Challenges for Decision Support Systems*

Key words

Qualitative Reasoning, Qualitative Optimization,
Groupware Technology, Knowledge Base,
Distributed Model Management

Sulin Ba
Andrew B. Whinston
The University of Texas at Austin

1. Introduction

During the past 25 years, great progress has been made in research and commercial applications of decision support systems (DSS). Conceived originally as the application of computing technology to support decision making, DSS research focused on the implementation of tools from operations research. By allowing end users to state business problems in a higher level language and use software to translate requests, build suitable models, access required databases, integrate and execute models, and finally provide answers to the user, decisions can be made in a more effective manner.

While there are still many challenges and opportunities within this paradigm, significant changes have taken place in the environment surrounding DSS that radically new approaches are required. These changes are based on end-user demand and technological changes. Instead of focusing on highly structured aspects of company operations that can be modeled using operations research tools, there is a need to analyze organization wide issues that incorporate a wide range of knowledge representations and data types. Companies face complex decision problems which must be resolved collectively by several individuals that involve multiple phases, including the specification of the problem, discovery and support for alternatives, eventual solution and implementation. There is, in effect, a complex process that underlies decision making that can be supported by computers.

Modern management of organizations in a complex environment requires managers to look at problems in a non-myopic fashion, recognizing all the significant interactions, including cross-functional relationships. One of the major purposes of an organization wide computing system is to provide a decision support tool for strategic decision making within a particular organization and to improve the organization's ability to make decisions that take into account all the relevant factors. Given a query describing a problem, the system should be able to

retrieve and explore relevant information, derive a valid answer to the query, and provide an explanation for the answer. The question is how to build such a decision support system that is focused and integrate diverse corporate wide knowledge to specific problems faced by a manager.

One of the motivations for building an organization wide decision support system is from Kaplan and Norton (1992, 1993). They show that organizational systems exhibit a significant degree of interdependency across functional areas, that requires decision makers to identify relevant relationships that contribute to a given corporate goal across functional boundaries. For example, senior managers need to consider all important operational measures to see whether improvement in one area may have been achieved at the expense of another. This decision-making process could be supported by an intelligent DSS that stores and manages organizational knowledge using appropriate model composition approaches. The system should be able to help managers focus their strategic vision and indicate to managers how they can improve performance from what happened last period, which means that the system functions as the cornerstone of a company's current and future success.

Haeckel and Nolan (1993) also point out that companies must be able to respond to threats in real time. They should be able to use an enterprise model to represent the operations of an entire business. Based on this model, expert systems, databases, software objects, hypertext files, and other technical components could be integrated to give top management the flexibility of modifying the business plan based on changes in external conditions, monitoring the performance of delegated responsibilities, etc. There are, however, many factors in organizations which cause difficulty in the decision making process. On the technical side, the complexity arises from the uncertainty and dynamics of new technologies and decision problems. On the organizational side, people have different opinions, values, desires, and motivations (Spetzler and Howard 1993). In most

* Invited Paper.

cases, much debate will precede any decision that is reached. Too many interrelated factors and alternatives exist, for which detailed explanations and "what if" questions need to be explored. Therefore, a DSS that can respond efficiently to this complexity is needed. The system should not only tell managers "how we do things around here" and "why we do things this way" but also show managers "how we change the way we do things".

One important fact that we have to recognize in developing a new DSS framework, is that companies are working more and more in a distributed environment. Bosman (1983) points out that, because decision support systems are integral participants in decision processes, the issue of coordination of participants is an important one for the DSS field. In effect, communication, collaboration, and coordination have become cornerstones of a company's activities (Ellis et al 1991). Effective collaboration demands that people share information. Good coordination enhances the effectiveness of communication and collaboration. New technologies in networking and distributed computing make it possible for organizations to achieve these goals. For example, the explosive growth of worldwide networks and their interconnection through Internet are already well recognized. Increasingly, the users of services on Internet are not engineers or programmers, but managers, customers, and students who want fast, accurate, and cheap access to this enormously informative knowledge pool for educational purposes and improved decision making (Hamalainen et al 1993). With millions of users on the Internet, communication services — such as electronic mail, mailing lists, and news and resource sharing services — such as file transfer and remote login, and resource discovery services for people, files, documents, and general network resources, become easily accessible, greatly increasing the amount of shared information.

Groupware technology, arising from the merging of computers and large information bases and communications technology, integrates a class of applications for groups and organizations to facilitate human interactions (Ellis et al 1991). CSCW (computer-supported cooperative work) systems are a recent technology intended to support group work on unstructured problems. They support not only group discourse tasks — by structuring the argumentation — but also provide a semi-formal documentation of the processes used to arrive at a decision.

DSS research has been focusing on query answering: users initiate a query and the system yields an output based on the modeling result. It is somehow a one-shot

process. However, we should recognize that decision support is a complicated collaborative problem-solving process. It is an augmentation of human capabilities in problem analysis and decision making. As we indicated above, decision making in most cases is a group activity. Individuals in the group express their positions on a certain issue and support their arguments with reasons for or against their positions. There are various phases in this decision making process. When a user first initiates a query, the system has to analyze what the problem is, define and select the best possible approaches. It seeks methods to resolve the differences among participants. Decision support could be reached either through unstructured discussion or formal modeling. Multiple problem solving methods (for example: spreadsheet, compositional modeling, influence diagrams, mathematical programming, etc.), might be needed to resolve differences and enhance the process of solving complex decision problems (Bhargava et al 1993).

In this paper, we intend to give an overview of some major issues that are facing DSS research. Some background on DSS and model management is provided in section 2. In section 3, we discuss the qualitative reasoning method which could provide a powerful technique for DSS to deal with unstructured or semi-structured qualitative organizational knowledge. Some data representation and qualitative optimization issues are also discussed. Section 4 focuses on the organizational knowledge base which is essential for a new DSS framework. Knowledge revision is discussed in section 5. In section 6, we stress that organizations are working more and more in a heterogeneous distributed computing environment, which must be recognized by a new DSS framework. Although we currently do not have solutions for all the problems, we do hope this paper will provide some direction for future DSS research.

2. A Brief Review on DSS and Model Management

As Bonczek et al (1984) point out, the DSS field is interdisciplinary in nature, drawing upon work in database management, language processing, operations research and modeling, artificial intelligence, formal logic, and other disciplines. Ideally, a DSS is intended to utilize knowledge about a particular application area to help decision makers solve ill-structured problems. It is always specifically oriented toward that kind of information processing activity which constitutes a decision process.

The traditional DSS framework has three major components: a language system (LS), a knowledge system (KS), and a problem processing system (PPS) (Bonczek et al 1981). The LS and KS are systems of representation, whereas the PPS is a software system which acts on the requests issued by the LS with respect to the current content of the KS. In other words, given a problem (stated with the LS) and problem-domain knowledge (represented in the KS), the PPS attempts to derive a solution which supports (enhances, facilitates, or makes possible) a decision-making process (Bonczek et al 1984). Besides many domain-specific PPSs, researchers have been working on developing a generalized problem processor (GPPS) which contains no application-specific knowledge embedded in its software. The study of GPPSs constitutes an important basis for DSS research because it affords a valuable background for understanding, comparing, developing, and discussing domain-specific PPSs.

Model management is an important area of DSS research. Model management systems constitute a class of software designed to support the construction, storage, retrieval, and use of models in the context of decision support systems (Balasubramanian et al 1992). The purpose of a model management system is to insulate the users of a DSS from the physical aspects of model base storage and processing. Research in model management has mainly focused on three topics: the structure of model bases, model base processing, and the organizational environment of model management systems (Blanning 1993). In terms of model base structure, one effort is structured modeling which provides a framework, not only for model structuring, but also for model base documentation (Dolk 1988), the development of libraries of reusable model components (Geoffrion 1987), and object-oriented model management (Lenard 1989). Modeling concepts are organized in a directed acyclic graph and mapped into the leaves of a tree representing a hierarchical decomposition of the modeling software (Geoffrion 1987). Another approach to model management is based on the relational approach to data management (Merrett 1984). Model management is the organization and processing of virtual relations (Blanning 1984a) which form the models on demand. Both approaches require a well-structured representation of model base, limiting the applicability of decision support tools.

A variety of techniques on model base processing have emerged during the past decade, most based on artificial intelligence, for interfacing models with each other,

integrating models with each other, helping users to construct models, and helping users to interpret model outputs. Besides the traditional command line and iconic input and printed and graphical output, user interfaces have begun to include natural language front ends that emphasize user-oriented dialogue (Blanning 1984b). Model integration discusses how to identify and properly combine models and other DSS components (such as data files and data analysis procedures) needed to answer a specific query. The integration of distributed model bases has become an important research issue (Muhanna 1990). In the model construction area, some intelligent systems have been developed for selecting a model from a set of existing models (Banerjee and Basu 1990, Klein 1986) and for constructing models when no existing model is appropriate to the task at hand (Hwang 1985). However, these systems are specialized to certain problem domains, such as selecting the well-structured and highly quantitative statistical technique, forecasting model, or econometric model.

As we indicated above, most decision support systems that have been developed to date focus on quantitative problems. Qualitative information is generally ignored. Intraorganizational relationships, however, play an essential role in any decision making process. Therefore, we need to develop a system that is more organization oriented and takes into account qualitative organizational knowledge, as well as other sources of information such as the environment of the organization (Elofson and Konsyski 1991), decision processes of individual managers (Carlson and Ram 1990a, 1990b), interactions between managers (Blanning 1991, Carley 1990), and messaging systems within organizations (Malone et al 1987).

3. Qualitative Analysis

The need to study qualitative analysis and reasoning arises in many fields, such as economics, management science, physics, and artificial intelligence. In any field that attempts to describe and/or predict the behavior of a dynamic system, the theoretical optimal solution would be to construct a completely explicit, fully determined and deterministic mathematical model, which could then be analyzed with any desired accuracy, if not always theoretically, at least by means of numerical algorithms. Any further problems which carry out such a task would then be related to the cost and speed of performing calculations to achieve the required accuracy.

In reality, however, our knowledge is often incom-

plete, which makes it difficult or impossible to construct such a model. It might be that the laws that govern the situation are not completely known, or the initial data is not completely specified, simply because it is too expensive to gather enough initial information. For a more accurate prediction or explanation, more information might be needed, resulting in a higher cost. The tradeoff of information cost and prediction quality is studied in Balakrishnan and Whinston (1991). Sometimes we might be in a situation where it is even considered desirable to keep the initial situation vague in order to obtain more general conclusions that apply not only to one specific initial situation, but to a class of such situations.

It would not be practical or possible to perform rigorous mathematical theorem proving in every situation that arises in model construction and analysis. What is needed is a more flexible mechanism that can accept both imprecise data and imprecise laws of evolution as input, and to deduce from them the consequences that are the best and most precise that can possibly be deduced from the given data.

Research in qualitative reasoning has produced a number of models that draw logical conclusions or enumerate possible states of behavior in the field of physics and engineering (Kuipers 1986; Williams 1991), with some recent models also incorporating partially quantitative data that can be used to include some states of behavior (Kuipers and Berleant 1990). The Rules-Constraints-Reasoning (RCR) algorithm, developed by Kiang et al (1993), deals with basically numerical data which may involve incomplete knowledge. It is a method of reasoning with imprecise knowledge aimed mainly at discrete dynamic systems. It proposes a model representation which is essentially an interval-based abstraction of difference equation systems. Instead of describing any quantity by giving its precise numerical value, it is assumed that this value is only known to belong to a given set. The description of that set then becomes precisely given numerical information that is further processed, which yields other sets at future times as predictions provided by this algorithm. In the next section, we give a brief review of RCR that will give readers a general background about this algorithm.

3.1 Rules-Constraints-Reasoning (RCR)

Developed by Kiang et al (1993), RCR is a method of reasoning with imprecise knowledge that is aimed mainly at discrete dynamic systems. It proposes a model representation which is essentially an interval-

based abstraction of difference equation systems. The following is a brief description of RCR (see Kiang 1990 for a detailed and formal description):

Quantities of interest are represented as discrete functions of time,

$$A_j(t), t \in T\{t_0, t_1, \dots\} \text{ and } j \in J = \{1, 2, \dots, n\},$$

whose exact forms are not known. In most applications we are only concerned with modeling a certain number of periods. For practical purposes, we may assume that T which denotes the set of time points is finite, that is, $T = \{t_0, t_1, \dots, t_m\}$. We also assume that at some $t A_j(t)$ is known to lie in some set E_{jt} which is a subset of R , a union of finitely many intervals. The sets E_{jt} are viewed as qualitative values which need to be understood as semi-numerical characterizations of qualitative information. They are usually represented by closed intervals on the real line:

$$E_{jt} = [a, b] = \{x \in R | a \leq A_j \leq b\}$$

which means that at time t , we have $a \leq A_j \leq b$. They basically carry the knowledge we have about quantities. The more precise our knowledge is, the smaller they will be.

The function $A_j(t)$ reflect the dynamics and interdependence of a system and are dependent on the whole history of the process, which includes their previous values $A_j(t-1)$, $A_j(t-2)$, etc., as well as previous values of other quantities of interest $A_k(t-1)$, $A_k(t-2)$, etc. ($k \neq j$). Information about these previous functions $A_j(p)$, $j=1, \dots, N$ and $p < t$, is imprecise. All we know is that they will produce a value that lies in some sets E_{jp} . Therefore, the set E_{jt} which constrains the value of $A_j(t)$ is a function of all previous sets E_{jp} .

There are three types of rules in RCR which are used as specific operations to find the sets E_{jt} : past rules, present rules, and constraints. Past rules are functional relationships which determine a quantity $A_j(t)$ solely in terms of previous quantities, that is, a certain A_j at a certain time t is a given function of some or all of the A_p at times q where $1 \leq p \leq n$ and $0 \leq q < t$. Present rules are similar but allow previous and additionally present quantities to determine a quantity $A_j(t)$, which means that q could also be equal to the present time t . Finally, constraints are further restrictions on the individual $A_j(t)$ expressed as simple equalities or inequalities. When rules are evaluated, the $A_j(t)$ are replaced with their corresponding covering sets E_{jp} . Given initial sets E_{j0} , the method computes all subsequent sets E_{jp} of the model horizon using a combination of constraint satisfaction and constraint propagation. Applying the past rules in period 0 propagates the initial information E_{j0} ,

$j=1, \dots, n$, to the next period, thus establishing new information about the quantities $A_j(1)$. Applying the present rules and the constraints of period 1 refines this information and ultimately determines the new sets E_{j1} , $j=1, \dots, n$. Since present rules and constraints are interdependent, several rounds of evaluation might be necessary until convergence. A convergence proof of the RCR method can be found in Kiang (1990).

The forms of the functional relationships used in rules are usually restricted for computational reasons. Only usual arithmetic operations and monotonic functions like exponential and logarithms are admissible. Interval computations are used to evaluate the rules. The functional relationships do not need to be specified completely. It is sufficient to provide monotonic upper and lower bounding functions (envelopes) on a quantity $A_j(1)$ from which intervals are constructed that cover the actual value. Where they are undefined, they are treated as providing no constraint. No monotonicity assumptions are made for the true, unknown functional relationships. For each quantity of interest, the method generates a series of intervals that show how it changes, that is, how its range of possible values changes from time to time (Kiang 1990; Hinkkanen et al 1993).

3.2 Data Representation in Qualitative Analysis

Suppose that data is represented in a monotonic form, for example, a certain variable is increasing or decreasing on a certain interval. A system taking this form of data will give its conclusions also in the form of monotonicity. Typically, there will be a huge number of possibilities for the resulting dynamic behavior, many of which may never occur in a practical situation. Therefore, the RCR algorithm uses numerical data representation dealing with sets (described by means of finitely many numbers), rather than listing all the states of behavior.

However, there may be situations where it is desirable to incorporate into the model description not only numerical data but logical type data as well. The question arises as to how to best handle such data. In recent years, a connection has emerged between logical deduction and integer programming (cf. Hooker 1988). By formulating problems where one needs to satisfy a set of logical conditions, as problems of solving a set of inequalities in integers, one can profitably solve the problem in a relatively brief period of time by some integer programming methods.

The RCR algorithm deals with qualitative reasoning when both the initial information and certain specifications of the system may not be completely

known. The uncertainty is expressed by saying that, a given time, a quantity, whose value need not be exactly known, belongs to a set of a suitable type. For computational purposes, the set should be describable by finitely many parameters. So, for a subset of the set of real numbers, unions of finitely many intervals seem suitable. In practice, intervals are used so far. The algorithm then amounts to a propagation of intervals, starting with those initially given. [This is not the same as the so called interval arithmetic. The idea of propagating information using intervals has also been considered by Davis (1987) mainly in connection with certain linear systems.] The interval obtained for a quantity at a given time then gives the best information about that quantity that can be obtained. The accuracy may be improved by using more than one interval, creating an issue of how much gain can be expected from making the computations longer and more complicated. This algorithm does not incorporate logical (non-numerical) information at this point.

It does not seem practical to produce all possible behaviors indicated by logical-type variables. Suppose we have a situation where ten intervals and two variables are studied. During each time interval considered, a logic variable can be true or false. There are $4^{10} = 1,048,576$ possible ways for this behavior to occur. Even if 99% of them are ruled out, more than 10,000 still remain. It seems doubtful that a human being inspecting the answer would be able to comprehend such an answer. Therefore, asking a question like "What are all the possible states of qualitative behavior that could arise from a given set of assumptions" does not seem very useful. It might be more practical to ask what the final result is after a certain period of time, in other words, if a statement is true or false at that time. Looking at the problem this way produces not only the advantage of cutting down the length of output, but also allows the kind of logical reasoning that could apparently be phrased in terms of integer programming.

We need to develop a unifying data representation language which can represent, in addition to precisely known numerical and logical data, qualitative knowledge and incompletely known data. This would include knowledge that is currently represented using various languages in logic. It seems that, in order to incorporate statements of logical type, it will be desirable to code all such information numerically, so that ultimately, the computational system will only have to deal with numbers. This should allow us to tie the numerical and logical components of the system together, making the

ranges of certain quantities or the bounds for certain functions dependent on the truth values of certain logical variables. The description of logical conditions should be attainable in many ways, and many possibilities could be allowed in the system for added flexibility. For example, in circumscription (Lifschitz 1993), logical conditions can be described by giving some main rules and then a number of exceptions. This is analogous to describing upper and lower bounds for functions by formulas that involve something simple most of the time, but a few different function forms at a number of "exceptional" intervals.

When introducing logic variables, we need to incorporate variables that take only integer values (possibly only 0 and 1 as in the case of logic), and therefore deal with the case when the most natural type of set containing some possible values of a variable is not a full interval of the real axis, but a discrete set. Data representation involving both numerical and logical (say 0 or 1 valued) or more generally integer valued variables could be handled using Cartesian products or finite unions of Cartesian products in case of several variables and many conditions. Thus, the case when a logical variable takes a value 1 and simultaneously, or conditionally on the logical variable being equal to 1, a numerical variable is known to lie on the interval $[4, 6]$, would be expressed using the Cartesian product $\{1\} \times [4, 6]$ (if the logical variable can be 0 in the particular situation considered and is 0, the numerical variable might belong to another interval, say $[3, 5]$, and the whole thing would be expressed using $(\{0\} \times [3, 5]) \cup (\{1\} \times [4, 6])$).

One of the challenges is to make sure that we can smoothly integrate genuinely numerical methods with those of integer programming in order to deal with logical information in a unified fashion. It is worth noting that one way to deal with integer information is to consider the corresponding "continuous" problem for real numbers and check at the end of the computation if any integer (or 0 or 1 valued) solutions were generated. Many methods of integer programming proceed using this idea.

Another important issue is the development of theoretical principles needed to develop a deductive system that could be used to start with given system specification and initial conditions expressed in terms of the unifying language, and deduce the state of the system at a future time.

3.3 Qualitative Optimization

Suppose that we are using the RCR algorithm to

predict the behavior of a partially known dynamic system. One of the variables, L , is initially restricted to an interval $[c, d]$. Consider a variable M , which may or may not be the same as L , at a given time t . The algorithm will predict that then the values of M will lie in an interval $[a, b]$. Now, the numbers a and b will be obtained by applying the RCR computations and they depend on the values of c and d .

Instead of considering c and d to be given and fixed, we now assume that they depend on an external variable x (which is not part of the model so far). Then all quantities of the model that were obtained by using c and d become functions of x . Hence the question arises as to how to optimize the values of M at time t . In an application to economics or management, the variable M might, for example, be related to profits, income, or the stability of the enterprise.

To optimize the interval $[a, b]$, one could obviously use many quantities. Suppose that we want to make the lower bound a as high as possible. In other words, we want to maximize a as a function of x . The calculation that yields a as a function of x is usually so complicated that it is not feasible to expect to find an explicit formula for a in terms of x . Here is the challenge of finding computational procedures to solve this extremal problem.

As we suggested in the last section, logical variables would be involved in many cases. Functions/quantities to be optimized, as described above, could depend on logical variables. In general, they would be expected to depend on both numerical and logical variables, resulting in combined or hybrid optimization.

The current directions taken by businesses offer motivation for the study of qualitative optimization. Many corporations are interested in or have undertaken enterprise modeling, to keep track of the development of the organization, financially and otherwise. Effectiveness demands the setting of concrete corporate/organizational goals. To achieve them, different lines of strategy can be considered. Alternative states of planning and behavior arise and must be determined. For example, the company may be simply asking whether or not a certain loan should be taken out, and this reduces to a question involving a logical variable.

Thus, there could be a number of logical and numerical variables whose values are to be chosen when the senior management plans the future of the organization. We can then use the (usually incompletely known) system and optimize some given quantity at a future time to determine the best configuration of logical and

numerical variables to be chosen at the present time. When constructing models like this for a large organization, the number of variables and functions involved in the model will obviously be very large and it will often be so complicated that it would prove difficult for someone to make decisions on a "common sense" basis. Therefore, it is important to have a computational system to assist in decision making, by giving the best conclusions that can be obtained, in view of the incomplete information given to the model, and to make choices that will optimize key quantities chosen with regard to the corporate goals.

Therefore, the theoretical development of an optimization approach to qualitative systems and the theoretical development of the foundations of robust optimization (optimization of a quantity in an incompletely known system) are needed. Logic variables could also be used as control variables with respect to which a given quantity in the system is to be optimized, the final result being a hybrid situation where the control variables can be of numerical or logical type, and where the concept of optimization might include a combination of optimizing a numerical quantity, and taking into account some logical variables for which a configuration deemed to be optimal is sought. This involves defining a preferential order among the configurations that one may obtain for such variables.

A simple example of qualitative and logic optimization would be able to show the flavor of the issues that we mean. Consider the financial situation of a company, described by the variables Cash $C(t)$, Sales $S(t)$, Inventory $I(t)$, Profit $P(t)$, Debt $D(t)$, and Stability $X(t)$, which is defined in terms of other quantities. Also there is the logic variable Raid $R(t)$, which is true if, at time t , the company buys another company of a size whose qualitative description is assumed to be given. We assume that the model gives equations or inequalities describing how these quantities are related to each other and in particular, how the condition $R(t)=1$ will affect the other quantities. Let us assume, for example, that we try to maximize the midpoint $q=(a+b)/2$ of the interval $[a, b]$ containing the possible values of $X(5)$ (stability after 5 years from the starting point). As control variables that may be varied in this experiment, we may take, for example, the limits for the intervals containing $I(0)$, $R(0)$, and perhaps some or all future values of $R(t)$ for $1 \leq t \leq 4$. Now q will depend on both numerical and logic variables. If we generalize and expand the model, and define X to depend also on some logic variables, then both the control variables and the

variables to be optimized would involve a mixture of numerical and logic variables. Thus, we need an algorithm able to handle such a very general situation.

4. Organizational Knowledge Base

As we mentioned in section 2, the three major components in a traditional DSS framework are the language system, the knowledge system, and the problem processing system. So far, knowledge contained in the KS of most DSS has been very quantitative and relatively well-structured. Those systems require the user to make all the modeling decisions, for example, which aspects of the problem being modeled should be included, at what level of detail, what time scale is appropriate, etc. They are more focused on quantitative modeling and are limited to well defined, highly structured problems, such as production scheduling and inventory control. However, "to remain generic in the face of technological advances, it is necessary that a framework does not enforce some preconceptions about what the KS can contain, what knowledge management capabilities the PPS can exercise, or what the elements of LS and PS can be." (Chang et al. 1993).

The qualitative and quantitative reasoning methods described in the previous sections can be taken into a DSS setting. A new DSS framework has to recognize that organizational knowledge is represented in different forms, some is quantitatively based, but other kinds of knowledge can only be represented in qualitative form. It is very complex in terms of the level of detail and level of analysis. It contains heterogeneous types which could be dynamic or static, qualitative or quantitative. For example, the knowledge base for a production and inventory distribution system could be quantitative, whereas some organizational relationships could be represented in qualitative form, such as "If you increase partnership, productivity will increase." There could also be statements which are expressed in logic form: "Don't cut off service to an elderly customer before x months." We therefore need to extend the traditional DSS framework to include the part of organizational knowledge which is qualitative in nature. The resulting knowledge base will be heterogeneous, incorporating both quantitative and qualitative knowledge and encompassing organizational processes, general knowledge of management theory, company specific expert knowledge, internal and/or external empirical data, and engineered knowledge derived from previously built models (Hinkkanen et al 1993). A main issue here is how to represent and organize this knowledge.

In an earlier work, Ba et al (1993) have proposed a system whose reasoning about a particular organization is based on a library of model fragments representing significant organizational processes and phenomena from different perspectives and different levels of detail. When modeling a certain organizational phenomenon, it is crucial to focus on the relevant aspects of the situation under investigation, that is, to include all the relevant objects and constraints, exclude irrelevant ones and ignore unnecessary details. Accomplishing this requires access to multiple sets of heterogeneous models which differ in several dimensions, some of which might even be mutually inconsistent. We need to address the issue of model representation and model organization, that is, we need a language for expressing relationships of different kinds, and for expressing underlying assumptions controlling their applicability. The model library should be organized as a set of model fragments encompassing general domain knowledge and organization-specific knowledge, and should be structured in a way that reflects the organization's topology. The former type of knowledge could be obtained from research results in the organizational behavior field, which tries to formulate theories about organizations in general, that is, to find relationships that help understand the behavior of wide classes of organizations. Since those relationships are supposed to hold for any particular organization of the class, they tend to be very qualitative in nature. Organization-specific information, on the other hand, is derived from historical data and experience accumulated within a particular company, and therefore, tends to be much more precise, and is often encoded in quantitative, management science/operations research (MS/OR) type models, like optimization models, simulation models, and forecasting models. Another important issue is model reusability, because many different modeling projects could benefit from using previously formulated models or parts thereof. This becomes even more compelling when working in an organization-wide environment, where many questions need to be analyzed using information originating from different sources across functional boundaries of the organization. The task of organizing organizational knowledge into semi-independent, reusable model fragments is a crucial one for enabling an enterprise wide DSS to compose useful, problem-specific new models by integrating relevant, existing model components under a variety of different modeling circumstances.

The observation that a model consists of more than just a set of relationships, because it always assumes a

particular modeling context, leads us to a definition of a model fragment where the modeling assumptions are explicitly and separately expressed from the actual relationships and constraints. Hence, we argue for a representation language which represents the underlying assumptions of a model fragment and its constituting relationships. Each model fragment would contain two sections, one containing the specification of modeling assumptions (conditions), and the other containing the actual constraints and relationships that apply if all conditions in the assumptions section are satisfied.

The domain model fragments would take the following form:

```
fragment <NAME> (output port)
{description of the functionality of the model fragment}
conditions
precondition-specifications
relations
relationship-specifications
end
```

where <NAME> is an expression designating an instantiation of the model fragment, *output port* is a list of the variables (quantities) which are computed by the model fragment, and which can be shared with other fragments. The conditions section contains *precondition-specifications*, which define the modeling assumptions that the instantiation of a model fragment depends on. We assume that internally, that is, within a single model fragment, the relationship specifications in the relations section are of a homogeneous type. Across model fragments, we permit heterogeneous relationship specifications by using several modeling languages (see the Appendix for an illustrative example of parts of a domain theory).

For example, in the following model fragment:

```
fragment MNF-1 (EXPENSE)
{manufacturing model determining total production cost}
conditions
consider (exists (SALES)),
model-type (qual),
simulation (dynamic)
relations
EXPENSE=FIX COST+VAR COST,
VAR COST=UNIT COST*SALES,
constant (UNIT COST)
constant (FIX COST)
end
```

EXPENSE is the output variable that will be computed

if the model fragment MNF-1 is activated. For this model fragment to be instantiated, however, three conditions have to be met: the composite model has to include in the same model a model fragment which has SALES as the output variable; all the relationships and the output are to be represented qualitatively; and the simulation process is dynamic. If this fragment is indeed activated, then two definitional relationships apply: $\text{EXPENSE} = \text{FIX COST} + \text{VAR COST}$, $\text{VAR COST} = \text{UNIT COST} * \text{SALES}$, with the variables represented qualitatively.

5. Knowledge Revision

We need to recognize that human (as well as organizational) knowledge is nonmonotonic. We humans make many assumptions about the world around us and assume that these assumptions are true. Later on, when a contradiction is encountered, we can smoothly accommodate corrections to the assumptions and easily discard old conclusions in favor of new evidence. This means that our beliefs of what is currently right depends on the current stage of our knowledge, thus they, in turn, change non-monotonically, too. Organizations are no exception. Decision makers base their actions on their current stage of knowledge and current set of beliefs, which must be updated continually, since both internal and external organizational environment change constantly. For example, a company might have decided to open another plant in Singapore. However, their most up to-date analysis shows that it would not be beneficial, since the Singapore government has just passed a new policy which imposes many restrictions on foreign plants in that particular business. In this case, the company would have to accommodate this new information and give up their original plan.

Ideally, decision makers would always strive to make rational decisions based on rational attitude. To say that an attitude (such as belief, desire, intent, or action) is rational is to say that there is an acceptable reason for holding that attitude. "Rational thought is the process of finding such acceptable reasons." (Doyle 1979) Whatever situations the decision makers may face, they should operate by constructing reasons for believing things, desiring things, or doing things. The current set of beliefs arises from the current set of reasons for beliefs, reasons phrased in terms of other beliefs. When action is taken, it is because some reason for the action can be found in terms of the beliefs. A DSS should work the same way reflecting constant update.

In an organizational environment, specific knowledge

may only be true in certain situations or contexts, or its truth is dependent on other knowledge being true. These dependencies have to be represented in the knowledge base, so that the system can get all the relevant information that is needed for a particular query. In the problem solving process, we might get some implausible predictions or inconsistent explanations, which will result in knowledge revision. Knowledge that was assumed to be true is revised or even discarded. We have to exploit the dependencies to determine what established conclusions must be revised. It is also important that a line of reasoning be presented to the decision maker as to explain the analysis done by the system.

With all the model fragments in our model knowledge base, we need to find a way to isolate all the coherent and adequate composite models for a particular query. Coherent means that all the assumptions and user-posed constraints are satisfied, whereas adequate means that the composite model is able to answer this particular query, taking into account all the assumptions derived from the query that need to be included in the model. Userposed constraints are usually represented as meta rules which would be specified as part of the model knowledge base. For example, we could have a rule which requires all the model fragments in one composite model to be of the same model type, which means that if the model type of one fragment is qualitative, then all the other fragments in this scenario model have to be qualitative also. The assumption-based truth maintenance system (ATMS) serves well for the knowledge revision purpose in the model formulation process. This system is based not only on justifications, but also on manipulating assumptions sets, which is an important property for modeling organizations. Since things change fast in organizations, many decisions are made based on certain assumptions. When the situation changes, a different set of assumptions may be needed for a problem. Therefore, we need a system which can respond to the different decision making situations taking into account all the relevant assumptions for a problem.

Most of the relationships in organizations are context-dependent. For a certain relationship to be true, some other relationships have to be true. The ATMS is a way of representing knowledge which reflects some kind of dependencies among different pieces of knowledge. With the ATMS, we can describe contexts in terms of environments and justifications.

In the ATMS, "an assumption is restricted to designate a decision to assume without any commitment as to

what is assumed." (de Kleer 1986). It does not depend on any justification. For example, in the conditions section of model fragment ACCT-1 (PRICE) (see Appendix: CORP Domain Theory), the condition "consider (exists (EXPENSE))" is an assumption which means that the model fragment that contains the quantity EXPENSE should be considered at the same time. A set of such assumptions is called an environment, which is logically a conjunction of assumptions. Assumptions are connected to model fragments via justifications. Every ATMS node associates with a model fragment, that is, each model fragment in the domain theory has a corresponding ATMS node and two model fragments which are the same — meaning that they have the same constraints and relationships — should be associated with the same ATMS node. An ATMS justification describes how a node is derivable from other nodes. In logical terms, justifications are propositional Horn clauses which could be written as:

$$X_1 \wedge X_2 \wedge \dots \wedge X_m \cdots > N.$$

where $X_1, X_2, \dots, X_m$ are the nodes from which N is derived (more specifically in our context, $X_1, X_2, \dots, X_m$ are the model fragments which must also be considered for model fragment N to be active, in other words, for N to be included in the composite model). For example, to activate the model fragment ACCT-1 (PRICE) which specifies in its assumptions section that for the relationships to hold, a qualitative dynamic simulation should be invoked. Hence, we have to consider the model fragments MNF-1 (EXPENSE) and MKT-1 (SALES), which together become a justification for ACCT-1 (PRICE), and are compatible with model fragment ACCT-1 (PRICE), in the sense that all three fragments are applicable in qualitative simulations (see the Appendix for the assumptions and relations of those model fragments). The reason for not choosing the model fragments MKT-2 (SALES) and MKT-3 (SALES) is because they do not fulfill the compatibility requirement. Since we have different kinds of knowledge in organizations, our knowledge base is highly heterogeneous. It therefore consists of heterogeneous model fragments from which we must choose those that are compatible at the execution time.

A node is said to hold in an environment if it can be derived from the justifications and the environment. Organizational processes and relationships often happen in various contexts, which means that each model fragment could be used in different environments (contexts) when we model different aspects of organizations. A node is labeled with the set of all environments in which

it holds. In other words, a model fragment is labeled with all the environments (contexts) in which it is applicable. An ATMS label is a set of environments associated with every node. In our context, each model fragment, which is a node in the ATMS, contains a conditions section and a relations section where the latter consists of the constraints and relationships that apply if all the modeling assumptions in the conditions section are satisfied. Therefore, the basic structure of an ATMS node in our system is a triple: $\langle \text{relations, label, justification} \rangle$, where the justification contains a set of nodes describing how a particular environment is derived. The ATMS would compute the label for the particular node.

Given a set of assumptions (including those derived from the query) and justifications, the task of the ATMS is to efficiently determine a set of coherent and adequate modeling environments. These modeling environments computed by the ATMS are what we need for building task-specific scenario models. The ATMS is incremental and updates only the changed contexts, which is crucial for the knowledge revision of the system when inconsistencies are detected in the prediction or explanation. In other words, those nodes whose label is empty, as opposed to the nodes whose label has at least one non-empty environment, must not be removed from the model knowledge base since future ATMS processing (in response to new justifications) may produce a consistent environment for it (see de Kleer 1986 for the ATMS algorithm). For example, we might add new model fragments to the model knowledge base which would result in a consistent environment for a model fragment that was not compatible with other fragments.

Suppose we have a query, "How does an increase in price affect net income?" In other words, we provide the system the constraint that the final scenario model must contain two model fragments: ACCT-1 (PRICE) and FIN-1 (NET INCOME) (see the Appendix). The ATMS would return all the combinations of assumptions (environments) consistent with the query which are all coherent and adequate models. Each environment would be a conjunction of modeling assumptions that need to be considered when building a model that reasons about the provided modeling terms. In the above example, the ATMS would return a set of minimal modeling environments like

$\{\{\dots\}, \dots, \{\text{consider}(\text{exists}(\text{PRICE})), \text{consider}(\text{exists}(\text{NET INCOME})), \text{consider}(\text{exists}(\text{SALES})), \text{consider}(\text{exists}(\text{EXPENSE})), \text{simulation}(\text{dynamic}), \text{model-}$

type(qual))

on which we would build the scenario model. The modeling environment given above in the last brackets means that, in our scenario model, the fragments which model the quantities PRICE, NET_INCOME, SALES, and EXPENSE have to be included. The modeling process has to be a qualitative dynamic simulation. The modeling environments need to be minimal because they could grow too large to be useful: the largest modeling environment for any fragment is the whole model knowledge base. We only need the environments that have all the relevant information.

In general, we have to expect multiple minimal modeling environments, since the same organizational processes and relationships may apply in different contexts. Hence, each of them could give rise to a different scenario model candidate. However, depending on the organization of the modeling assumptions, it is possible, upon checking the obtained modeling environment, that other objects have to be included in the scenario model if all relevant interactions are to be modeled appropriately. If the model includes too much information, effort will be wasted during analysis and the predictions or explanations will be unnecessarily complicated and hard to explain. If it includes too little information, its predictions may be unreliable and its explanations may not make much sense (Rickel 1992). Therefore, we need to apply some heuristics to choose the simplest model which is also coherent and adequate for answering the query. For example, we could select the model which has the fewest number of model fragments. What is simplest often depends on the details of the domain and the task. It is the user who has to make the final decision.

6. Heterogenous Distributed Model Management

The heterogeneous distributed modeling environment in our new DSS framework poses the question of how to handle model and/or data integration. For example, a company might already have several model management systems based on structured modeling approach or relational approach (Blanning 1993), and database management systems based on relational, hierarchical, or network models (Ram 1991). Ideally, we would like to have access to different model base and database systems which already exist, and to incorporate the existing data or model into our modeling procedure as building blocks. These model bases and databases, however, are often of different types. For example, the

design models in a large automobile company may reside in an object-oriented model base, machine data in a relational database such as IBM DB2, and tooling data in a hierarchical database such as IBM IMS. Suppose now we need to answer the question: When will a new automobile model be available if the design of a component is modified? The user would have to access more than one database and/or model base to get an answer since modifying a design changes the manufacturing schedule, inventory, and availability of the products that use that part. The heterogeneity of databases and model bases makes it difficult to answer this question. How to hide the heterogeneity of file systems, data models, database languages, model representation languages, and data semantics, as well as the hardware and operating systems on which the databases and model bases run provides a tough challenge for building our enterprise wide decision support systems.

The greater the diversity of the modeling languages is, which we allow for specifying relationships, the more expressive the entire knowledge base may become. The task of integrating model fragments, on the other hand, gets more complicated as the heterogeneity of the knowledge base increases. We need to select relevant model fragments and integrate them into a composite structure. In order to select relevant parts, the ATMS, which organizes the model fragments according to their explicitly stated conditions, assumptions, and dependencies, would use the modeling assumptions and user's query to compute a set of candidate composite models. Each candidate should be consistent, coherent, and adequate. The ATMS rules out inconsistencies in the underlying assumptions. Some heuristics would be used to evaluate the candidates and to select a particular model that is most useful for the task.

Different solvers are needed for different model fragments. The fragments included in the model computed by the ATMS would be sent to appropriate solvers. For example, a fragment with interval-valued variables could be sent to the RCR solver and a quantitative model fragment could be sent to a linear programming solver. The issue of model integration becomes critical here.

Model integration involves both the coupling of model component representations and the coupling of processes that manipulate the components. One important approach in model integration is message passing. For example, Kottemann and Dolk (1992) suggest using communicating messages between model components. All messages to and from model components pass

through a control procedure which defines dynamic procedural interactions between the model components. We could have a library which contains user-callable routines for passing messages, coordinating tasks, and modifying the execution procedures. This approach provides a promising direction for our future research. Another open issue is model execution. How is model execution done when there are multiple model types? Which part should be executed first? Which parts need to exchange information? Which information exchange should take place first? The idea is that the DSS should be able to generate a logical structure which controls and coordinates the execution of model fragments and also ensures the convergence of the composite model. Suppose we have an ATMS post-processor which could generate a dependency graph (see figure 1) that is able to coordinate the order of execution and the message passing between model fragments. From the underlying assumptions and justifications, the ATMS computes a set of modeling environments which are consistent and minimal. Suppose the one that we have chosen from this set includes four model fragments: ACCT-1, MNF-1, MKT-1, FIN-1, which have Price, Expense, Sales, and NetIncome as quantities respectively. The ATMS post-processor gives the above dependency graph which shows that the fragments Sales, Expense, and Price have to be considered in order to model NetIncome. More specifically, to model Price, we have to consider Expense first, which in turn depends on Sales. Therefore, this dependency graph would send the message that the model fragment MKT-1 (SALES) has to be executed first before the fragment MNF-1 (EXPENSE) is executed.

After the execution is completed, we'll get an explanation and/or a prediction presented to the user in a comprehensible way. This would also invoke the knowledge revision process which would add the composite model to the knowledge base for future use, especially

Fig. 1 The simplified dependency graph for query How does an increase in price affect net income ?

when we come across a similar situation. The newly generated model would be used as a building block and is available for reuse. As time progresses, feedback from analysis may force revision of the model if the model leads to inconsistencies, weak predictions, implausible predictions, or unreasonable explanations. If a fault is detected, that model (fragment) needs to be revised. Some additional assumptions or justifications might have to be added to the model, which might result in the ATMS' generating new environments for some model fragments. Eventually, this will build up a modeling history which enhances the system's future model building capability.

Some existing software systems offer the flexibility of partitioning a modeling task along lines of functions (Dongarra et al 1993): the ability to execute subtasks of a modeling procedure on the processor most suited to a particular function both enhances performance and improves utilization. In a heterogeneous distributed computing environment, resources should be accessed at three different levels: the transparent mode in which subtasks are automatically located at the most appropriate sites, the architecture-dependent mode in which the user may indicate specific architecture on which particular subtasks are to execute, and the machine-specific mode in which a particular machine may be specified. The PVM (parallel virtual machine), produced by the Heterogeneous Network Project specifically to facilitate heterogeneous parallel computing, was one of the first software systems to enable machines with widely different architecture and floating-point representations to work together on a single computational task. It supplies the functions to automatically start up tasks on the virtual machine and allows the tasks to communicate and synchronize with each other. In particular, PVM handles all message conversion that may be required if two computers use different data representations. These features well suit our purpose of model building in a distributed decision support environment.

7. Conclusion

Research in DSS has concentrated on the effective use of computer technology in aiding humans in the process of decision making. Though there has been considerable progress in developing concepts and tools for DSS, there is still considerable latitude for further progress. The main purpose of this paper is to point out the current stage of research in the DSS field and the direction that future DSS research should take. We discussed issues in using qualitative reasoning method to represent and

model organizational relationships and processes, more specifically, qualitative data representation and qualitative optimization problems. The knowledge base of a new DSS framework should be able to include heterogeneous types of knowledge, for example, quantitative knowledge, qualitative knowledge, logical statements, etc. Knowledge revision plays a significant role in the problem solving process, in the sense that decision makers must base their decisions on the most up-to-date information. With heterogeneous knowledge pieces in our knowledge base, model/data integration becomes critical. Future research in DSS should also recognize that decision support is a collaborative process which has different phases. Researchers need to investigate how computerized decision support could be more organizational oriented. In this paper, we mainly presented these issues at the suggestion level. A significant amount of research work will have to be done before these problems can be solved.

8. Appendix

The CORP Domain Theory:

```
fragment MKT-1 (SALES)
    {marketing model describing qualitative relationship between price and sales volume}
conditions
    consider(exists(PRICE)),
    model-type(qual),
    simulation(dynamic)
relations
    SALES=M-(PRICE)
end
fragment MKT-2(SALES)
    {marketing model describing quantitative relationship between price and sales volume}
conditions
    consider(exists(PRICE)),
    model-type(quant),
    simulation(static)
relations
    SALES=80,000-44,000*PRICE
end
fragment MKT-3(SALES)
    {marketing model describing semi-qualitative relationship between price and sales volume}
conditions
    consider(exists(PRICE)),
    model-type(qual-quant),
    simulation(dynamic)
relations
```

```
SALES(t)=[68000, 92000]
        -[40000, 48000]*PRICE(t)
end
fragment ACCT-1 (PRICE)
    {accounting model determining price}
conditions
    consider(exists(EXPENSE)),
    model-type(qual),
    simulation(dynamic)
relations
    PRICE=MARKUP+EXPENSE/SALES,
    constant(MARKUP)
end
fragment MNF-1 (EXPENSE)
    {manufacturing model determining total production cost}
conditions
    consider(exists(SALES)),
    model-type(qual),
    simulation(dynamic)
relations
    EXPENSE=FIX COST+VAR COST,
    VAR COST=UNIT COST*SALES,
    constant(UNIT COST)
    constant(FIX COST)
end
fragment FIN-1 (NET INCOME)
    {qualitative financial planning model to predict net income}
conditions
    consider(exists(SALES)),
    consider(exists(PRICE)),
    consider(exists(EXPENSE)),
    model-type(qual),
    simulation(dynamic)
relations
    NET INCOME=PRICE*SALES-EXPENSE
end
fragment MGT-1 (PRODUCTIVITY)
    {qualitative management model explaining overall productivity}
conditions
    consider(exists(PARTNERSHIP)),
    model-type(qual),
    simulation(dynamic)
relations
    PRODUCTIVITY=M+(PARTNERSHIP)
end
rules
    R-1: consider(performance(X)) and
```

model-type(qual) $\Rightarrow$ >simulation(dynamic).

R-2: model-type(qual) $\Rightarrow$ >solver(QSIM).

end

(Manuscript received Jan., 20, 1994)

References

- (1) Ba, S., K. R. Lang & A. B. Whinston: "Enterprise modeling and decision support." The Proceedings of Pan Pacific Conference on Information Systems. Kaohsiung, Taiwan. (1993)
- (2) A. Balakrishnan & A. B. Whinston: "Information issues in model specification." *Information Systems Research*, 2(4) 263-286. (1991)
- (3) P. Balasubramanian, T. Isakowitz, H. Johar, & E. Stohr: "Hyper model management systems." *Proceedings of the 25th Hawaii International Conference on System Sciences*. 462-472. (1992)
- (4) S. Banerjee & A. Basu: "A knowledge based framework for selecting management science models." Proceedings of the Twenty-Third Annual Hawaii International Conference on Systems Sciences III: *Decision Support and Knowledge Based Systems Track*. 484-493 (January 1990)
- (5) H. K. Bhargava, R. Krishnan, & A. B. Whinston: "On integrating collaboration and decision analysis techniques." Working paper. University of Texas at Austin. (1993)
- (6) R. W. Blanning: "A relational framework for model bank organization." *Proceedings of the IEEE Workshop on Languages for Automation*. 148-154 (November 1981a)
- (7) R. W. Blanning, "Conversing with management information systems in natural language." *Communications of the ACM*. 27(3) : 201-207. (1984b)
- (8) R. W. Blanning, "Organization design as expert system design." Proceedings of the Twenty-Fourth Annual Hawaii International Conference on Systems Sciences IV: *Organizational Systems and Technology Track*. 3-12 (January 1991)
- (9) R. W. Blanning, "Model management system: an overview." *Decision Support Systems*. 9(1) : 9-17. (1993)
- (10) R. H. Bonczek, C. W. Holsapple, & A. B. Whinston: "Future directions for developing decision support systems." *Decision Science* (October 1980)
- (11) R. H. Bonczek, C. W. Holsapple, & A. B. Whinston "Developments in decision support systems." *Advances in Computers*. 23 : 141-175. (1984)
- (12) R. H. Bonczek, C. W. Holsapple, & A. B. Whinston: Foundations of Decision Support Systems. Academic Press. (1981)
- (13) A. Bosman: "Decision support systems, problem processing and coordination." In *Processes and Tools for Decision Support*. H. G. Sol ed. North-Holland Publ. Amsterdam. (1983)
- (14) K. M. C. Carley: "Coordinating for success: trading information redundancy for task simplicity". Proceedings of the Twenty-Third Annual Hawaii International Conference on Systems Sciences IV: *Emerging Technologies and Applications Track*. 261-270 (January 1990)
- (15) D. A. Carlson & S. Ram "Hyperintelligence: the next frontier." *Communications of the ACM*. 33(3) : 311-321. (1990a)
- (16) D. A. Carlson & S. Ram: "Modeling organizations as a social network of distributed knowledge-based systems." Proceedings of the Twenty-Third Annual Hawaii International Conference on Systems Sciences III: *Decision Support and Knowledge Based Systems Track*. 271-280 (January 1990b)
- (17) A. M. Chang, C. W. Holsapple, & A. B. Whinston: "Model management issues and directions." *Decision Support Systems*. 9(1) : 19-37. (1993)
- (18) E. Davis: "Constraint propagation with interval labels." *Artificial Intelligence*, 32, 281-331. (1987)
- (19) de Kleer, J. "An assumption-based TMS." *Artificial Intelligence*, 28 : 127-162. (1986)
- (20) D. R. Dolk: "Model management and structured modeling: the role of an information resource dictionary system." *Communications of the ACM*. 31(6) : 704-718.
- (21) J. J. Dongarra, G. A. Geist, R. Manchek, & V. S. Sunderam: "Integrated PVM framework supports heterogeneous network computing." *Computers in Physics*. 7(2) : 166-175. (1993)
- (22) J. Doyle, "A truth maintenance system." *Artificial Intelligence*. 12 : 231-272. (1979)
- (23) C. A. Ellis, S. J. Gibbs, & G. L. Rein "Groupware: some issues and experiences." *Communications of the ACM*. 34(1) : 38-58. (1991)
- (24) G. Elofson & B. Konsynski "Delegation technologies: environmental scanning with intelligent agents." *Journal of Management Information Systems*. 8(1) : 37-62. (1991)
- (25) A. M. Geoffrion, "An introduction to structured modeling." *Management Science*, 33(5), 547-588. (1987)
- (26) S. H. Haeckel & R. L. Nolan: "Managing by wire." *Harvard Business Review*. 1221-132 (Sep-Oct 1993)
- (27) M. Hamalainen, R. Kalakota, & A. B. Whinston: "Interactive virtual documents from digital libraries: a pragmatic approach for designing future information systems." working paper. The University of Texas at Austin. (1993)
- (28) A. Hinkkanen, K. R. Lang, & A. B. Whinston "On the Usage of Qualitative Reasoning as Approach Towards Enterprise Modeling." forthcoming in *Annals of Operations Research*. (1993)
- (29) J. N. Hooker: "A qualitative approach to logic inference." *Decision Support Systems*. 4 : 45-69. (1988)
- (30) S. Hwang: "Automatic model building systems: a survey." *DSS-85 Transactions*. 22-32 (April 1985)
- (31) R. S. Kaplan & D. P. Norton "The balanced scorecard — measures that drive performance." *Harvard Business Review*. 70(1) : 71-79 (1992)
- (32) R. S. Kaplan & D. P. Norton "Putting the balanced scorecard to work." *Harvard Business Review*. 134-147 (Sept-Oct 1993)
- (33) M. Kiang: "Exploration of qualitative reasoning and its application to management." Ph. D. Thesis. University of Texas at Austin. (1990)
- (34) M. Y. Kiang, A. B. Whinston, & A. Hinkkanen "An interval propagation method for solving qualitative difference equation systems." in *Qualitative Reasoning and Decision Technologies*. ed. by N. P. Carrete and M. G. Singh. International Center for Numerical Methods in Engineering, Barcelona, Spain. (1993)
- (35) G. Klein: "Developing model strings for model managers." *Journal of Management Information Systems*. 3(2), 94-110. (1986)
- (36) J. Kottemann & D. R. Dolk, "Model integration and modeling languages: a process perspective." *Information Systems Research*. 3(1), 1-16. (1992)
- (37) B. J. Kuipers: "Qualitative simulation." *Artificial Intelligence*, 29, 289-338. (1986)
- (38) B. J. Kuipers & D. Berleant: "A smooth integration of incomplete quantitative knowledge into qualitative simulation." Tech. Report AI90-122, Artificial Intelligence Laboratory, University of Texas at Austin. (1990)
- (39) M. L. Lenard, "An object-oriented approach to model management." Proceedings of the Twenty-Second Annual Hawaii International Conference on Systems Sciences I. January. 507-515. (1989)
- (40) Lifschitz, V.: "Circumscription." manuscript for a chapter in *Handbook of Logic in AI and Logic Programming*, University of Texas at Austin. (1993)
- (41) T. W. Malone, K. R. Grant, F. A. Turbak, S. A. Brobst, & M. D. Cohen "Intelligent information sharing systems." *Communications of the ACM*. 30(5), 390-402. (1987)
- (42) T. H. Merrett: Relational Information Systems. Reston Publishing Co., Reston. (1984)
- (43) W. A. Muhanna: "Issues in distributed model management systems." *Proceedings of the Eleventh International Conference on Information Systems*. 231-242 (December 1990)
- (44) S. Ram: "Heterogeneous distributed database systems." *IEEE Computer*. 24(12), 7-10.
- (45) J. Rickel: "Automated modeling for answering prediction questions: exploiting interaction paths." Technical report. The AI Lab of the University of Texas at Austin. (1992)
- (46) C. Spetzler & R. Howard: "The effect of decision quality on management roles and processing." ORSA/TIMS Joint National Conference. Phoenix, Arizona. (1993)
- (47) B. C. Williams: "A theory of interactions: unifying qualitative and quantitative algebraic reasoning." *Artificial Intelligence*, 51, 39-94. (1991)

Andrew B. Whinston He is a professor in the Business School, Computer Science Department, Economics Department, and the Graduate School of Library and Information Science at The University of Texas at Austin, where he is also a fellow of the IC² Institute, Director of the Center for Information Systems Management. He also holds the Hugh Roy Cullen Centennial

Chair Business Administration. His research deals with enterprise modeling, management of large scale networks, decision support systems theory, distributed AI, and organizational modeling. He is the editor-in-chief of the journals *Decision Support Systems* and *Organizational Computing*, and an Associate Editor of most of the academic journals in the information systems field. He has authored or co-authored over 250 articles in such journals as *Management Science*, *Information Systems Research*, *Decision Support Systems*, *Operations Research*, *American Economic Review*, *ACM Transactions on Database Systems*, *Econometrica* and *Journal of Combinatorics* and coauthored 6 books. He received his Ph.D. in management from Carnegie Mellon University in 1962. He

served as an Assistant Professor at Yale University from 1961 to 1964 in the Department of Economics and the Cowles Foundation for Research in Economics. In 1964 he joined the faculty of the Economics Department at the University of Virginia as an Associate Professor and in 1966 was appointed as a Professor in the Business School at Purdue University with a joint appointment in Computer Science. He has been the Principal Investigator on contracts from the National Science Foundation, Office of Naval Research and Army Research Office as well as numerous grants from Companies such as Apple, N. C. R., IBM and Shell Development.

Sulin Ba

She is a second year Ph.D. student in the Department of Management Science and Information Systems at the University of Texas at Austin. She received her B. A. from Zhongshan University (China) in 1990. She got her Master's degree in Library and Information Science from the University of Texas at Austin in 1992. Her current research interests lie

in the areas of qualitative reasoning, enterprise modeling, model management, and organizational knowledge representation.