this research suggests that the cooperative system must utilize the inquiry "guaranteeing" concepts in order to generate "truthful" knowledge. These points are extremely important for designing systems that support semi-structured or

unstructured problems. The framework contributes to PDSS research by establishing a sound theoretical foundation for PDSS design.

2. The conceptual framework is the first one established for PDSS design research. The conceptual framework identifies essential elements of PDSS, delineates logical steps in developing and using a PDSS, specifies the roles played by individuals, discusses major issues to be addressed for PDSS development and use, establishes requirements for non-specific PDSS. By organizing these important issues in PDSS development, the conceptual framework facilitates a systematic development of PDSS research. For instance, the framework can show that CLSS (Khazanchi, 1991) is almost identical with MIND (Ramaprasad and Poon, 1985) in terms of all major PDSS elements. The main difference is in their display formats (graphics versus tabular). Such a comparison based on the analytical view of PDSS enables future research to avoid duplications and build upon extant research. Future research may find that the framework must be extended to include, for example, the display format variable. Without an initial framework, a systematic refinement of PDSS research is difficult.

3. The framework provides a broad perspective for PDSS designers by synthesizing a variety of sound theories and design concepts such as MIS frameworks, DSS models, a cooperative system concept, various causal modeling approaches, a process-oriented decision support approach, and the inquiry "guaranteeing" concepts. That is, the framework provides a refined Singerian view of PDSS design research.

4. The framework provides a conceptual basis for designing PDSS that overcome the limitations of the previous PDSS. Future research may further extend the modeling construct set, the diagnosis activities supported, and/or the human-computer cooperation into more sufficient ones.

CHAPTER V

CONCEPTUAL DESIGN OF PROBLEM DIAGNOSIS SUPPORT SYSTEM

The previous chapter postulated that non-specific PDSS can provide an effective environment for developing specific PDSS. A non-specific PDSS provides such an environment based on its capabilities to construct and manage various causal models, carry out and support various problem diagnosis activities, and dynamically interact with users. It is the responsibility of the system designers to develop a non-specific PDSS with such capabilities.

The purpose of this chapter is to develop the design specifications of the non-specific PDSS on the conceptual level. The first section identifies various types of views that constitute the knowledge base. The second section develops support functions for the problem diagnosis activities identified in the previous chapter. The third section discusses how the inquiry "guaranteeing" concepts may be incorporated into PDSS design. The last section summarizes this chapter.

5.1. Essential Views for Managerial Problem Diagnosis

The purpose of this section is to further detail the structure of knowledge base by identifying critical views for managerial problem diagnosis. The problem processing functions, to be discussed in the next section, operate on these views. This section presents nine view types: (1) an example data view; (2) problem state views; (3) goal state views; (4) performance discrepancy views; (5) structural model views; (6) statistical model views; (7) rule views; (8) alternative diagnosis views; and (9) a final diagnosis view. Many of these views are necessary to meet the problem processing requirements of the system. The focus of this section is to define the roles of the views and the dependencies among the views.

1. An example data view is a data table containing a set of examples or training cases. As stated in Section 4.3, this research assumes semi-structured or unstructured problems that are repetitive. An example data view stores the previous problem instances. The example data view helps the users to understand the general relationships among the variables pertinent to the problem. It also provides a basis for the system to conduct inductive learning. Because the preparation of the example data view requires much time and effort, the decision analysts should establish the view.

2. A problem state view refers to a data table that contains the information about the problem instance being evaluated. Assuming problem repetitiveness, a problem state view may contain information about the current state, at time t, as well as the information about its historical states, at time t - 1, ..., t - k. Therefore, a problem state view may consist of two parts: a current problem state view and a historical problem state view. In general, the users (i.e., decision analysts and decision makers) create the problem state views. It is assumed that the current problem state view has at least one unknown attribute value. One important reason for developing the causal models is to estimate these unknown values. Typically, one of these attributes tersely summarizes the current problem state; e.g., the conclusion being that the financial performance is "excellent." Such variables are called the "target" variables hereafter.

3. A goal state view bears the desired state of the problem instance at time t. Descriptive literature states that managers establish their goals based on planning, historical projection, and comparison (Pounds, 1969; Billings et al., 1980). Normative literature suggests managers to explicitly state their goals (Lyles, 1982). Typically, users establish the goal state views based on their expectations, i.e., aspiration level. If appropriate, the users may ask the system to

project the expected performances based on the historical problem state view. The users may interactively search through the example data view to locate the records containing comparable goal values.

Semi-structured or unstructured problems tend to have uncertain problem states and an ambiguous goal state. Consequently, PDSS must allow the users to develop and examine multiple problem state views and multiple goal state views to overcome the uncertainties and ambiguities.

4. A performance discrepancy view reports the gaps between a current problem state and a goal state. Because the term "problem" indicates the existence of differences between the way things are and the way one wants them to be, a performance discrepancy view, highlighting the gaps, helps the users to locate the problematic aspects. Because this view is strongly dependent upon the given problem state and goal state views, the users are not permitted to directly modify the view. The users may modify the performance discrepancy views through modifying the problem state and/or the goal state, however.

5. A structural model view, the cornerstone of the model base, shows the causal connections between pairs of the attributes and their causal impact signs. Structural models are also called digraphs (McLean and Shepherd, 1976), cognitive maps (Axelrod, 1976), cognitive lens (Khazanchi, 1991), or influence diagrams (Ramaprasad and Poon, 1985). Although the system attempts to help the user's structural modeling with the statistics measuring the dependencies between the attributes in the example data view, the users should exercise a great deal of judgment in establishing the cause-effect relationships.

6. A statistical model view shows the statistical relationship among the quantitative variables. Statistical models build upon the structural models. If a structural model is given, the system can identify all candidate dependent

variables and their explanatory variables. Under an interactive modeling mode, the system allows the users to include all the variables on the causal path leading to the dependent variables as explanatory. The users are responsible for specifying the statistical model structure; and the system fits the equation.

7. A rule view shows the relationships among the events associated with the numeric and categorical variables. The system is capable of deriving decision rules based on a set of given variables and an example data set. Alternatively, the users may edit the rules based on their subjective understanding of the problem structure. Statistical models and rules are "descriptive" models.

8. An alternative diagnosis view shows: (1) the estimated values of the target variables; and (2) the causes for discrepancies. PDSS must manage alternative diagnosis views, because the users may examine multiple discrepancy views and multiple causal models. Alternative diagnosis views support what-if analysis, what perhaps is the most critical DSS feature. To derive an alternative diagnosis view, users must specify: (1) a descriptive model view; and (2) a problem state view and/or a goal state view.

9. A final diagnosis view summarizes alternative diagnosis views and, thus, helps users to reach a final conclusion. This view has not been implemented in the prototype system. The system, however, helps users to compare and contrast alternative diagnoses by opening multiple windows.

In addition, PDSS may require a data dictionary maintain information about data fields, dependencies between the field, and conversion rules associated with the fields. Figure 5.1. summarizes the views and view dependencies. A dark rectangle indicates that the view is strongly dependent upon other views. A hidden black rectangle indicates that there can be alternative views. The dotted rectangle indicates that the view has not been implemented.

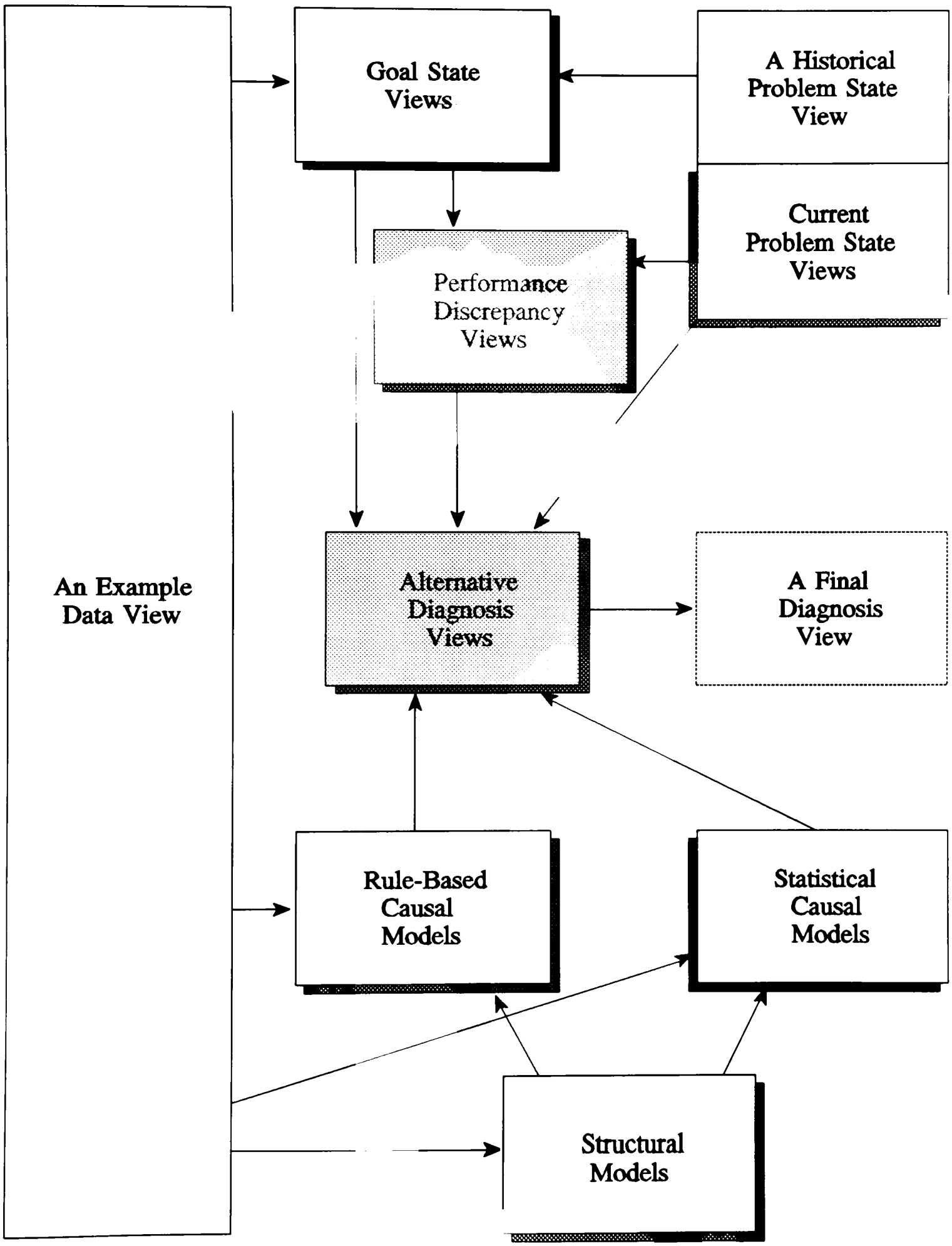


Figure 5.1. View Dependencies

The problem detection routine is supported by the problem state views, goal state views, and performance discrepancy views. During the model development routine, users formulate and validate the structural models, statistical models, and rules. A diagnosis view is generated by applying the models to the problem detection-related views. The adequacy of a diagnostic conclusion, thus, depends not only on the validity of the models but also on the appropriateness of the problem and goal state views.

5.2. Problem Diagnosis Support Functions

The purpose of this section is to develop a set of support functions for the problem diagnosis activities discussed in Section 4.5. The functions discussed in this section are available to the decision analysts and decision makers.

5.2.1. Problem Sensing Support

Problem sensing is an activity to examine the problem situation in terms of the available data. Problem sensing requires the identification of the problem state as well as the conception of a goal state. Thus, the system allows the users to create, retrieve, modify, save, and delete the problem state views and the goal state views. Because individuals establish the goal state views based on their own judgment, historical projection, and competitors' performances (Pound, 1969), PDSS can offer the following three goal formulation support functions.

Define goal allows the users to edit a goal state view based on their judgments. The system provides a template for the users to enter, for each attribute, a goal value, upper and lower goal boundaries, and goal directions associated with the boundaries. The users may copy the goal values from a problem state view and then modify them. The goal directions indicate whether a performance above or below the boundaries is either desirable or undesirable.

Project goal estimates the expected performance for the period *t* based on the historical problem state view. This function can utilize a variety of forecasting methods, such as a constant growth model, an exponential smoothing technique, or a more complex time series analysis model. The prototype supports the first two methods. Before using a historical goal projection, the users must examine whether the historical performances themselves were satisfactory and/or whether the trends can be extrapolated.

Search goal helps the users to identify appropriate performance standards from the example data view. This function identifies the records satisfying the given search conditions from the examples. Then it asks the users to provide a state view against which the examples are to be compared. Search goal, then, sorts the retrieved examples according to their statistical distances from the standard state view in terms of the specified attributes. In short, this function combines a data base operation, select, with a statistical k-th neighbor searching. This function is useful for establishing goals based on "competitors' performances."

5.2.2. Filtering Support

Filtering is an activity to select significant and relevant variables. In a sense, filtering is to dispose of irrelevant or insignificant information. Filtering reduces information loads for the users and problem spaces for the system. The system allows the users to filter out those variables not included in a structural model as irrelevant. After all, only those variables included in the causal models are relevant for drawing diagnostic conclusions, given that the user does not engage in any further model refinement with the variables filtered out.

However, it is more appropriate to consider filtering as an activity to assign priorities to the variables being examined. It is possible for the decision maker to

find a variable, once considered to be irrelevant, to be relevant upon further reflection. A PDSS, thus, should classify the variables in the data dictionary into operative or inoperative variables. Operative variables are the ones currently being considered; and inoperative variables are the ones that can be used to extend the causal models. The functions, filter out and filter in, enable the users to easily reduce and expand the problem space.

The system also helps the users to find the symptomatic variables. The significance of a variable is determined based on whether the current problem state value falls outside the goal boundaries. There are, thus, variables with and without significant deviations. Furthermore, depending on the goal directions, there are favorable deviations and unfavorable deviations. For instance, assume that a company's profit goal is $100,000 and that a manager expects the profit to be between $90,000 and $110,000. Then, an actual profit of $80,000 shows an unfavorable deviation. However, a profit of $120,000 may not be a problem, although the deviation is significant. The performance discrepancy view creation function thus classifies the variables into three groups: favorable, unfavorable, and insignificant manifestations. The performance discrepancy view provides a cognitive simplification to the users by highlighting the variables that require an immediate attention.

5.2.3. Labeling Support

Labeling is an activity to associate a symbol with a set of stimuli or other symbols. Thus, labeling can mean all kinds of symbol association. In fact, the above-discussed performance discrepancy view creation function supports labeling in the sense that it assigns the labels, {favorable, unfavorable, insignificant}, to the variables. Labeling, here, refers to the data conversion routines.

Discretization is a labeling method that assigns a set of symbols over a numeric domain. The function, discretize, allows the users to convert a numeric variable into a categorical variable. In general, the intervals covered by the symbol set are mutually exclusive. Obviously, there is no single best way to discretize a domain. In many knowledge-based systems, continuous variables are often "binarized"; i.e., any value that a variable takes on falls into either the subinterval (1) $X \leq V$ or (2) $X > V$. If a PDSS performs discretization interactively, the users may label a domain in many different ways, e.g., {high, average, low}, {very high, high, average, low, very low}, etc.

If discretization is useful for rule development, quantification is useful for statistical model development. Quantify allows the users to replace a set of labels with numeric values. In addition, PDSS may require a function that associates a set of labels with another set of labels. Associate enables the system to handle such common relationships as is-a, has, and part-of. PDSS also needs to allow the users to define new variables in relation with other numeric variables. For instance, the users may want to introduce various financial ratios while evaluating financial statements. The system may rank a numeric data field for nonparametric statistical analysis.

Figure 5.2 shows the five types of data conversion discussed above. It is generally believed that data transformation to one form or another tends to result in information loss. However, this assertion is valid only if all subsequent analyses use the transformed data. The system, thus, should not replace the original data set with the new data set, but utilize both sets of data. That is, labeling provides just another way of looking at the same data. The labeling functions introduce new data fields to the data dictionary.

Name	Revenue-1
ABC	3
DEF	4
GHI	2
JKL	6
MNO	5
PQR	1

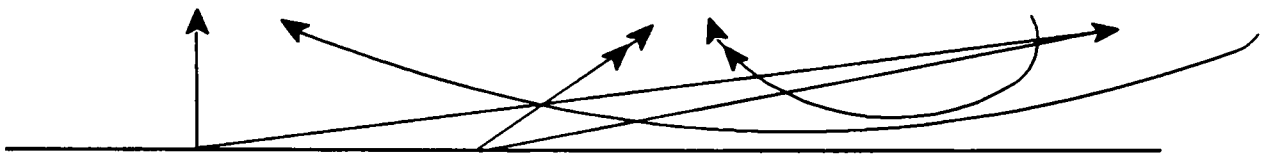
Rank Data Field

Name	Expenses-1
ABC	Average
DEF	Average
GHI	High
JKL	Low
MNO	Low
PQR	Very High

Discretized Field

Name	Profit
ABC	$5,000
DEF	$4,000
GHI	$3,000
JKL	$2,000
MNO	$4,000
PQR	$9,000

User-Defined Field



Name	Revenue	Expenses	Products	Competitiveness
ABC	$25,000	$20,000	Sausage	High
DEF	$22,000	$18,000	Orange	Average
GHI	$41,000	$38,000	Apple	Average
JKL	$12,000	$10,000	Onion	Low
MNO	$15,000	$11,000	Potato	High
PQR	$80,000	$89,000	Ham	Very high

Original Data Fields

Name	Products-1
ABC	Meat
DEF	Fruit
GHI	Fruit
JKL	Vegetable
MNO	Vegetable
PQR	Meat

Associated Field

Name	Competitiveness-1
ABC	70
DEF	50
GHI	50
JKL	30
MNO	70
PQR	90

Quantified Field

Figure 5.2. Five Types of Data Conversion

5.2.4. Structural Modeling Support

From a modeling perspective, causal variable identification corresponds to structural model development. For a user-driven structural modeling, the system allows the users to: (1) add and remove variables; and (2) modify the values indicating the relationships between the variables.

One desirable feature of PDSS is to automatically discover causal relationships between variables from the example data view (Billman, 1989). However, induction of the causal relationships from the experiential data alone is very difficult, to say the least. Therefore, this research limits the system's role in structural model development to detecting strong dependencies between the variables and presenting the information to the users for examination. Three traditional statistical analysis techniques can be used for this purpose.

The function correlation measures the dependency between two numeric variables. If the example data view contains observations at different points in time, t, t - 1, ..., t - k, the system also obtains the correlation between the increments; i.e., examines whether changes in a variable are correlated with changes in another variable over time. To measure the dependency between a categorical variable and a numeric variable, the system conducts a one-way factor analysis. The system performs the contingency table test to examine the dependency between two categorical variables. Again, these statistics should never be used as indicators of the cause-effect relationships. They even do not postulate whether one is affecting the other, or the other way around.

Once a structural model is built, the system can analyze the structural relationships. In fact, most of previous PDSS focus on supporting structural model analysis (Pracht, 1984; Ramaprasad and Poon, 1985; Jung, 1990; Khazanchi, 1991). Two common functions are affects (finds all the variables that

a specified variable affects), and affected (finds all the variables that affect the specified variable).

This research establishes the following additional structural model analysis functions: union affect (finds a sufficient set of variables that the specified variables affect), union affected, intersection affect (finds a necessary set of variables that the specified variables affect), and intersection affected. These functions are useful to examine, for example, what variables are affected by both "A" and "B" (intersection affect) and what variables are affected by either "A" or "B" (union affect). Incidentally, these operations were discussed by Burns and Winstead (1982) but have not been utilized by the previous managerial PDSS.

It is possible for a structural model to have causal cycles. There are three major approaches to handle causal cycles. The first approach replaces the variable cluster with a high-level conceptual variable. However, it can be quite difficult for the users to come up with an appropriate high-level conceptual variable.[8] The second approach removes the cycle by detailing the model structure. The difficulty is that the additional variables may be accompanied by additional cycles. The third approach, which is adopted in this research, considers the relationships as correlational, e.g., linear structural relations models. The implication of adopting this approach is that structural analysis and cause identification become slightly more complicated. No matter how the causal cycles are handled, the system must identify cycles for the users. Another important structural analysis function is one that identifies all causal paths between nodes.

---

[8] The high-level conceptual variable is similar to the "underlying factor" in the statistical factor analysis. Factor analysis requires intensive thinking on the part of the users. Johnson and Wichern (1982) state that the quality of a factor analysis depends on a WOW criterion. The criterion means that an application is deemed successful, when the investigator can shout "Wow, I understand these factors."

5.2.5. Model Development Support

Hypothesis formulation is an activity to describe the causal relationships among variables. From a modeling perspective, this activity corresponds to statistical causal model construction and rule induction. As stated, statistical models and rule-based models are built upon the structural causal models. This approach--delineating causal relationships first using structural models and describing the nature of relationships later--appears to be a simple but useful strategy for building causal models.

Although there are numerous statistical data analysis methods that can be employed for this purpose, the prototype system supports the least-squares regression modeling. A regression model fits the relationship between a numeric dependent variable and multiple numeric explanatory variables. The explanatory variables may include polynomial and interaction terms.

PDSS should also support rule induction. User-driven rules may describe the relationships between the events related to the numeric and/or categorical variables. System-driven rules describe the relationships among the categorical variables only; i.e., numeric fields must be discretized first. For the system-driven rule discovery, the ID3 algorithm (Quinlan, 1986) is perhaps the most widely used. ID3, however, must be revised to be applicable to the managerial problem diagnosis. In addition to the revised ID3, this research proposes a new rule induction algorithm. The algorithm, being a variant of ID3, is inspired by INFERULE (Uthurusamy et al., 1991) and PRISM (Cendrowska, 1987).

5.2.6. Model Validation Support

There are two major approaches to model validation in statistics. One is to: (1) split examples into a training set and a test set; and (2) use the test set to

5.2.5. Model Development Support

Hypothesis formulation is an activity to describe the causal relationships among variables. From a modeling perspective, this activity corresponds to statistical causal model construction and rule induction. As stated, statistical models and rule-based models are built upon the structural causal models. This approach--delineating causal relationships first using structural models and describing the nature of relationships later--appears to be a simple but useful strategy for building causal models.

Although there are numerous statistical data analysis methods that can be employed for this purpose, the prototype system supports the least-squares regression modeling. A regression model fits the relationship between a numeric dependent variable and multiple numeric explanatory variables. The explanatory variables may include polynomial and interaction terms.

PDSS should also support rule induction. User-driven rules may describe the relationships between the events related to the numeric and/or categorical variables. System-driven rules describe the relationships among the categorical variables only; i.e., numeric fields must be discretized first. For the system-driven rule discovery, the ID3 algorithm (Quinlan, 1986) is perhaps the most widely used. ID3, however, must be revised to be applicable to the managerial problem diagnosis. In addition to the revised ID3, this research proposes a new rule induction algorithm. The algorithm, being a variant of ID3, is inspired by INFERULE (Uthurusamy et al., 1991) and PRISM (Cendrowska, 1987).

5.2.6. Model Validation Support

There are two major approaches to model validation in statistics. One is to: (1) split examples into a training set and a test set; and (2) use the test set to

evaluate the models derived from the training set. The other approach is to reiterate the modeling process with subsets of the examples, e.g., bootstrap method (Efron, 1979) and statistical cross-validation method of Stone (1974). These techniques provide an unbiased estimator of the model errors.

This research does not attempt to estimate the unbiased error rates. Instead, all the cases in the example view will be used for the model development, and the system reports apparent errors, i.e., the differences between the actual values in the example set and the model estimates. That is, the system reports the residual errors for numeric target variables and the frequencies of "hit" cases and "missed" cases for the categorical target variables. This approach may be more suitable for non-scientific managerial problem diagnosis.

It is possible for the users to reevaluate a statistical model by comparing the signs of the regression coefficients with those signs postulated in the structural model. In general, the comparison might be more appropriate between a path-analytic statistical model and a structural model. Nevertheless, the comparison may remind the users of the risks of multicollinearity, model under-specification, and model over-specification.

One useful feature of PDSS is to verify the rules provided by the users. The user-provided rules are subject to various errors (see Bahill, 1991). Two most serious errors are inconsistency and incompleteness. Inconsistent rules generate conflicting conclusions. Incomplete rules fail to generate a conclusion. It is possible to detect such errors by chaining the rules backward and examining the problem spaces covered by the closured rules. The difficulty, however, is not much on detecting the errors, but on designing a way by which the users and the system can interactively correct the errors.

The prototype system avoids the problems instead of tackling them. For the rule completeness, the system inserts a rule with no premise for each of the variables appearing in the conclusion part. These rules will be fired only when all other rules fail. The system handles inconsistent rules based on the accuracies of the rules. The accuracy of a rule is calculated as the number of "hit" cases divided by the number of "hit" and "missed" cases.

5.2.7. Alternative Diagnosis Generation Support

Diagnosis generation has two meanings: (1) determination of the values of the target variables; and (2) identification of the causes. The function, generate diagnosis, determines the target value by: (1) identifying the explanatory variables from the descriptive model set; (2) obtaining their values from the current problem state or goal state; and (3) applying the descriptive models. This function performs backward chaining of rules and statistical model statements.

If either a problem state view or a goal state view alone is given, the system performs simple cause searching. That is, the system traces back its model application process and explains how the diagnostic conclusion has been reached. This type of cause searching can be done in terms of the structural model (Jung, 1990), statistical model (Paradice, 1986), or rules (Shortliffe, 1976).

If a performance discrepancy view is given, the system identifies causes based on the significance of the deviations. That is, the system: (1) identifies the explanatory variables; (2) selects those variables that show significant deviations in the same direction with the given variable; and (3) continues the above steps recursively, until no additional variable can be related to the identified variables.

5.2.8. Alternative Diagnosis Evaluation

Alternative diagnoses are possible due to two reasons: (1) by the multiple problem state and goal views; and (2) by the multiple causal models. The system allows the user to compare and contrast the alternative diagnosis views. If the alternative diagnosis views show a fairly consistent result, users may reach a conclusion quickly and easily. If they show significantly different results, the users may have to reiterate the problem diagnosis process.

5.2.9. Diagnosis Selection

The final diagnosis selection is the ultimate responsibility of the users. In general, the decision maker may have to consider many other factors that have not been, or could not have been, represented in the computer systems.

5.3. Cooperative Knowledge Generation

A PDSS acquires problem-specific knowledge in two primary ways. First, it acquires knowledge through interacting with the users. Second, it generates knowledge by applying the processing functions to the user-provided knowledge. Human decision makers exhibit various cognitive biases. It is very likely that a computer system also exhibits biases. Paradice and Courtney (1986) state that "computer-based systems have no inherent cognitive limitations. Nor do they exhibit the biases which characterize human decision making" (p. 54). In fact, computer systems are relatively free of computational biases, as they are built upon logic and mathematics, which have consistency, coherence, and precision.

However, the application of a computer-based system to semi-structured/ unstructured problem diagnosis is a rather different matter. The system's processing capabilities and problem-specific knowledge can be ultimately

attributed to the DSS designers and decision analysts. Consequently, the biases of the system designers and developers may become the biases of the system.

If a PDSS is potentially biased, how can it help the users to understand the problem correctly? In PDSS design research, two propositions are prevailing. One, decision analysts ought to be the domain experts; and, thus, the system should exhibit negligible biases (e.g., Chatfield, 1990). Two, decision makers must critically examine their own opinions and the system's recommendations to make correct judgment (e.g., Paradice, 1986). That is, the decision makers should accept the system's recommendations when the system is "right" and they are "wrong"; and reject the system's recommendations when the system is "wrong" and they are "right." To a certain degree, these assumptions are rather unavoidable for semi-structured or unstructured problems. This research incorporated the following components into the non-specific PDSS to "guarantee" the quality of the problem diagnosis process.

First, this research employs multiple techniques for goal projection and rule induction. All system-driven pattern extraction, and even human inductive reasoning, are based on certain assumptions. Therefore, individual induction methods may have different levels of validity depending on the situation. When a user is in a position of not being able to determine the "right" technique, the best way to conduct the inquiry is perhaps to apply all plausible techniques and evaluate their results altogether. Then, the user may synthesize or simply average[9] the extracted patterns. Multiple induction techniques provide

---

[9] Sanders and Ritzman (1990) report that an unweighted averages of the forecasts generated by two simple forecasting techniques, (1) simple exponential smoothing techniques and (2) linear regression model, performed better than the forecasts of single more complex models, such as (1) adaptive response rate exponential smoothing, (2) Holt's two parameter linear model, and (3) automatic

information that a single technique cannot provide: e.g., do different techniques yield a consensual conclusion, how much do their conclusions vary, and how much confidence can be attached to the final conclusion? Multiple induction techniques are necessary to arrive at a Lockean consensus. They can also generate conflicting results, prompting the user to examine the validity of the assumptions underlying the techniques, i.e., Hegelian evaluation.

Second, the system checks and maintains consistencies between a structural model and statistical models. Based on a given structural model, the system can determine: (1) candidate dependent variables; and (2) candidate explanatory variables for a chosen dependent variable. Therefore, the system can prevent users from formulating statistical models or rules that contradict with the structural model. In addition, the system allows the users to examine whether the signs of the regression coefficients are consistent with the signs maintained in the structural model. Groebner and Shannon (1985) state that "even if a regression model is significant, and if each independent variable is significant, decision makers should still examine the regression coefficients for reasonableness" (p. 590). The system helps the users to examine whether the regression coefficients have unexpected signs. These features help the users to maintain Leibnitzian consistencies among the causal models.

Third, the non-specific PDSS has the capability to manage alternative structural models and alternative descriptive model sets. Kantian, Hegelian, and

---

univariate adaptive estimation technique. Although the authors did not provide the theoretical elements that contribute to the improved accuracy, intuitions tell that a combination of models may be able to cancel out the errors associated with individual models, in a similar fashion that a portfolio of stocks reduces the non-systematic risks. However, one should not expect that the average of two "wrong" models will always produce a "right" estimation.

Singerian inquiries require alternative models. Thus, the system's capability to manage alternative models provides a facility in which the decision analysts and decision makers can pursue such inquiring strategies collectively or individually. The system allows to copy an existing model set under another name, so that the users can easily develop alternative model sets which differ only in part. In general, decision makers may achieve an effective inquiry by critically examining and refining the analyst-provided models. The system allows the decision makers to develop alternative models and compare the models with the analysts' models.

Furthermore, the system allows users to represent alternative statistical model statements in a descriptive model set. Thus, users can easily examine the effect of dropping a variable from the statistical model statement, for example. Such comparisons may help the users to better understand the problem space.

As shown in Figure 5.1, the system manages not only alternative model views but also alternative data views (problem state views, goal state views, etc.). The multiple overlapping data views can possibly reduce errors, as the weakness of a particular view may be depicted by other views. For example, a decision maker may realize that a goal based on the competitor's performance may not be realistic in light of the historical projection. The system also allows the users to perform what-if analysis. What-if analysis together with multiple views provides the users with a mechanism to deal with errors and uncertainties by supporting Kantian, Hegelian, and Singerian inquiries.

Fourth, the system allows users to build descriptive models based on structural models. The structural models are simple and useful to delineate the overall structure of the problem. However, as the users develop a deeper understanding of the problem, they study more complex relationships by building more intricate causal models (see Section 4.5). This research employs statistical

and rule-based modeling to capture such complex causal relationships. In this sense, this research supports a Singerian model refinement process. An agreement made on the structural model (e.g., price affects sales volume negatively) may be converted to a disagreement (e.g., price may or may not affect sales volume negatively) and developed into more refined hypotheses (e.g., price may not affect sales volume in a seller's market; otherwise, price affects sales volume negatively).

Finally, the non-specific PDSS has a capability to handle both quantitative and qualitative (categorical) data. This capability prevents the users from applying a wrong analysis technique to the given data.[10]

5.4. Summary

This chapter performed the conceptual design of PDSS. Consistent with the framework presented in Chapter IV, the conceptual design of PDSS with respect to three aspects: knowledge structure, problem processing, and user-system interaction. The conceptual design included: (1) identification of critical view types; (2) development of problem diagnosis support functions; and (3) incorporation of the inquiry "guaranteeing" concepts into PDSS design.

---

[10] For instance, Einhorn and Hogarth (1982) used a contingency table to illustrate the point that the causation does not necessarily imply correlation (p. 31). The 2 x 2 contingency table shows the following information. X denotes intercourse. Y denotes pregnancy. P(X & Y) = 20/200; P(X & ~Y) = 80/100; P(~X & Y) = 5/200; P(~X & ~Y) = 95/200. Note that P(~X & Y) reflect the measurement error. The authors calculated a correlation coefficient and concluded that r = 0.34 is not significant. For the given problem, however, it is more appropriate to run chi-square independence test (Conover, 1980). In fact, if the authors had applied the chi-square test, they might have clearly seen that X and Y are not independent (T=10.346; p-value < 0.001).

First, concerning the knowledge structure, this chapter identified the following views:

Example Data View	Stores historical data. Helps users to learn from observing examples. Allows the system to conduct inductive learning. Prepared by decision analysts.
Problem State View(s)	Stores information about the problem instance being evaluated. Consists of a current problem state and historical problem states. Developed by the users.
Goal State View(s)	Bears the desired state of the problem instance. Goal state values are based on expectations, historical projection, and and competitors' performances. Developed interactively.
Performance Discrepancy View(s)	Shows the gaps between the current problem state and goal state views. Locates the problematic aspects of the situation. Derived by the system based on the user's instructions.
Structural Model(s)	Shows the causal connections between pairs of the variables and their causal impact signs. Developed interactively.
Statistical Model(s)	Shows the relationships among numeric variables. Developed interactively.
Rule-based Model(s)	Shows the relationships among the events associated with the variables. Developed interactively.
Alternative Diagnosis View(s)	Estimates target variables' values and identifies causes for the discrepancies. Derived by the system based on the user's instructions. A basis for reaching the final diagnosis.

The problem state, goal state, and performance discrepancy views support the problem detection routine. During the model development routine, the users develop structural models, statistical models, and rules. During the model application routine, the users generate diagnostic conclusions and identify causes by applying the causal models to the problem detection-related views.

Second, concerning the problem processing, this chapter identified and proposed the following problem diagnosis support functions:

Problem Sensing	Create, retrieve, modify, save, delete problem and goal states Project goal (linear projection and exponential smoothing) Search goal (combines data retrieval and statistical searching)
Filtering	Filter out/in (removes/recovers variables from various views) Create, retrieve, modify, save, delete discrepancy views Evaluate significance (identify symptomatic variables)
Labeling	Discretize (converts numeric variables into categorical variables) Quantify (converts categorical variables into numeric variables) Associate (associates categorical variables) Define (defines new formula-based data fields)
Structural Modeling	Create, retrieve, modify, save, delete structural models Measure dependence (correlation, factor analysis, and contingency table test) Analyze structural relationships (affects, affected, union operation, intersection operation, cycle detection, path finding)
Statistical Modeling	Create, retrieve, modify, save, delete statistical models Identify candidate variables and fit statistical models Compare sign (cross-validate statistical and structural models)
Rule-Based Modeling	Create, retrieve, modify, save, delete rules Identify candidate variables and evaluate rule accuracy Induce rules (revised ID3 and attribute-oriented induction) Handle inconsistent and incomplete rules
Alternative Diagnosis Generation	Create, retrieve, modify, save, delete alternative diagnosis views Generate diagnosis (determines the target values) Identify causes (explains the diagnosis and identifies the causes)
Diagnosis Evaluation	Compare and contrast alternative diagnosis views

Finally, concerning the user-system interaction, the system supports and combines both user-driven and system-driven problem diagnosis. The system supports interactive goal formulation, structural model formulation, statistical model development and validation, rule formulation, and diagnosis generation. In many cases, active interactions between the user and the system, by themselves, tend to reduce errors and uncertainties, because they combine the user's and the

system's capabilities. This research utilized the inquiry "guaranteeing" concepts for the design of a non-specific PDSS as follows.

First, for system-driven induction, the system utilizes multiple algorithm. This may lead to a Lockean consensus or a Hegelian assumption evaluation. Second, the system checks and maintains Leibnitzian consistencies between a structural model and a descriptive model set. Third, the system manages multiple model views to support Kantian, Hegelian, and Singerian inquiries. In addition, it manages multiple problem state, goal state, and performance discrepancy views to support comparative analysis. During the problem diagnosis generation routine, the system supports what-if analysis. Fourth, the system builds descriptive models based upon structural models. Thus, users may be able to develop more intricate causal models as they develop a deeper understanding of the problem. In this sense, the system supports a Singerian model refinement process. The system also eliminates some of the common mistakes that the users can make, e.g., incorrect model structure and incorrect data analysis method application. These features help the users to reduce uncertainties, errors, and biases.

CHAPTER VI

SYMBOL-LEVEL DESIGN

The previous chapter performed the conceptual design of PDSS. The previous chapter identified the major view types, proposed various problem diagnosis support functions, and discussed how the inquiry "guaranteeing" concepts can be incorporated into the PDSS design. The conceptual design has been performed to satisfy the requirements of PDSS discussed in Chapter IV. This chapter continues to detail the system in terms of its structure and function.

The purpose of this chapter is to realize the conceptual design in terms of abstract symbols. The first section discusses the attribute structures of the views. The second section defines critical problem processing functions. The third section discusses the prototype system implementation environment. The prototype system is described in the Appendix. The final section summarizes this chapter.

6.1. Attribute Structures of the Views

The purpose of this section is to establish the attribute structures of the views discussed in Section 5.1. A simple data table structure was used to hold the example data view, problem state views, goal state views, and performance discrepancy views. A data table consist of two components: field definitions and the actual data occurrences. Because the field definitions will be shared by all data table views, they are maintained in a central location, i.e., data dictionary. Thus, the example data view is simply defined as:

$\{EV_{aet}\}$ is a set of example data values, where

$a = 1, 2, ..., A$, denotes the attribute or data fields; $e = 1, 2, ..., E$, denotes the problem instances; and $t = 1, 2, ..., T$, denotes time.

The system allows the users to examine one problem instance at a time. Thus, the problem state view is defined as

$\{PV_{at}\}$ is a set of problem state data values, where

$\{PV_{at} | t = 1, 2, ..., T - 1\}$ represent the historical problem state values; $\{PV_{aT}\}$ represents the current problem state values.

A goal state view represents the desired state of the problem instance *e* at time *T*. A goal state view has the following attributes:

$\{<GV_a, UV_a, UD_a, LV_a, LD_a>\}$ is a goal state view, where

$a = 1, 2, ..., A$ denotes the attributes; GV_a denotes a goal value; UV_a and LV_a denote an upper boundary value and a lower boundary value, respectively; UD_a and LD_a indicate whether a value above UV_a or below LV_a is desirable. $\{UD_a\}$ and $\{LD_a\}$ assume a domain value to be either "desirable" or "not desirable."

A performance discrepancy view shows whether the problem state is satisfactory in relation to the goal state with the following attributes:

$\{<PV_{aT}, GV_a, DV_a, DC_a> \mid a$ assumes a numeric domain$\}$.

DV_a may denote the difference of an attribute value between the current problem state and the goal state, i.e., $DV_a = PV_{aT} - GV_a$. DC_a denotes whether the attribute *a* is "problematic." DC_a is one of the following labels, {"favorable deviation," "insignificant deviation," or "unfavorable deviation"}. $\{DC_a\}$ are determined based on the logical relationships explained in Subsection 5.2.2.

Model views consist of three types of resources: structural models; mathematical statements; and rule statements. The structural models are stored in the following data structure:

$SM = <V, R>$ is a structural model view,

where $V = \{V_i \mid i = 1, 2, ..., n\}$ is a set of variables; and $R = \{R_{ij} \mid i = 1, 2, ..., n; j = 1, 2, ..., n\}$ is a set of values that denote the relationships between the

variables *i* and *j*. R_{ij} can assume a value of -1 (negative relationship), 0 (no relationship), +1 (positive relationship), or 1 (there is a relationship).

The statistical relationships are stored using the following structure:

$QM_y = <CD_y, SS_y>$, where

$CD_y = \{<V_{(ya)}, OP_{(ya)}, VV_{(ya)}>\}$ is a set conditions;

$SS_y = <V_y, \{<V_x, ß_x>\}>$ is a statistical equation.

$V_{(ya)}$ denotes an attribute name, i.e., a variable. $OP_{(ya)}$ denotes a relation operator, such as >, <, =, ≠, ≤, and ≥. $VV_{(ya)}$ is a domain value of the attribute, $V_{(ya)}$. V_y is the target quantitative variable. $\{V_x\}$ are the explanatory quantitative variables. $\{ß_x\}$ denote the least-square regression coefficients.[11]

A rule statement consists of conditions and conclusions. The prototype system represents a rule with the following structure:

$RL_r = <CD_r, CO_r>$, where

$CD_r = \{<V_{(ra)}, OP_{(ra)}, VV_{(ra)}>\}$ is a set of conditions;

$CO_r = <V_{(rz)}, VV_{(rz)}, PR_r>$ is a set of conclusions.

Although any form of propositions can be made in the conclusion part of the rule, the system restricts it to the form, $<V_{(rz)}, VV_{(rz)}>$, like many other rule-based classification systems. $V_{(rz)}$ is instantiated to $VV_{(rz)}$, if the conditions are satisfied. PR_r denotes the probability that the conclusion is true in the example set.

A descriptive model set has the following elements:

$DM = <SM, \{QM\}, \{RL\}>$, where

SM is a structural model; {QM} is a set of statistical model statements; and {RL} is a set of rules. An alternative diagnosis view consists of a discrepancy view (either {PV} or {GV} can be empty) and a descriptive model set.

---

[11] A statistical model view has many other attributes, e.g., the entries of the ANOVA table. Such attributes are not shown here to maintain simplicity.

6.2. System-Driven Problem Diagnosis Support Functions

The purpose of this section is to precisely define some of the critical system-driven problem processing functions discussed in Section 5.2 in terms of the symbols defined in the previous section. The implementation of the user-oriented function, such as editing, saving, retrieving, and deleting views, are not discussed, because their implementation is straightforward.

6.2.1. Goal Formulation

The prototype system supports two forecasting techniques: the least-squares regression analysis and the exponential smoothing technique. These techniques are applied over the historical problem states, $\{PV_{at} \mid t = 1, 2, ..., T - 1\}$, to forecast $\{GV_a\}$. For the exponential smoothing technique, the system identifies an optimal smoothing factor by incrementing α_a by 0.05 from 0.0 to 1.0.

The function, search goal, accepts a set of search conditions and a list of comparison criteria variables. It is possible for the users to assign weights for the comparison criteria variables. As a first step, the system retrieves the tuples $\{EV_{ae(T-1)}\}$ that satisfy the given data retrieval conditions. The system, then, calculates the distance between each retrieved example case e and the given state (used here is the historical problem state at time, T - 1), for example, as follows:

$$DS_e = \sqrt{\sum_{a=1}^{c} W_a * \left(\frac{EV_{ae(T-1)} - PV_{a(T-1)}}{Std(V_a)}\right)^2}$$

where DS_e is the weighted statistical distance; c is the number of comparison criteria variables; W_a is the weight for the variable a; and $Std(V_a)$ is the sample standard deviation of the variable a, obtained from $\{EV_{aet} \mid e = 1, 2, ..., E; t = 1, 2, ..., T\}$. If DS_e is close to zero, the entity e, at time T - 1, is similar to the historical problem state, $\{PV_{a(T-1)}\}$. Thus, the system displays the retrieved

$\{EV_{aet}\}$ sorted in the ascending order with respect to their DS_e. The user may examine them and retrieve a particular case $\{EV_{aET}\}$, for example, as a goal state.

6.2.2. Structural Model Construction

The data fields can be divided into categorical and numeric data fields. The categorical fields include the discretized fields, associated fields, and the original fields storing labels. The numeric fields include the quantified fields, the user-defined formula fields, the rank fields, and the original fields storing numbers. To facilitate structural model development, the system provides statistics measuring the dependencies between pairs of the data fields. The dependency between two attributes is measured, if neither field contains too many distinctive labels. For this purpose, three statistical analysis techniques are employed (Iman and Conover, 1989, p. 415, 388, 609).

First, if V_i and V_j are both numeric fields, the system calculates the correlation coefficient between them as:

$$Corr(V_i, V_j) = \frac{Cov(V_i, V_j)}{Std(V_i) * Std(V_j)}$$

where $Cov(V_i, V_j)$ is the sample covariance between $\{\Delta EV_{iet} \mid e = 1, 2, ..., E; t = 2, ..., T\}$ and $\{\Delta EV_{jet}\}$. Then, the system obtains, from the Student's t distribution table, the p-value for the null hypothesis that the increments of the two fields are not correlated. The system also calculates the correlation between the original values, i.e., between $\{EV_{iet}\}$ and $\{EV_{jet}\}$.

Second, if V_i and V_j are both categorical fields, the system tabulates the joint occurrences of $\{EV_{iet}\}$ and $\{EV_{jet}\}$ in a contingency table. Assuming that V_i has $r = 1, 2, ..., R$ classes (distinctive labels, or events) and V_j has $c = 1, 2, ..., C$ classes, the test statistic is defined as:

$$T = \sum_{r}^{R} \sum_{c}^{C} \frac{(O_{rc} - E_{rc})^2}{E_{rc}}$$

where O_{rc} is the observed count in row *r*, column *c*; and E_{rc} is the expected count. The reported probability is the p-value (from the χ^2 table) for the null hypothesis that the row classification is independent with the column classification.

Finally, if V_i is a categorical field and V_j is a numeric field (or vice versa), the system calculates the treatment sum of squares (TSS) and the error sum of squares (ESS) as:

$$TSS = \sum_{r}^{R} n_r(\bar{V}_{jr} - \bar{V}_j)^2 \quad and \quad ESS = \sum_{r}^{R} \sum_{c}^{n_r} (V_{jrc} - \bar{V}_{jr})^2$$

where *R* is the number of classes in V_i; n_r is the count of $\{EV_{iet}\}$ that belong to the class *r*; $\bar{V}_{jr}$ is the mean of $\{EV_{jet} \mid EV_{iet}$ indicates the class *r*$\}$; V_j is the average of the numeric field *j*; V_{jrc} is identical with $\{EV_{jet}\}$ except that the elements are rearranged according to $\{EV_{iet}\}$. The reported DP_{ij}, in this case, is the p-value (from the F table) for the null hypothesis that the numeric averages for the distinctive categories are equal to each other.

6.2.3. Structural Model Analysis

Once the user formulates a structural model using the statistics, the system can analyze the relationships among variables. For this purpose, the system internally constructs two matrices: an adjacency matrix and a reachability matrix. As stated, the structural model is $<V, R>$, where $R = \{R_{pq} \mid p = 1, 2, ..., n; q = 1, 2, ..., n; R_{pq} = -1, 0, +1, \text{ or } 1\}$. The adjacency matrix is defined as $RA = \{RA_{pq}\}$, where $RA_{pq} = 0$ if $R_{pq} = 0$; and $RA_{pq} = 1$, if $R_{pq} \neq 0$. The matrix shows the direct effect of variable *i* to variable *j*. The reachability matrix (RC) is defined (Warfield, 1976) as:

$$RC = B(\sum_{s=1}^{n}) RA^{B(n)}$$

where $B(\Sigma)$ means Boolean addition, and $RA^{B(n)}$ means Boolean multiplication of matrix RA to the power of *n*. Figure 6.1 presents an algorithm for generating the final reachability matrix. The algorithm maximally utilizes the transitivity of the binary reachability matrix. The algorithm is very fast and requires a single matrix for calculation.

The reachability matrix shows both direct and indirect effects. The row vector *i* of the reachability matrix, $\{RC_{i1}, RC_{i2}, ..., RC_{in}\}$ indicates whether variable *i* affects variable *j*, where $j = 1, 2, ..., n$. Likewise, the column vector *i* of the matrix, $\{RC_{1i}, RC_{2i}, ..., RC_{ni}\}$ indicates whether the variable *i* is affected by variable *j*. To be noted is that the reachability matrix does not discriminate direct effects from indirect effects. A simple way of examining the indirect effect of *i* to *j* is to multiply the row vector *i* by the column vector *j* after setting $RC_{ij} = 0$. If the resulting RC_{ij} equals to 1, then there is an indirect effect from *i* to *j*.

Once the reachability matrix is prepared, it is not difficult to implement the functions, union affect, intersection affect, union affected, and intersection affected. These functions accept a set of variables, $V^A = \{V_1, V_2, ..., V_a\}$ and output the list of variables, $V^Z = \{V_1, V_2, ..., V_z\}$ that affect, or are affected by, V^A, based on the reachability matrix. To derive V^Z for union affect, the system constructs a row vector $\{r_1, r_2, ..., r_n\}$. Each element of the row vector is set to be $r_k = RC_{1k} + RC_{2k} + ... + RC_{nk}$. Then, one or more variable in V^A affect $V^Z = \{V_k \mid r_k \neq 0; k = 1, 2, ..., n\}$. Intersection affect is similar to the affect-union, except that $r_k = RC_{1k} \bullet RC_{2k} \bullet ... \bullet RC_{nk}$. Then, each and every variable in V^A affect $V^Z = \{V_k \mid r_k \neq 0; k = 1, 2, ..., n\}$. Union affected and intersection affected can be implemented correspondingly.

Reachability Matrix Derivation

Inputs: An adjacency matrix, RA = $\{RA_{ij} | i = 1, 2, ..., n; j = 1, 2, ..., n\}$

Output: A reachability matrix, RC = $\{RC_{ij} | i = 1, 2, ..., n; j = 1, 2, ..., n\}$.

1. Copy RC = $\{RC_{ij}\}$ from RA = $\{RA_{ij}\}$ for $i = 1, 2, ..., n$, and $j = 1, 2, ..., n$.
2. Set Flag = CONTINUE
3. While Flag = CONTINUE
   - 3.1. Set Flag = STOP;
   - 3.2. For $i = 1, 2, ..., n$
     - 3.2.1. For $j = 1, 2, ..., n$
       - 3.2.1.1. If $RC_{ij} = 0$
         - 3.2.1.1.1. For $k = 1, 2, ..., n$
           - 3.2.1.1.1.1 If $RC_{ik} * RC_{kj} = 1$
             - 3.2.1.1.1.1.1. Set $RC_{ij} = 1$, and Flag = CONTINUE;
             - 3.2.1.3.1.1.2. Break the loop starting 3.2.1.1.1 and Go to 3.2.1.

Figure 6.1. An Algorithm for Reachability Matrix Derivation

Cycle Detection

Inputs: A structural model, SM = <V, R>.

Output: C = $\{C_1, C_2, ..., C_c\}$ storing the causal cycles.

1. Let $C_C = \{\}$ storing a causal cycle.
2. Obtain RC = $\{RC_{ij} | i = 1, 2, ..., n; j = 1, 2, ..., n\}$.
3. For each $V_i \notin C$ where $RC_{ii} = 1$
   - 3.1. For each V_j, where $RC_{ij} = 1$, and $RC_{ji} = 1$
     - 3.1.1. Push V_j into C_C;
   - 3.2. Push C_C into C; Set $C_C = \{\}$.

Figure 6.2. An Algorithm for Cycle Detection

The function, path, helps the user to understand the relationships between two specific variables. There are four ways that two variables, *i* and *j*, are related. First, *i* and *j* affect each other, if and only if $RC_{ij} = 1$ and $RC_{ji} = 1$. If this condition is satisfied, the system identifies and reports the causal cycle as $\{V_p \mid RC_{ip} = 1; RC_{pi} = 1; p = 1, 2, ..., n\}$. Figure 6.2 presented a simple algorithm that identifies all the causal cycles in the structural model.

Second, there is no causal path between *i* and *j*, if and only if $RC_{ij} = 0$ and $RC_{ji} = 0$. For this case, the system identifies the variables that affect, or are affected by both *i* and *j*. The variables that affect both i and j are $\{V_p \mid RC_{pi} = 1$ and $RC_{pj} = 1\}$; and the ones affected by both are $\{V_p \mid RC_{ip} = 1$ and $RC_{jp} = 1\}$.

Finally, *i* affects *j*, if $RC_{ij} = 1$ and $RC_{ji} = 0$. Likewise, *j* affects *i*, if $RC_{ji} = 1$ and $RC_{ij} = 0$. If either case is true, the system identifies and reports all causal paths between *i* and *j*. Path identification involves two major steps. As shown in Figure 6.3, the first step removes all the cycles from the structural model and constructs a condensed matrix. The second step, shown in Figure 6.4, identifies all causal paths between the given variables based on the condensed matrix.

The structural model stores the relationships only between the original and formula-based data fields. The quantified, discretized, and associated fields are "attached" to the data fields from which they are converted. The formula-based data fields are affected only by those variables used to define the fields. They can affect any other variables except unrelated formula-based fields.

6.2.4. Descriptive Model Development

If a structural model is given, the system initially identifies the candidate target variables as $\{V_y \mid RC_{xy} \neq 0; x = 1, 2, ..., n\}$. For statistical modeling, the list is adjusted to exclude all formula-based data fields and include appropriate

Cycle Removal

Input: A structural model, SM = <V, R>

Output: A condensed matrix $\{RZ_{ij}\}$ and lists of variables $C = \{C_1, C_2, ... C_c\}$

Procedures:

1. Obtain $RC = \{RC_{ij}\}$ and $C = \{C_1, C_2, ..., C_{c'}\}$ based on Figures 6.2 and 6.3
2. For each $V_i \notin C$, Push $C_C = \{V_i\}$ into C
3. Set $\{RZ_{ij} = 0 | \ i = 1, 2, ..., n; j = 1, 2, ..., n\}$
4. For each V_a in each C_i
   - 4.1. For each V_b in each Cj
     - 4.1.1. If $RZ_{ij} = 0$, then Set $RM_{ij} = R_{ab}$
     - 4.1.2. If $RZ_{ij} \neq 0$ and $R_{ab} \neq 0$, then Set $RM_{ij} = 1$

Figure 6.3. An Algorithm for Cycle Removal

Path

Input: A condensed matrix $\{RZ_{ij}\}$, $C = \{C_1, ...C_x, C_y, C_c\}$

Output: $P = \{P_1, P_2, ..., P_p\}$ storing all paths between C_x and C_y

Procedures:

1. Set $P_P = \{C_x\}$
2. Path(P_P)
   - 2.1. If $C_y \in P_P$, Put P_P into P and exit from the current Path(P_P)
   - 2.2. For each $\{C_n \mid C_n \in C; RZ_{(Ck)(Cn)} \neq 0\}$, where C_k is the last element in P_P
     - 2.2.1. Push C_n into P_P
     - 2.2.2. Perform Path(P_P) recursively
     - 2.2.3. Remove C_n from P_P

Figure 6.4. An Algorithm for Path Identification

quantified fields. Once the user selects a target variable from the list, the system identifies the candidate explanatory variables as $\{V_x \mid RC_{xy} \neq 0\}$ and adjusts the list. If the user provides a set of data retrieval conditions, the system fits the statistical model with only those examples that satisfy the given conditions.

The least-square regression coefficients, $\{\beta_x\}$, are estimated by $\mathbf{B} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$, where $\mathbf{X}$ is a matrix storing 1s (for the constant term) and the values of explanatory variables $\{V_x\}$; and $\mathbf{Y}$ is a column vector storing the values of V_y (In the implementation, the inverse of $\mathbf{X}'\mathbf{X}$ is obtained by the Gauss-Jordan elimination method). For each statistical model statement, the system maintains: (1) the conditions under which the model statement is applicable; (2) the list of target and explanatory variables; (3) the regression coefficients; (4) the standard errors of the regression coefficients; (5) the standard deviations of the variables; (6) the number of observations satisfying the given conditions; (7) the total sum of squares; and (8) the error sum of squares.

For system-driven rule discovery, ID3, having performed well in the chess end-game, is widely used. ID3 (Quinlan, 1986) builds a decision tree leading to the determination of an event based on the information entropy concept (see Cendrowska, 1987 for illustration). The decision tree is generated by setting the root node to be the entire set of examples; partitioning the examples according to the conditional information entropy, thus creating child nodes; and recursively partitioning each of the child nodes in the same fashion. Figure 6.5 introduces a few modifications to ID3, so that the algorithm can be applied to managerial problem diagnosis.

First, ID3 ignores the degrees of freedom associated with the explanatory variables. That is, the variables with large numbers of associated events (values or classes) tend to be used first for branching the decision tree (Hart, 1985). To

Revised ID3

Inputs: $\{EV_{ij} \mid i = 1, ..., x, \text{ and } y; j = 1, ..., \Sigma^M\Sigma^T\}$

Outputs: rules, $RL = \{RL_1, ..., RL_w\}$, modeling the events $\{V_{y1}, ..., V_{yr}\}$

Procedures:

1. Set $RL = \{\}$; $RL_w = \{\}$; and $V^X = \{V_1, V_2, ..., V_x\}$
2. For each V_x in V^X, obtain $P_x = P(\chi^2 > T_x; \text{d.f.} = (R\text{-}1)(C_x\text{-}1))$, where

$$T_x = \sum_{r=1}^{R} \sum_{c=1}^{C_x} \frac{(O_{rc} - E_{rc})^2}{E_{rc}}$$

   R is the number of the events, $\{V_{yr}\}$, related to y in $\{EVij\}$; C_x is the number of the events, $\{V_{xc}\}$; O_{rc} is the observed count of $\{EV_{ij} \mid EV_{xj} = c; EV_{yj} = r\}$; and E_{rc} is the expected count.
3. Identify V_x with the minimum P_x, where P_x is less than, for example, 10%
4. If V_x cannot be found (because $V^X = \{\}$ or the minimum P_x is unsatisfactory) or is unnecessary (because $R = 1$, i.e., homogeneous $\{EV_{yj}\}$)
   4.1. Identify the most frequent event V_{yr} from $\{EV_{yj}\}$ and put $<V_y, r, PR_r>$ into the conclusion part of RL_w, where PR_r is $\{EV_{yj} | EV_{yj} = r\} / \{EV_{yj}\}$
5. Else
   5.1. Remove V_x from V^X;
   5.2. For each event c associated with the identified explanatory variable V_x
      5.2.1. Put the statement, "$V_x = c$," into the condition part of RL_w
      5.2.2. Create a non-empty subset, $\{EV_{ij} | EV_{xj} = c; i = 1, ..., x\text{-}1, \text{ and } y\}$
      5.2.3. Recursively perform steps 2 - 5.2.3 with V^X, RL_w, and $\{EV_{ij}\}$ in steps 5.1, 5.2.1., and 5.2.2, respectively
      5.2.4. Put RL_w into RL; and set $RL_w = \{\}$

Figure 6.5. A Revised ID3 Algorithm

remedy the problem the revised algorithm determines the branching variables based on the contingency table test instead of the conditional information entropy.

Second, ID3 produces a decision tree. Decision trees are often "difficult to comprehend, manipulate, and explain" (Cendrowska, 1987, p. 354). The revised algorithm directly generates a set of rules instead of a decision tree.

Third, the revised algorithm denotes the terminal nodes, i.e, the conclusions of rules, with probabilities. Thus, the user can examine the probabilities to evaluate the accuracy of the rules. If multiple rules are true, the system chooses the conclusion made by the rule with highest accuracy.

One major drawback of ID3 is that it partitions the examples into subsets for all possible values of the branching variable. This is because ID3 it is designed to maximize the amount of information contributed by an explanatory variable to the determination of the target variable's values (Quinlan, 1986).

The goal of attribute-value oriented rule induction is to maximize the amount of information contributed by knowing the value of the explanatory variable to the determination of the target variable's values (Cendrowska, 1987; Uthurusamy et al., 1991). Thus, attribute-value oriented rule discovery does not subdivide the examples for unimportant attribute-value pairs. Figure 6.6 presents an attribute-value oriented rule induction algorithm, which is a variant of ID3.

6.2.5. Alternative Diagnosis Generation

The system determines the target variables' values based on: (1) a descriptive model set; and (2) a problem or goal state view. As stated in Section 6.1, a descriptive model set consists of statistical model statements and rules. For inferencing purpose, it is useful to treat statistical model statements as rules. The data in the problem or goal state view are facts over which these rules are fired.

An Attribute-Value Oriented Rule Induction Algorithm

Inputs: $\{EV_{ij} \mid i = 1, ..., x, \text{ and } y; j = 1, ..., \Sigma^M\Sigma^T\}$

Outputs: rules, $RL = \{RL_1, ..., RL_w\}$, modeling the events $\{V_{y1}, ..., V_{yr}\}$

Procedures:

1. Set $RL = \{\}$; $RL_w = \{\}$; and $V^X = \{V_1, V_2, ..., V_x\}$
2. For each attribute V_x in V^X

   2.1. For each event V_{xc} in $\{EV_{ij}\}$, obtain $P_{xc} = P(\chi^2 > T_{xc}; \text{ d.f.} = (R_{xc}-1))$, where

$$T_{xc} = \sum_{r=1}^{R_{xc}} \frac{(O_r - E_r)^2}{E_r}$$

   O_r is the observed count of the $\{EV_{ij} \mid EV_{xj} = c, \text{ and } EV_{yj} = r\}$.
3. Identify V_{xc} with the minimum P_{xc}, where P_{xc} is less than, for example, 10%
4. If V_{xc} is not found or is unnecessary

   4.1. Identify the most frequent event V_{yr} from $\{EV_{yj}\}$ and put $<V_y, r, PR_r>$ into the conclusion part of RL_w, where PR_r is $\{EV_{yj} | EV_{yj} = r\} / \{EV_{yj}\}$
5. Else

   5.1. Remove V_x from V^X

   5.2. Put the statement, "$V_x = c$," into the condition part of RL_w

   5.3. Create a non-empty subset, $\{EV_{ij} \mid EV_{xj} = c; i = 1, ..., x\text{-}1, \text{ and } y\}$

   5.4. Repeat steps 2 - 5.4 with V^X, RL_w, $\{EV_{ij}\}$ in steps 5.1, 5.2, and 5.3

   5.5. If $RL_w \neq \{\}$

      5.5.1. Set $V^X = \{V_1, V_2, ..., V_x\}$

      5.5.2. Remove $\{EV_{ij} \mid E_{ij}$ satisfies $RL_w\}$ from the original $\{EV_{ij}\}$

      5.5.3. Put RL_w into RL; and set $RL_w = \{\}$;

      5.5.4. Repeat steps 2 - 5.5.4 with the above V^X, $\{EV_{ij}\}$, and RL_w

Figure 6.6. An Attribute-Value Oriented Rule Induction Algorithm

The algorithm used to generate diagnostic conclusions is essentially a backward chaining of rules (Chabris, 1989) with the following modifications. First, the problem or goal state view is updated first by the discretization, quantification, and association rules and the formulas. Second, during the backward chaining, the following priorities are assigned to the different categories of rules: (1) user-provided formulas (highest priority); (2) statistical model statements; (3) decision rules; (4) quantification rules; (5) discretization rules; and (6) association rules.

Cause identification utilizes an algorithm essentially identical with the one used by Ata Mohamed (1985). One additional assumption employed in this research is that an unfavorable (favorable) deviation is explained by unfavorable (favorable) deviations in the explanatory variables.

6.3. Prototype Development Environment

The base language chosen for the prototype implementation is C++. As a superset of C language, C++ provides efficiency and flexibility. As an object-oriented language, C++ supports data encapsulation, inheritance, and polymorphism. C++ also has disadvantages. First, C++ is a typed language. Because the prototype system must manipulate strings as well as numeric values, C++ is less advantageous than non-typed languages. Second, C++ does not provide a rule interpreter usually found in more advanced development tools. Overall, however, the benefits of C++ outweigh its drawbacks.

Many C++ compilers generate Windows-compatible modules (Windows is a trademark of Microsoft). Windows provide several benefits to the programmer and users (McCord, 1992; Norton and Yao, 1992). For a window application development, the developers must discard the procedure-driven programming style

and embrace the event-driven programming style. Event-driven programming is effective for developing the applications that require frequent error handling and/or extensive user-system interaction (Barkakati, 1991).

The compiler used for the prototype implementation is Borland C++ (Borland C++ is a registered trademark of Borland International). Borland C++ provides a library that incorporates the advantages of object-oriented programming to Windows application development. dBASE III+ is used to create the example data view (dBASE III+ is a trademark of Ashton-Tate). The prototype system "imports" dBASE III+ files. This action was taken to concentrate on the development of more critical components of the system.

6.4. Summary

This chapter has developed the symbol-level design for PDSS. The inputs to the symbol-level design are the view types and diagnosis support functions established on the conceptual-level design. The outputs of the symbol-level design are the attribute structure of the views and the algorithms for the system-driven processing functions. This chapter also presented the prototype system implementation environment. Borland C++, Windows, and dBASE III+ were used for prototype system development and application.

CHAPTER VII

PROTOTYPE SYSTEM TESTING

The previous three chapters established the requirements of PDSS, the conceptual design of PDSS, and the symbol-level design of PDSS, respectively. The prototype system implementation serves two purposes. First, the actual system implementation verifies that the conceptual and symbol-level design performed in this research is, in fact, technologically viable. Second, the prototype system provides evidence that the conceptual and symbol-level design suffices as a basis for developing a system that satisfies the PDSS requirements.

The purpose of this chapter is to evaluate the prototype system's capabilities. Two problems were used to test the prototype system. The first problem is a bond rating problem; and the second is the blackline burning diagnosis problem. They are both relatively ill-structured problems. Scenarios are used to vividly describe the interactions between the user and the system.

7.1. Bond Classification Application

Bond ratings, in general, are determined by a firm's operating and financial states. They are also influenced by the rater's qualitative judgment about the firm's future ability to meet the scheduled payment on time. Portfolio managers and investors often use the ratings to evaluate the risk of investment in bonds.

Currently, Moody's bond rating system is most-widely used. It uses nine labels: "Aaa" (best), "Aa" (high quality), "A" (favorable), "Baa" (medium grade), "Ba" (speculative), "B" (lack desirability), "Caa" (poor), "Ca" (highly speculative), and "C" (extremely poor). In general, bonds rated in the top four categories are considered eligible for investment by the banks. For the ratings "Aa" through "B," Moody's system assigns a numerical modifier, 1, 2, or 3. The modifier 1 signifies

the highest end of its category. The rating system reflects only the investment quality factor and, thus, does not bear any indication for the future price of bonds.

7.1.1. Data Preparation

In order to obtain the example data view, the ratings of 100 outstanding corporate bonds were collected from Moody's Industrial Manual for the period of 1988 - 1991. The explanatory variables included such major financial statement items as current and fixed assets, current and long-term liabilities, owner's equities, and net income. The values of these explanatory variables were obtained from DISCLOSURE, a compact disk-based financial data base. A small C++ program was written to extract the data from the DISCLOSURE report, combine the data with the bond ratings, and produce an example data set. A few problem state views were established in the same fashion.

7.1.2. Demonstration of the System's Capabilities

Assume that a potential investor, named Alex, wants to evaluate a bond's quality. Further assume that its rating is not available from Moody's yet. Thus, he decided to assign "Moody's rating" himself. His goal is to capture the general characteristics of the rating system, i.e., develop a rule of thumb. He runs the prototype system. The first screen he sees is Figure 7.1.

He decides to examine the example cases to develop a "first-hand" feeling of the rating system. He selects Setting>>Example View from the main menu and retrieves several example data views as shown in Figure 7.2. Although the system provides a relatively easy way to compare the examples, he soon realizes that he may not comprehend the rating system by directly observing the raw data.

Thus, he decides to reduce the problem space. One hypothesis he considers is that larger firms may receive more favorable bond ratings. To

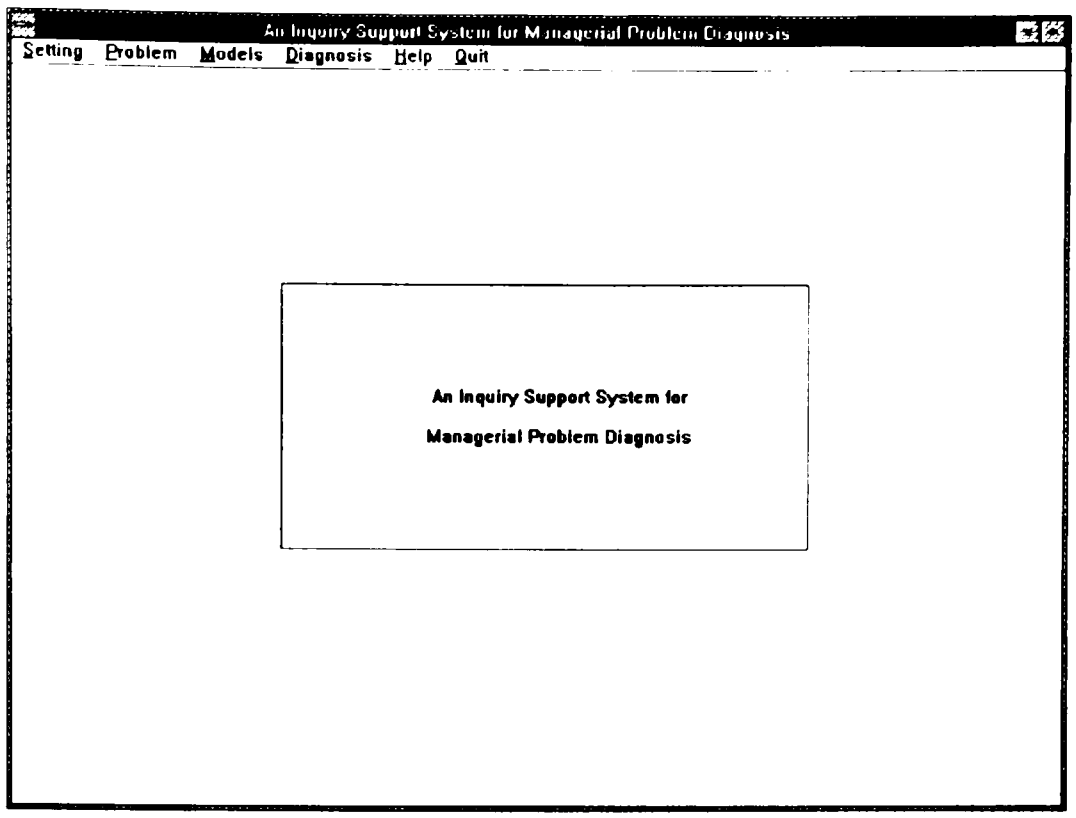


Figure 7.1. The Main Window

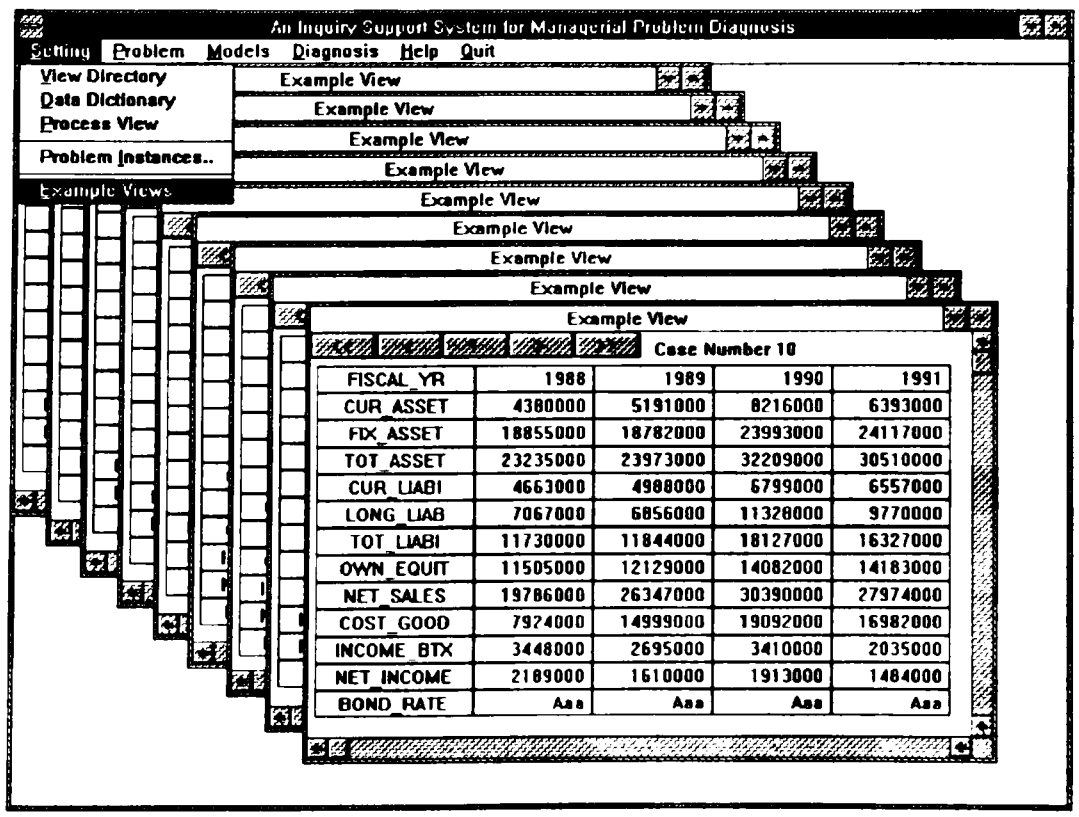


Figure 7.2. Example Data Views

represent this hypothesis, he opens a structural model view by choosing Model>> Structural from the main window. From the structural model window, he selects View>>Create New and creates a structural model.

The system copies all currently available variables from the data dictionary into the model. Figure 7.3 shows the model. He sets the relationship "tot-asset affects bond-rate" to "+1" by clicking the left button of the mouse. Then he clicks the right button at the same location. The system performs the analysis of variance for the hypothesis that the means of tot-asset for all categories of bond-rate are equal to each other. The dialog box in Figure 7.3 reports a p-value of 0.001 indicating that tot-asset is not likely to be independent with bond-rate.

He examines the relationships between bond-rate and all other variables similarly. He notes that bond-rate is best explained by net-income, income-btx, own-equit, and tot-asset. Based on the information, he decides to study the relationships between bond-rate and four variables, net-income, own-equit, tot-asset, and tot-liabi. He dropped income-btx, because it is highly correlated with net-income. He included tot-liabi, because it might negatively affect bond-rate. To simplify the problem, he selects Constructs>>Delete from the structural model window. As shown in the dialog box of Figure 7.4, he removes irrelevant variables. Upon returning to the window, he finds a much simpler model. He structures the relationships as shown at the upper left corner of Figure 7.4.

He studies structural relationships using Analysis>>Affects. As shown at the bottom of Figure 7.5, he examines what variables net-income affects. Net-income directly affects own-equit and bond-rate, and indirectly affects tot-asset and bond-rate. Thus, net-income affects bond-rate both directly and indirectly.

To further study this relationship, he selects Analysis>>Connected and chooses net-income and bond-rate, as shown in Figure 7.6. Figure 7.6 shows that

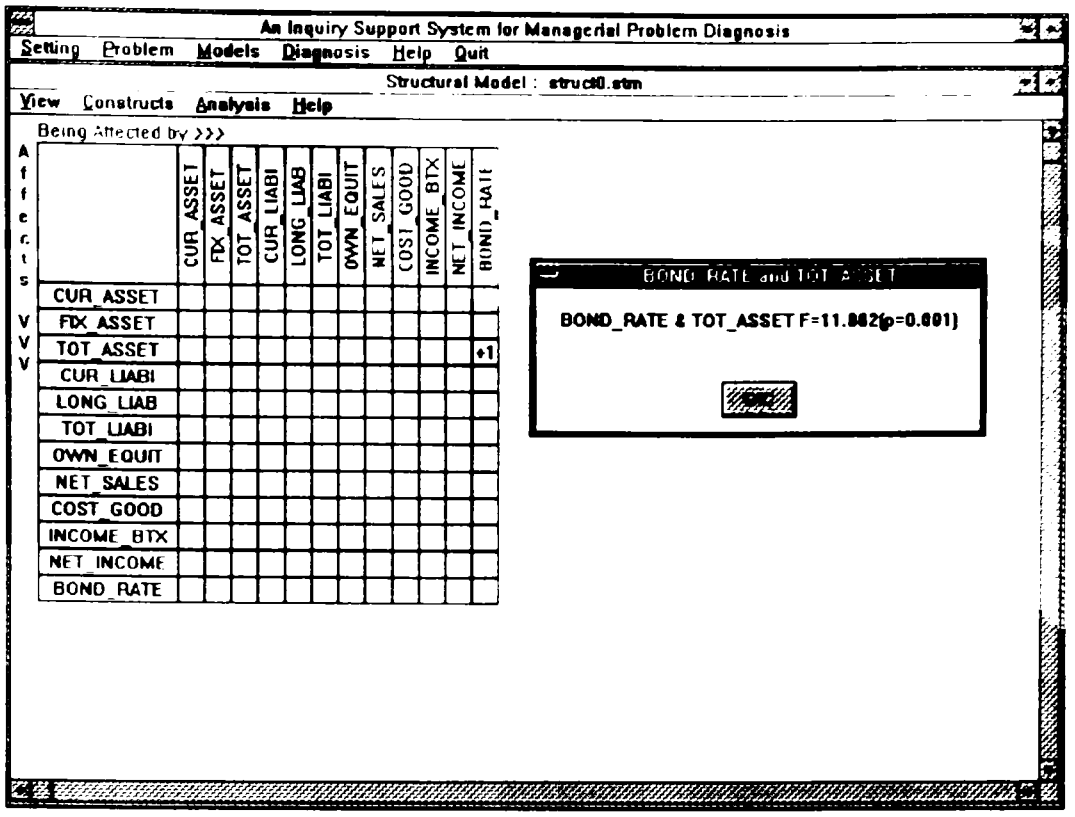


Figure 7.3. A Structural Model View and Data Analysis

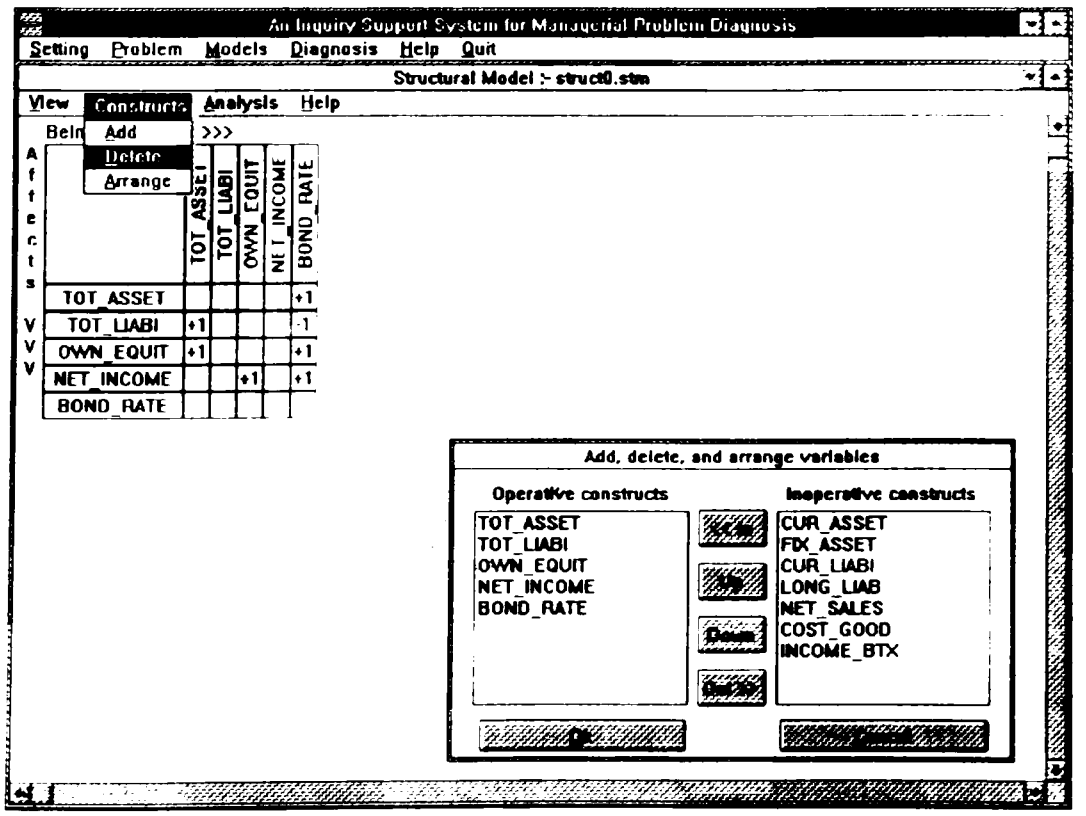


Figure 7.4. Structural Model Simplification

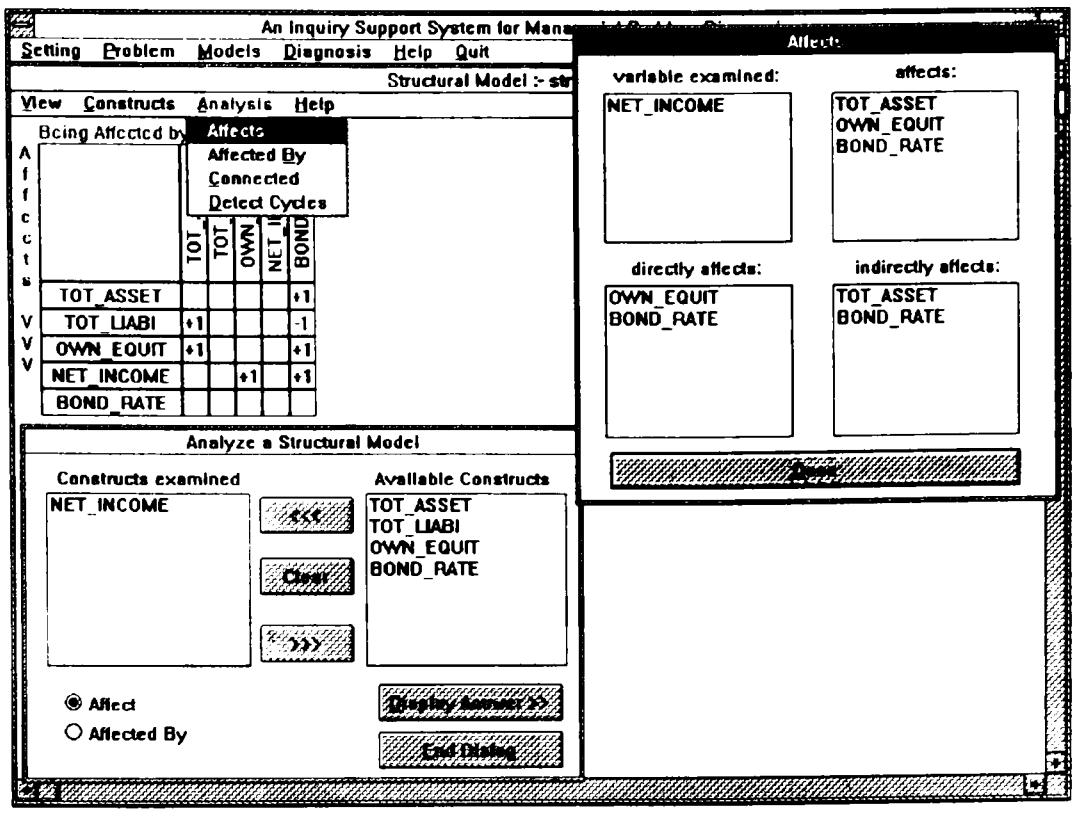


Figure 7.5. Structural Model Analysis

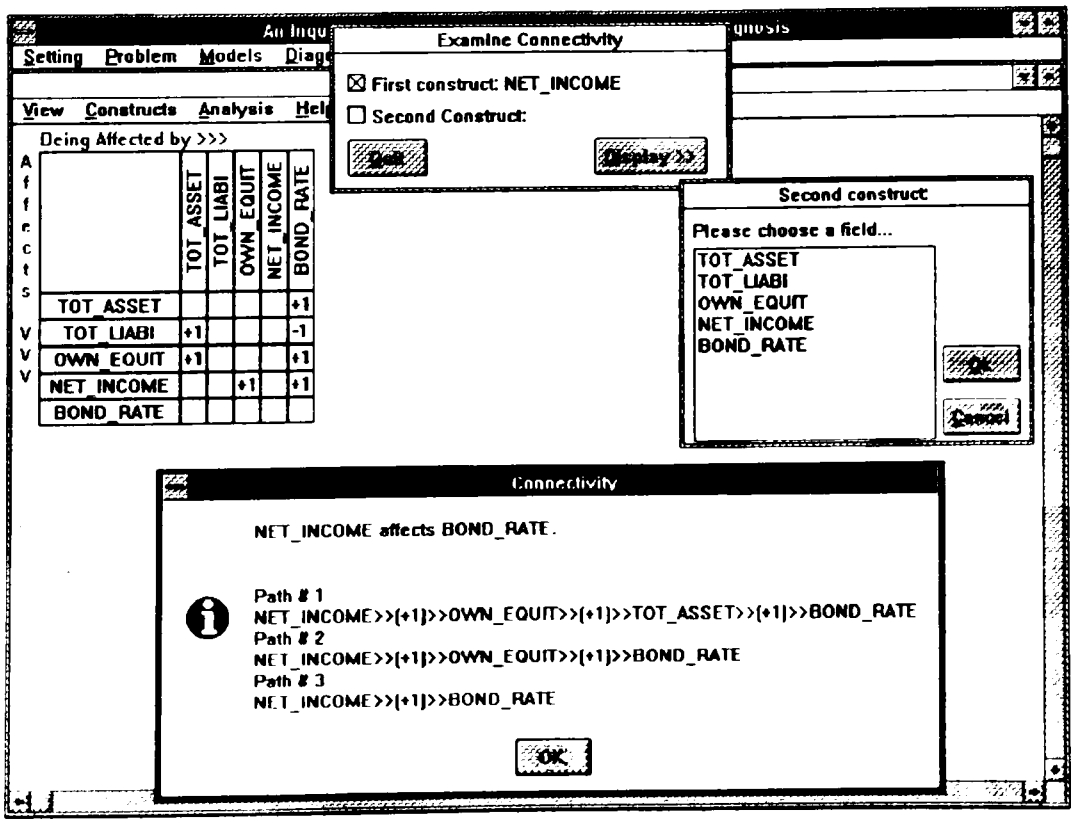


Figure 7.6. Structural Path Analysis

net-income has a direct positive impact and two indirect positive impacts on bond-rate. According to the model, an increase in net-income will improve bond-rate.

Alex continues structural path analysis. One relationship that attracted his attention is that tot-liabi also affects bond-rate both directly and indirectly. As Figure 7.7 shows, the direct effect is negative, while the indirect effect is positive. Thus, it is not possible to determine whether the total impact of tot-liabi to bond-rate is positive or negative based on the model. In fact, it may not be desirable for a firm to have too little or too much liability. Too little liability means an under-utilization of the tax shield, while too much liability indicates an increase in the firm's potential for bankruptcy. For this reason, Alex feels a need to introduce financial ratios indicating the capital structure of the firms.

He chooses Setting>>Data Dictionary (see Figure 7.2) from the main window. After opening the data dictionary, he performs the following tasks.

First, he creates several formula-based data fields by choosing Variable>> Create>>Define from the data dictionary menu. He defines two capital structure indicators. Leverage-t is defined as tot-liabi divided by tot-asset. Ldebt-ratio is defined as long-liab divided by (long-liab + own-equit). He also establishes a data field for return-on-asset (ROA), which is net-income over tot-asset.

Second, he wants to develop a statistical model that can estimate bond-rate. Thus, he quantifies bond-rate by choosing Variable>>Create>>Quantify. He assigns 5.0 to "Aaa1," 4.0 to "A3," 3.0 to "B," and so on. The name of the quantified field is bond-qn1.

Third, he discretizes numeric fields for system-driven rule discovery. Figure 7.8 shows the discretization routine. He chooses Variable>>Create>> Discretize and answers the dialog box shown in the middle. Then he edits the discretization rules as shown at the lower left corner. The "Quartile>>" button

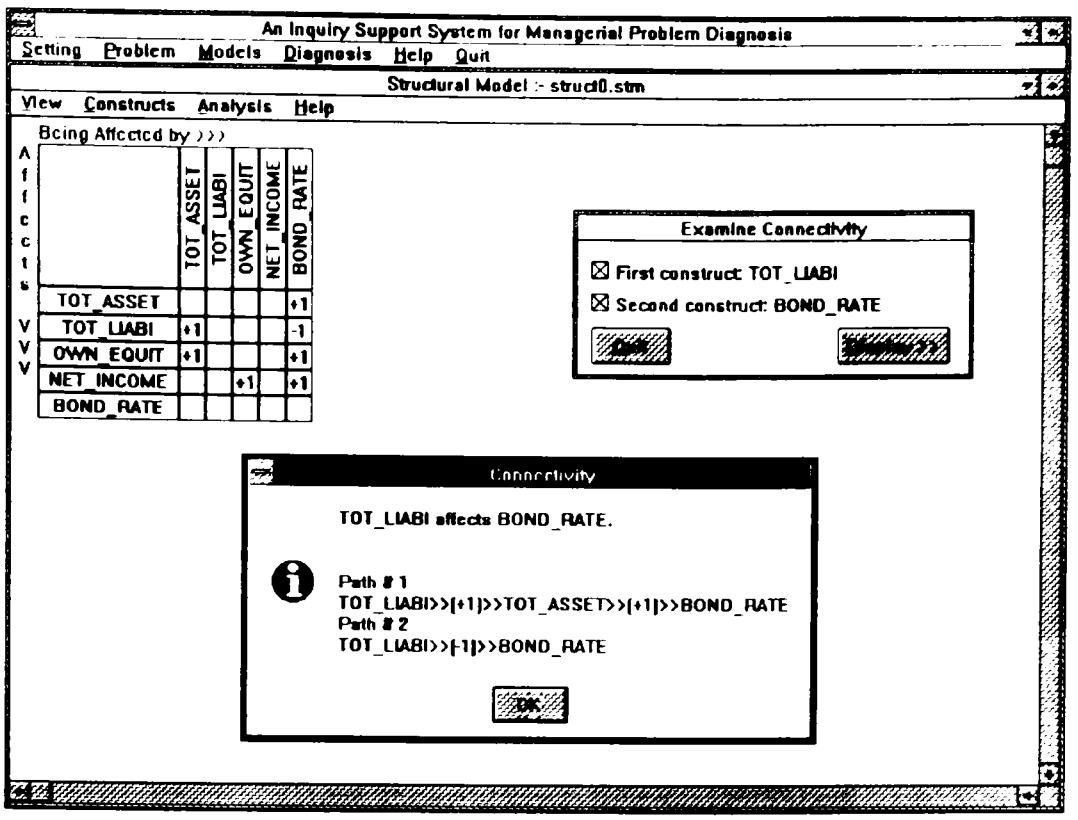


Figure 7.7. A Relationship with an Excitatory Path and an Inhibitory Path

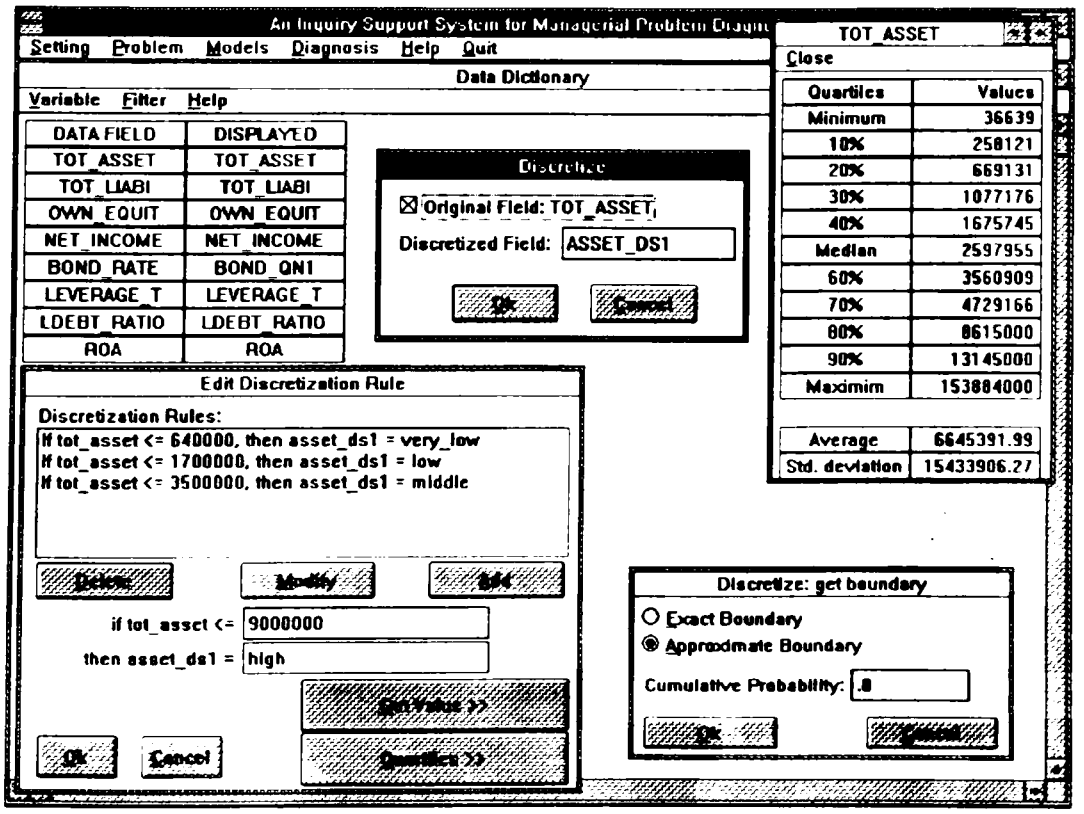


Figure 7.8. Discretization Routine

opens the window shown at the upper right corner. The "Get value>>" button opens the dialog shown at the lower right corner. For each numeric data field, he establishes five subintervals labeled as "very_low," "low," "middle," "high," and "very_high" by entering the probabilities, 0.2, 0.4, 0.6, 0.8, and 1.0, respectively.

Fourth, he establishes two categorical fields related to bond-rate. Bond-abc categorizes bond-rate into three classes, A, B, and C. Bond-yn indicates whether bonds are considered eligible for investment by the banks. The bond ratings, "Aaa," "Aa," "A," and "Baa," are "eligible."

Finally, he selects Filter>>Delete and filters out irrelevant data fields. The effects of data dictionary operations are global. Alex can study the example data in different ways using the converted fields, as shown in Figure 7.9. He can display converted data fields by changing the data dictionary setting.

Alex returns back to the structure model. He can easily switch back and forth between different views by using the "maximize" and "minimize" buttons. Upon returning to the structural model, he selects Constructs>>Add. He adds the formula-based variables into the model, one at a time, while establishing their relationships with the variables already in the model. During this process, he rearranges the variables in the order most comfortable to him.

As shown in Figure 7.10, he reexamines the relationship "tot-asset affects bond-rate." He now sees nine statistics (compare with Figure 7.3). He notes that all statistics are significant, except between increments of tot-asset and increments of bond-qn1. He suspects that the correlation is weak, because tot-asset and/or bond-qn1 do not change over time. He opens the example data view and finds that bond-rate is very stable over time.

At this point, Alex is satisfied with the system. The system helped him to conveniently examine the examples, identify key variables, simplify the problem in

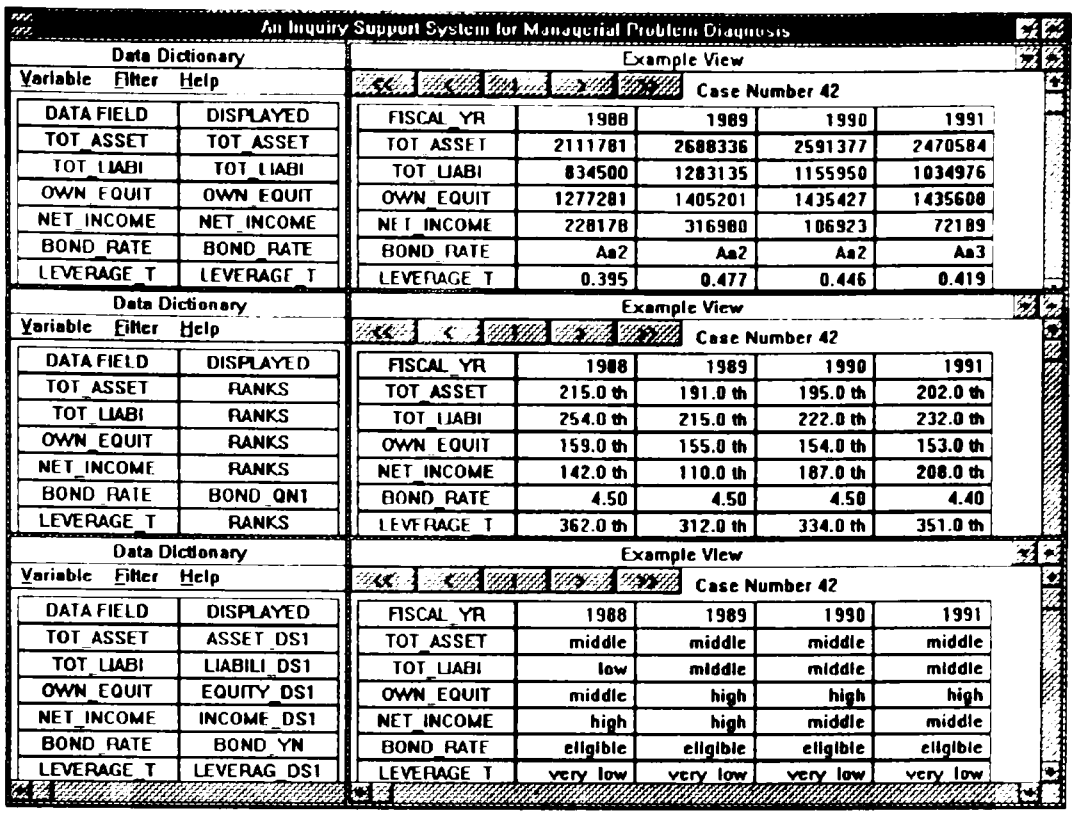


Figure 7.9. Alternative Ways of Examining the Example Data View

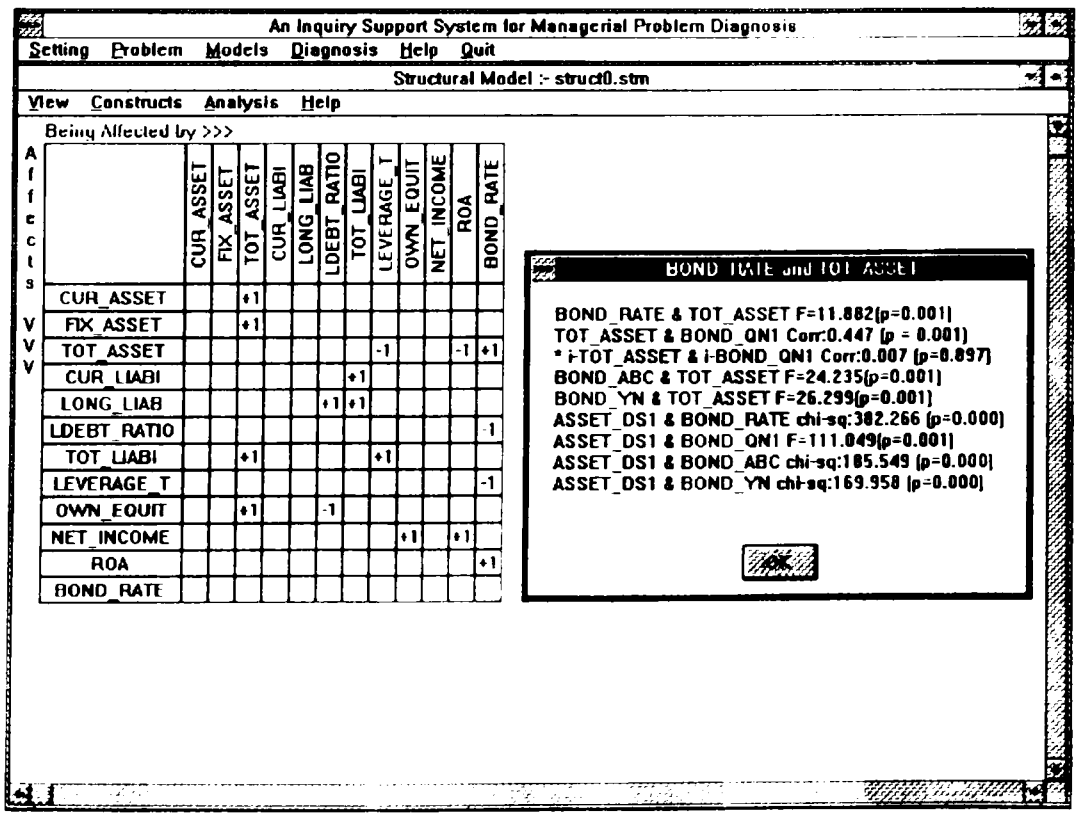


Figure 7.10. Structural Model Revision and Extended Data Analysis

the structural model, analyze structural relationships, define new variables, and realize that bond ratings are very stable over time. However, he is still wondering how tot-liabi might affect bond-rate.

Before developing a descriptive model, Alex notes that leverage-t and ldebt-ratio are competing concepts. Thus, he creates two alternative structural models: "leverage.stm" and "longdebt.stm" as shown in Figure 7.11.

He accesses Model>>Descriptive from the main window. Then he creates a new descriptive model set "leverage.qrm" using "leverage.stm." He creates a statistical model by using Create>>Equation, as shown in Figure 7.12. He clicks the "target variable" box. The dialog box at the right side states that tot-asset, tot-liabi, own-equit, and bond-qn1 are affected by other variable(s). He selects bond-qn1. The system now lists the variables that directly affect bond-qn1 in the box "explanatory variables" and all other affecting variables in the other list box.

He adds three variables, tot-liabi, own-equit, and net-income, to the explanatory variable set and presses the "Ok" button. He sees a statistical model statement in the descriptive model window (upper left corner of Figure 7.13). He moves the mouse on the model statement and clicks the left button of the mouse. The system opens a statistical model window. The window, however, cannot display the ANOVA table, because the model statement contains a functional relationship. Alex notes that tot-asset = own-equit + tot-liabi. Thus, he decides to remove tot-asset from the model. He chooses View>>Modify and opens the statistical model modification dialog shown at the top of Figure 7.13.

Alex takes out tot-asset and refits the model shown in Figure 7.14. He notes that the regular regression coefficients for tot-liabi, own-equit, and net-income are very small. This, however, does not imply that these variables have little impact on bond-qn1. In fact, the beta coefficients indicate that own-equit

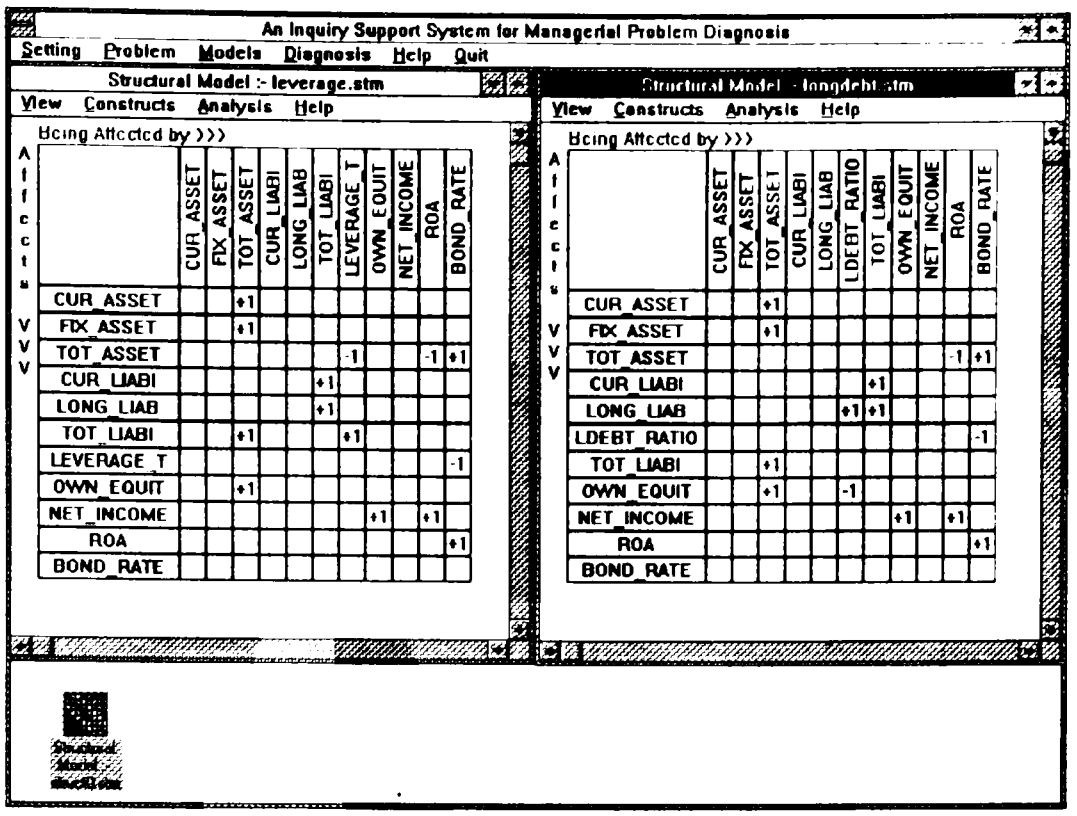


Figure 7.11. Alternative Structural Models

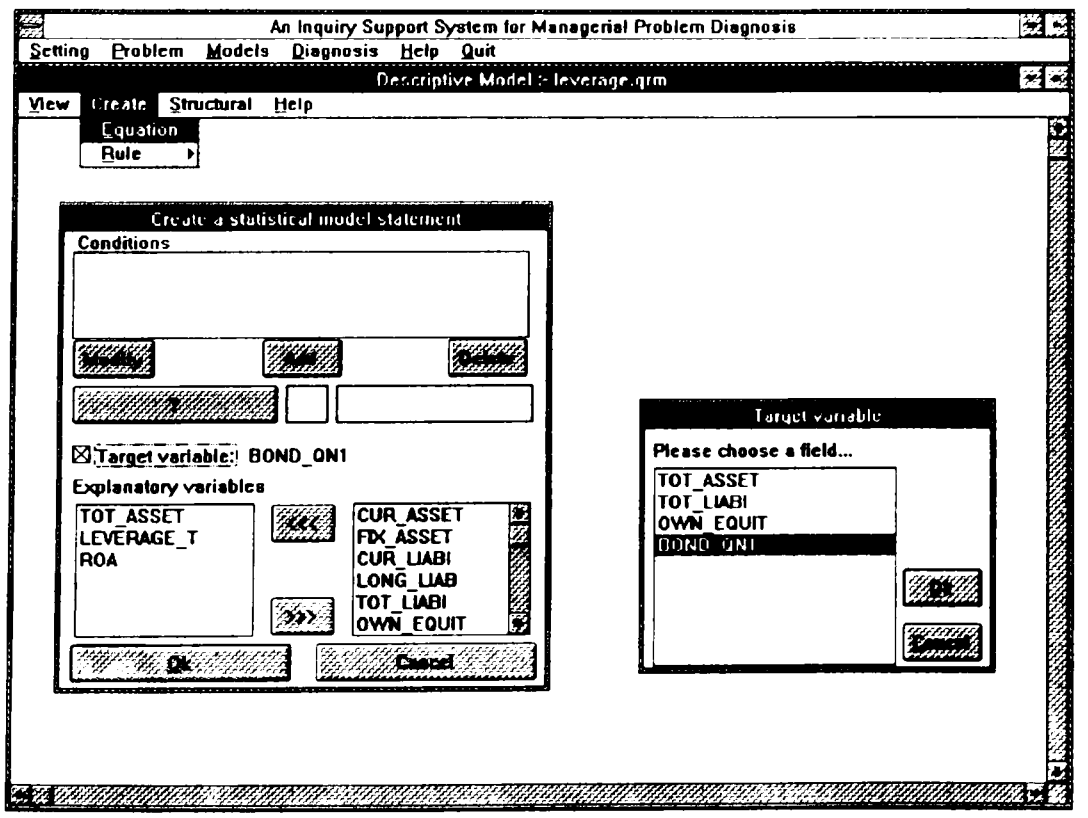


Figure 7.12. Statistical Model Construction

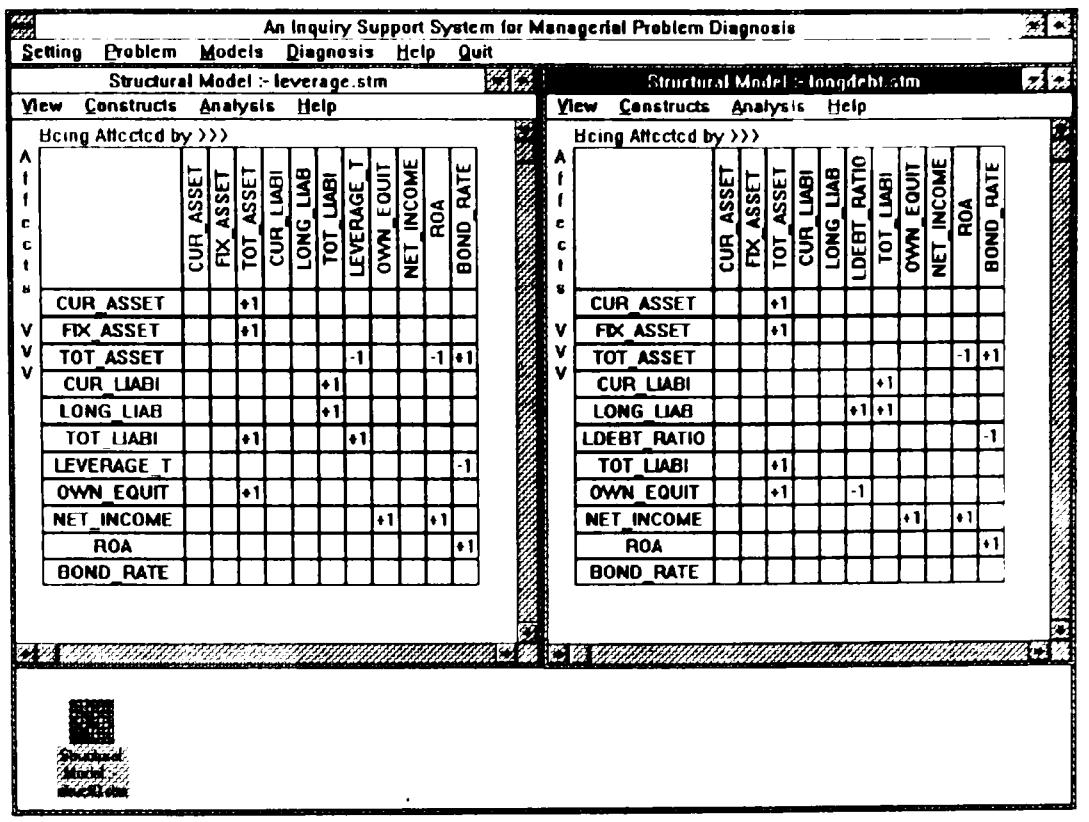


Figure 7.11. Alternative Structural Models

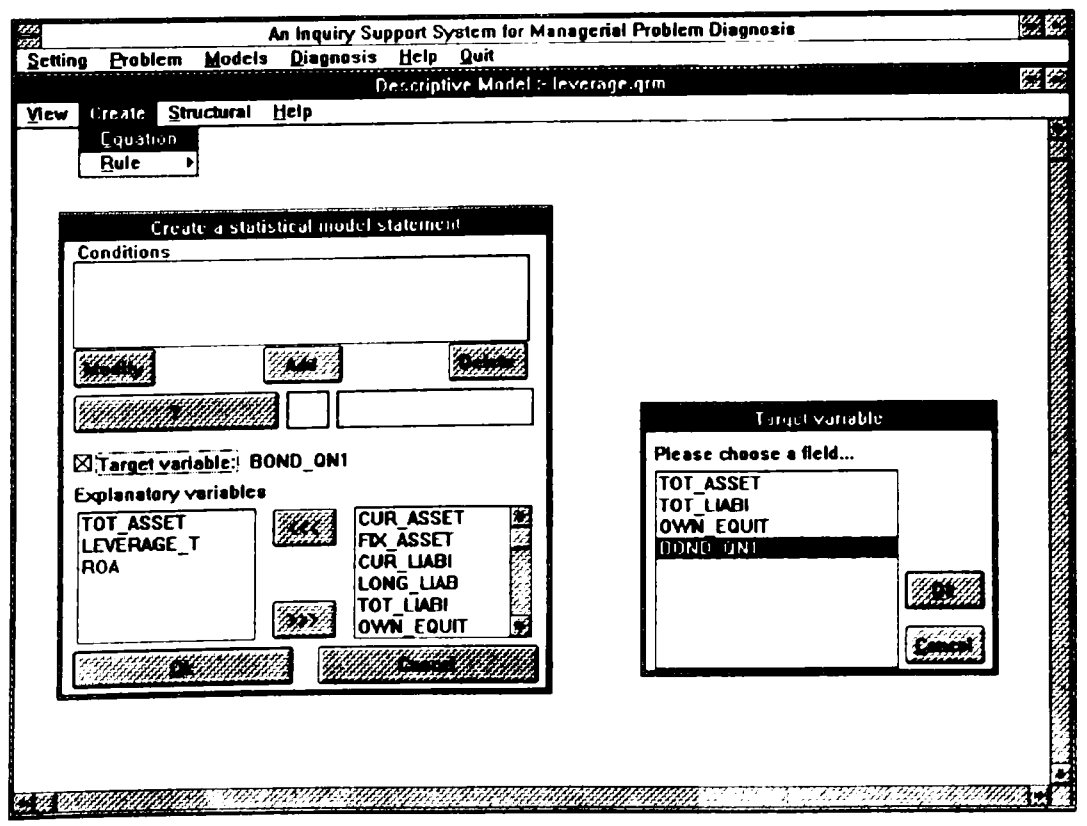


Figure 7.12. Statistical Model Construction

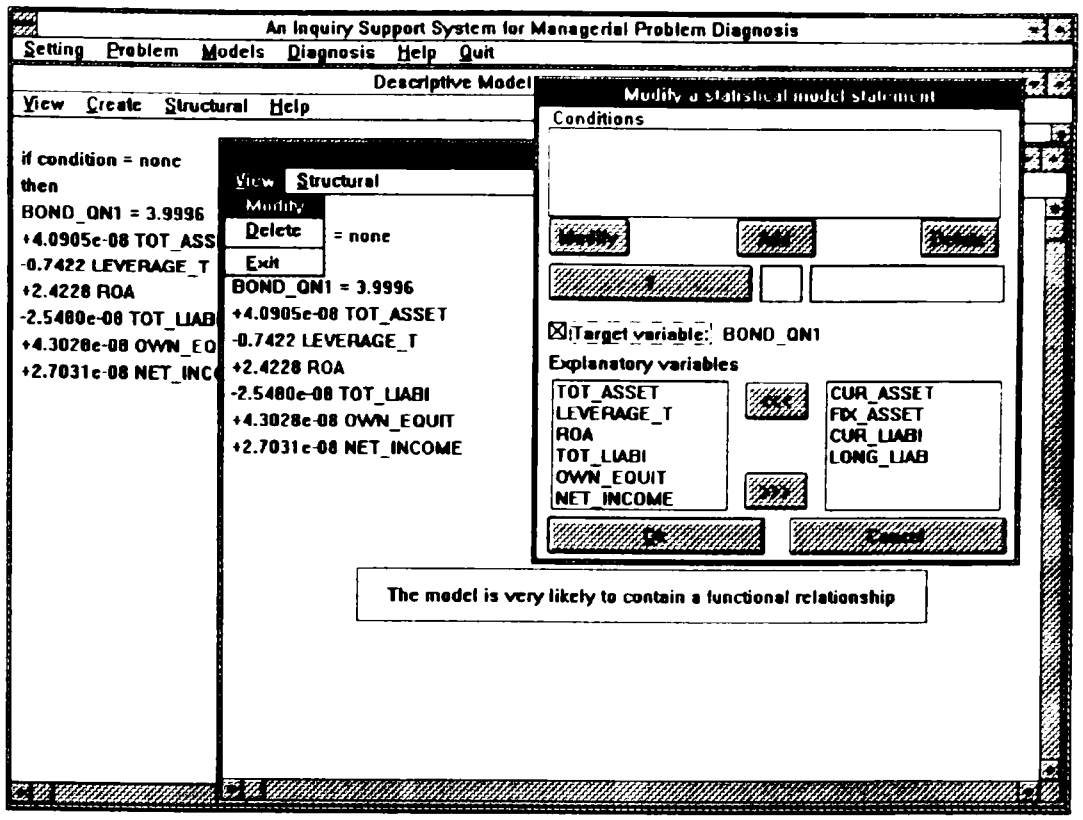


Figure 7.13. A Descriptive Model Window and a Statistical Model Window

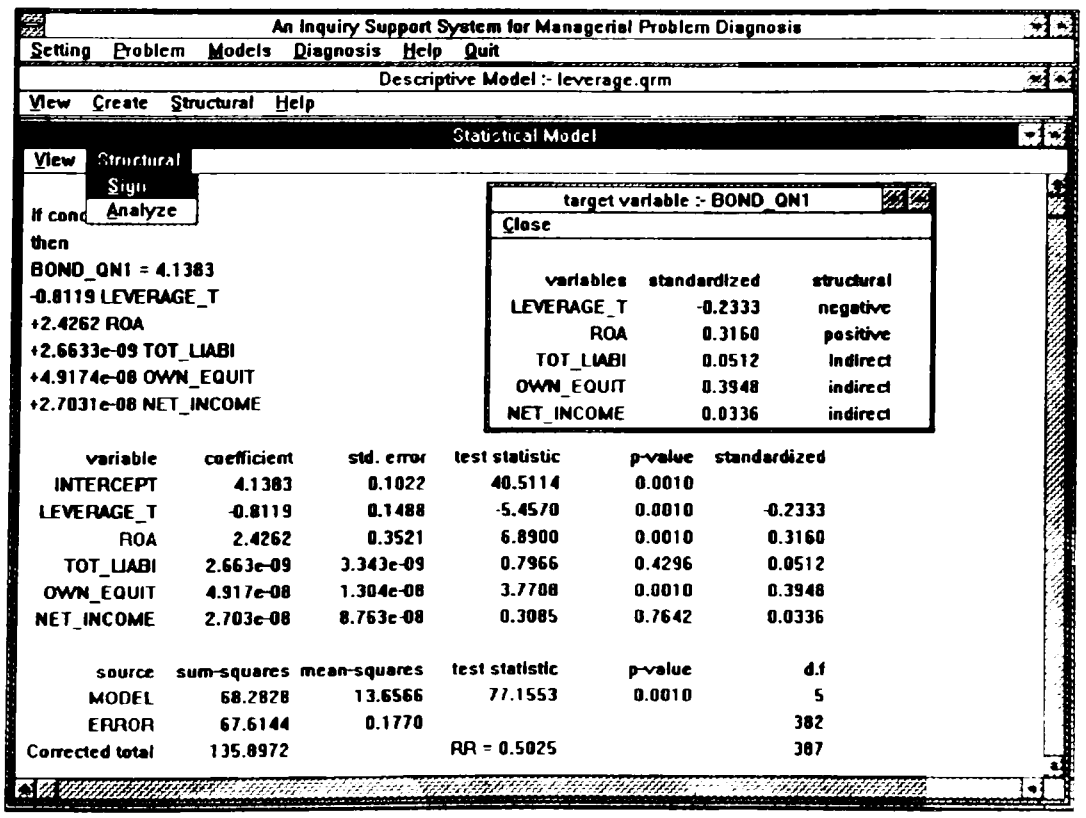


Figure 7.14. Cross-Validation of a Structural Model and a Statistical Model

has a greater impact on bond-qn1 than any other variable (assuming no multicollinearity). Alex chooses Structural>>Sign. The small window in Figure 7.14 reports that the coefficients of leverage-t and ROA are consistent with the structural model. He accesses Structural>>Analyze and examines the paths between bond-rate and explanatory variables. He finds that the sign of tot-liabi is the only one that contradicts with his "general" belief. He decides to drop tot-liabi, instead of net-income, from the model.

From Figure 7.15, he notes that net-income is still insignificant. Instead of modifying the model, he uses Create>>Equation and fits another equation with leverage-t, ROA, and own-equit. In this way, he can open multiple windows for comparison. He checks the signs of the regression coefficients once again and deletes the first model statement using View>>Delete.

He develops an alternative, but logically consistent, view of the model by replacing own-equit with tot-asset and tot-liabi (own-equit = tot-asset - tot-liabi). Figure 7.16 shows that tot-liabi affects bond-qn1 strongly in a negative direction. To be more precise, it is probable for the firms with larger tot-liabi to receive worse bond-rate. However, the structural model (Figure 7.7) indicates that it is also possible for the firms with smaller tot-liabi to receive better bond-rate.

Alex appreciates the benefits of building statistical models based on a structural model. First, the system identified candidate target and explanatory variables for him. This feature is especially useful when there are a large number of variables to be considered. Second, the system's structural path analysis helped him to evaluate the statistical model. The model development was, thus, guided more by his judgment than by the data alone. Finally, the system helped him to think about possible relationships as well as probable relationships. This enabled him to develop a more realistic picture of the problem.

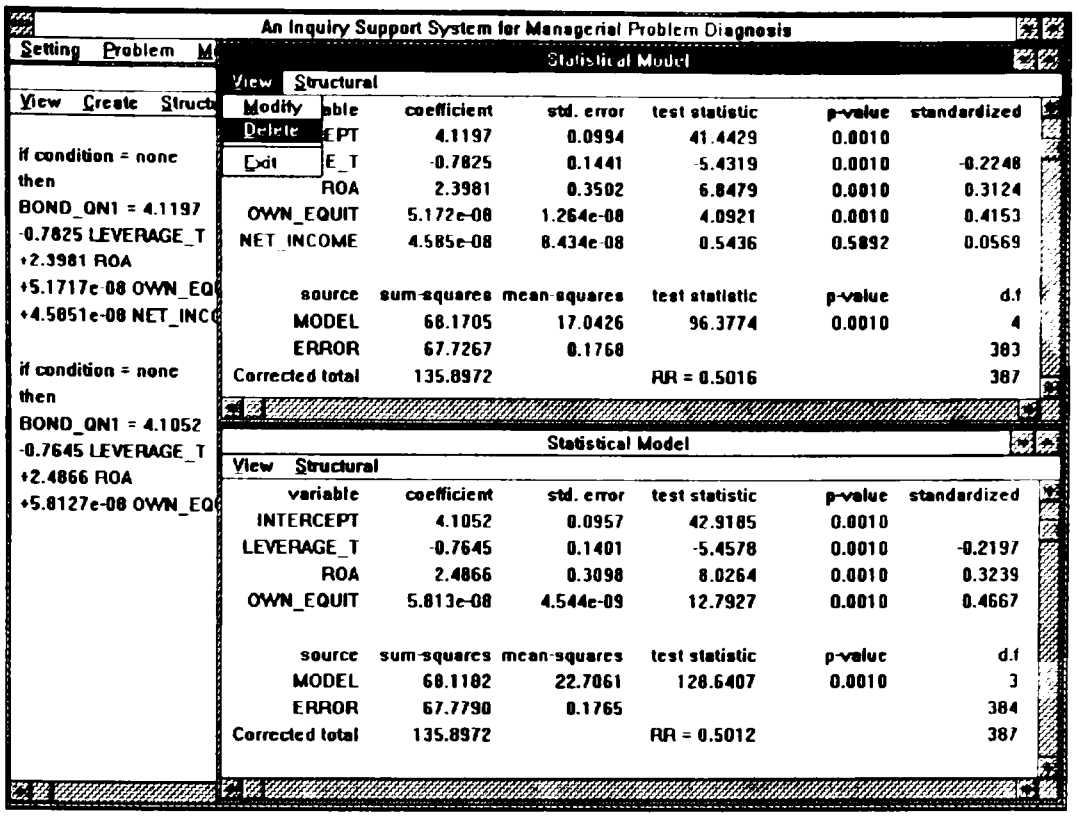


Figure 7.15. Statistical Model Comparison and Revision

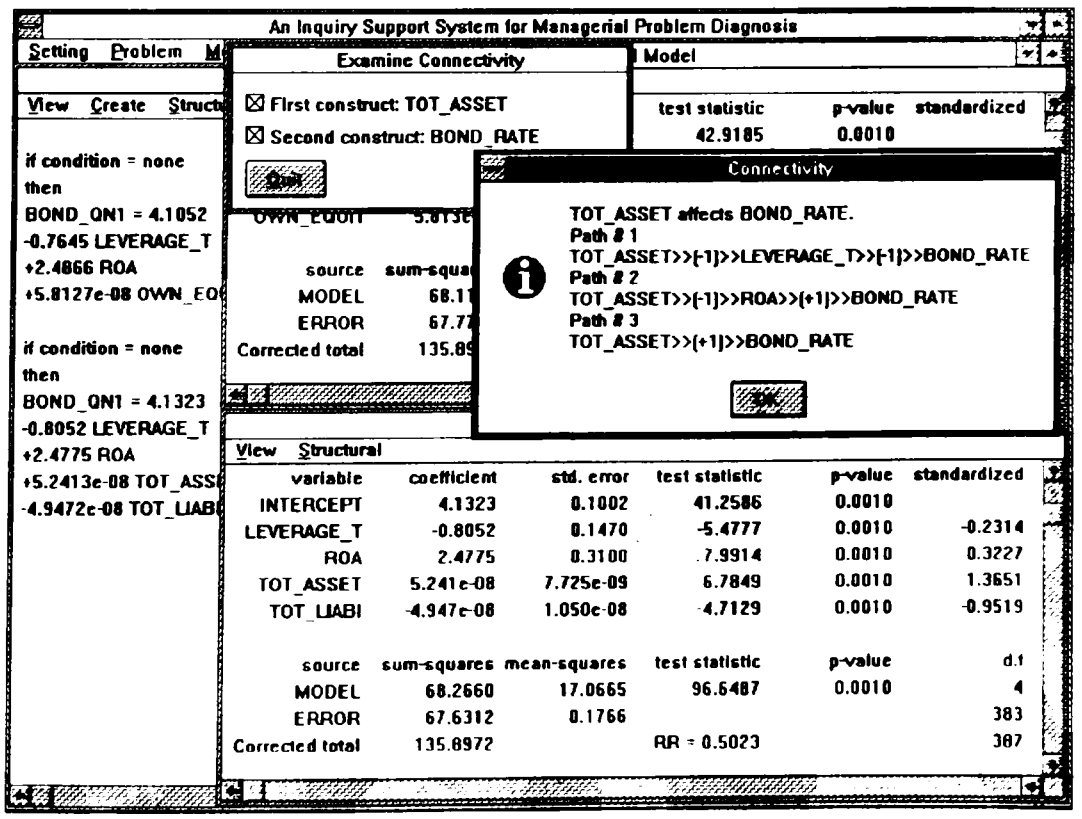


Figure 7.16. Model Evaluation through Variable Substitution and Structural Path Analysis

Alex believes that tot-liabi negatively affects bond-rate in general. He also believes that, in certain cases, tot-liabi may positively affect bond-rate. He is interested in knowing under what conditions tot-liabi can positively affect bond-rate. He suspects that an increase in tot-liabi may greatly hurt bond-rate, when a firm is heavily leveraged; but not necessarily when the ratio of long-liabi to tot-asset is very low. He examines the quartiles of leverage-t by opening the data dictionary (see Figure 7.8). He defines leverage-t less than 0.480 (bottom 20%) as "very low"; and leverage-t greater than 0.754 (top 20%) as "very high."

This time, he uses tot-liabi and own-equit combination to compare their effects. Figure 7.17 shows a positive impact, or more likely no impact, of tot-liabi on bond-qn1 when leverage-t is very low. However, when leverage-t is very high, tot-liabi greatly affects bond-qn1 in the negative direction. He also notes that the impact of own-equit on bond-qn1 becomes greater, when leverage-t is high. He may explore many different problem subspaces by changing the search conditions.

Alex develops an alternative statistical model using the structural model "longdebt.stm" (see Figure 7.11). However, he notices little difference between the two models shown in Figure 7.18.

Now, he selects Create>>Rule>>Induce and asks to induce decision rules for bond-rate. As shown in Figure 7.19, he uses three discretized variables corresponding to the variables shown at the bottom of Figure 7.15. The system induces 28 decision rules using the revised ID3 algorithm. He finds the accuracies of these rules unsatisfactory. He reasons that the rules have low accuracies, because the condition parts of the rules are not complex enough to discriminate different classes of bond-rate. He may have to establish a larger number of subintervals for each variable and/or include additional variables. He adds asset-ds1, liabili-ds1, and income-ds1 to the explanatory variable list.

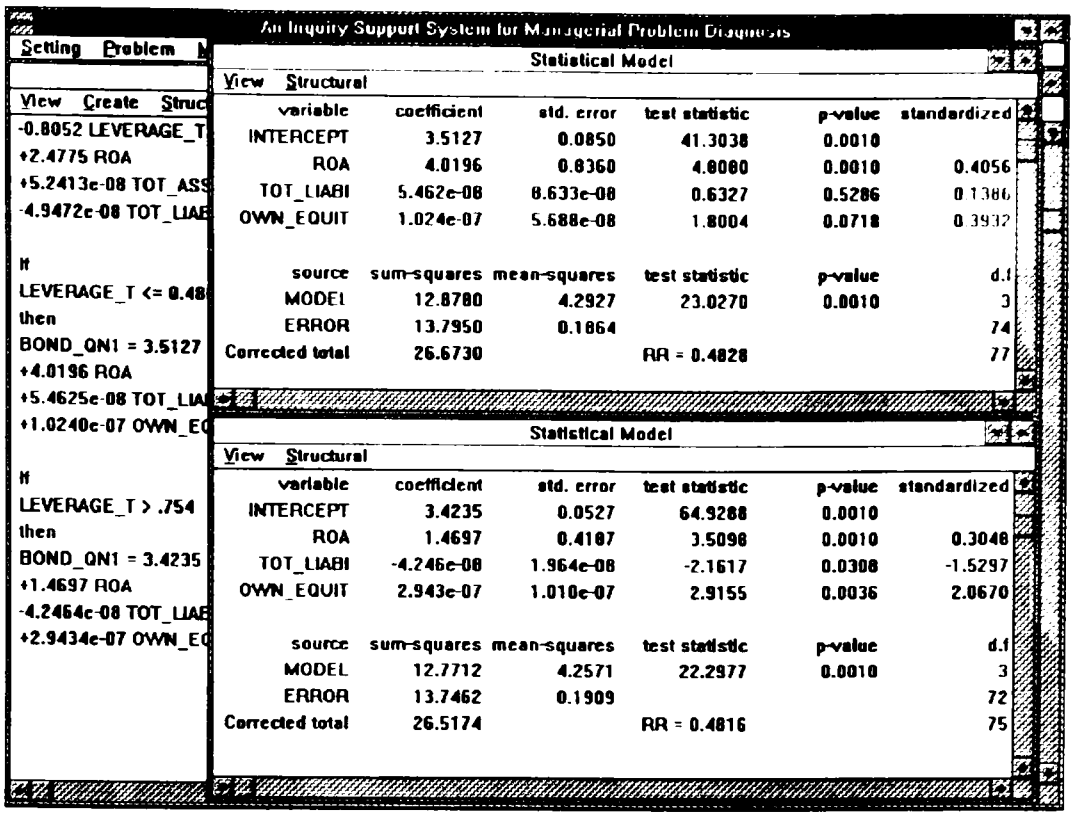


Figure 7.17. Exploration of Problem Subspace and Detection of Local Patterns

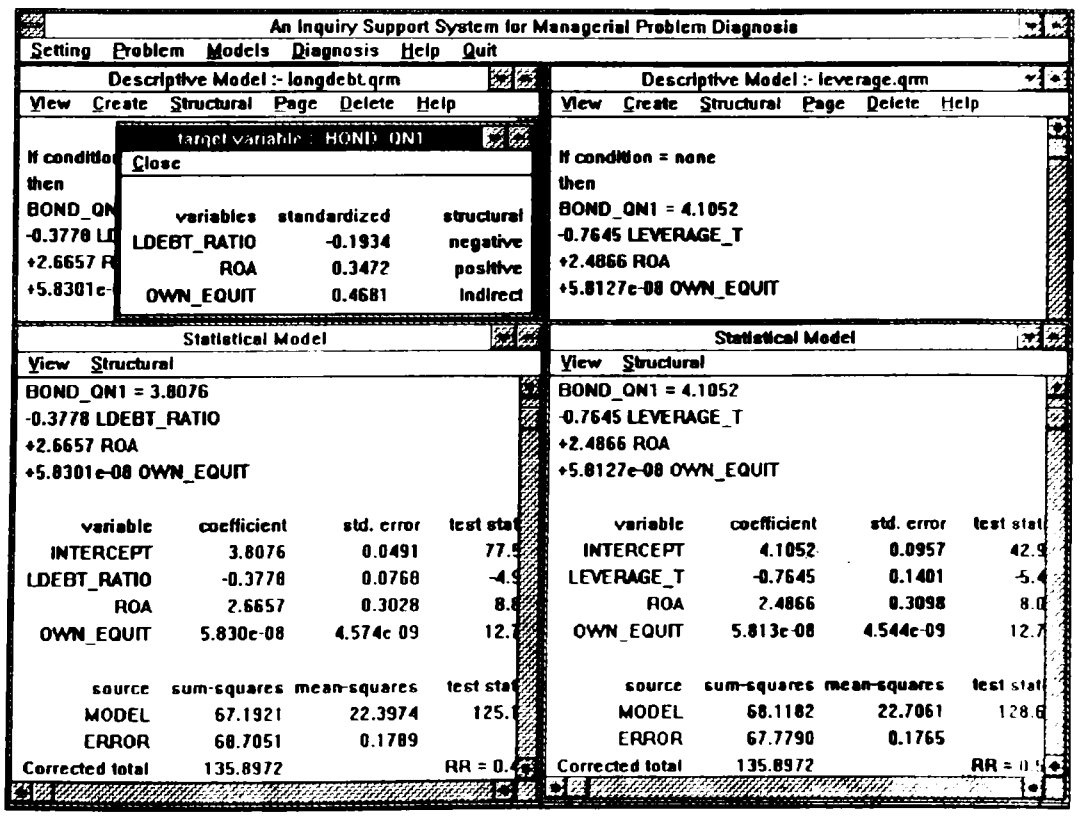


Figure 7.18. Comparison of Alternative Statistical Models

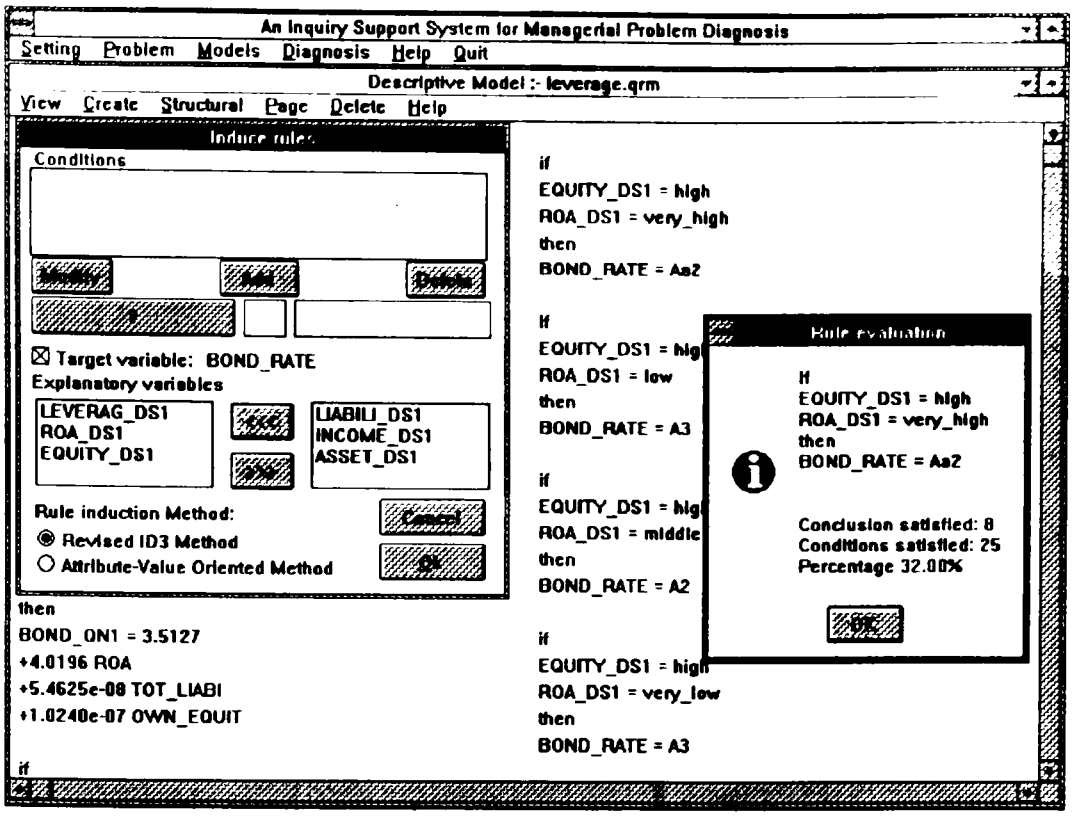


Figure 7.19. System-Driven Rule Induction (Revised ID3)

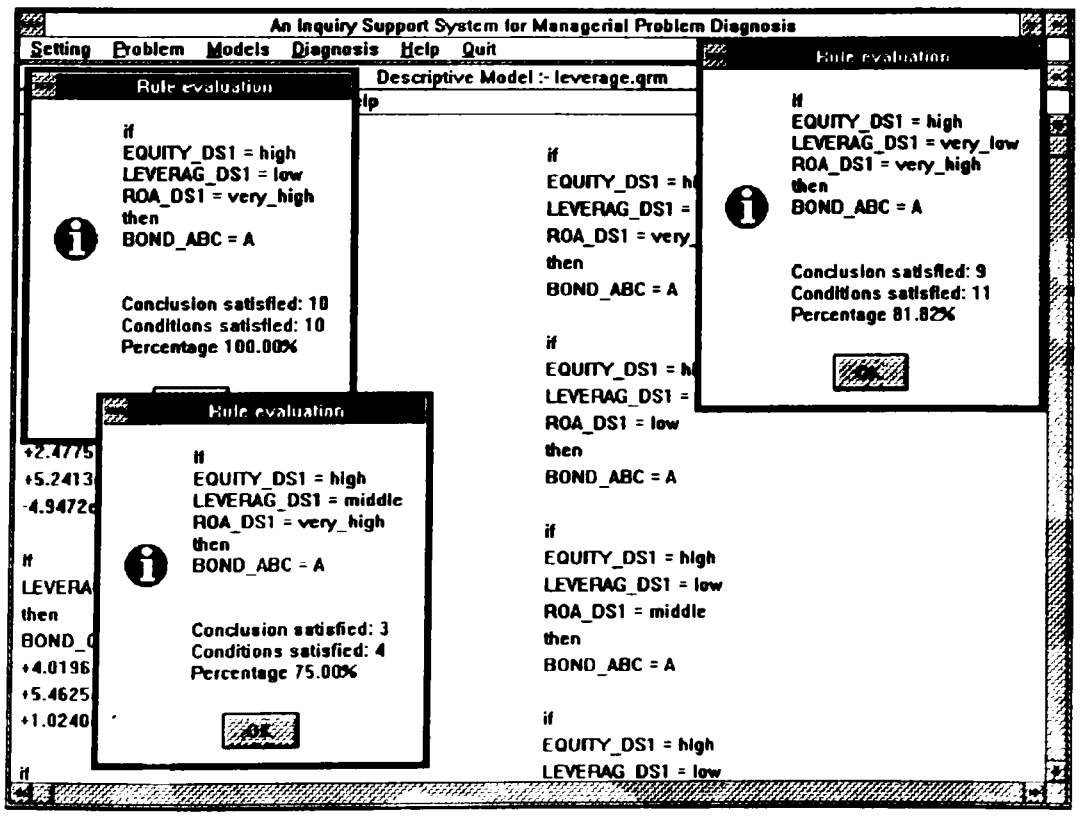


Figure 7.20. Simplified Rule Induction

With these variables, the system generates 41 rules. He notices a slight improvement in accuracy. They are, however, still unsatisfactory. Alex identifies another, less fundamental, reason for the low accuracy. He finds out that the domain of bond-rate consists of 18 different labels by using Variable>>Examine on the data dictionary window. It is difficult to expect that 28 to 41 rules can sufficiently discriminate 18 different values, especially when there is no apparent pattern. Thus, he decides to simply the problem by inducing rules for bond-abc instead of bond-rate. Bond-abc uses only three different labels, "A," "B," and "C."

Alex expects the number of induced rules to be smaller. Surprisingly, the system induces 57 rules. Figure 7.20 shows parts of the rules for bond-abc. He finds the rules much more accurate than those shown in Figure 7.19. While studying the rules, he finds that some of the induced rules are verbose. Using a rule modification dialog (to be shown in Figure 7.21), he notes that dropping leverag-ds1 from the rules shown in Figure 7.20 will make little difference.

Now, he applies the attribute-value oriented rule induction algorithm. The system generates 23 rules. As shown in Figure 7.21, these rules have only one or two attribute-value pairs in the condition part. In contrast, most of the rules in Figure 7.20 had three pairs. Thus, he thinks that the attribute-value oriented rule induction algorithm generates a smaller and simpler rule set (compare the message boxes in Figures 7.20 and 7.21). Their accuracies, however, are not as satisfactory as the rules generated by the revised ID3 (79.54% versus 86.60%).

Although Alex may have to continue modeling to detail his understanding of the problem structure, he now wants to apply the models he has developed. To open problem-specific views, he chooses Setting>>Problem Instance from the main window. He selects a problem instance. Then he chooses Problem >>Problem State and retrieves a problem state view shown in Figure 7.22.

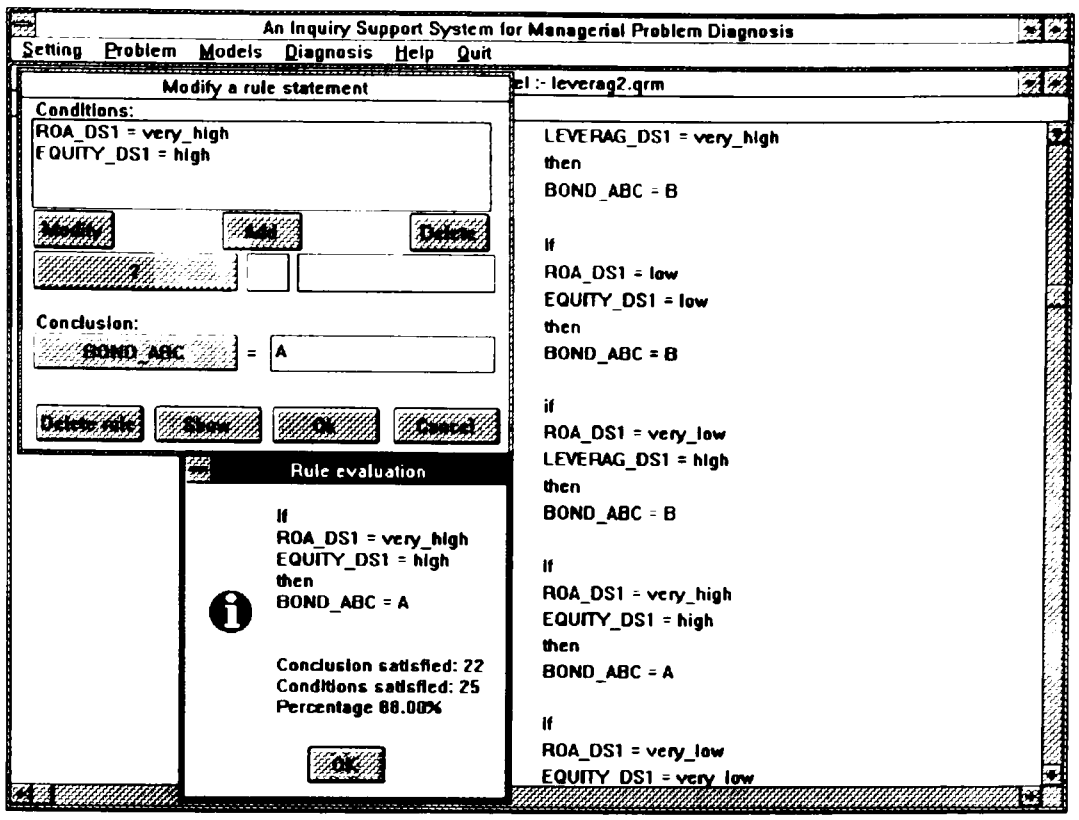


Figure 7.21. System-Driven Rule induction (Attribute-Value Method)

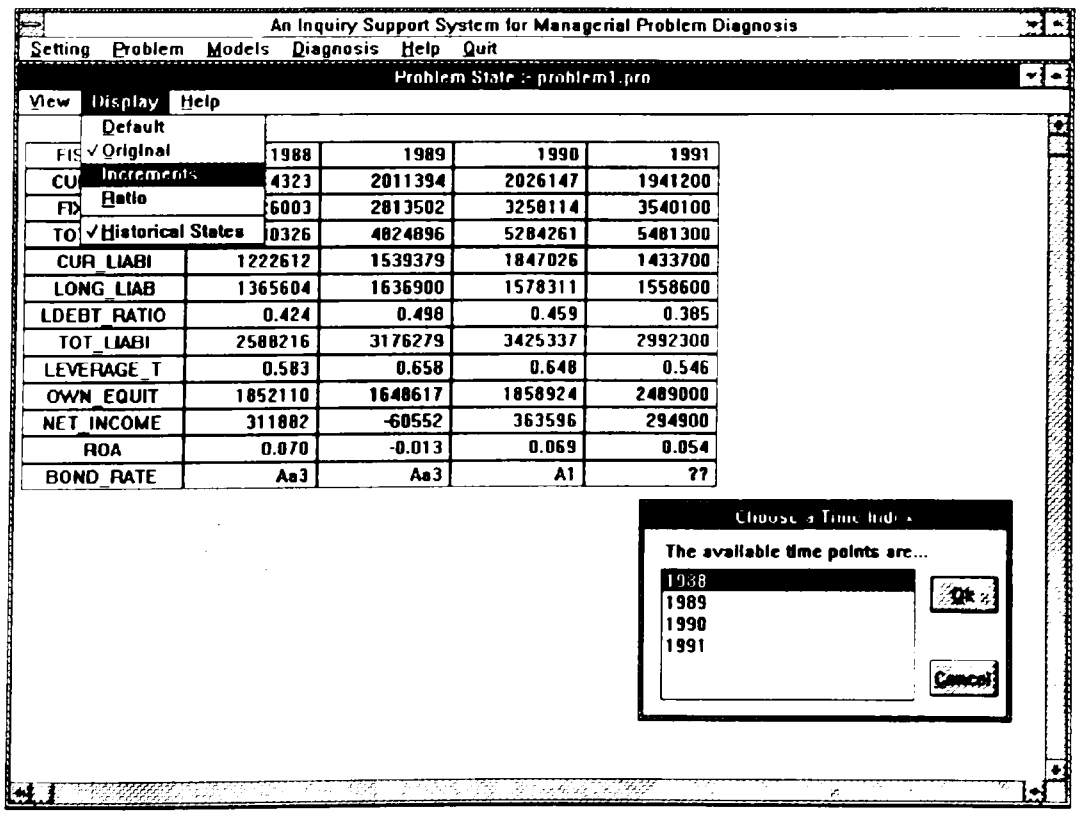


Figure 7.22. A Problem State View Window

He chooses Problem>>Goal State from the main window and creates a goal state view (Figure 7.23). Alex does not have a qualification to establish goals for the firm. However, he wants to know how the firm might have performed for the current period, if it had grown at a constant rate. Thus, he selects Goal Values>>Regression and asks the system to project the goal values. He notes that the system has projected bond-rate (based on the quantified data field, bond-qn1). If there are multiple quantified fields, he has to select one particular field from the data dictionary. He establishes the upper and lower boundaries as 90% and 110% of the projected goal values, respectively.

Alex also wants to compare the firm with other similar firms. He selects Goal Values>>Search. The system opens the dialog box shown at the upper left corner of Figure 7.24. He wants to examine those firms of which ldebt-ratio is between 30% and 60%, and tot-asset is similar. After entering the conditions, he presses the "Display" button. The system opens the dialog box shown at the upper right corner. When Alex chooses a view, the system presents the searched records in the list box shown at the lower left corner of Figure 7.24. He chooses the first record and presses the "Display" button. The system opens an example data view shown at the lower right corner. He presses the "Select the Record as a Goal" button and retrieves the example into the goal state view. He retrieves several comparable performance standards in this fashion.

Alex creates a performance discrepancy view by selecting Problem>> Discrepancy from the main menu. He instructs the system to derive Figure 7.25 from the problem state view in Figure 7.22 and the goal state in Figure 23. From Figure 7.25, he sees that, for the current period, the firm has performed better than historically projected in almost all aspects.

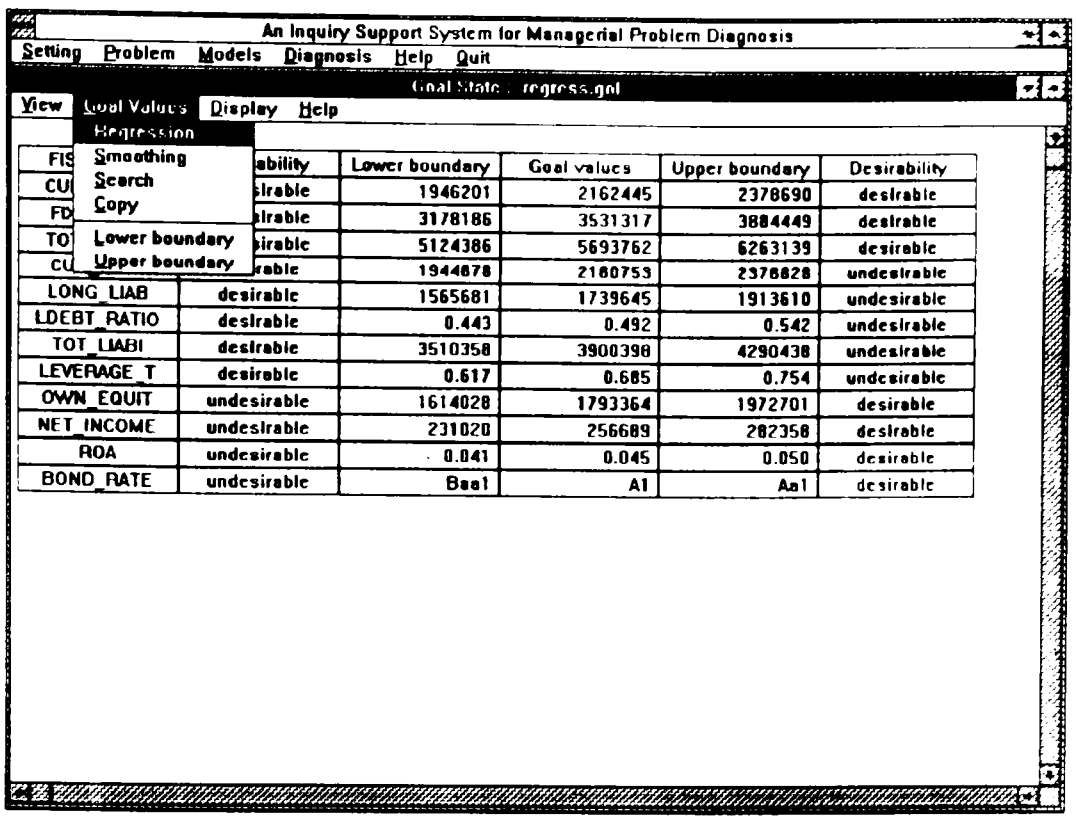


Figure 7.23. A Goal State View Window (Historical Projection)

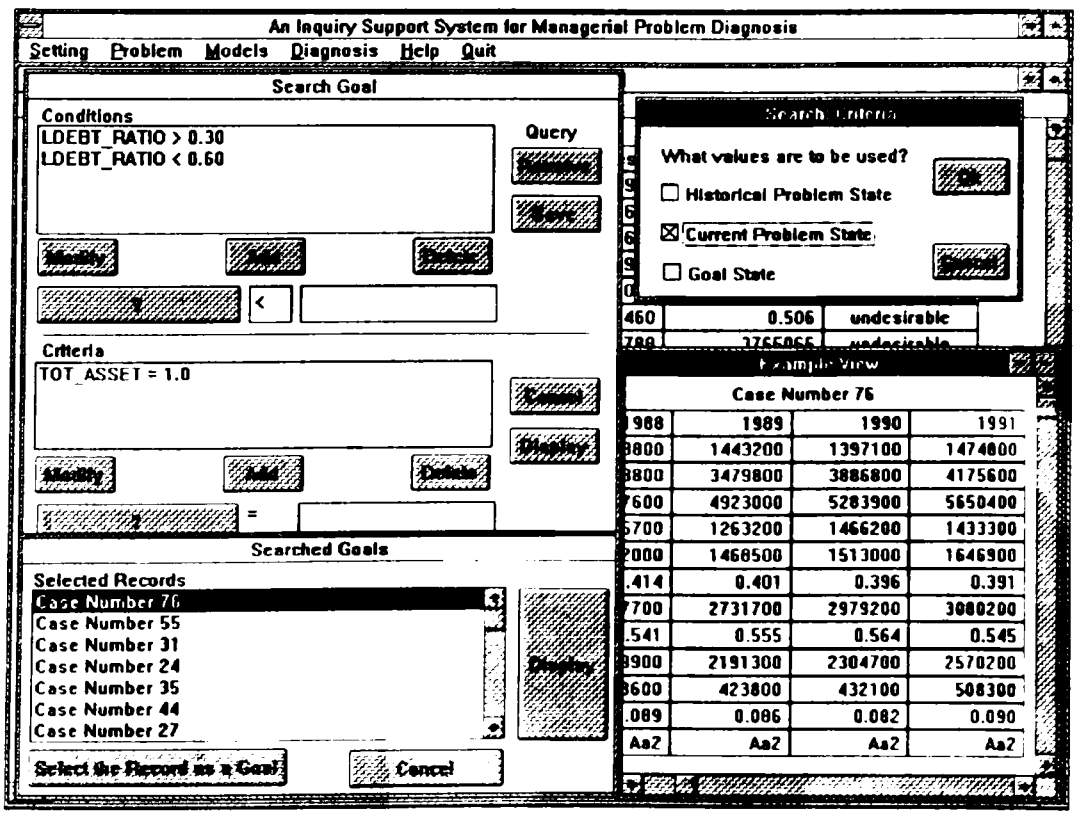


Figure 7.24. Searching Comparable Performance Standards

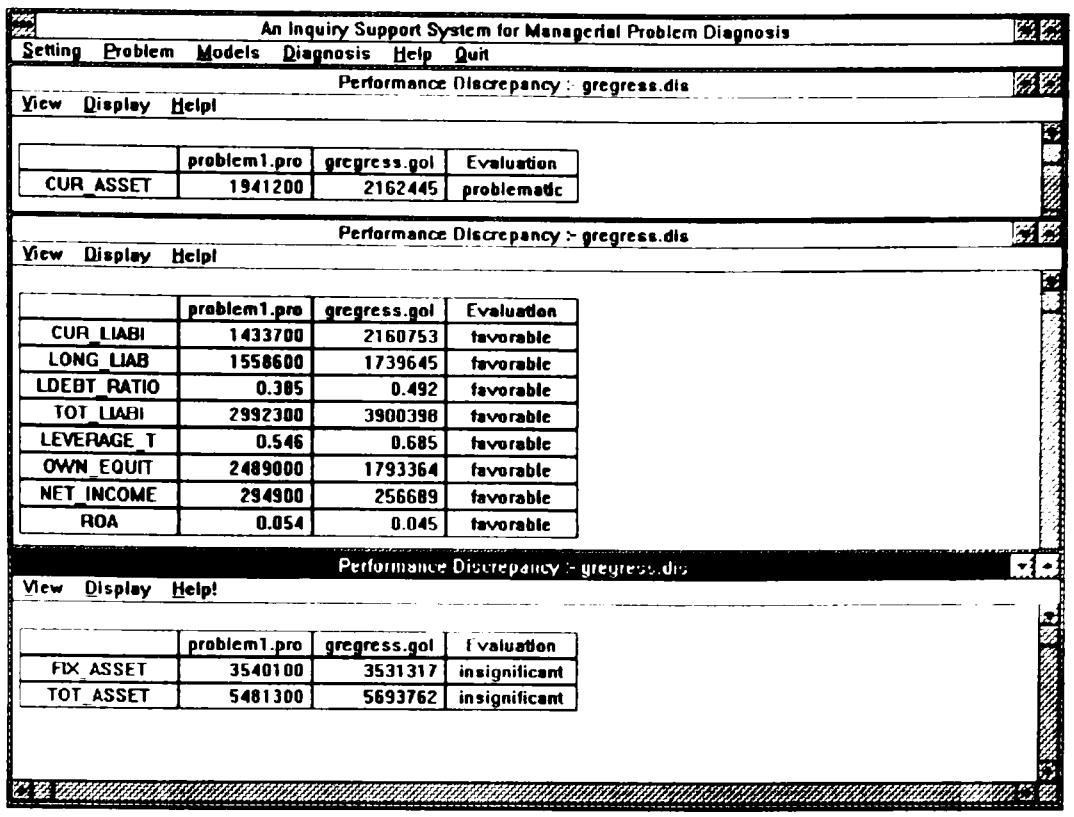


Figure 7.25. A Performance Discrepancy View Window

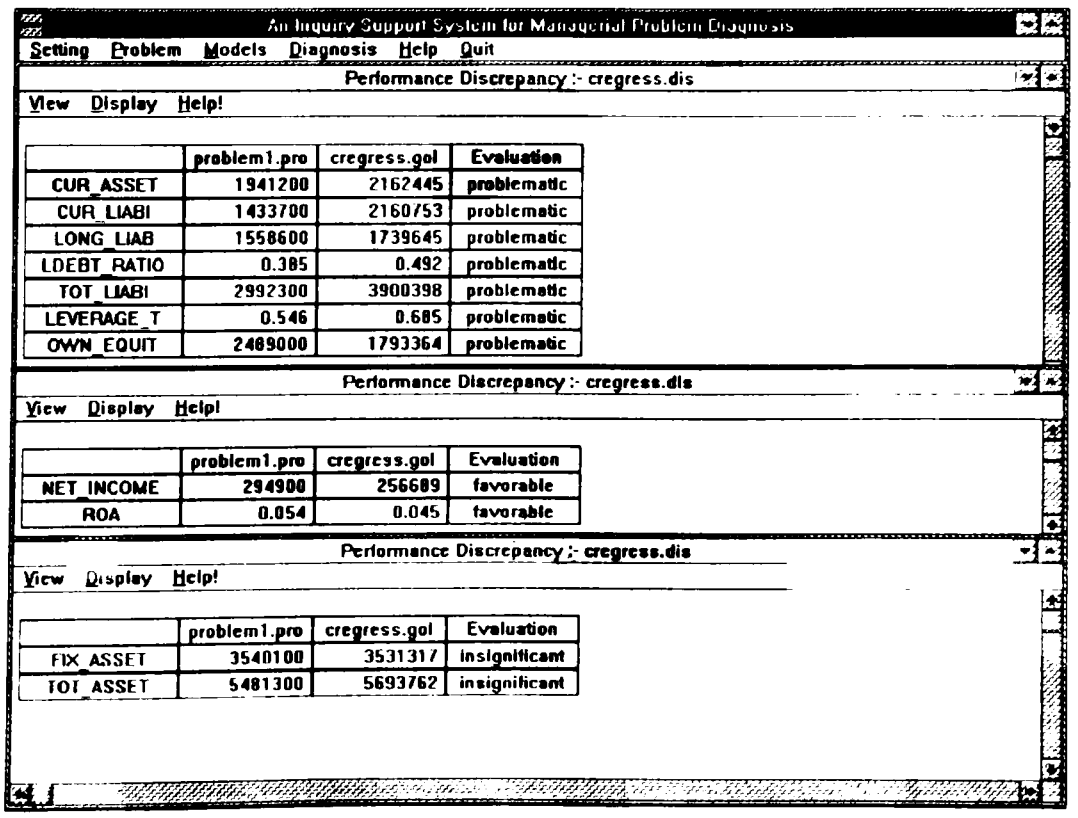


Figure 7.26. A Control-Oriented Problem Monitoring

Figure 7.23 states that a decrease in liabilities is desirable, while an increase in liabilities is undesirable. However, Figures 7.7 and 7.17 indicate such simplification may not be appropriate for certain situations. Alex modifies the goal directions in Figure 7.23 to indicate that any value outside the specified boundaries is undesirable, except for net-income, ROA, and bond-rate. Figure 7.26 shows a discrepancy view derived from this control-oriented monitoring.

One problem Alex notes is that the historical projections in Figure 7.23 are based on the historical problem states in Figure 7.22. Because there were only three previous states, historical projections may not be appropriate. Thus, Alex now compares the firm's performances with those of similar firms. Figure 7.27 shows a performance discrepancy view that compares the problem state in Figure 7.22 and the goal state established in Figure 7.24. By examining several such views, Alex finds out that the firm's overall performances are average. Among the four firms compared, two firms have the "Baa1" bond rating; one has "A2"; and the other one has "Aa2." Thus, Alex considers that the firm's rating is likely to be between "Baa1" and "Aa2."

Finally, Alex accesses Diagnosis > > Alternative. As Figure 7.28 shows, the system estimates the bond rating for the problem state as "A1," and that of the goal state also as "A1." The actual rating for the searched goal was "Aa2." Although the model failed to accurately estimate the bond rating for the goal state, its estimate for the problem state, "A1," was within "Baa1" and "Aa2." Now, Alex believes that the firm's rating is very likely to be between "Baa1" and "Aa2.

Alex presses the right button of the mouse on the cell bond-rate in the "problem1.pro" column. The system pops up the dialog box shown at the right upper corner of Figure 7.29. When he runs the descriptive models for bond-abc, the system answers that the problem state should receive an "A" rating. Because

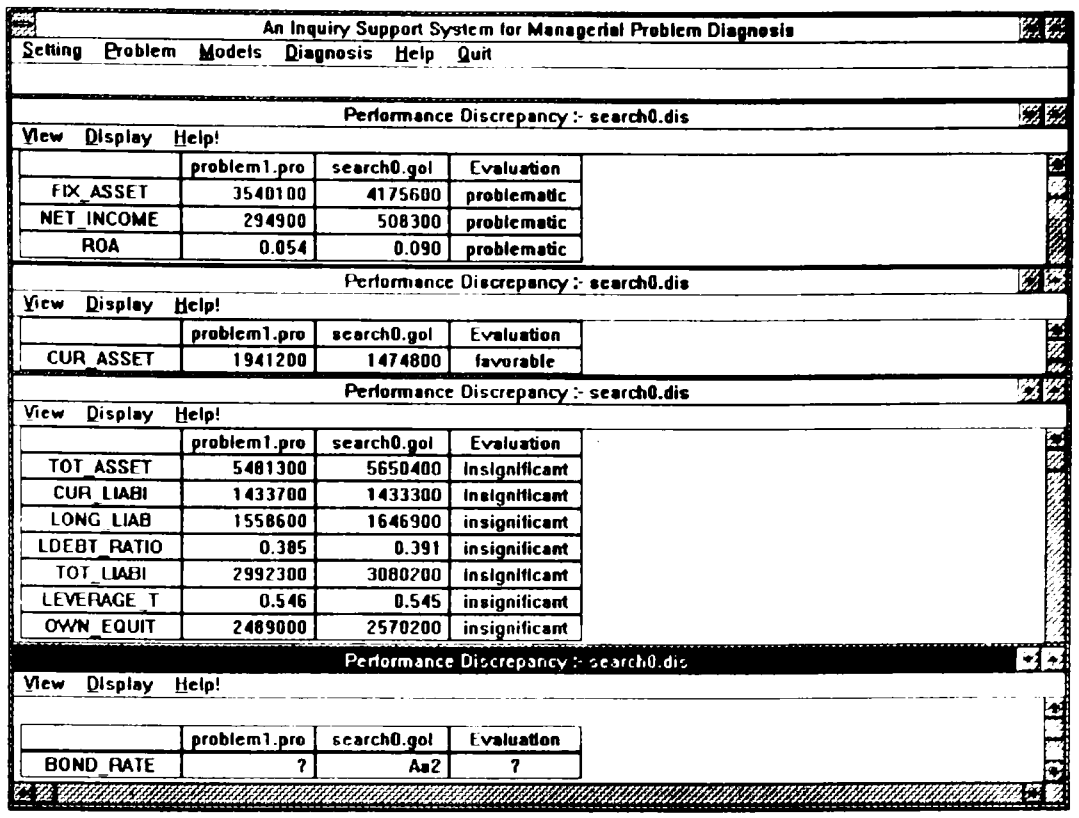


Figure 7.27. Performance Comparison

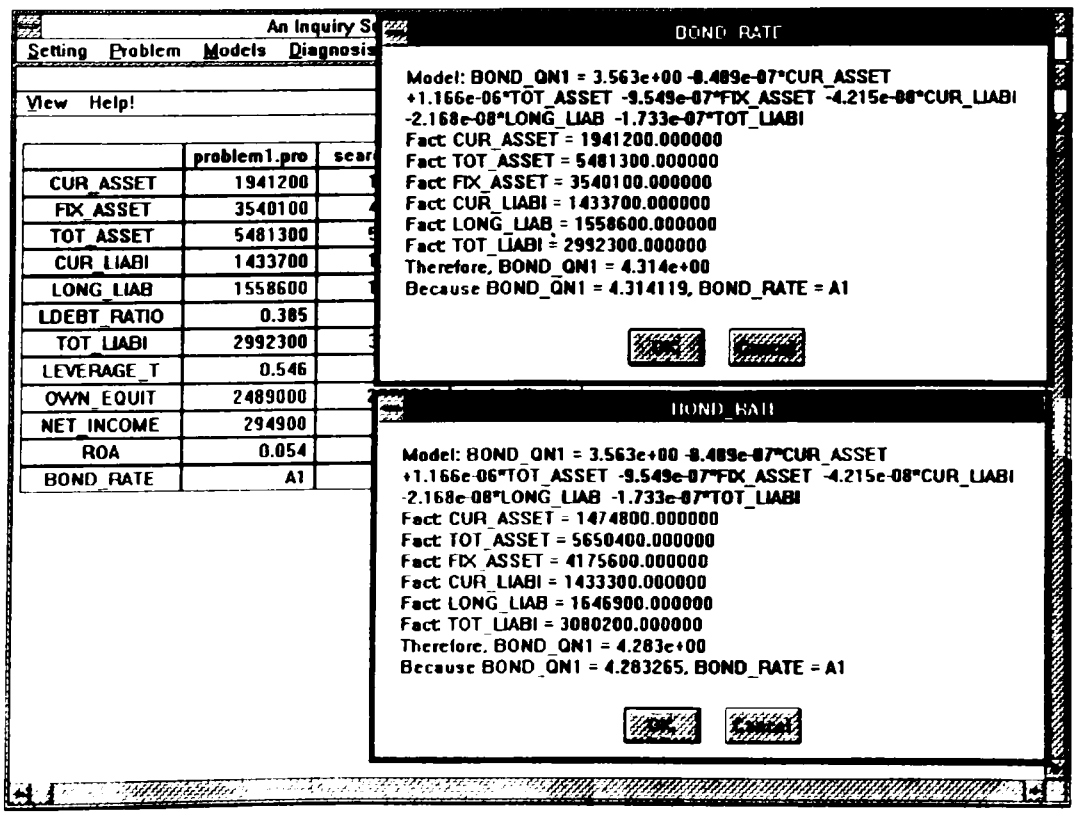


Figure 7.28. An Alternative Diagnosis View

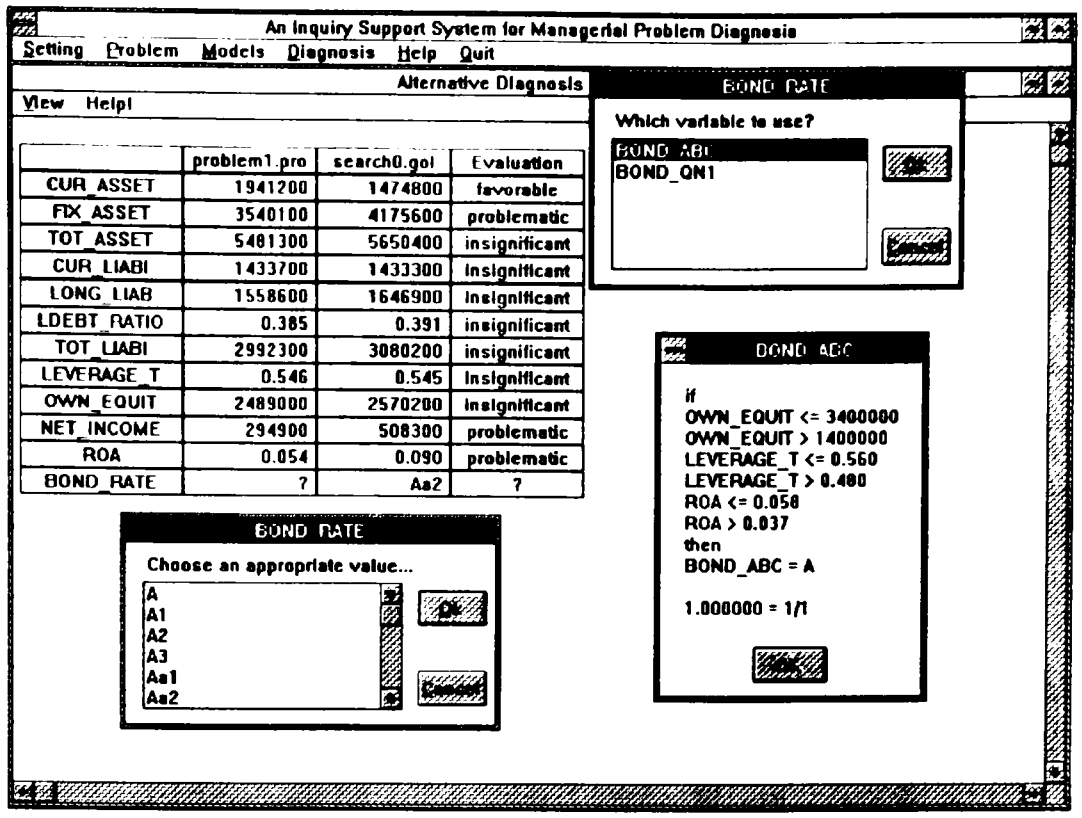


Figure 7.29. An Interactive Diagnosis Generation

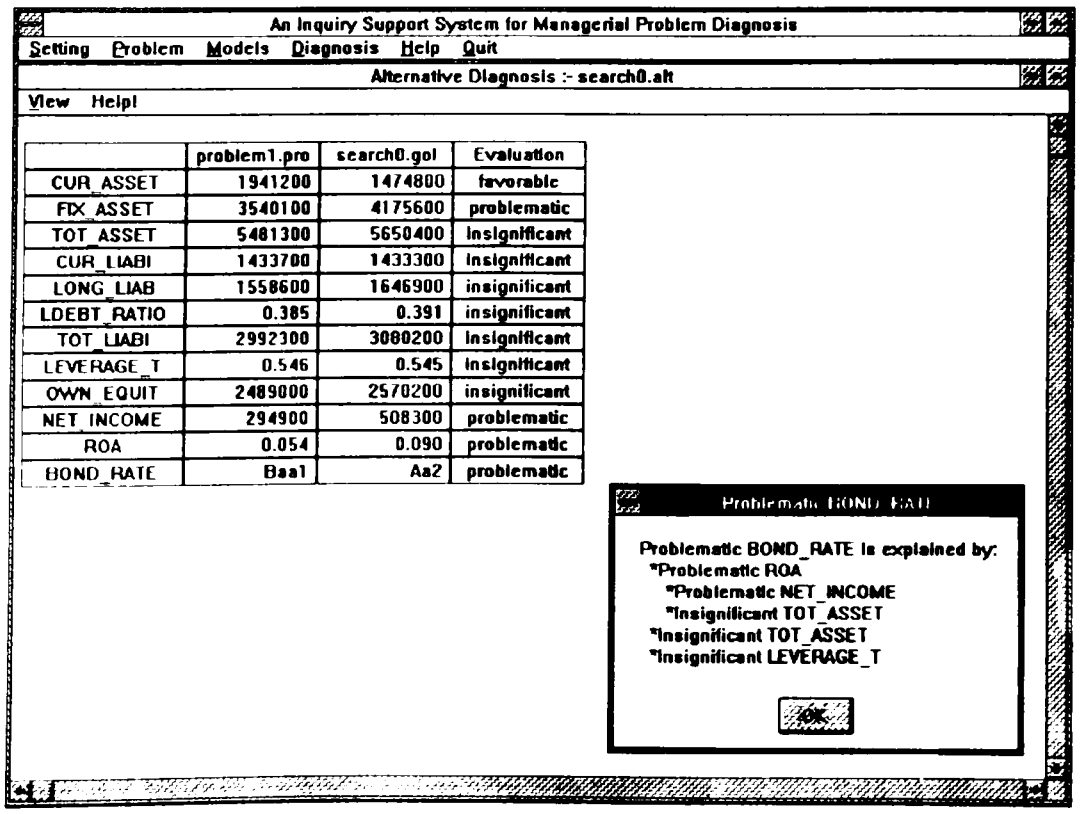


Figure 7.30. Cause Identification

the system cannot determine an exact value for bond-rate, it opens the dialog box shown at the lower left corner and asks Alex to choose an appropriate value.

Alex can change the problem state and goal state values by pressing the left button of the mouse on the screen. The effects of these changes can be examined by pressing the right button of the mouse on the cell of which value must be estimated. When he presses the right button on the cells in the last column, the system opens a message box, shown in Figure 7.30, helping him to identify the causes of the discrepancy.

7.1.3. Evaluation of the System

Alex developed structural, statistical, and rule-based models using the system. The system helped Alex to develop descriptive models systematically and easily based on a structural model. The system also helped Alex to think about, and deal with, possible relationships as well as probable relationships. Alex developed complex statistical models by dissecting the problem space with a set of conditions. The system treated these statistical models as rules and used them in generating diagnoses. The system demonstrated that PDSS can provide a more powerful support by integrating structural, statistical, and rule-based modeling.

Using the prototype system, Alex has established goal state, examined problematic aspects, delineated structural relationships, analyzed the structural relationships, formulated statistical models, cross-validated structural and statistical models, learned from examining system-driven rules, and generated a diagnosis. The system supported various problem diagnosis activities.

Finally, Alex studied the problem interactively with the system. During structural modeling, Alex utilized various dependence statistics. The system developed descriptive models interactively. Moreover, the system helped him by

comparing the signs of the coefficients and performing structural analysis. Alex used two rule induction algorithms and compared their performances. The system managed multiple model sets and allowed Alex to represent alternative model statements. Alex could perform what-if analysis on the alternative diagnosis view windows. Finally, the system helped Alex to refine models (from structural to descriptive, and from unconditional to conditional).

7.2. Prescribed Burning Diagnosis Application

Prescribed burning is an effective tool to improve range management. Prescribed fires can eliminate noxious brush, increase herbage yield and forage availability, and control various diseases (Wright, 1982). Although humans have probably used fire for many thousands of centuries, it is still difficult to precisely predict fire behaviors on an open field. Thus, prescribed burning is usually conducted in two phases: blackline burning and headfire burning. Blacklines are the areas burned around the headfire area to reduce the risk of fire escape.

7.2.1. Background

Andrew is a graduate student majoring in range management. He is taking a course in fire ecology. His homework assignment is to model under what conditions blackline burning is safe. His class was provided with the previous prescribed burn data. Andrew has experiences using structural modeling systems, statistical packages, and rule-based development tools.

7.2.2. Demonstration of the System's Capabilities

Andrew needs to establish an example data set before using the prototype system. He creates a data base file using dBASE III+. Then he opens a utility program shown in Figure 7.31 and imports the dBASE III+ example data file.

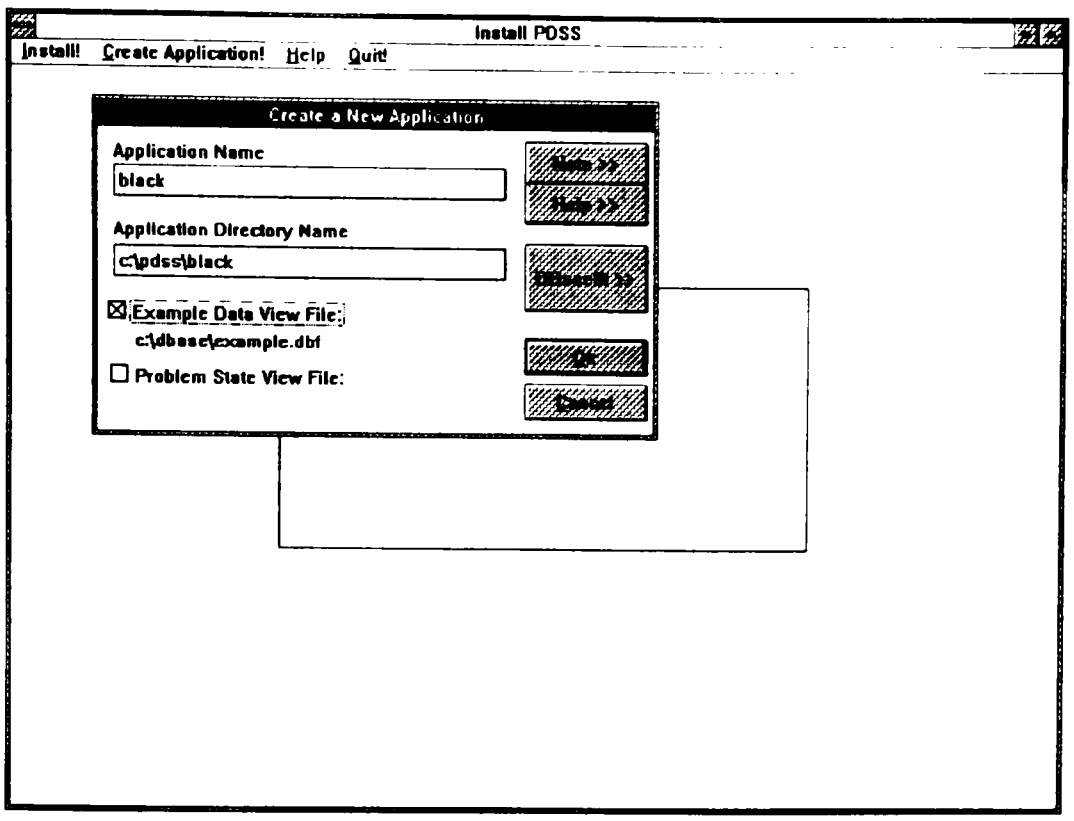


Figure 7.31. A New Application Creation

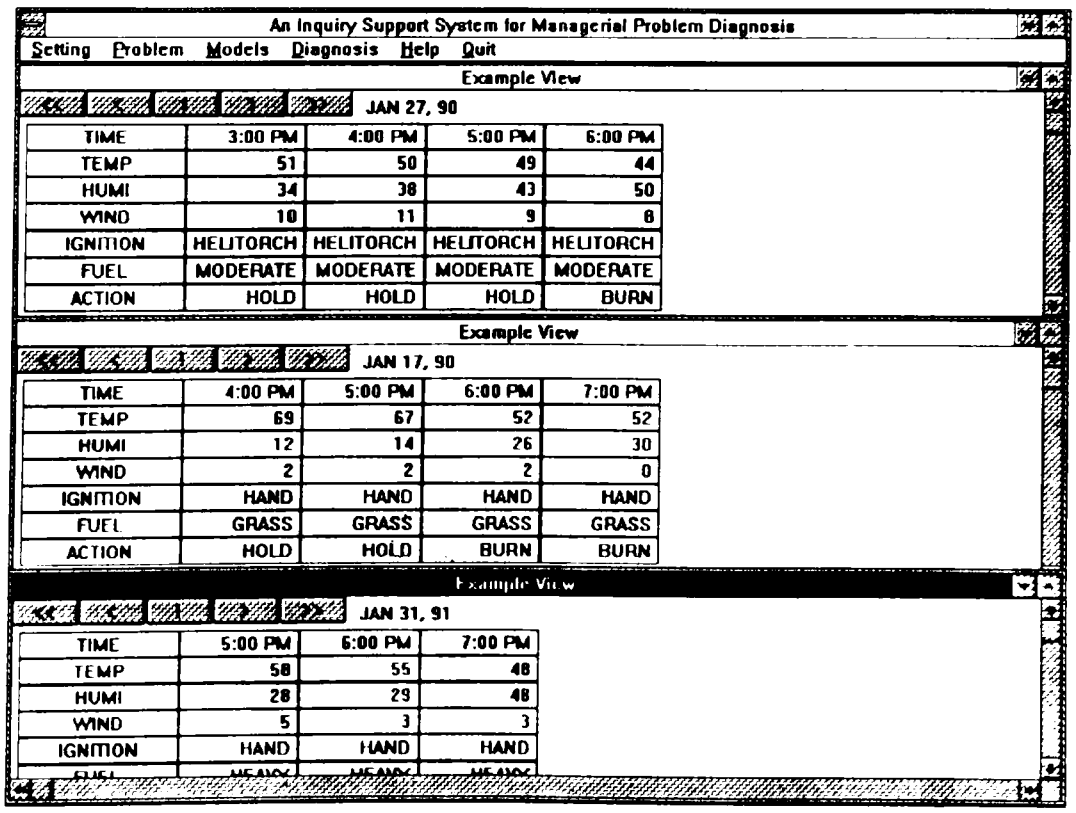


Figure 7.32. Example Data Views for the Blackline Problem

After creating the example data set, he runs the prototype system. He first examines the previous burn data, as shown in Figure 7.32.

He notes that ignition (ignition method), fuel (type of the natural fuel to burn), and action (to burn or not) are categorical variables. He quantifies them in the data dictionary as follows: hand-heli quantifies (i.e., indexes) ignition (hand crew = 0; helicopter = 1); fuel-type quantifies fuel (grass = 0; moderate brush = 1; heavy brush = 2); and hold-burn quantifies action (hold = 0; burn = 1).

He develops a structural model shown in Figure 7.33. He examines how temp (air temperature) affects action. The system indicates that temp affects action both positively and negatively. Andrew remembers using a system called CLSS (Khazanchi, 1991). If CLSS had been used, the system might have told him that temp affects action negatively (because the direct path between them is negative). In fact, it is true that an increase in temp often makes action to be "not to burn," because a fire may get out of control when air temperature is too hot. However, there are cases where an increase in temp makes action to be "burn." For example, if temp is too low, a fire may not start. Under such a condition, an increase in temp will actually improve the chance of effectively conducting prescribed fire. Thus, Andrew realizes that simplification is a useful strategy; however, it may hide a "true" relationship from him. He also notes that the support provided by MIND (Ramaprasad and Poon, 1985), Jung (1990), and Khazanchi (1991) is limited to this stage of problem conceptualization.

Andrew now develops statistical models. He finds that the system is easier to use than other statistical analysis packages. The system, however, supports regression analysis only. The window at the bottom of Figure 6.34 shows that a decision to burn depends on wind speed and ignition method. One problem he notes, however, is that the dependent variable and one of the two independent

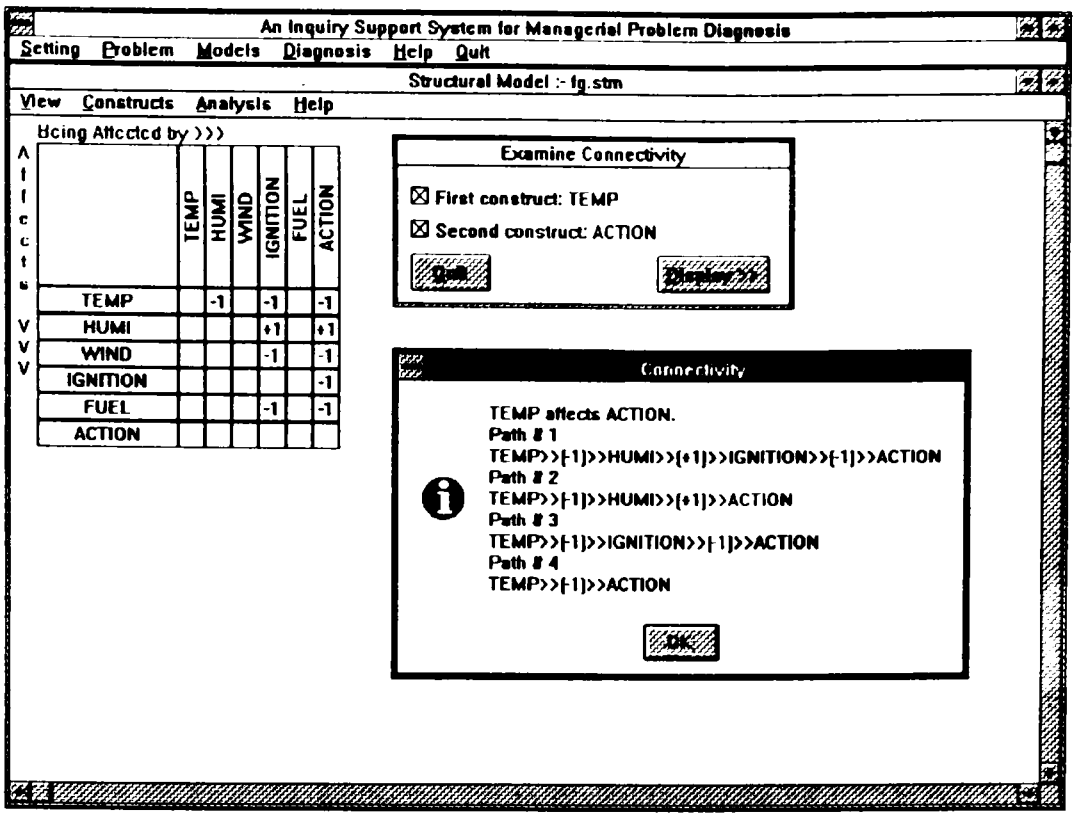


Figure 7.33. A Structural Model for the Blackline Problem

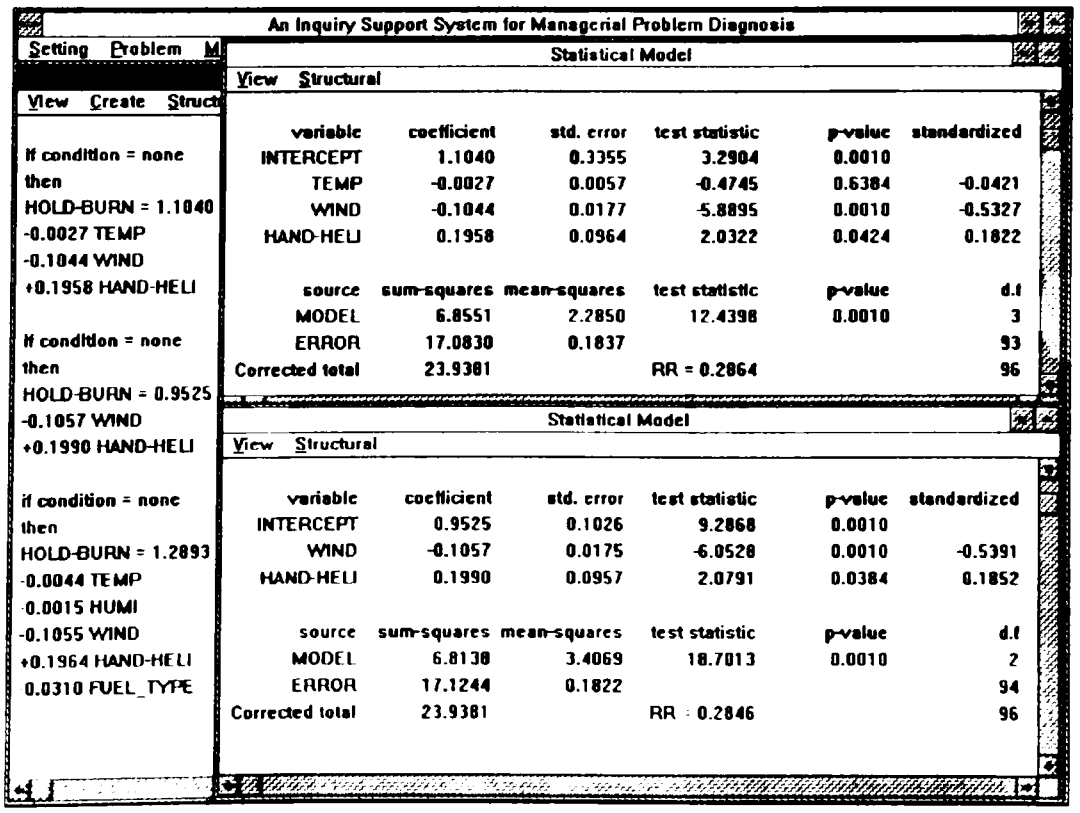


Figure 7.34. Statistical Models for the Blackline Problem

variables are categorical. Thus, it is not likely that the model is precise. Another problem is that regression analysis assumes that the dependent variable is a linear function of the independent variables. This assumption may not be inappropriate for the problem, however. Andrew knows a few PDSS that assume linear relationships among variables (e.g., Ata Mohamed, 1985).

After discretizing the numeric variables, Andrew develops two rule sets. ID3 produces 34 rules for action. He finds these rules very accurate (Figure 7.35). For instance, if the ignition method is hand crew, fuel type is moderate brush, relative humidity is between 30% and 40%, and air temperature is between 50 and 60 degrees in Fahrenheit, then burning is possible only if wind speed is less than 6 miles per hour. He learns such useful rules by examining the rules induced the revised ID3 algorithm. Andrew compares the system with VP-Expert, a commercial rule-based system development shell. Andrew remembers that VP-Expert converts each example case into a rule. The prototype system, however, extracts patterns for him, not just changes the representation format. Now, Andrew applies the attribute-value oriented algorithm similarly. He finds the three rules induced by the algorithm almost useless (Figure 7.36).

At this point, Andrew concludes that neither too high nor too low air temperature is desirable for blackline burning. The same is true for relative humidity. High wind speed will deter the decision to burn. These factors, however, must be considered altogether for each type of natural fuel and each type of fire ignition method. Based on this conceptualization, Andrew summarizes the decision rules (partly shown in Figure 7.35) by drawing diagrams. He drew six diagrams for three type of fuel and two types of ignition method (see Figure 7.37). Andrew presented the diagrams to his class and received a favorable response from his professor.

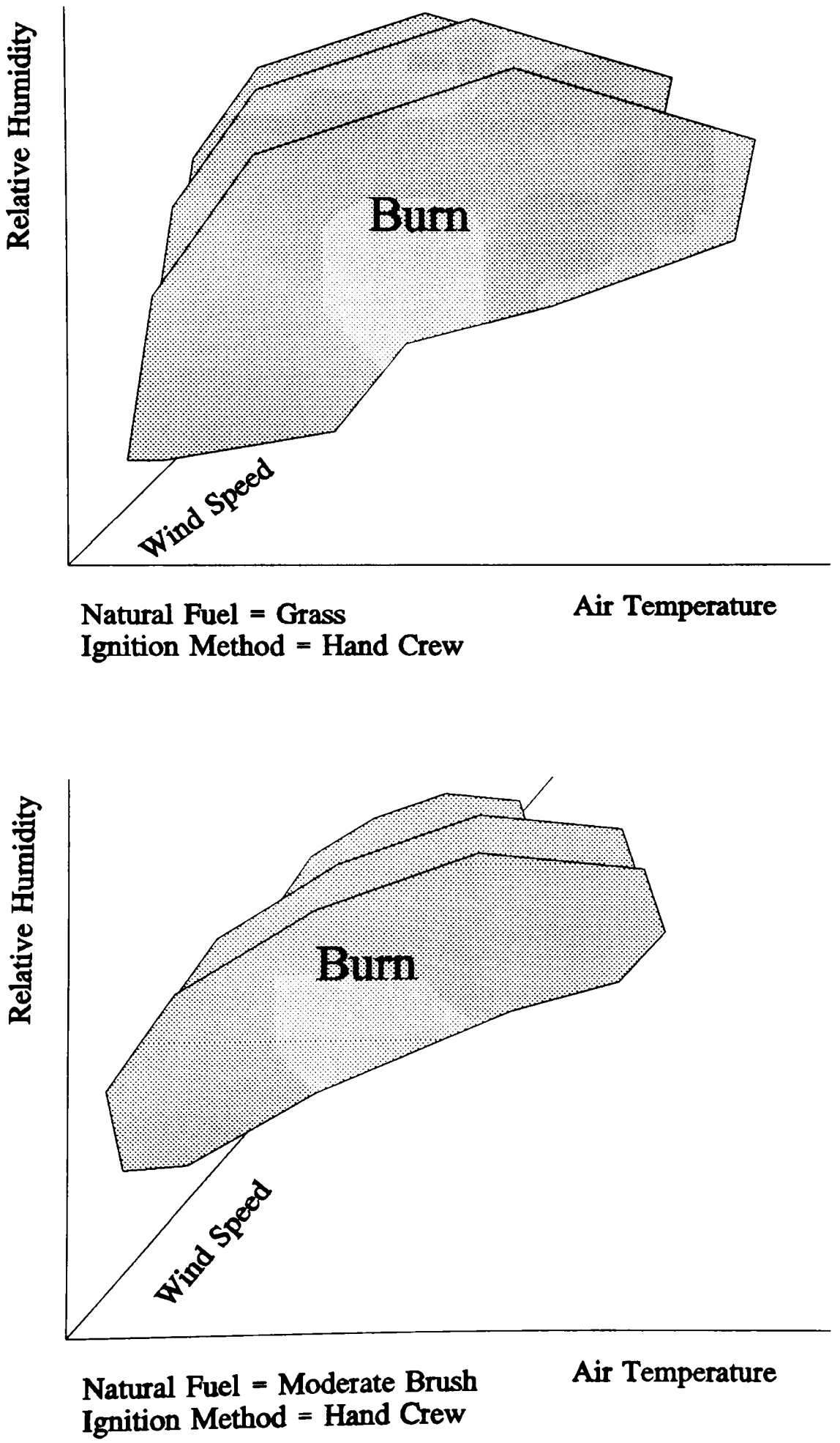


Figure 3.37. User's Conceptual Models of the Blackline Problem

7.3. Summary

This chapter has applied the prototype system to two different problems. The prototype system has demonstrated that PDSS can interactively support structural, statistical, and rule-based modeling. The benefits of combining these modeling approaches in a single system is that: (1) users can systematically detail their understanding of a problem by building descriptive models upon a structural model; (2) the system can simplify user's model structure specification routine; (3) users can check the appropriateness of statistical models by comparing them with a structural model; and (4) users may explore complex relationships by using rules and conditional statistical models.

The prototype system also has demonstrated that PDSS can support the entire problem diagnosis process. The prototype system has helped the users to project and search goal states, locate problematic aspects, structure a problem, analyze structural relationships, build and validate statistical models, develop and represent decision rules, generate diagnoses, and identify causal variables. Thus, the system could support iterative, retroductive problem diagnosis process.

The prototype system has demonstrated that a tighter cooperation can be achieved between the users and the system. The system provided interactive decision support for various problem diagnosis activities, such as goal searching, structural model development, statistical modeling, rule formulation, and diagnosis generation. Moreover, the system provided several mechanisms to ensure the quality of the problem diagnosis process. For instance, the prototype system utilized two induction algorithms. Each of the algorithms was more useful for a problem, but not for the other problem. The system managed multiple views, cross-validated models, and provided a mechanism where users can refine their conceptualization of the problem.

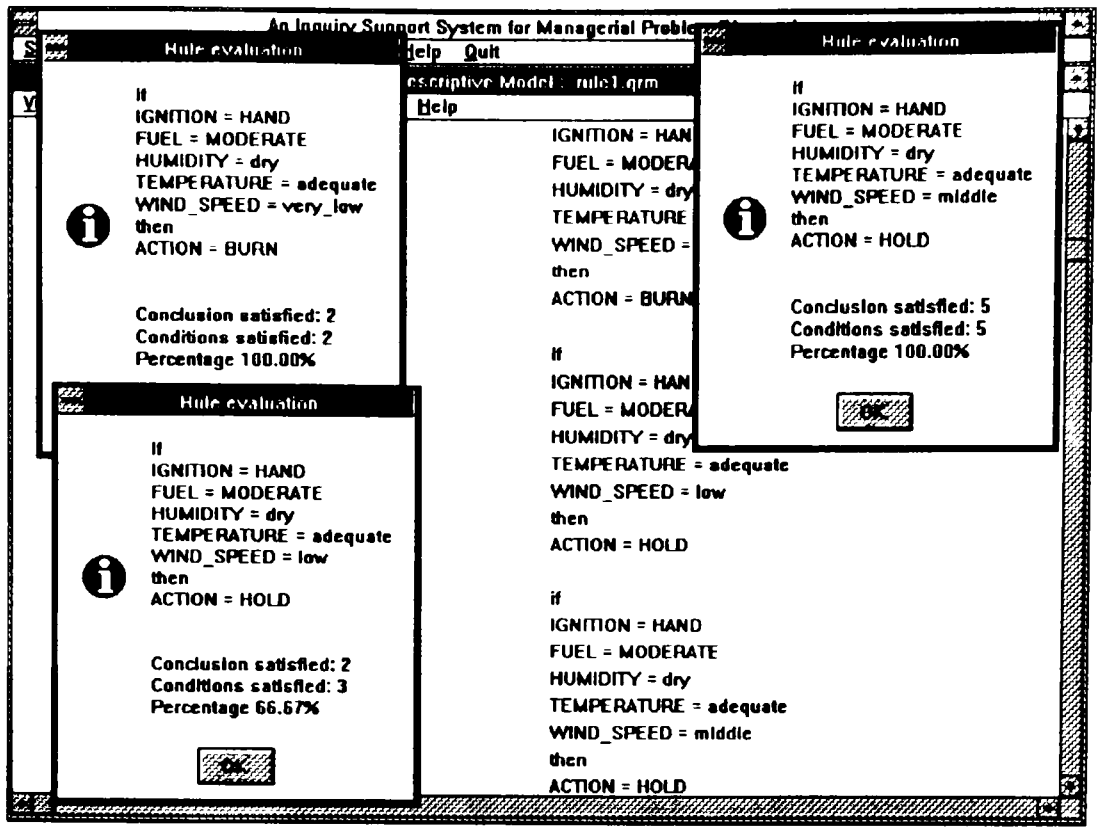


Figure 7.35. Rule Induction for the Blackline Problem (Revised ID3)

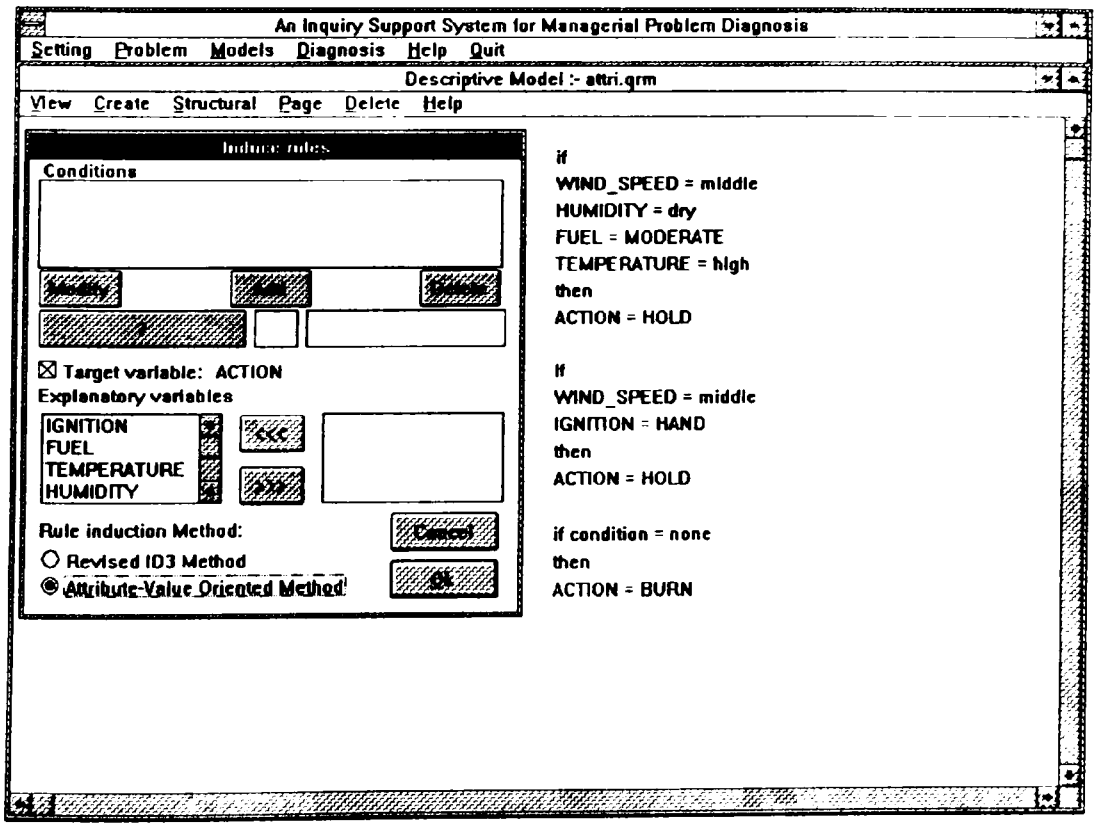


Figure 7.36. Rule Induction for the Blackline Problem (Attribute-Value Method)

CHAPTER VIII

EVALUATION OF THE RESEARCH

The purpose of this chapter is to evaluate the validity of the research. The research is evaluated with respect to four types of validity: face validity, internal validity, construct validity, and external validity.

Section 8.1 examines the face validity of the research. Face validity deals with the issue of whether the research is intuitively valid (makes sense) on its outset (at its face value). Face validity is subjective in nature. To improve objectivity, face validity is examined in light of other literature.

Section 8.2 examines internal validity. Internal validity addresses whether the research has examined what it was supposed to examine; i.e., whether the prototype system followed the symbol-level specifications; whether the symbol-level design followed the conceptual-level specifications; and whether the conceptual-level design is based on the conceptual framework. If the system implementation is consistent with the symbol-level design and the conceptual-level design, the prototype system demonstrates that the design concepts formulated within the conceptual framework are, in fact, viable.

Section 8.3 examines construct validity. Construct validity is concerned with the plausibility of the elements and relationships in the theory; i.e., how valid the model structure is. The construct validity of the design is assessed through comparing the structure and functions of the prototype system with those of the previous PDSS.

Section 8.4 discusses the external validity of the research; i.e., generalizability and effectiveness. The discussion includes the generalizability of the system and design concepts, and the validity of the conceptual framework.

8.1. Face Validity: Overall Assessment of the Research

The major impetus of this research was the recognition that the previous PDSS suffer several shortcomings. Then, is it worth developing a PDSS that overcomes the shortcomings? To answer this question, the first issue to be addressed is whether problem diagnosis is a critical aspect of decision making. If it is, an improvement in PDSS may have a relatively great impact on the overall decision making process.

There is a general consensus in the management literature that problem diagnosis is a critical decision making phase (Ackoff, 1974; Mintzberg et al., 1976; Schwenk and Thomas, 1983; Yadav and Korukonda, 1985; Adams et al., 1990). An exemplary statement emphasizing the importance of problem diagnosis is:

> knowledge about relationships between events is a critical aspect of human knowledge. Knowing whether events are related, and how strongly they are related, enables individuals to explain the past, control the present, and predict the future. (Crocker, 1981, p. 272)

However, it is also possible to find some exceptional statements undermining the importance of problem diagnosis. The following statement contains perhaps the harshest criticism on emphasizing problem diagnosis:

> diagnosis is a discretionary activity irrelevant to the solving of many problems. ... situations where a cause is not continuing, e.g., a tornado-flattened factory; matters of evaluative judgment, e.g., which house should we buy; or of alternative generation, e.g., what should we name this new product: none of these requires causal diagnosis. (Smith, 1989, p. 965)

At least four points can be made to rebut the criticism. First, diagnosis may be a discretionary activity for the structured problems but not for the unstructured or semi-structured problems (Mintzberg et al., 1976). Second, the identification of immediate causes is not the single most important activity in problem diagnosis. What is perhaps more critical is the identification of the causal structure. For example, decision makers must identify those factors to be

considered in purchasing a house, in naming a product, or in building a factory that can better sustain tornados. Third, it appears that Smith (1989) defined "diagnosis" narrowly to be the activity of identifying the immediate causes. Even so, the author admits that "diagnosis is critical for the solution of most problems that requires the generation of alternatives" (Smith, 1990, p. 631). Finally, there is an obvious inconsistency between the last quoted statement and "diagnosis is a discretionary activity irrelevant to the solving of ... problems ... of alternative generation" (Smith, 1989, p. 965).

There is little doubt that problem diagnosis is an important phase of decision making. The simple reason is that individuals may not be able to solve a problem effectively without first understanding the problem. Problem diagnosis is also a difficult process (Anderson and Janson, 1979; Lyles and Mitroff, 1980). The difficulty arises in that individuals have cognitive limitations (Miller, 1956; Tversky and Kahneman, 1974; Argyris, 1976; Hogarth, 1980; Sage, 1981) and that problem diagnosis involves a variety of complex tasks that can quite possibly overload the cognitive processing (Ackoff and Emery, 1972; Taylor, 1975; Einhorn and Hogarth, 1986; Lyles, 1987; Ramakrishna and Brightman, 1986). Adams et al. (1990) reported that DSS users perceive problem identification to be the most critical and difficult decision process that requires the most thought.

Given that problem diagnosis is a critical and difficult process, the next issue to be addressed is whether a support system can improve the process. A fundamental premise of MIS/DSS research is that a well-designed support system, when used appropriately, can improve the decision maker's performances (Mason and Mitroff, 1973; Sprague, 1980). In other words, a poorly-designed support system or an inappropriate use of the system may result in no effect or even a negative effect on the decision makers' performances. In fact, Sharda et al.

(1988) reviewed twelve DSS laboratory studies and found seven studies reporting significant improvement, four studies reporting no effects, and one study reporting a negative impact of the DSS.

Then, the question becomes what attributes a support system must have to promise, on its own part, its effectiveness. The conceptual framework of this research established that there are at least three essential elements of DSS: knowledge base, problem processing, and interaction capabilities. This conceptualization is consistent of an object-oriented system design (Howard, 1988), IDEF methodology (see Bravoco and Yadav, 1985), knowledge-based system design (Newell, 1982), and DSS framework (Bonczek et al., 1980).

First, concerning the knowledge base structure, many of the previous PDSS relied on the structural modeling construct. Although structural models have been often used in the areas of foreign policy analysis, environmental psychology, and organizational theory (see Ford and Hegarty, 1984), the modeling construct has limited representational capabilities; e.g., too simple to model even the rat's behavior in solving a maze problem. As Axelrod points out, "a cognitive map is a specific way of representing a person's assertion about some limited domain" (Axelrod, 1976, p. 55, emphasis added).

The structural model is a basic mechanism to represent objects and their relationships. This research combined structural modeling with statistical modeling and rule-based modeling. Thus, this research extended the structural modeling in three major ways. First, the relationships are no longer limited to pairwise ones. Second, the relationships between the variables are no longer limited to {-1, 0, +1}. Third, it is possible to represent, in a systematic way, the relationships among events as well as those among variables. Consequently, it is possible to model complex relationships observed in the management domain.

Second, concerning the problem processing capabilities, the previous PDSS had focused on supporting different problem diagnosis activities. They devoted little effort to studying the overall problem diagnosis process and designing a system to support the entire range of the diagnosis activities. In other words, previous PDSS have been developed in a fragmented fashion. An argument can be made that the fragmented PDSS, as a group, can support the overall problem diagnosis process. The matter is not that simple, however. First of all, interconnecting independent systems is not an easy task. More importantly, a system is a set of elements that work together; i.e., the whole is greater than the sum of its part (Ackoff, 1971; Checkland, 1976). A simple collection of the systems may not support the interactive, retroductive problem diagnosis process.

The process-oriented decision support has been emphasized by many theoretical DSS researchers (Keen and Scott Morton, 1978; Sprague and Carlson, 1982; Parker and Al-Utaibi, 1986; Weber, 1986; Elam and Konsynski, 1987; Weber and Konsynski, 1987; Nunamaker et al., 1988; Finlay and Martin, 1989; Adams et al., 1990). However, process-oriented DSS have been seldom designed and developed, because they require extensive research efforts (because of the wide spectrum of activities to be supported). In terms of the research scope, the face validity of this research is more credible than the previous PDSS.

Finally, concerning the user-system interaction, one common belief is that the support system should automate the decision process. Automation is certainly an effective way to handle structured problems. Under the DSS context, however, complete system-driven processing is often impossible and impractical. Semi-structured or unstructured problems involve uncertainties and ambiguities that require human judgment. On the other hand, a heavily user-driven system may under-utilize the system's capabilities. For example, a program that delivered

simple heuristics to the users had no different effect on the user's performance than a paper-and-pencil medium (Cats-Baril and Huber, 1987).

The point is that an effective PDSS should combine the strengths of two systems to compensate for the limitations of individual systems. Unfortunately, only a very few previous PDSS support interactive problem diagnosis. Because this research emphasizes dynamic user-system interactions throughout the problem diagnosis process, it has better face validity than previous PDSS that support either user-driven or system-driven problem diagnosis.

In summary, this section examined the overall significance of the research. First, it was argued that the face value of designing a PDSS is significant, because the problem diagnosis is a critical and difficult aspect of decision making. Then, it was suggested that a support system, on its own part, must have certain properties to improve the diagnosis process. The three critical elements are knowledge base structure, problem processing, and interaction capabilities. This conceptualization is intuitively sound and consistent with other information system design concepts. It has been shown that this research improved PDSS in all of the three aspects. Thus, this research has a significant face validity.

8.2. Internal Validity: Evaluation of the Design Process

In empirical research, internal validity is concerned with whether the experiment effects have been measured without serious contamination. Internal validity here refers to whether the design process has been carried out in an unbiased fashion. An information system design is usually carried out on three levels; i.e., system requirement definition, functional specification, and programming levels (Langefors, 1973; Ross and Schoman, 1977).

8.2.1. System Requirement Definitions

The framework development in this research encompassed the system requirement definitions. As discussed in the previous section, three aspects of requirements were examined. The knowledge base structure requirements were established through illustrating the structural model's limitations and arguing why statistical and rule-based modeling constructs are necessary for PDSS. The processing requirements are based on the diagnosis activities identified from the problem diagnosis literature. The interaction requirements stated that a support system must allow users to fully utilize TELL and ASK operations. Interactive problem solving is a basic premise of DSS research. In addition, user-system interaction must improve the quality of problem diagnosis process. This is a teleological proposition of DSS research. To improve the quality of problem diagnosis process, PDSS must utilize the inquiry "guaranteeing" concepts. This statement is sound from an epistemological standpoint.

8.2.2. Conceptual Design

First, consistent with the knowledge structure requirements, the conceptual level design of PDSS acknowledged structural, statistical, and rule-based model views. Careful studies of the problem diagnosis activities, however, revealed that PDSS may require other views than the causal models. They include: (1) problem state views; (2) goal state views; (3) performance discrepancy views; (4) alternative diagnosis views; and (5) an example data view. The necessities of the first four views have been discussed in descriptive and prescriptive problem diagnosis literature (see Section 5.1). An example data view is a direct product of the research assumption of problem repetitiveness.

After the view identification, this research established the dependencies among the views. The dependencies, in a sense, are similar to the input-output relationships. However, the derivation of many views requires dynamic user-system interactions. Other views (e.g., rules and goals) may be derived either by the users or by the system.

Second, the conceptual design included the identification, refinement, and development of support functions for various problem diagnosis activities. The support functions were based on descriptive and prescriptive literature on problem diagnosis, available techniques, insights, analysis, and synthesis. Problem detection routine was supported by goal projection, goal searching, filtering based on relevancy and significance, and data conversion and association functions. The modeling routine was supported by data analysis, structural analysis, regression modeling, and rule induction modules. The system supported the model application routine through providing alternative diagnosis views. Thus, the conceptual design satisfies the problem processing requirements fairly well.

To be noted is that the support functions may not be sufficient for certain problems. In fact, they can be only a very small fraction of the necessary problem diagnosis support functions. One problem associated with this line of inquiry, however, is that it is difficult to precisely delineate the necessary and sufficient sets of support functions. One common and partial solution to this problem is to comparatively evaluate the systems; i.e., relative sufficiency. The next section performs this task.

Finally, the task allocation between the users and the system was carried out based on common sense. For instance, the system should not ask the users to provide such difficult information (for the users) as the coefficients of causal impact. If technologically possible, the system's processing functions were

designed to overlap with the tasks that the users can possibly perform without the system's aid. For instance, the users may edit a set of rules by themselves or ask the system to induce them. In this way, the users may assign an appropriate role to the system as the situational factors warrant. The conceptual design indicated an interactive PDSS by defining the users' and system's roles in deriving various views and performing various problem diagnosis activities.

The conceptual design established that the system: utilizes multiple algorithms for goal projection and rule induction; maintains a consistency between a structural model and a descriptive model set; manages multiple model views, multiple problem sensing views, and alternative diagnosis views; refines structural models into descriptive models; and partitions a general hypothesis into more specific hypotheses. These capabilities provide a basis for arriving at a Lockean consensus, achieving a Leibnitzian consistency, facilitating Kantian inquiries, invoking Hegelian assumption evaluation, and supporting a Singerian model refinement process. Thus, some of the inquiry "guaranteeing" concepts have been incorporated into the PDSS design.

8.2.3. Symbol-Level Design and Prototype System Development

The symbol-level design established the attribute structures of the major views and detailed the algorithms for critical system-driven diagnosis support functions. The prototype system implementation was based on the symbol-level design. During prototyping, the correctness of system components had been extensively tested. For instance, the structural model analysis modules were tested with several hypothetical cases. The statistical modeling component was tested with textbook examples for correctness. The rule induction modules were verified by comparing the system-induced rules with hand-derived rules. Because

the verification had been done step-by-step, the system was capable of handling all reasonable exceptions. For instance, the regression analysis module reported an error when it could not obtain the inverse of a matrix (see Figure 7.13).

The prototype system demonstrated a successful implementation of all system capabilities envisioned on the conceptual design level. The prototype system managed and related structural models, statistical models, rules, and other useful views. The system also showed that all envisioned support functions were implemented and working correctly.

In summary, the conceptual design satisfied the system requirements; and the prototype system showed that the conceptual design was viable. In the previous section, it was stated that designing a PDSS that overcomes the limitations of previous PDSS is significant. The prototype system demonstrated that it is possible to overcome these limitations. Because the prototype system implementation is based on the symbol-level and concept-level design, the design performed in this research provide a basis for developing stronger inquiry support systems for managerial problem diagnosis.

8.3. Construct Validity: Evaluation of the Design

Construct validity addresses the issue of whether a theory provides a clearer picture of the reality; i.e., whether the variables are adequately defined and whether the relationships among the variables are plausibly established. In a design research context, the construct validity may be interpreted as a measure of the strengths of the design. The objective of this section is to comparatively evaluate the significance of this research. Figure 8.1 provides an overview of the managerial PDSS.

Pracht (1984)	A structural modeling support system employed in an empirical research. The system allows the users to save, retrieve, and modify the structural model. The system can analyze the model.
Ata Mohamed (1985)	An automatic cause-finding system. The system stores an extended structural model of which arcs are labeled with the causal impact coefficients. The user inputs the problem state values. The system identifies "symptoms" and the "causing" variables based on an extended structural model.
Ramaprasad & Poon (1985)	An interactive extended structural modeling system. The user categorizes the variables into decision, environmental, and goal elements. The values of the relationships are from -3 (very negative) to +3. The system can analyze, compare, and integrate the models.
Paradice (1986)	An interactive statistical modeling support system. The users enters a model statement. The system fits the model. The users may accept or reject the model. The system can analyze the model.
Billman (1989)	An automatic pattern discovery system. The system analyzes how the changes in variables are related. If the changes meet certain patterns, the system reports the relationships to the user.
Jung (1990)	An extended structural modeling support system. The user enters a structural model. The system identifies all causal cycles and ask the users to remove them. The system rearranges the variables into several layers. Then, the weights of arcs are readjusted to maintain the network equilibrium. The system can explain how strongly the variables are related.
Khazanchi (1991)	An interactive structural modeling system. The user edits a structural model. The system analyzes the structural model. The system can compare and integrate several structural models.
Current Research	An interactive problem diagnosis support system. The system supports structural, statistical, and rule-based modeling, goal formulation, data conversion, problem expansion/reduction, hypothesis formulation, classification, and cause-identification.

Figure 8.1. An Overview of Managerial Problem Diagnosis Support Systems

Figure 8.2 shows the modeling constructs adopted by the managerial PDSS. A majority of the systems utilize the structural modeling construct only. The systems developed by Pracht (1984) and Khazanchi (1991), for example, manage simple structural models denoting the arcs with the values of -1, 0, and +1. Ramaprasad and Poon (1985) used a slightly extended scale of {-3, -2, -1, 0, +1, +2, +3}. Jung (1990) extends the structural model by denoting the arcs with weights ranging from -1 to 1. Ata Mohamed (1985) assigns causal impact coefficients to the arcs. This research used -1, 0, +1, and 1 to label the arcs. The label "1" was necessary to mark those relationships that cannot be easily categorized into either "+1" or "-1." Paradice (1986) overcomes the limitations of the structural models by allowing the users to fit a variable with different sets of explanatory variables. Thus, his system can represent alternative model statements for a given variable.

	Structural Model	Statistical Model	Rules
Pracht (1984)	Simple		
Ata Mohamed (1985)	Extended		
Ramaprasad (1985)	Extended		
Paradice (1986)	Simple	Yes	
Billman (1989)	Implicit		
Jung (1990)	Extended		
Khazanchi (1991)	Simple		
Current Research	Simple	Yes	Yes

Figure 8.2. Modeling Constructs Adopted and Supported by Managerial PDSS

Because the PDSS developed in this research manages structural, statistical, and rule-based models, its representational capability is superior to the previous PDSS. This capability provides a significant advantage, because the system can handle not only the variable relationships but also the event relationships.

While representational capability is an important aspect of information systems, it must be coupled with processing capability. For instance, a paper-and-pencil system may represent a variety of things. However, it has no capability to process the represented objects. Thus, it is necessary to examine the systems' capabilities to build, analyze, and apply the models. Figure 8.3 show the reasoning methods utilized or supported by the PDSS. Inductive reasoning refers to whether the system helps to induce models. Deductive analysis refers to whether the system can analyze models and produce inference about the models. Deductive interpretation refers to whether the system can apply the models to particular instances and draw conclusions about the specific cases.

	Inductive Reasoning	Deductive Analysis	Deductive Interpretation
Pracht (1984)		Yes	
Ata Mohamed (1985)		Yes	Yes
Ramaprasad (1985)		Yes	
Paradice (1986)	Yes	Yes	
Billman (1989)	Yes		
Jung (1990)		Yes	Yes
Khazanchi (1991)		Yes	
Current Research	Yes	Yes	Yes

Figure 8.3. Reasoning Methods Employed by Managerial PDSS

Figure 8.3 shows the superiority of the system developed in this research. Unlike previous PDSS, the system is capable of supporting all three reasoning methods. Because managerial problem diagnosis process is retroductive (Dutton et al., 1983), the system promises better support.

Figure 8.4 examines the problem diagnosis routines being supported by managerial PDSS. With an exception of Ata Mohamed (1985), all the previous PDSS focus on structural model development. They are not particularly concerned with how the developed models can be applied. This, however, is not surprising for the systems developed by Pracht (1984), Ramaprasad and Poon (1985), Billman (1989), and Khazanchi (1991), because a simple structural model does not have a mechanism to interpret individual cases. That is, simple structural models are purely conceptual. From the point that models are built to interpret, explain, and predict events, a system that handles the specific problem instances should have a higher validity than those that do not.

	Problem Sensing	Model Development	Diagnosis Generation
Pracht (1984)		Yes	
Ata Mohamed (1985)	Yes		Yes
Ramaprasad (1985)		Yes	
Paradice (1986)		Yes	
Billman (1989)		Yes	
Jung (1990)		Yes	
Khazanchi (1991)		Yes	
Current Research	Yes	Yes	Yes

Figure 8.4. Problem Diagnosis Routines Supported by Managerial PDSS

Figure 8.5 shows the major inputs and outputs of managerial PDSS. Together with Figures 8.1 to 8.4, it shows the systems' overall functionalities. It also shows the interactions between the users and the system. The prototype system developed in this research supports extensive user-system interactions.

One point made clear by Figures 8.2 to 8.5 is that the PDSS developed in this research is more powerful, comprehensive, and cooperative than the previous PDSS. The system: (1) has powerful representational capability; (2) supports all major phases of problem diagnosis process; and (3) combines user-driven problem diagnosis and system-driven problem diagnosis. Two additional points must be made and further examined, however.

First, the system developed in this research is not a superset of the previous PDSS. Each research has its own niche; and this research did not cover all aspects of problem diagnosis support that one may pursue.

	User Inputs	System Outputs
Pracht (1984)	A structural model	Structural analysis
Ata Mohamed (1985)	A problem state	Cause identification
Ramaprasad (1985)	Structural models	Structural analysis
Paradice (1986)	Model specifications	Statistical models & analysis
Billman (1989)	Example Data	Structural models
Jung (1990)	A structural model	Analysis & cause identification
Khazanchi (1991)	Structural models	Structural analysis
Current Research	Example data Structural models Model specifications Problem & goal states	A structural model Structural analysis Statistical models & rules Diagnosis generation

Figure 8.5. Major Inputs and Outputs of Managerial PDSS

Second, although it is desirable for a PDSS to provide extensive support, the extensiveness itself does not guarantee the significance of the research. The system must provide unique advantages.

In the following, each of the previous PDSS are compared, within its own research scope, against the PDSS developed in this research. Pracht's system (1984) is a modeling tool employed in an empirical study. All of its capabilities are encompassed by the current research.

Ata Mohamed's system (1985) can identify the symptoms and causes for a specific problem instance. If a variable shows an abrupt change from the previous time period, the variable is flagged. That is, the performance in the previous time period serves as a goal for the current period. This approach may be acceptable for problems that should maintain a static equilibrium, but not for the problems that change dynamically. To overcome this difficulty, the current research allows the users to project, search, and edit the goal state values. The current research covers and refines all of its major functionalities.

Ramaprasad and Poon (1985) categorize the nodes of a structural model into three classes: decision, environmental, and goal variables. The distinction between the decision and environmental elements is especially useful, because it can tell whether a variable is within the decision maker's control. The current research does not cover this functionality. MIND can also compare and integrate several structural models. This function also has not been implemented in this research to avoid duplication.

Most of the major functionalities of Paradice's system (1986) are also supported in the current research. One difference is that Paradice's system provides users with the standardized coefficients, whereas this research provides both standardized and non-standardized regression coefficients. Paradice's system

is capable of providing advice; e.g., to increase one unit of sales volume, increase two units of advertising. The current research does not provide such advice, because there can be an infinite number of possible advice.

Billman's system (1989) provides three qualitative induction methods for structural model development. The system, thus, can suggest three different structural models for a given problem. The current research, however, provides more traditional statistics describing the pairwise relationships and leaves the construction of structural model to the users. Billman assumes numeric data only.

Jung (1990) adopts the neural network concept for managerial problem diagnosis. Neural network modeling has not been examined in the current research, because it is strongly system-driven. Compared to the depth-first cycle detection algorithm suggested by Jung, the algorithm in Figure 6.2 is much faster. Jung's system can identify all paths leading to or coming from a node. Because the number of such paths can be extremely large, the current research allows the users to examine all paths between two nodes, instead.

One unique feature of Khazanchi's system (1990), shared with Ramaprasad and Poon (1985), is its capability to compare and integrate multiple structural models. One limitation of Khazanchi's system is that it displays one relationship at a time. For instance, if a variable affects "A," "B," "C," and "D," then the user has to go through four screens to get the information. In addition, the user has to identify variables with their identification numbers during the causal model analysis. Such user interface impose an unnecessary cognitive burden on the users. The current research overcomes such limitations.

The above analysis indicates that the prototype system's functionality is rather comprehensive even in light of other PDSS design research. The current research designed and successfully implemented the following unique capabilities.

1. The system allows users to easily represent and modify problem states and goal states. The system helps the users to analyze the current problem state by comparing and contrasting it with the historical performances. The system allows the users to define, project, and search the performance goals. It can identify symptomatic variables.

2. The system supports filtering activity. Users may filter out and reintroduce the variables. This feature supports problem reduction/expansion.

3. The system allows users to deal with both quantitative and categorical data. This is evident in that the system supports both statistical and rule-based modeling. In addition, the system allows the users to interactively define, discretize, quantify, and categorize variables. Thus, the users have more flexibility in statistical and rule-based model development.

4. The system helps users' structural model development with such statistics as correlation between increments, analysis of variance, and chi-square. Conventionally, these analysis techniques have been used for hypothesis testing. This research utilizes them for hypothesis formulation.

5. The system provides an extended set of structural model analysis functions. For instance, the system can identify the variables that are affected by both "A" and "B." It can also identify whether a variable affects another variable directly, indirectly, or both directly and indirectly. The system can explain how two variables are related. The system is unique in that it finds all paths between two variables even when the paths contain cycles.

6. The system allows the users to delineate the relationships between the variables in a structural model and then further explore the relationships with statistical analysis techniques and rule-based modeling. This systematic way of deriving descriptive models is an effective way to deal with complex problems. In

1. The system allows users to easily represent and modify problem states and goal states. The system helps the users to analyze the current problem state by comparing and contrasting it with the historical performances. The system allows the users to define, project, and search the performance goals. It can identify symptomatic variables.

2. The system supports filtering activity. Users may filter out and reintroduce the variables. This feature supports problem reduction/expansion.

3. The system allows users to deal with both quantitative and categorical data. This is evident in that the system supports both statistical and rule-based modeling. In addition, the system allows the users to interactively define, discretize, quantify, and categorize variables. Thus, the users have more flexibility in statistical and rule-based model development.

4. The system helps users' structural model development with such statistics as correlation between increments, analysis of variance, and chi-square. Conventionally, these analysis techniques have been used for hypothesis testing. This research utilizes them for hypothesis formulation.

5. The system provides an extended set of structural model analysis functions. For instance, the system can identify the variables that are affected by both "A" and "B." It can also identify whether a variable affects another variable directly, indirectly, or both directly and indirectly. The system can explain how two variables are related. The system is unique in that it finds all paths between two variables even when the paths contain cycles.

6. The system allows the users to delineate the relationships between the variables in a structural model and then further explore the relationships with statistical analysis techniques and rule-based modeling. This systematic way of deriving descriptive models is an effective way to deal with complex problems. In

addition, the system maintains a consistency between a structural model and descriptive models. The system help users to cross-validate structural and statistical models. Structural analysis for the validation of statistical models often helps users to study possible relationships as well as probable relationships.

7. The system helps users to study complex relationships by building conditional statistical models. This capability, however, is not unique to the prototype system, because many statistical analysis packages can fit statistical equations by retrieving appropriate data sets from the data base. However, the way the prototype system allows the users to conveniently represent, examine, and modify the statistical models is unique. Furthermore, the system treats the statistical model statements as rules and "runs" them to generate diagnoses.

8. The system can induce rules from the examples based on a structural model. The prototype system extracts patterns from the examples, rather than merely changing the representation format. The system allows the users to edit the induced rules or their own rules.

9. The system can generate alternative diagnoses. Diagnosis generation includes determination of target values and identification of causes. The system allows the users to perform what-if analysis.

In addition to the unique capabilities discussed above, the system provides a very friendly user interface. For instance, the users can build regression models simply by choosing the relevant variables from dialog boxes. The system does not impose a particular decision process upon the users. Such a feature is necessary to support a variety of decision processes (Sprague and Carlson, 1982).

In conclusion, the system is relatively comprehensive in terms of its modeling constructs, diagnosis support functions, and user-system interaction. The system provides many useful capabilities that are unique.

8.4. External Validity

One aspect of external validity is how generalizable the system is. As demonstrated in Chapter 7, the system can be adapted to a variety of problems. One factor that limits the system's applicability is the particular view structures that the prototype system adopted. Specifically, the system employs a simple data table to represent a problem state, for example. In real decision making situations, the decision makers may have a variety of information in different format; e.g., market share information or regional sales data. The limited data management capability is not a unique problem for the current research. In fact, among the examined PDSS, only Ata Mohamed (1985), Paradice (1986), and Billman (1989) utilize data tables. They also assume a simple data table structure, and the contents are limited to the numeric values. The prototype system, however, handles both numeric and categorical values.

The other point is the generalizability of the design. One strength of this research is that the design is well grounded in problem diagnosis and information system literature. The framework was built upon MIS/DSS research frameworks and sound design concepts. Throughout this chapter, it has been shown that the views and support functions are sound in light of descriptive and normative problem diagnosis literature. Logical reasons were provided for having each of the views and support functions. Thus, the views and support functions are likely to be necessary and useful. It is certainly possible for the users to find certain views or support functions irrelevant for a particular problem. It is also probable that the users find some desirable functions missing from the system. Overall, the design concepts are valuable for many problem diagnosis situations.

The conceptual framework, especially concerning the system's requirement definitions, must be evaluated, as well. There can be a variety of ways to validate

a conceptual framework; e.g., logical argument, ad hoc analysis, etc. One way is through examining its capability to categorize and contrast various research that can be framed within the conceptual foundation (e.g., Ives et al., 1980). The framework developed in this research was satisfactory in this regard, as shown in the previous section (Figures 8.2 to 8.5).

An ultimate test for this research is to examine whether the final product, i.e., the implemented system, can actually improve the decision maker's performances in an experimental setting. This, however, could not be done in this research because of limited research time and resources. To be noted is that a single experiment may not be able to determine the effectiveness of the system. As the MIS frameworks indicate, the effectiveness of an information system is determined by a variety of factors such as an individual's characteristics, problem characteristics, and decision environment characteristics. One additional factor introduced in this research is the specific roles played by the decision analysts and decision makers. Unfortunately, the current research could not have addressed such empirical issues.

8.5. Summary

Problem diagnosis is a critical and difficult aspect of decision making. To improve the problem diagnosis process, a support system must have strong capabilities to represent various causal models, support problem diagnosis activities, and dynamically interact with the users. The face validity of this research is favorable, because it develops a more powerful, comprehensive, and cooperative support system to improve one of the most critical and difficult phases of the decision making process.

The conceptual design satisfies the system requirements. The prototype system implementation is based on symbol-level design and conceptual design. The prototype system satisfies the system requirements and demonstrates that the conceptual and symbol-level design is viable. Thus, the prototype system provides a proof that the conceptual design established a foundation for developing strong inquiry support systems for managerial problem diagnosis.

The conceptual design is relatively comprehensive with respect to modeling constructs, reasoning methods, diagnosis activities being supported, and user-system interaction. The design provides many unique advantages.

The prototype system can be applied to a variety of different problems. The design concepts are either grounded in descriptive and prescriptive literature or justified. In conclusion, this research is significant, appropriately executed, relatively comprehensive, and fairly generalizable.

CHAPTER IX

SUMMARY AND CONCLUSIONS

Problem diagnosis, the process of discovering and constructing the causal structure of a problem, is a critical aspect of decision making. It is a complex process requiring various activities. The process of inquiring into a problem situation becomes very difficult when problem complexity exceeds the inquirer's cognitive capacity. The inquirer, thus, often requires a supporting tool or a cooperative partner. MIS/DSS have unique qualifications to provide such a cooperative inquiry support system. The objective of this research was to establish a conceptual framework for PDSS development and design an inquiry support system for managerial problem diagnosis that overcomes the limitations of the previous PDSS.

9.1. Summary of the Research

The major impetus of this research was the recognition that the previous PDSS suffer several shortcomings. First, the majority of these systems employ structural modeling as a sole basis of supporting problem diagnosis. Although structural models are useful, they have limited ways of representing and making inferences about object relationships. Second, they focus on different phases of the problem diagnosis process. Thus, they cannot support the entire problem diagnosis process which is iterative and retroductive. Finally, they rarely combine user-driven and system-driven problem diagnosis.

The basic premise of this research was that the shortcoming of the previous PDSS can be overcome by: (1) formulating a sound conceptual framework for PDSS development; (2) carefully deriving the requirements of PDSS; and (3) designing PDSS that satisfy the requirements. This research has pursued two

major objectives: (1) to establish a conceptual framework for PDSS development; and (2) to design a system that overcomes the limitations of the previous PDSS by satisfying the PDSS requirements established in the conceptual framework.

Underlying the framework is the cooperative inquiry system concept. The concept is a synthesis of problem diagnosis literature, MIS/DSS frameworks, a cooperative system concept, and the inquiry "guaranteeing" concepts, among others. The cooperative inquiry system concept indicated that: (1) problem diagnosis effectiveness is a function of the cooperation achieved by the users and the PDSS for the given situation; (2) from a cognitive perspective, PDSS must have a knowledge base that easily accommodates the users' mental models and problem processing functions that facilitate the users' natural problem diagnosis processes; (3) from a normative perspective, PDSS must have a knowledge base that extends the users' mental models and problem processing functions that can alter the users' suboptimal problem diagnosis processes; and (4) PDSS must utilize both user-driven and system-driven problem diagnosis and have mechanisms that "guarantee" the quality of the inquiry process.

The conceptual framework also identified essential elements of PDSS, delineated logical steps in developing and using PDSS, specified the roles played by individuals, and discussed major issues to be addressed for PDSS development. PDSS designers must develop the system's generic capabilities to: (1) manage various causal models; (2) support various problem diagnosis activities; and (3) cooperatively interact with the users. The framework, thus, established the requirements of non-specific PDSS from these three perspectives.

First, the knowledge base structure of PDSS is analogous to the decision maker's mental modeling constructs. Thus, PDSS must provide modeling constructs that are comfortable to the users. Three most widely used causal

models are structural, statistical, and rule-based models. This research argued that they also constitute a necessary set of modeling constructs for PDSS. They are necessary for modeling relationships among numeric and categorical variables.

Second, the problem processing capabilities of PDSS are analogous to the decision maker's problem diagnosis activities. This research studied problem diagnosis literature and identified the following routines and activities that are inherent in, and essential for, successful problem diagnosis: (1) problem detection routine (sensing, filtering, and labeling activities); (2) causal model development routine (variable identification, model construction, and model validation activities); and (3) causal model application routine (diagnosis generation, cause identification, and diagnosis evaluation activities).

Finally, cooperative PDSS should be interactive to combine both user-driven and system-driven problem diagnosis. Moreover, PDSS must reduce, or at least attempt to reduce, users' errors and biases in conceptualizing problems by utilizing the inquiry "guaranteeing" concepts.

The conceptual design specifications of PDSS were performed to satisfy the PDSS requirements. On the conceptual design level, this research identified various types of views necessary for PDSS, developed a set of support functions for the problem diagnosis activities, and incorporated several mechanisms that may guarantee the quality of the problem diagnosis process. The views and critical functions were then described by a set of precisely defined symbols. The symbol-level specifications were used for prototype system implementation.

The prototype system demonstrated that it can not only manage but also help users to develop structural models, statistical models, and rules. The system allowed the users to formulate, save, retrieve, copy, modify, and delete problem state views, goal state views, performance discrepancy views, structural models,

statistical model statements, rules, and alternative diagnosis views. It supported and combined user-driven and system-driven problem diagnosis throughout the entire problem diagnosis process with such functions as goal projection, goal searching, filtering, discretization, quantification, association, symptom identification, correlation analysis, analysis of variance, contingency table test, structural analysis, cycle detection, path identification, conditional statistical analysis, cross-validation of structural and statistical models, rule induction, rule evaluation, diagnosis generation, cause identification, and what-if analysis. The prototype system employed multiple algorithms for Lockean inquiries, maintained a Leibnitzian consistency between structural and descriptive models, managed multiple models for Kantian and Hegelian inquiries, and supported a Singerian model refinement by building descriptive models based on the structural models.

The focus of this research was on supporting semi-structured or unstructured problem diagnosis process. Problem diagnosis is one of the most critical and difficult aspects of the decision making process. The conceptual framework for PDSS development has a significant face validity, as it has been built upon problem diagnosis literature, MIS/DSS frameworks, and sound design concepts such as the cooperative system concept, multiple causal modeling approaches, a process-oriented decision support approach, and the inquiry "guaranteeing" concepts. The prototype system supported structural, statistical, and rule-based modeling instead of relying on the structural modeling alone. It supported the entire problem diagnosis process instead of focusing on a certain aspect of problem diagnosis. Finally, the system combined both user-driven and system-driven problem diagnosis and utilized the inquiry "guaranteeing" concepts. Therefore, the prototype system overcame all the major limitations of the previous PDSS.

In conclusion, the conceptual framework and the conceptual design of PDSS provide a sound basis for developing more powerful, more comprehensive, and more cooperative PDSS. The prototype system overcame the limitations of the previous PDSS by integrating structural, statistical, and rule-based modeling approaches, supporting the entire problem diagnosis process, and achieving a more cooperative inquiry.

9.2. Contributions of the Research

The major contribution of this research is the formulation of a conceptual framework for PDSS development. The fragmented PDSS research, to date, may be attributed to the lack of a conceptual foundation for PDSS development. The risks are twofold. First, PDSS research efforts may overlap without synthetic effects, resulting in waste of research resources. Second, the essences of PDSS may become obscure so that a cumulative development of knowledge is impossible. This research presented the first conceptual framework for PDSS development. The framework reduces the risks by structuring important issues related to the development of PDSS. The framework contributes to a systematic development of PDSS research.

The framework is significant in terms of its paradigm orientation. Traditionally, PDSS research has emphasized either normative or cognitive support. This research pointed out the risks involved in either approach and suggested to adopt the cooperative inquiry system concept as a basis of PDSS design. The framework contributes to PDSS research by establishing a sound theoretical foundation for PDSS design.

The framework provides a broad perspective for PDSS designers by synthesizing a variety of sound theories and design concepts such as MIS

frameworks, DSS models, a cooperative system concept, multiple causal modeling approaches, process-oriented decision support, and the inquiry "guaranteeing" concepts. That is, the framework provides a refined Singerian view of PDSS design research to future PDSS designers.

This research is significant in that it relates and combines structural, statistical, and rule-based modeling techniques for the purpose of problem diagnosis support. The majority of the previous PDSS supported only structural modeling. DSS research (inclusive of PDSS research) has long proposed the integration of the rule-based modeling with the traditional quantitative modeling (Ata Mohamed, 1985; Paradice, 1986; Turban and Watkins, 1986; White, 1990). By integrating the three causal modeling approaches, this research has provided ways to: (1) handle both numeric and categorical data; (2) represent event relationships as well as variable relationships; (3) deal with complex, non-linear relationships; and (4) systematically develop more intricate causal models.

This research is significant in that it realizes a process-oriented PDSS. The concept of process-oriented DSS is simple. To support a decision process, the designer must study the decision process first and then develop support functions to facilitate the studied process. Previous PDSS research has paid more attention to the technological aspect of the system and less attention to the problem diagnosis process. Consequently, they are deficient in supporting various problem diagnosis activities that are inherent in and essential for successful problem diagnosis. This research showed that a process-oriented PDSS is viable and provides better support.

This research is also significant in that it emphasizes cooperative problem diagnosis. This research established the views and support functions from both descriptive and prescriptive perspectives. The system supports both user-driven

and system-driven problem diagnosis. A majority of previous PDSS emphasize either normative or cognitive decision support. Normative systems impose a particular decision model onto the decision makers. Thus, they have difficulties in defending their positions, like any Leibnitzian system does. On the other hand, so-called "cognitive modeling" support systems are, very often, little more than simple structural modeling systems. This research designed a cooperative PDSS to which the users can assign an appropriate role.

This research is significant in that it utilizes inquiry "guaranteeing" concepts for PDSS design. First, this research utilized two Lockean rule induction algorithm. It has been shown that each of the algorithms was more useful for a problem but not for the other problem. Second, this research maintained a Leibnitzian consistency between a structural model and a descriptive model set. Third, this research supported multiple model views to support Kantian, Hegelian, and Singerian inquiries. Fourth, this research supported a Singerian model refinement process by building descriptive models based on structural models. By incorporating these inquiry "guaranteeing" modules into PDSS design, this research improves the quality of PDSS.

This research also has made some practical contributions to PDSS development. The reachability matrix derivation algorithm (Figure 6.1) finds a final reachability matrix very efficiently by utilizing the matrix's special characteristics (Warfield, 1976). Together with this algorithm, the cycle detection algorithm (Figure 6.2) identifies cycles from the adjacency matrix. This method is much faster than the depth-first or width-first causal cycle searching method (instead of traversing the digraph one node at a time, Figure 6.1 traverses over a collection of traversed paths). Two rule induction algorithms (Figures 6.5 and 6.6) make a practical contribution to the pattern discovery in data base.

Another contribution of this research is the prototype system itself. Because all the majority support functions are currently operational, individuals can readily apply the system to a variety of problems. Users may find many of the prototype system's unique capabilities useful; e.g., goal searching, filtering, dependence statistics for structural modeling, statistical modeling based on a structural model, conditional statistical models, rule induction, diagnosis generation using both statistical model statements and rules, and what-if diagnosis analysis.

9.3. Recommendations for Practitioners

Managers who deal with complex problems in their everyday routine may want a decision aid that can simplify their problems. In general, structural models can greatly help decision makers to simplify a problem by delineating the variables pertinent to the problem and the cause-effect relationships between the variables. Structural modeling support systems are especially useful to analyze which variables affect, or are affected by, other variables.

When decision makers employ a certain modeling construct, their ways of thinking can be influenced (and often bounded) by the modeling construct, however. When a singed structural model (using +1, -1, and 0) is used, decision makers may be forced to think that there are only three types of relationships, i.e., positive, negative, and no relationships. To be noted is that "+1" indicates a monotonic increase, and that "-1" indicates a monotonic decrease. That is, structural models assume linear or curvilinear relationships. Therefore, they cannot adequately represent nonlinear relationships.

Although simplification is a useful strategy, decision makers must keep themselves remind of the risk involved. A simplified picture may hide "true"

relationships. If the cost of making such an error is high, decision makers must employ a system that can truly handle, not hide, the complex relationships. This research recommends that decision makers employ structural models only to delineate and analyze logical connections among variables. Although some structural modeling support systems categorize the impacts of a change in a variable on another variables into either positive or negative, such simplification may generate misleading information. In many cases, the impact of a variable on other variables can be both positive and negative depending on other variables. Thus, causal impact analysis must be done based on descriptive models that can handle such complex relationships. This research recommends that decision makers build statistical models and rules based on structural models.

Decision makers (and/or decision analysts) are encouraged to develop multiple causal models for a given problem. All system-driven model induction, and even human inductive reasoning, are based on certain assumptions. In general, such assumptions should be carefully tested before applying the induction technique. In practice, however, those assumptions can be very implicit and difficult to test. If such is the case, decision makers may have to apply all readily-available, plausible techniques and evaluate their results altogether. The model results may be evaluated comparatively, averaged, synthesized, or developed into alternative diagnoses. A cost-benefit analysis should determine the number of alternative models to develop and examine.

9.4. Limitations of the Research

One major limitation of the research originates from the assumption that the cause-effect relationship is the single most important information for successful problem diagnosis. The basis for stating that structural, statistical, and

rule-based model statements are a necessary set of modeling constructs is that there are objects and object relationships, and that an object assumes its domain values. Although the importance of causal relationships cannot be overemphasized, there are many other relationships that decision makers are interested in. Therefore, the structural, statistical, and rule modeling may not be enough to sufficiently represent a realistically complex problem.

Another major limitation of the research is the limited user-system interaction types. Specifically, the system does not allow user programming. This research developed support functions for each diagnosis activity. Because a problem diagnosis routine consists of several activities, the users may want to combine several functions in a specific order, so that the system can perform the routine without interacting with the users. This research decomposes the diagnosis process into routines and then activities, but the activities cannot be combined together into routines and ultimately into a process without active user interaction. Thus, the decision analysts may have little means to custom-tailor the system to specific decision situations except through providing different causal models and problem-specific views.

9.5. Suggestions for Future Research

This research has taken a broad perspective for PDSS development. Many research topics can be derived from this research. First, this research has focused on supporting managerial problem diagnosis. It may be possible to integrate PDSS capabilities with other capabilities and support an entire decision process. Future research may incorporate into PDSS design a systematic way to design different effects on the causal models and represent/evaluate alternative solutions (Then, the system can be called "a process-oriented DSS").

Second, future PDSS research may refine the PDSS design framework. This research, having focused on developing the capabilities of a non-specific PDSS, could not address many issues related to specific PDSS development and use processes. Future research may address such issues as: (1) what the major characteristics of the decision environment factors are (e.g., time constraints and stress level); (2) what additional attributes of PDSS must be considered to reflect such factors (e.g., presentation formats and interaction protocol); (3) how a PDSS can effectively adapt to changing problem situation and decision environment; (4) what roles decision analysts and decision makers should play in that process; (5) how to manage various causal models formulated by different individuals; etc.

Third, one challenging future research is to enrich the knowledge base by including more relationships than the causal relationship. AI research has identified and used several common relationships; e.g., has, is-a, instance-of, and is-part-of. These relationships have well-define properties, thus, are amenable to matrix manipulation (Burns et al., 1989). Baldwin (1989) proposed a few other relationships commonly used in the management domain, e.g., is-responsible, flows, achieved-by, and requires. It appears that they also have the properties of transitivity, anti-symmetricity, and irreflexiveness. The structural model analysis functions are equally applicable to each of the relationship types.

The challenge is not much on representing the relationships or reasoning within each relationship type but on: (1) identifying the common and critical relationships; and (2) establishing the relationships over the relationships. One valuable future research project is, therefore, to identify those critical predicates and define clauses over the predicates. In the process of identifying additional relationship types, one may have to establish different object types as well, because certain relationships may hold only between certain object types.

Fourth, future research may expand the problem processing functions into a more comprehensive set. This research did not utilize purely system-driven modeling approaches (e.g., neural networking), because users may have difficulties to understand such models. Future research may develop a way to effectively communicate the knowledge embedded in such system-driven models to users.

A PDSS may manipulate causal models (declaratively represented in the knowledge base) in the same fashion that it manipulates data. Future research may attempt to improve PDSS's model management capabilities. For instance, a PDSS that allows users to retrieve relevant statistical models and rules on a graphically illustrated structural model may greatly enhance user's understanding.

Fifth, DSS research in general, and PDSS research in particular, call for an in-depth research to incorporate the inquiry "guaranteeing" concepts into a computer-based system. Recent development in model management systems indicates that models will be accessed more frequently by more individuals. Thus, the costs of keeping "incorrect" models in the model base are likely to increase. Thus, more research are required to "guarantee" the quality of decision models.

Finally, an empirical research may be conducted to refine PDSS design. The empirical research must address at least two issues (see Adams et al., 1990): (1) what problem diagnosis activities are most critical in actual settings; and (2) what capabilities of PDSS are considered most essential by the users. This research has identified problem diagnosis activities from theoretical literature and designed PDSS capabilities based on literature, available techniques, insights, synthesis, etc. An empirical study may indicate whether the PDSS developed in this research provides an appropriate level of support (thus improves problem diagnosis effectiveness) and what other capabilities the system requires to better support the actual managerial problem diagnosis process.

REFERENCES

Ackoff, R.L., "Management Misinformation Systems," Management Science, Vol. 14, No. 4, pp. B147-B156, 1967.

Ackoff, R.L., "Towards A System of Systems Concept," Management Science, Vol. 17, No. 11, pp. 661-671, 1971.

Ackoff, R.L., Redesigning the Future, John Wiley, New York, 1974.

Ackoff, R.L., The Art of Problem Solving, John Wiley, New York, 1978.

Ackoff, R.L., "The Art and Science of Mess Management," Interfaces, Vol. 11, No. 1, pp. 20-26, 1981.

Ackoff, R.L. and Emery, F.E., On Purposeful Systems, Aldine-Atherton, Chicago, 1972.

Ackoff, R.L., Gupta, S.K., and Minas, J.S., Scientific Method: Optimizing Applied Research Decisions, John Wiley, New York, 1962.

Adams, D.A., Courtney, J.F., and Kasper, G.M., "A Process-Oriented Method for the Evaluation of Decision Support System Generators," Information and Management, Vol. 19, pp. 213-225, 1990.

Alter, S.L., "A Taxonomy of Decision Support Systems," Sloan Management Review, Vol. 19, pp. 39-56, 1977.

Anderson, J.C. and Janson, M.A., "Methods for Managerial Problem Cause Analysis," Interfaces, Vol. 9, No. 5, pp. 121-128, 1979.

Andriole, S., "The Design of Microcomputer-Based Personal Decision-Aiding Systems," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 12, pp. 463-469, 1982.

Argyris, C., "Single-Loop and Double-Loop Models in Research on Decision Making," Administrative Science Quarterly, Vol. 21, No. 3, pp. 363-377, 1976.

Arinze, B., "A Contingency Model of DSS Development Methodology," Journal of MIS, Vol. 8, No. 1, pp. 149-166, 1991.

Ata Mohamed, N.H., "A Knowledge-Based Decision Support System for Managerial Problem Recognition and Diagnosis," Unpublished D.B.A. Dissertation, Texas Tech University, 1985.

Axelrod, R., The Structure of Decision: Cognitive Maps of Political Elites, Princeton University Press, Ewing, NJ, 1976.

Bahill, A.T., Verifying and Validating Personal Computer-Based Expert Systems, Prentice Hall, Englewood Cliffs, NJ, 1991

Bahm, A.J., Philosophy: An Introduction, John Wiley, New York, 1953.

Baldwin, D., "Principles of Design for a Multiple Viewpoint Problem Formulation Support Systems," Unpublished Ph.D. Dissertation, Texas Tech University, 1989.

Banville, C. and Landry, M., "Can the Field of MIS be Disciplined?," Communications of the ACM, Vol. 32, No. 1, pp. 48-60, 1989.

Barkakati, N., Object-Oriented Programming in C++, Macmillan, Camel, IN, 1991

Barnes, J.H., "Cognitive Biases and Their Impact on Strategic Planning," Strategic Management Journal, Vol. 5, pp. 129-137, 1984.

Bartee, E.M., "A Holistic View of Problem Solving," Management Science, Vol. 19, No. 4, pp. 439-447, 1973.

Benbasat, I. and Taylor, R.N., "Behavioral Aspects of Information Processing for the Design of Management Information Systems," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 12, No. 4, pp. 439-450, 1982.

Bennett, J. and Hollander, C., "DART: An Expert System for Computer Fault Diagnosis," Proceedings of International Joint Conference on Artificial Intelligence, pp. 843-845, 1981.

Billings, R.S., Milburn, T.W., and Schaalman, M.L., "A Model of Crisis Perception: A Theoretical and Empirical Analysis," Administrative Science Quarterly, Vol. 25, No. 2, pp. 300-316, 1980.

Billman, B.H., Automated Discovery of Causal Relationships in Managerial Problem Domains, Unpublished Ph.D. dissertation, Texas A&M University, 1989.

Blanning, R.W., "A Relational Framework for Model Management in Decision Support Systems," DSS-82 Transactions, Province, RI, 1982.

Blanning, R.W., "Knowledge Acquisition and System Validation in Expert Systems for Management," Human Systems Management, Vol. 4, pp. 280-285, 1984.

Bonczek, R.H., Holsapple, C.W., and Whinston, A.B., "Future Directions for Developing Decision Support Systems," Decision Sciences, Vol. 11, pp. 616-631, 1980.

Bonczek, R.H., Holsapple, C.W., and Whinston, A.B., "A Generalized Decision Support System Using Predicate Calculus and Network Data Base Management," Operations Research, Vol. 29, No. 2, pp. 263-281, 1981.

Bouwman, M.J., "Human Diagnostic Reasoning by Computer: An Illustration from Financial Analysis," Management Science, Vol. 29, No. 6, pp. 653-672, 1983.

Bravoco, R.R. and Yadav, S.B., "Requirement Definition Architecture -- An Overview," Computers in Industry, Vol. 6, pp. 237-251, 1985.

Browing, D., Ontology and the Practical Arena, The Pennsylvania State University Press, University Park, PA, 1990.

Brown, C., "Causal Reasoning in Performance Assessment: Effect of Cause and Effect, Temporal Order, and Covariation," Accounting, Organizations and Society, Vol. 10, No. 3, pp. 255-266, 1985.

Bui, T. and Jarke, M., "Communications Requirements for Group Decision Support Systems," Proceedings of the 19th Annual Hawaii International Conference on System Sciences, Honolulu, pp. 524-533, 1986.

Bunge, M., Method, Model, and Matter, Reider Publishing, Boston, MA, 1973.

Bunge, M., Causality and Modern Science, Dover, New York, 1979.

Burns, J.R., "A Specification Language for Generating Intelligent Discrete Next-Event Simulations," Information and Decision Technologies, Vol. 16, pp. 3-13, 1990.

Burns, J.R. and Winstead, W.H., "An Input/Output Approach to the Structural Analysis of Digraphs," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 12, No. 1, pp. 15-24, 1982.

Burns, J.R. and Winstead, W.H., "M-Labeled Digraphs: An Aid to the Use of Structural and Simulation Models," Management Science, Vol. 31, No. 3, pp. 343-357, 1985.

Burns, J.R., Winstead, W.H., and Haworth, D., "Semantic Nets as Paradigm for Both Causal and Judgmental Knowledge Representation," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 1, pp. 58-67, 1989.

Burrell, G. and Morgan, G., Sociological Paradigms and Organizational Analysis, Heinemann, New York, 1979.

Campbell, K., Metaphysics: An Introduction, Dickenson, Encino, CA, 1976.

Cats-Baril, W.C. and Huber, G.P., "Decision Support Systems for Ill-structured Problems: An Empirical Study," Decision Sciences, Vol. 18, pp. 350-372, 1987.

Cendrowska, J., "PRISM: An Algorithm for Inducing Modular Rules," International Journal of Man-Machine Studies, Vol. 27, pp. 349-370, 1987.

Chabris, C.F., Artificial Intelligence & Turbo C, Dow Jones-Irwin, Homewood, IL, 1989

Chandrasekaran, B. and Goel, A., "From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, No. 3, pp. 415-424, 1988.

Chatfield, A.T., "A User Learning Based DSS Implementation Methodology," Unpublished Ph.D. Dissertation, Texas Tech University, 1990.

Checkland, P.B., Systems Thinking, Systems Practice, John Wiley, New York, 1981.

Checkland, P.B., "Towards a Systems-Based Methodology for Real-World Problem Solving," in J. Beishon and G. Peters, Systems Behavior, 2nd ed., Open University Press, Bristol, PA, 1976.

Chen, P.P.S, "The Entity-Relationship Model - Toward a Unified View of Data," ACM Transactions on Database Systems, Vol. 1, No. 1, pp. 9-36, 1976.

Cherrington, D.J., Personnel Management: The Management of Human Resources, Brown Co., Dubuque, IA, 1983.

Chervany, N.L., Dickson, G.W., and Kozar, K.A., "An Experimental Gaming Framework for Investigating the Influence of Management Information Systems on Decision Effectiveness," Management Information Systems Research Center, Working Paper 71-12, University of Minnesota, 1971.

Churchman, C.W., The Design of Inquiring Systems, Basic Books, New York, 1971.

Conover, W.J., Practical Nonparametric Statistics, 2nd ed., Wiley, New York, 1980.

Cooper, R.B., "Review of Management Information Systems Research: A Management Support Emphasis," Information Processing and Management, Vol. 24, No. 1, pp. 73-102, 1988.

Cooprider, J.G. and Henderson, J.C., "Technology-Process Fit: Perspectives on Achieving Prototyping Effectiveness," Journal of MIS, Vol. 7, No. 3, pp. 67-87, 1991.

Cosier, R.A., "Dialectical Inquiry in Strategic Planning: A Case of Premature Acceptance?," Academy of Management Review, Vol. 6, No. 4, pp. 643-648, 1981.

Courtney, J.F., Paradice, D.B., and Ata Mohammed, N.H., "A Knowledge-Based DSS for Managerial Problem Diagnosis," Decision Sciences, Vol. 18, pp. 373-399, 1987.

Cowan, D.A., "Developing a Process Model of Problem Recognition," Academy of Management Review, Vol. 11, No. 4, pp. 763-776, 1986.

Cowan, D.A., "Executive's Knowledge of Organizational Problem Types: Applying a Contingency Perspective," Journal of Management, Vol. 14, No. 4, pp. 513-527, 1988.

Crocker, J., "Judgment of Covariation by Social Perceivers," Psychological Bulletin, Vol. 90, No. 2, pp. 272-292, 1981.

de Kleer, J. and Williams, B.C., "Diagnosing Multiple Faults," Artificial Intelligence, Vol. 32, pp. 97-130, 1987.

Dery, D., "Decision-Making, Problem-Solving and Organizational Learning," OMEGA, Vol. 11, No. 4, pp. 321-328, 1983.

Dolk, D.R., "Data as Models: An Approach to Implementing Model Management," Decision Support Systems, Vol. 2, pp. 73-80, 1986.

Dos Santos, B.L. and Holsapple, C.W., "A Framework for Designing Adaptive DSS Interfaces," Decision Support Systems, Vol. 5, pp. 1-11, 1989.

Dutton, J., Fahey, L., and Narayanan, V., "Toward Understanding Strategic Issue Diagnosis," Strategic Management Journal, Vol. 4, pp. 307-323, 1983.

Efron, B., "Bootstrap Methods: Another Look at the Jackknife," The Annals of Statistics, Vol. 7, pp. 1-26, 1979.

Einhorn, H.J. and Hogarth, R.M., "Prediction, Diagnosis, and Causal Thinking in Forecasting," Journal of Forecasting, Vol. 1, pp. 23-36, 1982.

Einhorn, H.J. and Hogarth, R.M., "Judging Probable Cause," Psychological Bulletin, Vol. 99, No. 1, pp. 3-19, 1986.

Elam, J.J. and Konsynski, B., "Using Artificial Intelligence Techniques to Enhance the Capabilities of Model Management Systems," Decision Sciences, Vol. 18, pp. 487-501, 1987.

Elam, J.J. and Mead, M., "Designing for Creativity: Considerations for DSS Development," Information and Management, Vol. 13, pp. 215-222, 1987.

Evans, J.R., "A Review and Synthesis of OR/MS and Creative Problem Solving," OMEGA, Vol. 17, No. 6, pp. 499-524, 1989.

Fales, E., Causation and Universals, Routledge, New York, 1990.

Feldman, J., "Beyond Attribution Theory: Cognitive Processes in Performance Appraisal," Journal of Applied Psychology, Vol. 66, pp. 127-148, 1981.

Finlay, P.N. and Martin, C.J., "The State of Decision Support Systems: A Review," OMEGA, Vol. 17, No. 6, pp. 525-531, 1989.

Fischler, M.A. and Firschein, O., Intelligence: The Eye, the Brain, and the Computer, Addison-Wesley, Reading, MA, 1987.

Fiske, S. and Linville, P., "What Does the Schema Concept Buy Us?," Personality and Social Psychology Bulletin, Vol. 6, pp. 543-557, 1980.

Flanagan, O., The Science of the Mind, The MIT Press, Cambridge, MA, 1991.

Fogg, C.D., Diagnostic Marketing: Finding and Fixing Critical Problems, Addison-Wesley, Reading, MA, 1985.

Ford, J.D. and Hegarty, W.H., "Decision Makers' Beliefs About the Causes and Effects of Structure: An Exploratory Study," Academy of Management Journal, Vol. 27, No. 2, pp. 271-291, 1984.

Freeland, J.R. and Stabell, C.B., "Allocation of Managerial Effort: An Investigation of the Relationship between Decision Strategies, Environment, and Performance," Behavior Science, Vol. 23, pp. 234-239, 1978.

Gallupe, R.B., DeSanctis, G., and Dickson, G.W., "Computer-Based Support for Group Problem-Finding: An Experimental Investigation," MIS Quarterly, Vol. 12, pp. 277-296, 1988.

Geoffrion, A.M., "An Introduction to Structured Modeling," Management Science, Vol. 33, No. 5, pp. 547-586, 1987.

Getzels, J.W., "Problem Finding: A Theoretical Note," Cognitive Science, Vol. 3, pp. 167-172, 1979.

Ghiaseddin, N., "An Environment for Development of Decision Support Systems," Decision Support Systems, Vol. 2, pp. 195-212, 1986.

Glymour, C., Theory and Evidence, Princeton University Press, Ewing, NJ, 1980.

Gorry, G.A. and Scott Morton, M.S., "A Framework for Management Information Systems," Sloan Management Review, Vol. 13, No. 1, pp. 55-70, 1971.

Groebner, D.F. and Shannon, P.W., Business Statistics: A Decision-Making Approach, Merrill Publishing, Columbus, OH, 1985.

Haak, S., Philosophy of Logics, Cambridge University Press, New York, 1978.

Hale, D.P. and Kasper, G.M., "The Effect of Human-Computer Interchange Protocol on Decision Performance," Journal of MIS, Vol. 6, No. 1, pp. 5-20, 1989.

Harary, F., Norman, R.Z., and Cartwright, D., Structural Models: An Introduction to the Theory of Directed Graphs, John Wiley, New York, 1965.

Hart, A.E., "Experience in the Use of an Inductive System in Knowledge Engineering," in M.A. Bramer, Research and Development in Expert Systems, Cambridge University Press, pp. 117-126, 1985.

Hasher, L. and Zacks, R., "Automatic and Effortful Processes in Memory," Journal of Experimental Psychology: General, Vol. 108, pp. 356-388, 1979.

Hogarth, R.M., Judgement and Choice: The Psychology of Decision, John Wiley, New York, 1980.

Hogarth, R.M., "Generalization in Decision Research: The Role of Formal Models," IEEE Transactions on System, Man, and Cybernetics, Vol. 16, No. 3, pp. 439-449, 1986.

Hogarth, R.M. and Makridakis, S., "Forecasting and Planning: An Evaluation," Management Science, Vol. 27, pp. 115-138, 1981.

Howard, G.S., "Object Oriented Programming Explained," Journal of Systems Management, pp. 13-19, July 1988.

Huber, G.P., "Cognitive Style as a Basis for MIS and DSS Design: Much Ado About Nothing?" Management Science, Vol. 29, No. 5, pp. 567-582, 1983.

Hwang, S., "Automatic Model Building Systems: A Survey," DSS-85 Transactions, The Institute of Management Sciences, Province, RI, 1985.

Iman, R.L. and Conover, W.J., Modern Business Statistics, John Wiley, New York, 1989

Inhelder, B., de Caprona, D., and Cornu-Wells, A., Piaget Today, Lawrence Erlbaum, Hillsdale, NJ, 1987.

Ireland, R., Hitt, M.A., Bettis, R.A., and de Porras, D.A., "Strategy Formulation Processes: Difference in Perceptions of Strength and Weakness Indicators and Environmental Uncertainty by Managerial Level," Strategic Management Journal, Vol. 8, pp. 469-485, 1987.

Ives, B., Hamilton, S., and Davis, G.B., "A Framework for Research in Computer-Based Management Information Systems," Management Science, Vol. 26, No. 9, pp. 910-934, 1980.

Janis, I.L. and Mann, L., Decision Making: A Psychological Analysis of Conflict, Choice, and Commitment, The Free Press, New York, 1977.

Johnson, R.A. and Wichern, D.W., Applied Multivariate Statistical Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Jung, D., "Design of Inexact Reasoning Systems for Managerial Problem Diagnosis," Unpublished Ph.D. Dissertation, Texas Tech University, 1990.

Keen, P.G.W., "MIS Research: Reference Disciplines and a Cumulative Tradition," in Proceedings of the First International Conference on Information Systems, Philadelphia, pp. 9-18, 1980.

Keen, P.G.W., "Decision Support Systems: The Next Decade," Decision Support Systems, Vol. 3, pp. 253-265, 1987.

Keen, P.G.W. and Scott Morton, M.S., Decision Support Systems: An Organizational Perspective, Addison-Wesley, Reading, MA, 1978.

Kelley, H., "The Process of Causal Attribution," American Psychologists, Vol. 28, pp. 107-113, 1973.

Kepner, C.H. and Tregoe, B.B., The New Rational Manager, Princeton Research Press, Princeton, NJ, 1981.

Khazanchi, D., "Subjective Understanding of Ill-Structured Problems: An Information Systems Perspective," Unpublished Ph.D. Dissertation, Texas Tech University, 1991.

Kiesler, S. and Sproull, L., "Managerial Response to Changing Environments: Perspectives on Problem Sensing from Social Cognition," Administrative Science Quarterly, Vol. 27, pp. 548-570, 1982.

Klein, G., "Developing Model Strings for Model Managers," Journal of MIS, Vol. 3, No. 2, pp. 94-110, 1986.

Klein, G., Konsynski, B., and Beck, P.O., "A Linear Representation for Model Management in a DSS," Journal of MIS, Vol. 2, No. 2, pp. 40-54, 1985.

Klein, G. A. and Weitzenfeld, J., "Improvement of Skills for Solving Ill-Defined Problems," Educational Psychology, Vol. 13, pp. 31-41, 1978.

Klien, H.K. and Hirschheim, R.A., "Fundamental Issues of Decision Support Systems: A Consequentialist Perspective," Decision Support Systems, Vol. 1, pp. 5-24, 1985.

Klien, H.K. and Hirschheim, R.A., "A Comparative Framework of Data Modelling Paradigms and Approaches," The Computer Journal, Vol. 30, No. 1, pp. 8-15, 1987.

Kohen, M., "Are MIS Frameworks Premature?," Journal of MIS, Vol. 2, No. 3, pp. 92-100, 1986.

Kolodner, J.L. and Kolodner, R.M., "Using Experience in Clinical Problem Solving: Introduction and Framework," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 17, No. 3, pp. 420-431, 1987.

Landry, M., Pascot, D., and Briolat, D., "Can DSS Evolve Without Changing Our View of the Concept of 'Problem'?," Decision Support Systems, pp. 25-36, 1985.

Lang, J.R., Dittrich, J.E., and White, S.E., "Managerial Problem Solving Models: A Review and A Proposal," Academy of Management Review, pp. 854-866, 1978.

Langefors, B., Theoretical Analysis of Information Systems, Auerback Publishers, Philadelphia, 1973.

Leavitt, H.J., "Beyond the Analytic Manager," California Management Review, Vol. 17, No. 3, pp. 5-12, 1975.

Levesque, H.J., "Foundations of a Functional Approach to Knowledge Representation," Artificial Intelligence, Vol. 23, No. 2, pp. 155-212, 1984.

Liang, T.P., "Integrating Model Management with Data Management in Decision Support Systems," Decision Support Systems, Vol. 1, pp. 221-232, 1985.

Liang, T.P. and Jones, C.V., "Meta-Design Considerations in Developing Model Management Systems," Decision Science, Vol. 19, pp. 72-92, 1988.

Licklider, J.C.R., "Man-Computer Symbiosis," IRE Transactions on Human Factors in Electronics, Vol. 1, No. 1, pp. 4-11, 1960.

Little, J.D.C., "Research Opportunities in the Decision and Management Science," Management Science, Vol. 32, No. 1, p. 1, 1986.

Lucas, H.C., Jr., "A Descriptive Model of Information Systems in the Context of the Organization," Data Base, Vol. 5, No. 2, pp. 27-36, 1973.

Lucas, R., "Organizational Diagnostics: Translating the Political into the Technical," Journal of Management, Vol. 13, No. 1, pp. 135-148, 1987.

Lyles, M.A., "Formulating Strategic Problems: Empirical Analysis and Model Development," Strategic Management Journal, Vol. 2, pp. 61-75, 1981.

Lyles, M.A., "Defining Strategic Problems: Subjective Criteria of Executives," Organization Studies, Vol. 8, No. 3, pp. 263-280, 1987.

Lyles, M.A. and Mitroff, I.I., "Organizational Problem Formulation: An Empirical Study," Administrative Science Quarterly, Vol. 25, pp. 102-119, 1980.

Lyles, M.A. and Thomas, H., "Strategic Problem Formulation: Biases and Assumptions Embedded in Alternative Decision-Making Models," Journal of Management Studies, Vol. 25, No. 2, pp. 131-145, 1988.

Lyles, R.I., Practical Management Problem Solving and Decision Making, Van Nostrand Reinhold, New York, 1982.

MacCrimmon, K.R. and Taylor, R.N., "Decision Making and Problem Solving," in M.D. Dunnette, Handbook of Industrial and Organizational Psychology, Rand McNally College Publishing, Chicago, pp. 1397-1453, 1976.

Mackie, J.L., "Causes and Conditions," American Philosophical Quarterly, Vol. 2, No. 4, pp. 245-264, 1965.

March, J.G. and Simon, H.A., Organizations, John Wiley, New York, 1958.

March, J. and Feldman, M., "Information in Organizations as Signal and Symbol," Administrative Science Quarterly, Vol. 26, pp. 171-186, 1981.

Markus, M.L. and Robey, D., "Information Technology and Organizational Change: Causal Structure in Theory and Research," Management Science, Vol. 34, No. 5, pp. 583-598, 1988.

Mason, R.O. and Mitroff, I.I., "A Program for Research on Management Information Systems," Management Science, Vol. 19, No. 5, pp. 475-485, 1973.

Mason, R.O. and Mitroff, I.I., Changing Strategic Planning Assumptions, John Wiley, New York, 1981.

McCord, J.W., Developing Windows Applications with Borland C++3, SAMS, Carmel, IN, 1992.

McKeeney, J.L. and Keen, P.G.W., "How Managers' Minds Work," Harvard Business Review, Vol. 52, No. 3, pp. 79-90, 1974.

McLean, M. and Shepherd, P., "The Importance of Model Structure," Futures, pp. 40-51, February 1976.

Meredith, J.R., "Reconsidering the Decision Making Approach to Management," OMEGA, Vol. 12, No. 4, pp. 347-352, 1984.

Miller, G.A., "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," Psychological Review, Vol. 63, No. 2, pp. 52-60, 1956.

Miller, P.H., Theories of Developmental Psychology, Freeman, San Francisco, 1983.

Mintzberg, H., Raisinghani, D., and Theoret, A., "The Structure of 'Unstructured' Decision Processes," Administrative Science Quarterly, Vol. 21, pp. 246-275, 1976.

Mischel, W., "On the Interface of Cognition and Personality: Beyond the Person-Situation Debate," American Psychology, Vol. 34, pp. 740-754, 1979.

Mitroff, I.I. and Betz, F., "Dialectical Decision Theory: A Meta-Theory of Decision-Making," Management Science, Vol. 19, No. 1, pp. 11-24, 1972.

Mitroff, I.I., Emshoff, J.R., and Kilmann, R.H., "Assumption Analysis: A Methodology for Strategic Problem Solving," Management Science, Vol. 25, No. 6, pp. 583-593, 1979.

Mitroff, I.I. and Featheringham, T.R., "On Systemic Problem Solving and The Error of the Third Kind," Behavioral Science, Vol. 19, pp. 383-393, 1974.

Mitroff, I.I. and Kilmann, R.H., Methodological Approaches to Social Science, Jossey-Bass, San Francisco, 1978.

Mock, T.J., "A Longitudinal Study of Some Information Structure Alternatives," Database, Vol. 5, pp. 40-45, 1973.

Moody's Industrial Manual, Moody's Investor Service, New York, 1991.

Moore, J. and Chang, M., "Meta-Design Consideration in Building DSS," in J. Bennett, Building Decision Support Systems, Addison-Wesley, Reading, MA, 1983.

Nadler, G., "Human Purposeful Activities for Classifying Management Problems," OMEGA, Vol. 11, No. 1, pp. 15-26, 1983.

Naraynan, V.K. and Fahey, L., "The Micro-Politics of Strategic Formulation," Academy of Management Review, Vol. 7, No. 1, pp. 25-34, 1982.

Naylor, T.H., "Decision Support Systems or Whatever Happened to M.I.S.?," Interfaces, Vol. 12, No. 4, pp. 92-94, 1982.

Newell, A., "Heuristic Programming: Ill-structured Problems," in J. Aronofsky, Progress in Operations Research, John Wiley, New York, pp. 361-414, 1969.

Newell, A., "The Knowledge Level," Artificial Intelligence, Vol. 18, No. 1, pp. 87-127, 1982.

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

Newell, A. and Simon, H.A., "Computer Science as Empirical Inquiry: Symbols and Search," Communications of the ACM, Vol. 19, No. 3, pp. 113-126, 1976.

Nisbett, R. and Ross, L., Human Inference: Strategies and Shortcomings of Social Judgment, Prentice-Hall, Englewoods Cliffs, NJ, 1980.

Norton, P. and Yao, P., Borland C++ Programming for Windows, Bantam, New York, 1992

Nunamaker, J.F., Applegate, L.M., and Konsynski, B., "Computer-Aided Deliberation: Model Management and Group Decision Support," Operations Research, Vol. 36, No. 6, pp. 826-848, 1988.

Nunamaker, J.F., Chen, M., and Purdin, T.D.M, "Systems Development in Information Systems Research," Journal of MIS, Vol. 7, No. 3, pp. 89-106, 1991.

O'Leary, D., "Expert System Prototyping As A Research Tool," in E. Turban and P. Watkins, Applied Expert Systems, Elsevier Science, New York, pp. 17-31, 1988.

Paradice, D.B., "Causal Modeling as a Basis for the Design of an Intelligent Business Problem Formulation System," Unpublished Ph.D. Dissertation, Texas Tech University, 1986.

Paradice, D.B. and Courtney, J.F., "Controlling Bias in User Assertions in Expert Decision Support Systems for Problem Formulation," Journal of MIS, Vol. 3, No. 1, pp. 52-64, 1986.

Parker, B.J. and Al-Utaibi, G.A., "Decision Support Systems: The Reality That Seems Hard to Accept?," OMEGA, Vol. 14, No. 2, pp. 135-143, 1986.

Patel, V.L. and Groen, G.J., "Knowledge Based Solution Strategies in Medical Reasoning," Cognitive Science, Vol. 10, pp. 91-116, 1986.

Peng, Y. and Reggia, J.A., Abductive Inference Models for Diagnostic Problem-Solving, Springer-Verlag, New York, 1990.

Piaget, J., The Origins of Intelligence in Children, International University Press, New York, 1952.

Piaget, J., Understanding Causality, Norton, New York, 1974.

Pounds, W.F., "The Process of Problem Finding," Industrial Management Review, Vol. 11, No. 1, pp. 1-19, 1969.

Pracht, W.E., "An Experimental Investigation of a Graphical Interactive Problem Structuring Aid for Decision Support Systems," Unpublished Ph.D. Dissertation, Texas Tech University, 1984.

Quinlan, J.R., "Induction of Decision Tree," Machine Learning, Vol. 1, No. 1, pp. 81-106, 1986.

Ramakrishna, H.V. and Brightman, H.J., "The Fact-Net Model: A Problem Diagnosis Procedure," Interfaces, Vol. 16, No. 6, pp. 86-94, 1986.

Ramaprasad, A. and Mitroff, I.I., "On Formulating Strategic Problems," Academy of Management Review, Vol. 9, No. 4, pp. 597-605, 1984.

Ramaprasad, A. and Poon, E., "A Computerized Interactive Technique for Mapping Influence Diagrams MIND," Strategic Management Journal, Vol. 6, pp. 377-392, 1985.

Rangaswamy, A. and Federowicz, J., "Domain-Independent Decision Aids for Managerial Decision Making," Proceedings of Fourth International Conference on Information Systems, Houston, pp. 7-21, 1983.

Reiter, R., "A Theory of Diagnosis from First Principles," Artificial Intelligence, Vol. 32, pp. 57-95, 1987.

Reitman, W.R., "Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-Defined Problems," in M.W. Shelly and G.L. Bryan, Human Judgements and Optimality, John Wiley, New York, pp. 282-315, 1964.

Rogers, W., Ryack, B., and Moeller, G., "Computer-Aided Medical Diagnosis: Literature Review," International Journal of Bio-Medical Computing, Vol. 10, pp. 267-289, 1979.

Ross, D.T. and Schoman, K.E., "Structured Analysis for Requirements Definition," IEEE Transactions on Software Engineering, Vol. 3, No. 1, pp. 6-15, 1977.

Rowe, C., "Analyzing Management Decision-Making: Further Thoughts after the Bradford Studies," Journal of Management Studies, Vol. 26, No. 1, p. 29, 1989.

Sage, A.P., Methodology for Large-Scale Systems, McGraw-Hill, New York, 1977.

Sage, A.P., "Behavioral and Organizational Considerations in the Design of Information Systems and Processes for Planning and Decision Support," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 11, No. 9, pp. 640-677, 1981.

Said, K.E., "A MIS for Problem Detection, Diagnosis, and Evaluation," Managerial Planning, pp. 4-8, March/April 1978.

Sanders, N.R. and Ritzman, L.P., "Improving Short-Term Forecasts," OMEGA, Vol. 18, No. 4, 1990.

Schoennauer, A.W.W., Problem Finding and Problem Solving, Nelson-Hall, Chicago, 1981.

Schwenk, C.R., "Cognitive Simplification Processes in Strategic Decision-Making," Strategic Management Journal, Vol. 5, pp. 111-128, 1984.

Schwenk, C.R., "Information, Cognitive Biases, and Commitment to a Course of Action," Academy of Management Review, Vol. 11, pp. 298-310, 1986.

Schwenk, C.R., "The Cognitive Perspective on Strategic Decision Making," Journal of Management Studies, Vol. 25, No. 1, pp. 41-55, 1988.

Schwenk, C. and Thomas, H., "Formulating the Mess: The Role of Decision Aids in Problem Formulation," OMEGA, Vol. 11, No. 3, pp. 239-252, 1983.

Sharda, R., Barr, S.H., and McDonnell, J.C., "Decision Support System Effectiveness: A Review and An Empirical Test," Management Science, Vol. 34, No. 2, pp. 139-159, 1988.

Shortliffe, E., Computer-Based Medical Consultation: MYCIN, American Elesevier, New York 1976.

Shortliffe, E., Buchanan, B., and Feigenbaum, E., "Knowledge Engineering for Medical Decision Making: A Review of Computer-Based Clinical Decision Aids," Proceedings of IEEE, Vol. 67, No. 9, pp. 1207-1226, 1979.

Simon, H.A., Administrative Behavior, Macmillan, New York, 1957.

Simon, H. A., The New Science of Management Decisions, Harper & Row, New York, 1960.

Simon, H.A., "The Structure of Ill Structured Problems," Artificial Intelligence, Vol. 4, pp. 181-201, 1973.

Simon, H. A. and Barenfeld, M., "Information-Processing Analysis of Perceptual Processes in Problem Solving," Psychological Review, Vol. 76, pp. 473-483, 1969.

Simon, H.A. and Newell, A., "Human Problem Solving: The State of the Theory in 1970," American Psychological Association, pp. 39-50, 1971.

Smart, C. and Vertinsky, I., "Design for Crisis Decision Units," Administrative Science Quarterly, Vol. 22, pp. 640-657, 1977.

Smith, G.F., "Towards a Heuristic Theory of Problem Structuring," Management Science, Vol. 34, No. 12, pp. 1489-1506, 1988.

Smith, G. F., "Defining Managerial Problems: A Framework for Prescriptive Theorizing," Management Science, Vol. 35, No. 8, pp. 963-981, 1989.

Smith, G.F., "Heuristic Methods for the Analysis of Managerial Problems," OMEGA, Vol. 18, No. 6, pp. 625-635, 1990.

Sprague, R.H., "A Framework for the Development of Decision Support Systems," MIS Quarterly, Vol. 4, pp. 1-26, 1980.

Sprague, R.H. and Carlson, E.D., Building Effective Decision Support Systems, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Sprague, R.H. and Watson, H.J., "A Decision Support System for Banks," OMEGA, Vol. 4, pp. 657-671, 1976.

Stone, M., "Cross-Validation Choice and Assessment of Statistics Predictions," Journal of the Royal Statistical Society, Vol. 36, pp. 111-147, 1974.

Sundstrom, G.A., "Process Tracing of Decision Making: An Approach for Analysis of Man-Machine Interactions in Dynamic Environments," International Journal of Man-Machine Studies, Vol. 35, pp. 843-858, 1991.

Szolovits, P. and Pauker, S., "Categorical and Probablistic Reasoning in Medical Diagnosis," Artificial Intelligence, Vol. 11, No. 2, pp. 115-144, 1978.

Taylor, R.N., "Perception of Problem Constraints," Management Science, Vol. 22, No. 1, pp. 22-29, 1975.

Taylor, S. and Crocker, "Schematic Bases of Social Information Processing," in E. Higgens, et al., Social Cognition: The Ontario Symposium, Lawrence Erlbaum, Hillsdale, NJ, 1983.

Thomas, H., "Strategic Decision Analysis: Applied Decision Analysis and Its Role in the Strategic Management Process," Strategic Management Journal, Vol. 5, pp. 139-156, 1984.

Todd, P. and Benbasat, I., "Process Tracing Methods in Decision Support System Research: Exploring the Black Box," MIS Quarterly, Vol. 11, pp. 493-512, 1987.

Tsai, R.J., "An Investigation on the Impact of Task Characteristics and Cognitive Style on Cognitive Process in a Decision-Making Environment Information Systems," Unpublished Ph.D. Dissertation, University of North Texas, 1991.

Turban, E. and Watkins, P.R., "Integrating Expert Systems and Decision Support Systems," MIS Quarterly, Vol. 10, pp. 121-135, 1986.

Tversky, A. and Kahneman, D., "Judgement under Uncertainty: Heuristics and Biases," Science, Vol. 185, pp. 1123-1131, 1974.

Uthurusamy, R., Fayyad, U.M., and Spangler, S., "Learning Useful Rules from Inconclusive Data," in G. Piatetsky-Shapiro and W.J. Frawley, Knowledge Discovery in Databases, The AAAI Press, Menlo Park, CA, 1991.

Volkema, R.J., "Problem Formulation in Planning and Design," Management Science, Vol. 29, pp. 639-652, 1983.

Volkema, R.J., "Problem Formulation as A Purposive Activity," Strategic Management Journal, Vol. 7, pp. 267-279, 1986.

Wallace, W.A., Causality and Scientific Explanation: Classical and Contemporary Science, The University of Michigan Press, Ann Arbor, MI, 1974.

Warfield, J.N., Societal Systems: Planning, Policy and Complexity, John Wiley, New York, 1976

Weber, E.S., "Systems to Think With: A Response to 'A Vision for Decision Support Systems'," Journal of MIS, Vol. 2, No. 4, pp. 85-97, 1986.

Weber, E.S. and Konsynski, B.R., "Problem Management: Neglected Elements in Decision Support Systems," Journal of MIS, Vol. 4, No. 3, pp. 64-81, 1987.

Weber, E.U. and Coskunoglu, O., "Descriptive and Prescriptive Models of Decisionmaking: Implications for the Development of Decision Aids," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 20, No. 2, pp. 310-317, 1990.

Weber, R., "Toward a Theory of Artifacts: A Paradigmatic Base for Information Systems Research," Journal of Information Systems, pp. 3-19, 1987.

Weick, K., "Cognitive Processes in Organizations," in B. Staw, Research in Organizational Behavior, JAI Press, Greenwich, CT, 1979.

White, C.C., "A Survey on the Integration of Decision Analysis and Expert Systems for Decision Support," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 20, No. 2, pp. 358-364, 1990.

Woods, D.D., "Cognitive Technologies: The Design of Joint Human-Machine Cognitive Systems," AI Magazine, Vol. 6, No. 4, pp. 86-92, 1986.

Woolley, R.N. and Pidd, M., "Problem Structuring -- A Literature Review," Journal of Operations Research Society, Vol. 32, pp. 197-206, 1981.

Wright, P., "The Harassed Decision Maker: Time Pressure, Distractions, and the Use of Evidence," Journal of Applied Psychology, Vol. 59, No. 5, pp. 555-561, 1974.

Wright, H.A. and Bailey, A.W., Fire Ecology, John Wiley, New York, 1982.

Wright, H., Burns, J.R., and Chang, H., "An Expert System for Prescribing Burning," Rangelands, Vol. 14, No. 5, pp. 286-292, 1992.

Yadav, S.B. and Baldwin, D., "The Process of Research Investigations in Artificial Intelligence -- An Unified View," Working Paper, ISQS Dept., Texas Tech University, 1990.

Yadav, S.B. and Korukonda, A., "Management of Type III Error in Problem Identification," Interfaces, Vol. 15, pp. 55-61, 1985.

Young, L.F., "Right-Brained Decision Support Systems," Data Base, pp. 28-36, 1983.

Zmud, R.W., "Individual Differences and MIS Success: A Review of Empirical Literature," Management Science, Vol. 10, pp. 966-979, 1979.

APPENDIX: USER'S GUIDE TO THE PROBLEM DIAGNOSIS SUPPORT SYSTEM

A.1. Introduction

ISS/MPD is a software designed to support managerial diagnosis for semi-structured or unstructured problems that are repetitive. ISS/MPD supports various problem diagnosis activities including symptom detection, causal model development, and diagnosis generation. ISS/MPD helps you to systematically develop structural models, statistical models, and decision rules.

A.2. Hardware and Software Requirements

1. IBM PCs or compatible with SVGA card and monitor;
2. A hard disk with at least 1 MB of free space;
3. Microsoft Windows version 3.0 or later (with a mouse input device); and
4. dBASE III+ (for new application development)

A.3. Software Package

ISS/MPD consists of two executable programs. First, ISS_INST.EXE is a program that allows you to create new applications. This program accepts dBASE III+ data files and creates example data set and (optional) problem instances. Second, ISS_MPD.EXE is the main program that runs the applications. An application should be established in a subdirectory of the directory where ISS_MPD.EXE is located. The system stores problem instances under the subdirectories of the application directory. For example, if ISS_MPD.EXE is located in C:\ISS, an application can be stored in C:\ISS\APPLICA1. Then, the system may store a problem instance in C:\ISS\APPLICA1\INSTANC1.

A.4. System Description

This section describes the prototype system. Because the prototype system has many functions, it is not possible to illustrate each individual function in detail. The following describes the system's menu structure, key dialog, and user input methods.

1. MainWindow: organizes various views.

   Setting>>
   - 1.1. View Directory: Execute ViewDirectoryDialog
   - 1.2. Data Dictionary: Open a DataDictionaryWindow
   - 1.0. Process View: (Show the actions you have taken: Not implemented)
   - 1.3. Problem Instance: Execute ProblemInstanceDialog
   - 1.4. Example View: Open an ExampleViewWindow

   Problem>>
   - 1.5. Problem State: Open a ProblemStateWindow
   - 1.6. Goal State: Open a GoalStateWindow
   - 1.7. Discrepancy: Open a DiscrepancyWindow

   Model>>
   - 1.8. Structural: Open a StructuralWindow
   - 1.9. Descriptive: Open a DescriptiveWindow

   Diagnosis>>
   - 1.10. Alternative: Open a DiagnosisWindow
   - 1.0. Summary: (Develop a final diagnosis view: Not implemented)

   Help>>
   - 1.11. Menu Items: Explain the menu items on the main window
   - 1.12. About: Display information about the program

   Quit>>
   - 1.13. Exit to Windows: Save the application and terminate this program
   - 1.14. Exit to DOS: Save the application and exit to DOS

1.1. ViewDirectoryDialog: lists all available views and opens a chosen view. The system shows all available views in a list box. When you select a view from the list, the system opens the view in an appropriate window.

1.2. DataDictionaryWindow: creates new variables and performs filtering.

Variable>>

1.2.1. Create>>Define: Execute DefineDialog

DefineDialog: When you enter a new field name and its formula, the system creates a formula-based data field and updates examples.

1.2.2. Create>>Discretize: Execute DiscretizeDialog

DiscretizeDialog: You may add, delete, or modify discretization rules. When you push "Ok" button, the system creates a discretized data field and updates the example data view. The following windows can be opened during DiscretizeDialog.

GetValueDialog: If you press "Get Value" button, the system allows you to obtain an (exact or approximate) interval boundary by entering a cumulative probability.

QuartileWindow: If you press "Quartile" button, the system opens a quartile window showing the quartiles (10% increments), average, and standard deviation of the variable being discretized.

1.2.3. Create>>Quantify: Execute QuantifyDialog

QuantifyDialog: You may add or modify quantification rules. The system creates a quantified data field and updates the examples.

1.2.4. Create>>Associate: Execute AssociateDialog

AssociateDialog: You may add or modify association rules. The system creates an associated data field and updates the examples.

1.2.5. Examine: Display the field definition of a chosen variable

1.2.6. Exit: Close this window

Filter>>

1.2.7. Remove: Execute FilterDialog

FilterDialog: The system displays both operative and inoperative variables. You may move variables between the operative and inoperative variable lists. You can rearrange the display locations of the operative variables.

1.2.8. Insert: Execute FilterDialog

1.2.9. Perform: Execute PerformFilteringDialog

PerformFilteringDialog: When you choose a structural model, the system performs filtering based on the chosen structural model

1.2.10. Reset: Reset the data dictionary to include all the data fields

1.2.11. Arrange: Execute FilterDialog

Help>>

1.2.12. Menu Item: Explain the menu items on the data dictionary window

1.2.13. Screen: Explain the mouse input to the window's client area

Screen Input>>

1.2.14. Change Default Display: If you press the left button of the mouse on the cells in the second column (this column shows the variables being displayed) of the data dictionary window, the system changes the data dictionary setting to display discretized, quantified, associated, or original data fields.

1.3. ProblemInstanceDialog: opens a problem instance.

A list box shows all available problem instances. If you select a problem instance, the system automatically closes all instance-related data windows (i.e., problem state, goal state, discrepancy, and diagnosis views). If these views have been modified, the system asks you to save the views before closing, however. Until you select a problem instance, you cannot access the instance-related views.

1.4. ExampleViewWindow: displays examples as specified in the data dictionary.

1.4.1. Fast Backward (<<): Skip backward 20% of the total examples

1.4.2. Backward(<): Display the next example case

1.4.3. Exclamation (!): Close this window

1.4.4. Forward (>): Display the previous example case

1.4.5. Fast Forward (>>): Skip forward 20% of the total examples

1.5. ProblemStateWindow: provides a problem state view.

View>>

1.5.1. Retrieve: Retrieve a problem state view

1.5.2. Create New: Create a new problem state view
1.5.3. Save: Save the problem state view
1.5.4. Write: Write the problem state view under another name
1.5.5. Delete: Delete a problem state view
1.5.6. Exit: Close this window

Display>>

1.5.7. Default: Display as specified in the data dictionary
1.5.8. Original: Display the original data field values
1.5.9. Increments: Display increments
1.5.10. Ratio: Display in ratio
1.5.11. Historical States: Turn on/off the display of the historical states
TimeBaseDialog: The system asks you to choose a base time index. This dialog is used in 1.5.9, 1.5.10. and 1.6.9.

Help>>

1.5.12. Menu Items: Explain the menu items on the problem state window
1.5.13. Screen: Explain the mouse input to the window's client area

Screen Input>>

1.5.14. Change Value: If you press the left button of the mouse on a current problem state value, the system will open a small edit window allowing you to modify the value.

1.6. GoalStateWindow: provides a goal state view.

View>>

1.6.1. Retrieve: Retrieve a goal state view
1.6.2. Create New: Create a new goal state view
1.6.3. Save: Save the goal state view
1.6.4. Write: Write the goal state view under another name
1.6.5. Delete: Delete a goal state view
1.6.6. Exit: Close this window

Goal Values>>

1.6.7. Regression: Project goals using the linear regression technique
1.6.8. Smoothing: Project goals using the exponential smoothing technique
1.6.9. Search: Execute SearchGoalDialog

SearchGoalDialog: The system provides a box listing data query statements. You can edit, add, modify, and delete the statements. The system also allows you to edit the searching criteria in a similar fashion. "Display" button executes ListExampleRecordDialog.
ListExampleRecordDialog: The system shows the retrieved examples in a list box. When you press "Display" button after choosing an example, the system opens an ExampleViewWindow (with <<, <, >, >> operations disabled). You can select a particular record as a goal state.

1.6.10. Copy: Copy a goal state from a problem state for modification

1.6.11. Lower Boundary: Execute BoundaryDialog

1.6.12. Upper Boundary: Execute BoundaryDialog
BoundaryDialog: The system establishes the lower or upper goal boundaries as a percentage of the goal values

Display>>

1.6.13. Default: Display as specified in the data dictionary

1.6.14. Original: Display the original data field values

1.6.15. Difference: Display the differences between goal state values and the goal boundary values

1.6.16. Ratio: Display the ratios of the boundary values to the goal values

Help>>

1.6.17. Menu Items: Explain the menu items on this window

1.6.18. Screen: Explain the mouse input to the window's client area

Screen Input>>

1.6.19. Change Value: If you press the left button of the mouse on the cells in the first or last column (goal direction columns), the system changes their values. If a cell in the middle columns (goal state and goal boundary columns) is pressed, the system opens an edit window allowing you to modify the value.

1.7. DiscrepancyWindow: provides a performance discrepancy view.
View>>

1.7.1. Retrieve: Retrieve a performance discrepancy view

1.7.2. Create New: Create a new performance discrepancy view based on a problem state view and a goal state view
1.7.3. Save: Save the performance discrepancy view
1.7.4. Write: Write the performance discrepancy view under another name
1.7.5. Delete: Delete a performance discrepancy view
1.7.6. Update: Update a performance discrepancy view
1.7.7. Exit: Close this window

Display>>

1.7.8. Default: Display as specified in the data dictionary
1.7.9. Original: Display the original data field values
1.7.10. Difference: Display the differences between the problem state and the goal state
1.7.11. Ratio: Display the ratios of the problem state to the goal state
1.7.12. Problematic: Display/hide problematic variables
1.7.13. Favorable: Display/hide variables with favorable deviations
1.7.14. Insignificant: Display/hide insignificant variables
1.7.15. Unknown: Display/hide variables with unknown deviations

Help>>

1.7.16. Menu Items: Explain the menu items on this window
1.7.17. Screen: Explain the mouse input to the window's client area

1.8. StructuralWindow: provides a structural view.

View>>

1.8.1. Retrieve: Retrieve a structural model
1.8.2. Create New: Create a new structural model
1.8.3. Save: Save the structural model
1.8.4. Write: Write the structural model under another name
1.8.5. Delete: Delete a structural model
1.8.6. Exit: Close this window

Variables>>

1.8.7. Add: Add a variable to the structural model
1.8.8. Delete: Delete a variable from the structural model
1.8.9. Arrange: Change the display locations of variables

Analysis>>

1.8.10. Affects: Execute AnalyzeDialog with Operation = Affects

1.8.11. Affected By: Execute AnalyzeDialog with Operation = Affected
AnalyzeDialog: When you choose a set of variables, the system identifies all the variables that affect, or are affected by, the chosen variable(s).

1.8.12. Connected: Execute ConnectivityDialog
ConnectivityDialog: When you choose two variables from the system-provided list, the system explains how they are related.

1.8.13. Detect Cycles: Identify all causal cycles in the structural model

Help>>

1.8.14. Menu Items: Explain the menu items on this window

1.8.15. Screen: Explain the mouse input to the window's client area

Screen Input>>

1.8.16. Change Value: If you press the left button of the mouse on the matrix cells, the system changes the values in the sequence of (0) -> (1) -> (+1) -> (-1) -> (0).

1.8.17. Dependence Statistics: If you press the right button of the mouse on the matrix cells, the system provides you with the statistics measuring the dependence between the variables.

1.9. DescriptiveWindow: displays statistical model statements and rules.

View>>

1.9.1. Retrieve: Retrieve a descriptive model set

1.9.2. Create New: Create a new descriptive model set

1.9.3. Save: Save the descriptive model set

1.9.4. Write: Write the descriptive model set under another name

1.9.5. Delete: Delete a descriptive model set

1.9.6. Exit: Close this window

Create>>

1.9.7. Equation: Execute CreateEquationDialog
CreateEquationDialog: You can edit, add, modify, and delete data query statements. When you press "target variable" box, the system

identifies all candidate target variables (if a structural model is given). When you choose a target variable, the system identifies all variables that affect the target variable. You can include a subset of the variables as explanatory.

1.9.8. Rule>>Edit: Execute EditRuleDialog

EditRuleDialog: The system helps you to edit a rule statement. After stating the rule, you may press "Show" button to see how accurate the rule is over the example data.

1.9.9. Rule>>Induce: Execute InduceRuleDialog

InduceRuleDialog: This is similar to CreateEquationDialog, except that you must specify which rule induction algorithm to use. You have two choices: revised ID3 method and attribute-value oriented method.

Structural>>

1.9.10. Retrieve: Retrieve a structural model into the descriptive model set

1.9.11. Create New: Create a structural view of the descriptive model set

1.9.12. Save: Save the currently embedded structural model

1.9.13. Write: Write the structural model under another name

1.9.14. Delete: Remove the structural model

Page>>

1.9.15. Next: Display the next page of the descriptive model view

1.9.16. Before: Display the previous page of the descriptive model view

1.9.17. Start: Display the first page of the descriptive model view

1.9.18. End: Display the last page of the descriptive model view

1.9.19. Current Page: This menu item tells the current page

Delete>>

1.9.20. Equations: Delete all statistical models

1.9.21. Rules: Delete all rules

Help>>

1.9.22. Menu Items: Explain the menu items on this window

1.9.23. Screen: Explain the mouse input to the window's client area

Screen Input>>

1.9.24. Open a StatisticalModelWindow: If you press the left button of

your mouse on a statistical model statement in the descriptive model window, the system opens a StatisticalModelWindow.

1.9.25. Execute ModifyRuleDialog: If you press the left button of your mouse on a rule, the system executes ModifyRuleDialog.

1.10. DiagnosisWindow: provides an alternative diagnosis view.

View>>

1.10.1. Retrieve: Retrieve an alternative diagnosis view

1.10.2. Create New: Create an alternative diagnosis view based on a descriptive model view and a performance discrepancy view

1.10.3. Save: Save the alternative diagnosis view

1.10.4. Write: Write the alternative diagnosis view under another name

1.10.5. Delete: Delete an alternative diagnosis view

1.10.6. Update: Update the alternative diagnosis view

1.10.7. Exit: Close this window

Help>>

1.10.8. Menu Items: Explain the menu items on this window

1.10.9. Screen: Explain the mouse input to the window's client area

Screen Input>>

1.10.10. Change Value: If you press the left button of our mouse on a cell in the first two columns (problem state and goal state values), the system opens an edit window for you to change the value.

1.10.11. Generate Diagnosis: If you press the right button of our mouse on a cell in the first two columns, the system estimates the variable's value based on the descriptive model set.

1.10.12. Find Causes: If you press the right button of our mouse on a cell in the last column (displays discrepancy), the system identifies causes for the deviations.

1.11. StatisticalModelWindow: displays a statistical model.

View>>

1.11.1. Modify: Execute ModifyEquationDialog

ModifyEquationDialog is a CreateEquationDialog with filled-in dialog box contents (see 1.9.7).

1.11.2. Delete: Delete the statistical model statement

1.11.3. Exit: Close this window

Structural>>

1.11.4. Signs: Open a SignComparisonWindow

SignComparisonWindow: This window compares the signs of the regression coefficients with the signs in the structural model.

1.11.5. Analyze: Execute ConnectivityDialog (see 1.8.12)

1.12. ModifyRuleDialog: allows you to modify a rule.
ModifyRuleDialog is identical with EditRuleDialog with one additional option, i.e., delete the rule (see 1.9.8).