Sun Yat-sen Management Review
Vol.2, No.1, March 1994

Enterprise Modeling and Decision Support

Sulin Ba
Department of Management Science and Information Systems
The University of Texas at Austin, USA

Karl Reiner Lang
Department of Management Science and Information systems
The University of Texas at Austin, USA

Andrew B. Whinston
Department of Management Science and Information systems
The University of Texas at Austin, USA

ABSTRACT

This paper analyzes the computational requirements of enterprise modeling, and presents a framework for reasoning with enterprise-wide models. Enterprise modeling studies complex systems which are normally only partially known, and focuses on studying the interactions among the components of the system, and the underlying modeling assumptions thereof. We discuss some major design issues for the next generation of enterprise modeling systems (EMS), that is, organizational-wide computing systems, and indicate the direction future research could take to support enterprise-wide problem formulation and problem solving. The focus of this paper is chiefly on investigating the question of how can we do model building in an enterprise-wide environment, and on proposing ideas which we see as promising steps towards accomplishing this difficult endeavor.

Keywords: Enterprise Modeling Systems, Decision Support, Artificial Intelligence

1. Introduction

Enterprise modeling has recently emerged as a very active research area. An enterprise model is a structural description of an organization, respectively a part of an organization, in terms of key variables and essential relationships only. Such a model takes a global view of an organization, and is typically based on a rather coarse level of description. The focus of enterprise modeling is on representing and analyzing both the quantitative and the qualitative properties of the underlying system, and thus differs from traditional management science approaches like mathematical programming and statistical forecasting. The main challenge is how to build enterprise-wide models which are focused on the issues we are dealing with. We need a flexible enterprise modeling sysem which builds models as needed in response to queries posed by the end-user. We propose a model building strategy which avoids looking at a problem in a myopic fashion, but one which also realizes that the construction of a single monumental enterprise-wide model would be impractional.

Typical applications of enterprise modeling are the diagnosis of the performance of a firm, predicting the behavior of an organization over time, testing the implications of theories about organizations, and supporting strategic business decisions. Hinkkanen et al (1993a) suggest a qualitative reasoning approach to modeling enterprises, and present a prototype system which they apply to qualitatively predict the financial behavior of a particular company over time.

A framework which conceptualizes both qualitative and quantitative modeling extends the quantitative modeling paradigm in traditional decision support systems (DSS). The motivation for taking a qualitative perspective has various reasons. Firstly, for many problems there is simply not enough information available to formulate a quantitative model. We call this situation modeling systems with incomplete knowledge or information. Secondly, although in many cases information is available, it might be inexact or carry uncertainties. This is called modeling systems with imprecise knowledge or information. Modeling partially known systems encompasses both cases, which can also occur together. Thirdly, even if it were possible to acquire complete and precise knowledge, the modeler is often not really interested in the details of the system, in other words, the modeler prefers a qualitative description. The latter case adopts a point of view which is typical for a top level man-

agement perception of an organization, and is especially suited for addressing strategic business issues. Fourthly, research in, for example, organizational science and economics, pursues as one of its main goals the development of general theories about certain classes of firms. In this context, one might be interested in a qualitative framework that allows us to abstract knowledge from a collection of particular organizations, which might be fairly specific, into more general descriptions, retaining the qualitative information that represents only significant distinctions and characterizes all members of the class. Analyzing such a generalized, qualitative model draws implications that hold for all specific, possibly quantitative, models that are instances of this class.

In business related areas like organizational science, management, business communication, and others, qualitative approaches are widely used in order to develop theories (Monge 1990). However, in most cases qualitative descriptions are presented as purely verbal formulations, that is, in an informal manner, Despite the usefulness of verbal formulations, additional more formal methods are often desired in order to overcome certain vaguenesses in describing complex systems. Semiformal graphical techniques like organigrams and influence diagrams are a step in that direction, but still fall short when it comes to reasoning about complex systems. Qualitative reasoning attempts to provide a framework that allows us to model dynamic systems in qualitative terms. These reasoners are based on a qualitative mathematics that was mainly developed in economics (Moore et al 1993). Although, methods like qualitative simulation (Kuipers 1986) have been successfully applied to simple problems, mostly from physics and engineering, they fail on more complex and realistic problems like modeling an enterprise. The reason for the limited applicability of purely qualitative reasoning methods is their weak inferential power, that is, their intrinsic inability to derive strong, precise predictions from weak, purely qualitative problem specifications. Hence, topics like incorporating quantitative information and, especially, model building have become particularly important items on the agenda of current research in qualitative reasoning. Davis (1987), Kuipers and Berleant (1990), Kiang et al (1993), and Williams (1992) have extended the view on qualitative reasoning by intergrating qualitative and quantitative information into one reasoning framework which enables stronger predictions at the expense of higher information cost. Falkenhainer and Forbus (1991), Addanki et al (1991), and Weld (1992) have developed modeling approaches which support reasoning with multiple models in an effort to construct more accurate and task-specific models from a

collection of descriptions encompassing general domain knowledge and model fragments that describe certain phenomena from different perspectives and at different levels of detail. However, existing systems are still either too weak in their predictive strength or too restricted in their range of possible applications to serve as complete enterprise modeling systems. Other techniques such as stochastic processes, time series analysis, or fuzzy set techniques follow a non-deterministic paradigm, and are beyond the scope of this paper. Besides those computational difficulties, current qualitative reasoning systems offer only primitive user interfaces, for example model specification as LISP expressions, which further limits their acceptance among practitioners.

Next, we discuss some major design issues for the next generation of enterprise modeling systems (EMS), that is, organization-wide computing systems, and indicate the direction future research could take to support enterprise-wide problem formulation and problem solving. The focus of the paper is chiefly on investigating the question of how can we do model building, and how can we extract the relevant parts of the model to support specific analysis of a corporate issue in an enterprise-wide environment. We also propose some ideas which we see as promising steps towards accomplishing this difficult endeavor.

Using different sets of assumptions and various kinds of knowledge, ranging from general, qualitative knowledge to specific and precise numerical models, managers analyze organizational questions from different perspectives and at different levels of detail. Given a particular task, model building is guided by the selection of an appropriate perspective and level of detail, a modeling decision for which little support is found in current decision support system technology. 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 contraints, but also to exclude irrelevant ones and to ignore unnecessary details. For example, answering a question like "How do customers see our company?" doesn't require us to consider the production process at an operational level, or to consider individual pieces of machinery or product units. Modern management of organizations in a complex and turbulent environment require managers to look at problems in a non-myopic fashion which also recognizes all the significant interactions, including cross-functional relationships. Traditional DSS systems do not support this kind of focused attention, and require the modeler to make all the adequate modeling decisions. Another important issue is the one of 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. While more recent model management systems (MMS) do provide features for supporting model reusability and model integration, they still require the modeler to decide which model pieces are relevant for the task, and if their level of representation is appropriate to answer the posed question. Muhanna and Pick (1992), Krishnan et al (1991), and Dolk and Kottemann (1993), for example, present model control languages which significantly increase the flexibility and productivity of the model building process by providing access to multiple models and solvers and by enabling the user to select a collection of existing model from a model library, and to assemble a composite model by linking several models. However, those systems merely offer a language for representing the modeler's decision which model components to include in the analysis, and how to integrate the different components of the composite model. There is no explicit management of the underlying modeling assumptions of a particular analysis, and no facility to prevent the modeler from specifying incompatible, incoherent, inappropriate, or inaccurate model compositions.

There are two, essentially disjoint, research efforts, one based in the artificial intelligence community and the other in the decision support systems community, which study model building and reasoning with multiple models. This paper draws upon both, and develops a synergistic framework for future enterprise modeling systems. Designing an organization-wide modeling system which satisfies the above requirements necessitates explicit reasoning about the model's underlying assumption, and its range of applicability. We propose a model building strategy which is inspired by research in the qualitative reasoning field, and is particularly influenced by compositional modeling, first presented by Falkenhainer and Forbus (1991). While current DSS research emphasizes quantitative modeling, work in qualitative reasoning has given more attention to reasoning about purely qualitative knowledge. We see organizational-wide reasoning systems as decision tools for strategic and operational management. Especially when exploring strategical questions, it is essential to be able to include qualitative knowledge into the analysis. We envision a system whose reasoning about a particular organization is based on a library of model components representing interesting organizational phenomena from different perspectives and different levels of detail. Given a query,

such a system would generate a composite model that is derived from the specific needs of the query.

Accomplishing this requires access to multiple sets of heterogeneous model fragments 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. We use simplifying assumptions as a class of assumptions to explicitly represent modeling commitments in terms of abstraction level, approximation, perspective, level of detail, and granularity. Reasoning about those assumptions should enable a system to identify a suitable collection of commpatible model fragments, and to compose an appropriate model for a query. 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 understanding 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-specfic 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 a quantitative, management science/operations research (MS/OR) type of models like optimization models, simulation models, and forecasting models. The task of organizing organizational knowledge into semi-independent, reusable model fragments is a crucial one for enabling an EMS to compose useful, problem-specific models by integrating relevant, existing model components under a variety of different modeling circumstances.

Selecting the model fragments to compose an appropriate model for answering a given query requires modeling decisions along several dimension. What is the best set of variables to be included in the model? What level of detail is appropriate? Which are the relevant organizational phenomena for studying the posed question? From what perspective should the problem be viewed? What kinds of approximations and abstractions should be allowed? Even the most carefully organized model fragment library won't provide enough information to find an answer to all these questions. Therefore,

we assume that the query itself has to supply the missing pieces of information, and has to provide clues which narrow the focus of the model composition process, and sufficiently constrains an appropriate set of modeling assumptions. In order to help generation parsimonious answers, and to ease model simulation, another class of modeling assumptions called operation assumptions is introduced. Operating assumptions narrow the scope of the model space search, and delimit different ranges of behavior. For example, a question asking for the key factors which affect the future performance of the company should contain a hint that allows the system to infer if the question refers to short-term performance, or to long-term performance. A short-term analysis could penalize investments whose payoffs materialize only in the long run. Long-term analyses usually suggests less detail or a higher level of abstraction, because of their more strategical nature, and because uncertainties about the future increase over time. Hence, when building a model in response to the above question, we make an operating assumption on the planning horizon which narrows its scope, and guides the search for adequate yet simple models.

The real-world phenomenon under study is often referred to as a scenario, and the model representing it is then called a scenario model. The compositional modeling framework requires first the building of a general-purpose domain theory that describes a variety of organizational objects, activities, and processes, and a collection of interesting scenario descriptions. The domain theory is represented as a set of model fragments, each describing an independent aspect from a particular viewpoint. It contains general organizational laws and rules as well as relationships that are very specific to a particular company, and thus consolidates the organizational behavior way of thinking with the management science school of thought. Given a domain theory and a scenario description, the model formulation problem can be defined as selecting the model relevant model fragments, and by composition generate a scenario model which is coherent and most useful for answering a specific query about the scenario's behavior.

Model composition would start by analyzing the query, and then identifying relevant model fragments by matching terms of the query with the objects represented in the model fragments of the domain theory while checking the conditions that govern the applicability of the model fragments. From an initial set of relevant model fragments, the system would compose a final model, which eliminates any inconsistencies among modeling assumptions that might occur across some of the initially selected model fragments. The second crite-

ria that drives model composition is the goal of finding the most useful model for the task, where most useful refers to the property of being the simplest possible model that is comprehensive and appropriate.

2. Reasoning with Multiple Models

This section overviews major approaches to the model building processes which are based on reasoning with multiple models, that is, model building methods which construct composite, problem-specific models from a collection of predefined model fragments. Researchers in the DSS and AI communities have, basically unaware of each other, proposed serveral frameworks which provide partial solutions to this formidable problem. We argue that cross-fertilizing ideas from both research fields will achieve significant progress in answering many of the open research questions impeding the development of complete, enterprise-wide model management systems. While the DSS and AI paradigms diverge in their application domains, management are engineering, respectively, they are faced with basically the same underlying model building issues. Work in the two areas differ also in other aspects. Model management in the DSS field can be seen as a natural extension of previous work in management science and operations research, and has advanced mathematical modeling from a state where modeling was an uncoordinated task, whose success depended mainly on the technical skills and expertise of the modeler, to a state where systems actually know about certain types of mathematical models and appropriate solvers. While restricted to mathematical programming models, statistical forecasting model, and perhaps discrete-event simulation models, DSS research has made considerable progress in solver integration. AI research, on the other hand, has pretty much ignored the issue of how to integrate different solution algorithms employed in a composite model, but has addressed other important issues which in turn have been neglected in DSS research, the explicit representation of modeling assumptions, and the usage and exploration of qualitative knowledge to name the two extremely important ones. The next two sections summarize the work presented in the AI and DSS literature.

2.1 Building Composite Models in Artificial Intelligence

DeKleer and Brown (1984) propose component connection modeling as a framework which is intended as a tool for reasoning about loosely coupled,

dynamical physical systems. DeKleer and Brown's framework rests on the no-function-in-structure principle which says that we can decompose a complex system into a structure of context-free components and interconnections. A domain dependent component library would supply the modeler with a standard set of independent building blocks from which a particular scenario model can be built. In the physics and engineering domain such a library would contain basic devices like pipes, tanks, valves, springs, electrical circuits and so on. The structural description of the basic components consist exclusively of qualitative relationships which are represented as a set of confluence equations. Model Integration is achieved by connecting component with each other through terminal points which represent shared variables, a task which requires explicit specifications from the modeler. Components communicate by applying input signals to terminal points, and propagating output signals from terminal point to connected components. The output signals produced by a component depends not only on the input signals but also on the active set of assumptions. The supporting context is described separately as a set of global, class-wide assumptions that determines the function of a component as a part of the system. The explicit representation of the underlying assumptions determines which devices are compatible and what kind of interactions are admissible. Class-wide assumptions correspond to ontological commitments. Modeling an electrical circuit system, for example, demands that variables are expressed in terms of voltage and current, and the behavior of its constituent component be governed by physical laws like the Kirchoff laws. In a business environment, one would have to consider other dimensions, and would describe variables in terms of dollars and bushel, for example. Reasoning about fluid flow systems requires an explicit assumption if flows are considered laminar or turbulent, or, in a network flow formulation one would need an assumption if flows yield gains (e.g., interests on an financial investment) or losses (e.g., spoilage during a produce shipment). Another type of assumption concerns the treatment of the dynamics of the system, that is, it explicates if we are dealing with a equilibrium or non-equilibrium system. The quasistatic approximation assumption, the standard case in component connection modeling, states that the system is always in or near equilibrium, that is, the system returns quickly to equilibrium after a disturbance. This implicit assumption allows one to ignore intermediate non-equilibrium states which simplifies the simulation process, but excludes more general modeling situations. DeKleer and Brown do not discuss strategies of when and how to

switch between different sets of assumption. In order to apply the component connection approach to enterprise modeling we must develop a theory of business ontologies with coherent laws and assumptions, a task that needs further research.

Compositional modeling, recently proposed by Falkenhainer and Forbus (1991), presents perhaps the most complete model building framework for reasoning with multiple models, We believe that developing an enterprise-wide modeling system could significantly benefit from adopting the compositional modeling strategy. For this reason, and because compositional modeling contains several ideas which are still novel to the DSS community, we present a somewhat more detailed introduction to this new and exciting model building approach. Despite its historical roots, which lie in the realm of modeling systems from the engineering and physics domain, it pursues a much more versatile approach to modeling complex systems in general. In our view, compositional modeling presents a well suited groundwork for designing an enterprise modeling system. Accepting the fact that system engineers draw upon a vast body of knowledge ranging from qualitative rules of thumb to precise numerical models in order to describe predict and explain the behavior of complex systems, independent of a particular domain, compositional modeling does not commit to a specific style of reasoning but offers provisions for several kinds of knowledge representation and linkages to different solvers. In other words, knowledge representation and solution methods are kept separate, and can be configured to suit a specific problem domain, thus providing the capability of quantitative, qualitative and hybrid methods of reasoning. The main contribution of compositional modeling, besides offering multiple styles of reasoning, is its flexible approach to automatically compose task appropriate models from a model base taking account of diverse aspects of problem analyses such as selective attention, different levels of approximation and abstraction, explicit representation of simplifying assumptions and operation assumptions, modeling different perspectives of an artifact, and expressing different levels of detail.

The main characteristics of compositional modeling can be summarized as the capability of providing access to multiple models pertaining to a particular problem domain, forming an appropriate model for each specific analysis, and expressing explicit representations of underlying modeling assumptions. Given (1) a model library that contains a collection of building blocks, called model fragments, representing the available domain knowledge, (2) a description of

a particular problem, called a scenario, and (3) a specific query, the compositional modeling system's main task is then to generate a scenario model which can be solved in order to give a satisfactory answer to the question raised by the query.

The idea of composing a scenario model as needed imposes a great challenge to model management. It is impossible to maintain a monolithic model that represents all facets of a whole system. Instead, we organize our domain knowledge as a collection of heterogeneous model fragments and assumptions constraining their use. An important organizing principle in building a model library is that the model fragments form a structural part-of hierarchy which is used to identify related fragments. A model fragment must contain more than just a set of relationships describing objects and their interactions, it must also explicitly encode underlying assumptions which tell under which conditions it is actually applicable. Model fragments should be designed as modular, semi-independent, reusable, and possibly mutually incomsistent building blocks. Additionally, we need to maintain sets of class-wide assumptions each describing commitments and conditions of a particular perspective of a scenario. By matching the assumptions of the active set of class-wide assumptions with the assumptions explicated in the model fragments it is possible to retrieve only those model fragments for composing a scenario model that are applicable and related to a given task. Which fragments should be considered for building a scenario model depends on the active set of class-wide assumption. A crucial presupposition of compositional modeling is that the query posed provides enough clues to identify which objects and processes, and thereby which model fragments, need to be considered and what the appropriate set of assumptions could be.

A scenario model generated in response to a query has to be coherent and most useful. The former requires a scenario model to be consistent with the assumptions and principles of the underlying domain theory, the latter refers to the tradeoff between information cost and relevance to the query in terms of sufficiency and minimality (Balakrishnan and Whinston 1991). Sufficiency means that the answer to the query is (a) not just correct but also about the user's question and provides his/her with the sought information (aboutness), and (b) in addition satisfactorily detailed and accurate (accuracy). Minimality, on the other hand, calls for a parsimonious response, and forbids extraneous details. In general, there is no unique scenario model but several candidate models satisfying the above criteria. An assumption based

truth maintenance system (ATMS), cf. deKleer (1986), can be used to find sets of suitable modeling assumptions, called modeling environment, each implying a different candidate scenario model. The complete model composition algorithm consists of the following four stages.

(1) Query Analysis:

Identify objects, quantities, processes, and relations of interest, and thereby identify model fragments that need to be considered. Using an ATMS, compute a collection of (partial) model environments representing suitable modeling assumptions.

(2) Object Expansion:

Include additional objects (and corresponding model fragments) implied by the partial model environments using the part-of hierarchy that structures the model fragments in the model library.

(3) Candidate Completion:

Complete the partial model environments computed in (1) by incorporating new assumptions that evolved from object expansion.

(4) Candidate Evaluation and Selection:

Select the best candidate scenario model by applying some heuristic rules that ensure minimality.

(5) Model Simulation:

After composing the final scenario model it needs to be transformed into a proper input format suitable for an appropriate solver.

Another notable AI approach to reasoning with multiple models is the graphs of models framework by Addanki et al (1991), which expresses physical domains as graphs where the nodes represent models and the edges represent the assumptions that have to be changed in order to switch between different models. Finally, Weld (1992) introduces a model management system which reasons about one dimension of modeling assumptions. Given a query it selects through refinement techniques a model with an appropriate level of detail, which might be qualitative or quantitative.

2.2 Model Integration in Decision Support Systems

The current stream of the DSS literature argues for an approach in which model integration is achieved by relating existing models to each other, and

thus creating higher level structures. Kottemann and Dolk (1993) propose an object oriented, integrated modeling environment based on the structured modeling language (SML), Geoffrion (1987), where an overall task such as production and distribution planning is decomposed into several interacting subtasks, and each subtask is modeled individually. They present a model control language which allows the user to specify a collection of predefined models as communicating processes. A cost accounting model, for example, could calculate product prices as its output which would be sent as an input to a forecasting model, which in turn would predict a demand which could be sent to a production planning model, and so forth. Muhanna and Pick (1992) report on a model management system called SYMMS that offers a model description and configuration language which enables the modeler to reuse and connect predefined models. For example, a production planning model which is formulated as a linear program could be coupled to a forecasting model which would provide demand figures as the right hand side parameters of the linear program. However, the matching of the shared variables, in this case the demand variables, which establishes the model-model linkage needs to be done explicitly by the user by writing a control module which pairs the output of the forecasting model with an input port of the production, and thus providing the desired coupling link. Another example is Krishnan et al (1991) who developed the system ASCEND which allows for (re)configuring models that exhibit a common, compatible structure. For example, a production model and a distribution model which are both formulated as transportation models with warehouses as common nodes could be merged into a transshipment model that suggests a production schedule, shipments from production plants to warehouses, and shipments from warehouses to customers.

While all these approaches support model reuse and model integration they leave essential modeling decisions up to the modeler, who is responsible for the selection of relevant model components and for checking their compatibility, for the identification and matching of shared variables, and for sequencing and synchronizing the model solving process. These modeling decisions made by the user are based on a set of assumptions which is only implicitly expressed in the composite model, and there is very little support for ensuring the sufficiency and the consistency of the assumptions being used. We believe that these limitations can only be overcome by explicitly representing the underlying modeling assumptions which are necessary to compose a model. Bonczek et al (1981) first suggested the use of first-order predicate logic for

stating the conditions which imply the application of certain model units, an idea which is also used by Falkenhainer and Forbus (1991), and which we shall employ for organizing the domain knowledge in our model base. Bhargava and Krishnan (1991) suggest a different approach, in which they advocate the explicit representation of the assumptions underlying the reasoning process with multiple mathematical programming formulations. They present a language for a model management systems based on defeasible reasoning, a form of nonmonotonic reasoning, that allows tentative model formulations to be defeated when new information becomes available.

3. Organization of the Model Knowledge Base

In this section, we discuss the organizing principles of the model knowledge base underlying the compositional modeling framework. We view the model base as a repository of organizational knowledge whose purpose is to provide a resource of sharable and reusable models for helping to better understand, explain and predict organizational phenomena in a variety of different situations. In order to achieve the necessary depth and versatility, the model base needs to contain knowledge of different knows which should include quantitative, qualitative, and hybrid forms of information.

The main challenge of designing such a mode knowledge base is to decompose the vast body of knowledge available into semi-independent fragments in a manner that allows us to assemble new, task-specific models under a wide range of scenarios. Merging relationships from multiple model fragments into an integrated, composite model requires not only a careful approach of grouping relationships into independently meaningful units, but also an explicit treatment of the modeling assumptions which describe when they apply.

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 different representation languages to represent the underlying assumptions of a model component and its constituting relationships, which we describe in the next two sections. Each model fragment would contain two sections, one that contains the specification of modeling assumptions as predicates, and the other containing the actual constraints and relationships that apply if all predicates of the model assumption section are true. The constraint

section, finally, would specify a set of constraints and relationships that are imposed by the model fragment if its assumptions are validated. Before a compositional algorithm can actually search the model base and identify task-specific, relevant model fragments, it needs sufficient information to be able to evaluate the predicates in the model assumption section. This extra information needs to be derived from the query, or needs to be inferred from meta rules which would be specified as a part of the model base. These meta fules would rule out incoherent and inconsistent combinations of modeling assumptions, and would also imply additional conditions as a consequence of modeling assumptions that have been already established.

3.1 Representation of Modeling Assumptions

We distinguish between several types of modeling assumptions. (1) Conditions providing structural and topological information on the organization considered. The entire organizational system should be organized into linked subsystems which can consist of other systems or primitive objects (individual variables). Part-of relations can be used to state, for example, that the manufacturing department is part of the company, and that plant X is part of manufacturing. Is-a relations can be used to express that widget 2000 is a new product, and that a new product is also, in a more general sense, just a product. (2) Simplifying assumptions which shift focus to a particular perspective of the organization, and indicate if a cost analysis. a productivity analysis, or if some other kind of analysis is desired. (3) Ontological assumptions which take a certain view on the organization, and select an appropriate method of description. Should the organization be viewed as a collection of employees who are working towards a common, cooperate goal? Should the organization be described as a collection of interacting subunits such as functional departments where the interaction might be represented as information flows, cash flows, or material flows? (4) Granularity assumptions determine the level of detail for a given analysis. A production scheduling analysis may require the consideration of each worker and piece of machinery involved in the manufacturing process of the products. A strategical marketing study might need a more aggregated view, and suggest a study in terms of product groups without explicitly considering any details of the manufacturing process. (5) Approximations are mainly used to simplify a model for computational benefits. Linearity assumptions, for example, abound in all modeling contexts. (6)

Abstractions are used to reduce the complexity of phenomena. Inventory models typically assume that new batches of products arrive at the beginning of a planning period, and are consumed at a constant, continuous rate, although product arrivals and departures are actually discrete events. (7) Operating assumptions help to focus model simulation by choosing, for example, if a static or a dynamic analysis is appropriate, or if a qualitative, a quantitative, or a hybrid form of model is desired. We think that these seven different types of modeling assumptions, which could be represented as first-order logical predicates, cover most distinctions which are implicitly made when modelers formulate traditional, monolithic models. As suggested by Bonczek et al (1981) and Falkenhainer and Forbus (1991), when using first order predicate logic to represent the assumptions underlying a particular model component, one would specify each model component as a logical implication, where the set of the relationships would be the consequence, and the modeling assumptions, expressed as a conjunction of predicates, would be taken as the antecedent.

3.2 Representation of Model Constraints

We allow different representational forms to specify relationships or constraints, including QSIM-type qualitative constraints, quantitative algebraic and differential equations, and RCR rules and constraints as a hybrid form which can be used to integrate models that are formulated at different levels of details. RCR models might also be useful for specializing purely qualitative model fragments, and incorporating numerical, company-specific information in the form of bounds in order to improve preciseness. In the following we discuss the different kinds of relationships which we want to include into our framework, and suggest how we would represent them in the model fragment.

3.2.1 Purely Qualitative Relationships

Theories in management encompass usually general statements which apply to whole classes of organizations. Hence, management theories try to discover commonalities among all organization (of a certain class) with general validity, which can sometimes only tenuously be described as certain trends, influences or tendencies. A widely used practice in research areas such as organization science, management, and behavioral information systems is to use qualitative descriptions in order to formulate causal and functional relationships as general propositions. The abundance of uncertainties and vaguenesses,

which are actually very characteristic of organizational knowledge, often inhibit the specification of precise quantitative models. Qualitative statements are typically based on hypothesized monotonic relationships of the form if variable X is increased (or decreased) then variable Y will increase (or decrease). For example, (a) Cooprider (1990) states the qualitative proposition "increasing the level of partnership among organizational units leads to an increase in the productivity of the entire organization", and (b) Huber (1990) hypothesizes that "For a highly centralized organization, use of computer-assisted communication and decision-support technologies (i.e., information technology (IT) leads to more decentralization.". Each of these two propositions expresses, verbally, a monotonic relationship between two variables which is a very common form in the OS/MT literature. Relationships of this kind could very well be represented in the QSIM modeling language as M^+/M^- constraints, and stored in an organizational knowledge base. Thus, we would specify relationship (a) as a QSIM constraint

(a) PRODUCTIVITY = M^+(PARTNERSHIP),

and relationship (b) similarly as

(b) DECENTRALIZATION = M^+(IT).

However, while relationship (a) is formulated as a generally applicable statement, relationship (b) is conditioned on the assumption that we are operating in a highly centralized organization. This means that we need a corresponding predicate in the modeling assumptions section of the model fragment as well, we may be done by specifying CENTRALIZED (ORGANIZATION).

3.2.2 General Qualitative Relationships

Monge (1990), and Weick (1989), for example, have observed that the lack of appropriate conceptual and computational tools to model inexactly, vaguely, or qualitatively specified systems has lead to the dominance of purely qualitative, typically verbal, formulations of organizational theories. But in order to test and verify theories more formalized methods are needed. Monge outlines some specifications of a representation language for expressing dynamic theories of organizational processes. Monge's proposal is rather a list of requirements than a completely formalized language, but it illustrates that qualitative-quantitative hybrids to formally describe variables and relationships are key concepts for developing a successful organizational computing

system. Monge addresses six dimensions of dynamic behavior when theorizing about individual dynamic variables independently: continuity, magnitude, rate of change, trend, periodicity, and duration. To model cause-effect relationships among two or more variables, he presents a typology of five components: history of variables, lag, rate of change, magnitude of change, and permanence of change.

When describing organizational processes, all quantities (variables) are perceived as functions of time. Continuity, the first dimension, distinguishes between continuous-time variables like organizational climate and discontinuous-time variables like payments. The second dimension, magnitude, refers to the amount of a variable across time. Rate of change specifies how fast a variable changes over time. Trend can be either positive or negative meaning that in the long run a variable increases or decreases. Constants and variables that increase and decrease randomly are said to be trendless. If a variable follows a regularly repeating pattern, periodicity denotes the amount of time that elapses between the occurrence of repeating values. The sixth dimension, duration, refers to mainly discontinuous-time variables that alternate between zero and non-zero values, and measures the length of time a variable has a non-zero value. Duration might be constant or change over time.

Monge proposes five types of dynamic cause-effect relationships involving two or more variables as the basic component for theorizing about organizational processes. The first component is the history of related variables, that is, their record of past values. The history might be known empirically or might be hypothesized. The lag specifies the amount of time until a change in the causal variable shows an effect in the outcome variable. Rate of change refers to the rapidity of change in the causal variable and the rapidity with which its effect occurs in the outcome variable. Magnitude specifies the amount of change in related variables. Finally, permanence indicates how an induced effect continues over time, that is, if it is a permanent or a merely a temporary effect. Ba et al (1993) are currently working on an extension to the RCR language in order to incorporate Monge's suggestions.

Some purely qualitative relationships obtained from qualitative management theory can actually be refined with respect to particular companies under consideration. For example, the relationship "increasing promotional expenditure causes increasing sales volumes," which would be specified as

(c) $\mathrm{SALES} = \mathrm{M}^{+}(\mathrm{PROMOTIONAL\_EXP})$,

which, in this formulation, is a purely qualitative relationship which simply says that sales will monotonically increase with higher promotional expenditures.

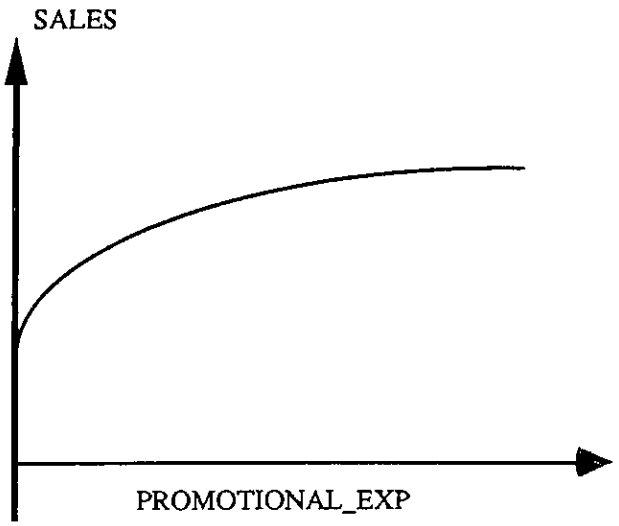


Figure 1: Purely Qualitative Relationship

Relationship (c) could be specialized, if needed for better accuracy, to mirror the company's specific experiences and projections, and be restated more precisely by including specific ranges of the expected increase. Using the RCR language for representing such semi-qualitative constraints, we could restate the above relationship as

$$\begin{aligned}(\text{c1})\text{SALE} = {} & [\text{lb}(\text{PROMOTIONAL\_EXP}, \\ & \text{ub}(\text{PROMOTIONAL\_EXP})],\end{aligned}$$

where lb(PROMOTIONAL_EXP) denotes a lower bounding function, and ub(PROMOTIONAL_EXP) denotes an upper bounding function of the qualitative relationship between promotional expenditure and sales. In order to get specific bounds on relationship (c1) competent experts could then specify ranges for this relationship, as shown in Figures 2a and 2b.
Using interval analysis it is possible to reconcile inconsistent range specifications by taking the union of intervals. In this case, we might get a compromise formulation, shown in Figure 3, which would then be added to the model base as a new fragment.

4.1 Specification of the Domain Theory

First of all, we need a domain theory describing the specific company under study. Constructing an exhaustive domain theory would be, of course, a tremendously time consuming and costly project in itself. Exploiting results from database technology, we can organize the model knowledge base, which contains the domain theory specification, in a somewhat analogous way. Similar to a database schema definition, the domain theory presents a complete, global view on the entire organization, and would be developed and managed by a central, specialized administrator. Particular views on the organization are defined as scenario descriptions, which correspond to database subschema definitions. Users would work with scenario descriptions suited for their particular task, and would issue queries with respect to a certain scenario.

For our purpose, it shall suffice to merely sketch out such a domain theory. Recall, that a model fragment definition consists of two major parts: one that contains the conditions under which the model fragment is applicable (called the modeling assumptions section), and the other that actually encodes the relationships of the model fragment (called the relations section). Thus, a model fragment becomes a candidate for a scenario model, if the conditions of a model fragment are satisfied. The following shows parts of the domain theory of the CORP corporation, and uses relationships proposed by Blanning (1987), Huber (1990), and Cooprider (1990). The complete domain theory would obviously be much more elaborate. For the sake of simplicity, we have left out some of the details in the relationships section which would be necessary in order to render the composite model solvable by a selected solution method such as QSIM or RCR. Domain model fragments are of the 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 this model fragment, output port is a list of the variables which are computed by the model fragment, and which can be shared with other fragments. Variable names are written in upper case letters throughout the model definitions.

The conditions section contains precondition-specifications, which define the modeling assumptions that an instantiation of a model fragment depends on. The consider clause is used to describe certain perspectives, such as an organizsational performance perspective, which we would specify as consider (performance). Perspectives can be further focused, and a specification like consider (performance (financial)) would imply a financial perspective on the organization. Granularity assumptions enforce a certain level of detail. The clause exist (EXPENSE), that is, total manufacturing cost (expense) needs to be considered when reasoning about net income in the model. Other modeling assumptions are specified in a similar way, like operating assumption about the type of relationships (e.g., qualitative or quantitative) in the model, or the kind of analysis (e.g. dynamic or static) the model could be used for. Lastly, the relations section contains relationship specifications, which would be constraints of a particular modeling language. We only assume, that internally, that is, within a single model fragment, the relationships are of a homogeneous type. Across model fragments, we permit heterogeneous relationship specifications by using several modeling languages.

Besides the definition of model fragment, a domain theory also contains rules which constrain the use of the model fragment, and thus help to eliminate potential model candidates. For example, it could be a reasonable assumption, that qualitative performance studies always entail a dynamic analysis, a rule which we have included as rule R-1 in our domain theory in a separate rules section. Another rule, R-2, selects QSIM as the only solver for purely qualitative scenario models.

CORP Domain Theory:

fragment MKT-1 (SALES) {marketing model describing qualitative relationship between price and sale volume}

conditions

consider(exists (PRICE)),

consider(performance (financial)),

model-type(qual), simulation(dynamic)

relations

SALES=M^-(PRICE)

end

fragment MKT-2 (SALES)

{marketing model describing qualitative relationship between price and sale volume}

conditions

consider(exists (PRICE)),

consider(performance (financial)),

model-type(quant), simulation(static)

relations

SALES=80000–44000*PRICE

end

fragment MKT-3 (SALES)

{marketing model describing semi-qualitative relationship between price and sale volume}

conditions

consider(exists (PRICE)),

consider(performance (financial)),

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)),

consider(performance (financial)),

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)),

consider(performance (financial)),

model-type(qual), simulation(dynamic)

relations

EXPENSE=FIX_COST+VAR_COST,

VAR_COST=UNIT_COST*SALES,

constant(UNIT_COST)

constant(FIX_COST)

end

fragment MNF-1 (EXPENSE)

{manufacturing model determining total production cost}

conditions

consider(exists (SALES)),

consider(performance (financial)),

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)),

consider(performance(financial)),

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)),

consider(performance(overall)),

model-type(qual), simulation(dynamic)

relations

PRODUCTIVITY=M^+(PARTNERSHIP)

end

fragment MGT-2 (DECENTRALIZATION)

{qualitative management model describing the effect of the usage of information technology on the company's overall structure}

conditions

consider(exists(IT)),

centralized(organization)

consider(performance(overall)),

model-type(qual), simulation(dynamic)

relations

DECENTRALIZATION=M^+(IT)

rules

R-1: consider(performance(X)) and model-type(qual)=>simulation(dynamic).

R-2: model-type(qual)=>solver(qsim)

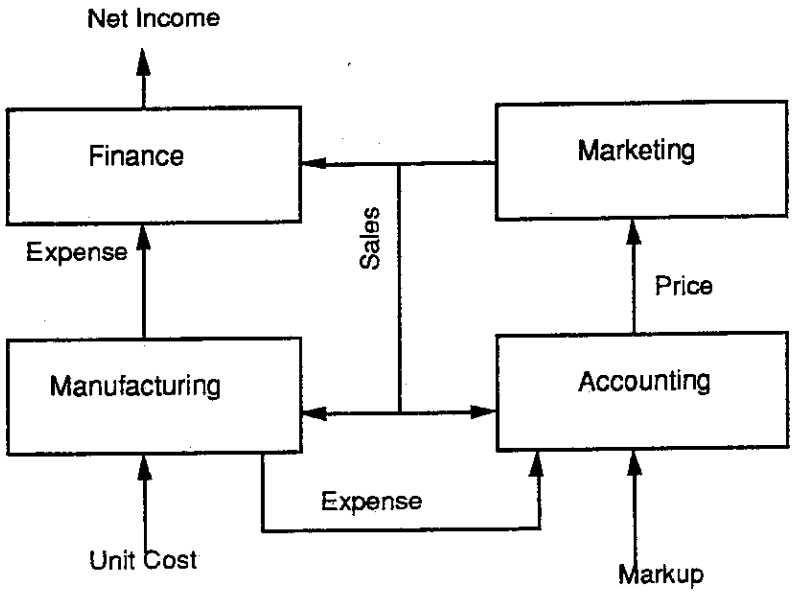
end

4.2 Specification of a Scenario Description

For a given enterprise we would have multiple scenario descriptions, each representing a specific viewpoint on the organization. A scenario description provides us with a structural configuration of the objects constituting the specific scenario being investigated, and, in this regard, some topological information about the CORP company. Usually, it also carries additional information about the system's behavior, such as initial states, and several simplifying assumptions that are reasonable for the given scenario. Kaplan and Norton (1992), for example, suggest a variety of perspectives, which are modeled as interacting organizational units where, depending on the particular view, different activities and phenomena are emphasized. Views would include scenario descriptions in terms of cash flows and material flows, scenarios representing customer perspectives, innovation and learning perspectives, and numerous internal business perspectives. The particular scenario, which is depicted below, represents a global, highly aggregated view on the organizational subunits, and their interactions, which determine the financial performance of the organization.

4.3 Query Analysis

Unless queries are restricted in some way, they can vary tremendously in form and content. Without loosing task and domain independence, we should restrict the query language to convey only basic information. Let us suppose a user posed the query "How does an increase in price affect net income?" Conceptually based on natural language processing, a query elaboration procedure would analyze the issued query, and derive from it a set of ground expressions which would be passed on to the model composition module of the enterprise modeling system for evaluation. In absence of such a sophisticated query analyzer, we could simply devise a primitive query language which basically lists a number of ground expressions which permit the system to identify objects, quantities and relations of interest, where each of these has a referent in the model knowledge base. Hence, let us consider the simplified query {increase (Price), Quantity (amount-of (NetIncome))}, whose ground expressions increase (Price) and Quantity (amount-of (NetIncome)) provide the input to the model composition module.



Department(Finance)	link(Marketing, Finance, Sales)
Department(Marketing)	link(Manufacturing, Finance, Expense)
Department(Manufacturing)	link(Manufacturing, Accounting, Expense)
Department(Accounting)	link(Marketing, Manufacturing, Sales)
cash(NetIncome)	link(Marketing, Accounting, Sales)
cash(Expense)	link(Accounting, Marketing, Price)
units(Sales)	link(,Manufacturing, UnitCost)
price(Price)	link(,Accounting, Markup)
price(UnitCost)	link(Finance,, NetIncome)
percentage(markup)	

Figure 4: A Scenario Description of the CORP Corporation

4.4 Model Composition

The query indicates that we need a scenario model which computes net income. While the one ground expression quantity (amount-of (NetIncome)) of the query does not hint to either a qualitative or a quantitative modeling approach, the other ground expression provides a clear clue for a qualitative analysis. Since the increase operator indicates a desired direction of change without further specification, it suggests a qualitative model for investigating this effect on net income in the given scenario. Now, we could try to enumerate all possible combinations of model fragments, and to prune out those which

either violate some of the modeling assumptions or prove to be irrelevant or insufficient regarding the query. However, this approach would be computationally too costly, considering the many combinations of model fragments, which would typically occur in an enterprise-wide environment. The number of modeling assumptions, on the other hand, tends to be much smaller, and suggests better computational alternative by reasoning about combinations of modeling assumptions first, and then to select and to integrate a suitable set of model fragments. The task is to derive an appropriate modeling environment, that is, to find a set of modeling assumptions consistent with the preconditions of those model fragments which are going to be involved in the constitution of the scenario model. Since we want to find a scenario model which is most useful, that is, which is sufficient and, at the same time, coherent and parsimonious, we should try to compute the minimal modeling environment.

Falkenhainer and Forbus (1991) advocate the application of assumption-based truth maintenance systems, and in particular the ATMS presented by deKleer (1986), as a natural way to manipulate sets of modeling assumptions. The ATMS maintains a dependency network where the nodes represent statements received from the problem solver. never inspected by the ATMS, and treated as data instead. Dependencies between nodes are represented as justifications. The justifications are also supplied by the problem solver, and never examined by the ATMS. Assumptions are nodes which don't depend on any justifications, and a set of such assumptions is called an environment. A node holds in an environment if it can be derived from the justifications and the environment. Nodes are labeled with the set of all environment in which the node holds, and simply called the node's label. Concisely stated, a node consists of triple < datum, label, justification>. Given a new datum and its justification, the ATMS computes the label which contains all consistent and minimal environment in which the new datum can be justified.

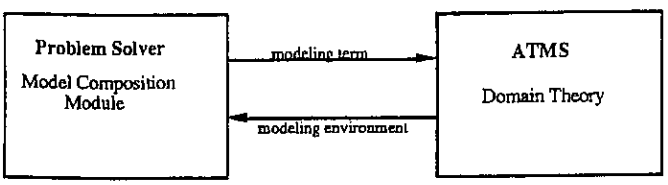


Figure 5: Reasoning with Modeling Assumptions

Figure 5 shows an ATMS application to compositional modeling where

the ATMS is employed to compute adequate modeling environments, which we need for building task-specific scenario models. In our case, the problem solver would be the model composition module which would supply the ATMS with the modeling terms derived from the query, by taking the conjunction of the query's ground expression, for example, (increase (Price), Quantity (amount-of (NetIncome))). Model terms are any terms that could principally appear in a query. Each model term is maintained as the datum of a node in the ATMS dependency network. Given a specific model term, such as (increase (Price)), (Quantity (amount-of (NetIncome))), or the conjunction (increase (Price) and Quantity (amount-of (NetIncome))), the ATMS would return a set of minimal modeling environments. Each environment would be a minimal conjunction of modeling assumptions, which needs to be considered when building a model that reasons about the provided model terms. In general, we have to expect multiple, minimal modeling environment. Hence, each of them could give rise to a different scenario model candidate. In the above example, in which the problem solver supplies the ATMS with the model term (increase (Price) and Quantity (amount-of (NetIncome))), the ATMS would return a set of minimal model environments like

> {{.....},.....,{consider (performance (financial)), consider (exist (Price)). Consider (exist (NetIncome)), consider (exist (Sales)), consider (exist (Expense)), simulation (dynamic), model-type (qual)}, }

on which we would build the scenario model. However, depending on the particular design of the dependency network and the organization of the modeling assumption, 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. After such object expansion and candidate completion steps, we would finally arrive at a stage of the model building process where we would face a collection of complete scenario model candidates from which we would have to choose a final scenario model. Falkenhainer and Forbus discuss some heuristics for this extremely difficult task, which could be used for evaluating candidate models, and selecting the best or most appropriate one. In our example, the final scenario model to determine the change in net income after increasing price would look something like

scenario: NetIncome

constant(UNIT_COST)

constant(FIX_COST)

constant(MARKUP)

SALES=M^{-}(PRICE)

PRICE=MARKUP+EXPENSE/SALES,

EXPENSE=FIX_COST+VAR_COST,

VAR_COST=UNIT_COST*SALES,

NET_INCOME=PRICE*SALES-EXPENSE

end.

This, of course, is not a completely specified model yet, one which could be solved as it is. However, it shows the complete specification of the mode's constraints, the core part of the final scenario mode. To complete the model specification, we still need to provide an initial state of the system, that is, we need to specify that price is initially increasing, and we need to specify the current, qualitative values of the other involved variables. While the initial state value of price would be derived from the query, we would obtain the other values from an organizational database. Which details remain to complete the scenario model specification depends on the solver selection. In our example, we have used a QSIM kind of language to represent qualitative relationships. If we chose QSIM as the solver of the scenario model, we would already have a complete constraints specification for a QSIM model, but we would need to include the initial state specification, and we would also need to include appropriate quantity space definitions. All this extra information required for completing the specification of particular scenario models could be kept in the fragment definitions. However, conceptionally, we prefer a solution where this information is maintained separately, because it is context independent and can be reused in different settings and scenarios. Thus, we suggest to set up a database as a part of the model knowledge base where we store and maintain information like current value of system variables, possible quantity space definitions, and quiddity information, that is information about the intended interpretation of the variables. Bhargava et al (1991) suggest a representation of quiddity information, and also discuss the, more practical yet very difficult, problems of possible variable name violations, and the problems of dealing with variables expressed in different dimensions and measurement units.

The integration of the relationships in the above example scenario model was relatively straightforward. Because we have used only one qualitative representation language in our illustration, and the query indicated a qualitative scenario model, we could simply merge the relations sections of the involved model fragment. In a situation like this, all we have to worry about is to check the composite model for possible inconsistencies among the constraints, and the possible removal of redundant constraints. Fortunately, both tasks could be left to the solver, which, presumably would not be affected (much) by redundant constraints, and which would detect inconsistent constraints by failing to solve the model.

The greater the diversity of the modeling languages is, which we allow for specifying relationships, the more expressive the entire model knowledge base becomes. The task of integrating model fragments, on the other hand, gets more complicated as the heterogeneity of the model knowledge base increases. Since, in our model integration approach, we are favoring the merging of model components, like in compositional modeling, over communicating messages between model components, the issue of compatibility becomes more severe. We need to carefully check the compatibility of different model fragments before we actually attempt to convert and then to merge the relationships into a new, composite model. We suggest a model transformation mechanism based on the qualitative abstraction principle, which was developed in Hinnanken et al (1993b). The basic idea of qualitative abstraction is, that we can take two (or more) models, identify the one with the highest abstraction level, and then abstract the other one to this higher level. To illustrate this method, let us suppose that we need to integrate the two models

$$\text{M1}: \quad \text{SALES} = [68000, 92000] + [40000, 48000] * \text{PRICE}, \text{and}$$
$$\text{M2}: \quad \text{PRICE} = 1.1 + 3,340,653/\text{SALES}.$$

Now, we see that model M2 is a precisely stated quantitative model, and that model M1 is a semi-qualitative model formulation. Hence, we identify model M1 as the one with the highest abstraction level among the participating models, and select its representation language as the target language of the composite model, say model M3. In this example, we would transform model M2 into an equivalent formulation, model M2', using the semi-qualitative target language, which, in this case, resembles the RCR language. Thus, we get

$$\text{M2}': \text{PRICE} = [1.1, 1.1] + [3340653, 3340653]/\text{SALES}$$

were PRICE and SALES are now viewed as interval-valued variables. Model integration is achieved by merging models M1 and M2' into a new model

$$\begin{aligned} \text{M3}: \quad & \text{SALES} = [68000, 92000] + [40000, 48000]^{*}\text{PRICE} \\ & \text{PRICE} = [1.1, 1.1] + [3340653, 3340653]/\text{SALES}. \end{aligned}$$

The usage of interval representations as sort of a super language which can subsume quantitative and qualitative relationships bears some resemblance to an embedding language of the kind presented in Bhargava and Kimbrough (1993). Obviously, this technique has its limitations, and we cannot expect to merge any arbitrary combination of language representations, neither can we expect to retain all information when we perform the conversion through an abstraction mechanism. For example, it would be hard to integrate a SAS model with a linear program formulation in GAMS. A rule base which would determine an embedding language for a given model combination, and which would infer incompatible combinations of language representation could be used to prevent us from attempting to merge incompatible model fragments. This rule base could be incorporated into the ATMS managing the modeling assumptions, or could be kept as a separate tool in the model knowledge base.

A related issue is the one of trying to combine a static and a dynamic model formulation. This could be ruled out by simply using the information that the two fragments are conditioned on different modeling assumptions, namely, the operating assumptions simulation (static) versus simulation (dynamic). Attempting to reformulate dynamic models as a static one might be inappropriate in most cases, but, on the other hand, generalizing a static model into a dynamic model is often a reasonable thing to do. For example, model

$$\text{M1}: \text{SALES} = [68000, 92000] + [40000, 48000] * \text{PRICE},$$

which is a static one, could be generalized, and reformulated as

$$\text{M1}': \text{SALES(t)} = [68000, 92000] + [40000, 48000] * \text{PRICE(t)}.$$

In absence of a better dynamic model in the model base, model M1′ could serve as a valuable piece in a dynamic scenario model which needs to have sales included.

5. Discussion and Conclusion

In this paper, we have outlined a novel, comprehensive framework for future enterprise-wide modeling systems. While some parts of our framework

are on a mainly suggestive level, it does present, to our knowledge, the first comprehensive model building framework which addresses the important issues in organizational modeling in one unifying approach. We believe that future research in model building and model management for decision support in organizational environments requires more attention to related model building and model reasoning research in artificial intelligence. One purpose of this paper is to bring to bear some of the stimulating results obtained from the AI community, and to indicate how they can be incorporated into the DSS research on model building. Among the new features we have proposed we want to highlight those two, which, we strongly feel, map out the most promising future research directions. First, the possibility of both qualitative and quantitative model formulations, which introduces a new level of versatility to organizational model building, and which should widen the scope of computer supported decision tools considerably. Second, the application of a compositional modeling strategy to automatically build task-specific secnario models, which liberates users from having to specify special modules for controlling the modeling integration process.

We conclude by mentioning several, more specific issues which need further investigation. The success of compositional modeling depends heavily on a clear organization of the domain theory. We still need a better understanding of how to decompose an organizational environment into semi-independent components in a manner which really reflects how different parts of the organization work together to accomplish the common, corporate goals. More research needs to be done to develop comprehensive taxonomies and ontologies which are necessary as a conceptual basis for the formulation of organizational descriptions which, truly, can be considered as interpretable knowledge units that can be used to synthesize new knowledge. Work in organizational behavior and management, such as Orton and Weick (1990) and Kaplan and Norton (1992), could help to structure organizational knowledge more systematically. A realistic organizational setting would take place in a distributed environment. Thus, one would like to extend our framework to a distributed computing system, where the model knowledge base is managed across functional organizational units and geographical locations.

References

1. Addanki, S., R. Cremonini, and J.S. Penberthy, "Graphs of Models." Artificial Intelligence, 51, pp.145–178.
2. Ba, S., K.R. Lang, and A.B. Whinston "Modeling Organizational Processes with RCR." Center for Information Systems Management, Technical Report, The University of Texas at Austin, 1993.
3. Balakrishnan, A., and A.B. Whinston "Information Issues in Model Specification." Information Systems Research, 2(4), 1991, pp.263–286.
4. Bhargava, H.K, and S.O. Kimbrough "Model Management: An Embedded Languages Approach.". forthcoming in Decision Support Systems, 1993.
5. Bhargava H.K., S.O. Kimbrough, and R. Krishnan "Unique Names Violations, a Problem for Model Integration or You Say Tomato, I say Tamahto." ORSA Journal on Computing, 3(2), 1991, pp.107–120.
6. Bhargava, H.K., and R.Krishnan "Reasoning with Assumptions, Defeasibility, in Model Formulation." working paper, The H. John Heinz III School of Public Policy and Management, Carnegie Mellon University, 1991.
7. Blanning, R.W. "A Relational Theory of Model Management." in Holsapple, C.W., and A.B. Whinston(eds.), Decision Support Systems: Theory and Application, Springer Verlag, 1987.
8. Bonczek, R.H., C.W. Holsapple, and A.B. Whinston Foundations of Decision Support Systems, Academic Press, 1981.
9. Cooprider, J.G. "Partnership Between Line and I/S Managers: A Management Model." Ph.D. Thesis, MIT Sloan School of Management, Cambridge, MA, September 1990.
10. Davis, E. (1987) "Constraint Propagation with Interval Labels." Artificial Intelligence, 32, 1987, pp.281–331.
11. deKleer, J., and J.S. Browne "A Qualitative Physics Based on Confluences." Artificial Intelligence, 24, 1984, pp.7–83.
12. deKleer, J. "An Assumption-Based TMS." Artificial Intelligence, 28, 1986, pp.127–162.

13. Dolk, D.R., and J.E. Kottemann "Model Integration and a Theory of Models." Decision Support Systems, 9(1), 1993, pp.51–63.

14. Falkenhainer, B., and K.D. Forbus "Compositional Modeling: Finding the Right Model for the Job." Artificial Intelligence, 51, 1991, pp.95–144.

15. Forbus, K.D. "Qualitative Process Theory." Artificial Intelligence, 24, 1984, pp.85–168.

16. Geoffrion, A.M. "An Introduction to Structured Modeling." Management Science, 33(5), 1987, pp.547–588.

17. Granof, M.H. Financial Accounting Principles and Issues, Pretice-Hall, Englewood Cliffs, N.J., 1985.

18. Hinkkanen, A., K.R. Lang, and A.B. Whinston "On the Usage of Qualitative Reasoning as Approach Towards Enterprise Modeling." forthcoming in Annals of Operations Research, 1993a.

19. Hinkkanen, A., K.R. Lang, and A.B. Whinston "Qualitative Modeling And Decision Support: On the Representation of Organizational Systems and the Improvement of their Performance." forthcoming in Proceedings of Quardet '93: III IMACS International Workshop On: Qualitative Reasoning And Decision Technologies, Barcelona, Spain, 16–18 June 1993.

20. Huber, G.P. "A Theory of the Effects of Advanced Information Technologies on Organizational Design, Intelligence, and Decision Making." Academy of Management Review, 15(1), 1990, pp.47–71.

21. Kaplan, R.S., and D.P. Norton "The Balanced Scorecard-Measures That Drive Performance." Harvard Business Review, 70(1), 1992, pp.71–79.

22. Kiang, M.Y., A. Hinkkanen, and A.B. Whinston "An Interval Propagation Method (Rules Constraints Reasoning) for Computing Qualitatively Defined Systems." Working Paper, MSIS Department, The University of Texas at Austin, 1993.

23. Krishnan, R., P. Piela, A. Westernberg "On Supporting Reuse in Modeling Environments." working paper, School of Urban and Public Affairs and Engineering Design, Carnegie Mellon University, 1991.

24. Kuipers, B. "Qualitative Simulation." Artificial Intelligence, 29, 1986, pp.289–338.

25. Kuipers, B., and 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.

26. Monge, P.R. "Theoretical and Analytical Issues in Studying Organizational Processes." Organization Science, 1, 1990, pp.406–430.

27. Moore, J., K.R. Lang, and A.B. Whinston "Computational Systems for Qualitative Economics." working paper, Center for Information Systems Management, The University of Texas at Austin, 1993.

28. Muhanna, W.A., and R.A. Pick "Meta-Modeling Concepts and Tools for Model Management: A Systems Approach." forthcoming in Management Science, 1992.

29. Orton, J.D., and K.E. Weick "Loosely Coupled Systems: A Reconceptualization." Academy of Management Review, 15(2), 1990, pp.203–23.

30. Weick, K.E. "Theory Construction as Disciplined Imagination." Academy of Management Review, 14(4), 1989, pp.516–32.

31. Weld, D.S. "Reasoning About Model Accuracy." Artificial Intelligence, 56, 1992, pp.225–300.

32. Williams, B.C. "A Theory of Interactions: Unifying Qualitative and Quantitative Algebraic Reasoning." Artificial Intelligence, 51, 1991, pp.39–94.