Enterprise Modeling

Mark S. Fox and Michael Gruninger

■ To remain competitive, enterprises must become increasingly agile and integrated across their functions. Enterprise models play a critical role in this integration, enabling better designs for enterprises, analysis of their performance, and management of their operations. This article motivates the need for enterprise models and introduces the concepts of generic and deductive enterprise models. It reviews research to date on enterprise modeling and considers in detail the Toronto virtual enterprise effort at the University of Toronto.

An *enterprise model* is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise. It can be both descriptive and definitional—spanning what is and what should be. The role of an enterprise model is to achieve model-driven enterprise design, analysis, and operation.

From a design perspective, an enterprise model should provide the language used to explicitly define an enterprise. We need to be able to explore alternative models in the design of enterprises spanning organization structure and behavior. To reason about alternative designs for enterprises, we need to reason about different possible sets of constraints for enterprises within the model. We need to ask the following questions: Can a process be performed in a different way, or can we achieve some goal in a different way? Can we relax the constraints in the enterprise such that we can improve performance or achieve new goals?

We also need to be able to determine the impact of changes on all parts of the enterprise. For example, how will relaxation of policies affect the quality of products or services provided by the enterprise? How will the purchase of a new machine affect the activities that are performed? Will we need to retrain people in the enterprise to give them the skills to use the machine? How will changing the activities change resource consumption?

From an operations perspective, the enterprise model must be able to represent what is planned, what might happen, and what has happened. It must supply the information and knowledge necessary to support the operations of the enterprise, whether they be performed by hand or machine. It must be able to provide answers to questions commonly asked in the performance of tasks.

In this article, we motivate the need for an enterprise model and introduce the concept of a generic enterprise model (GEM). We then extend the concept of a GEM to a deductive enterprise model (DEM) and then briefly review research to date. Next, we discuss criteria for selecting a GEM or a DEM and then review the Toronto virtual enterprise (TOVE) DEM effort at the University of Toronto.

Motivation: Integrating the Enterprise

To remain competitive, enterprises, regardless of whether they are industrial, financial, governmental, or something else, must produce products and services that are "of consistently high quality throughout the product/service's life, customized to local market needs, open in that they may be integrated with other products/services, environmentally benign, and technically advanced" (Nagel and Dove 1991, p. 7). The key to achieving these capabilities is "agility" (Nagel and Dove 1991, p. 7). Agility implies the ability to "continuously monitor market demand; quickly respond by providing new products, services, and information; quickly introduce new technologies; and quickly modify business methods" (Nagel and Dove 1991, p. 9). The problem is how to design and operate an agile enterprise.

Nagel (1993, p. 77) provides four strategic principles of agility: (1) "Use an entrepreneurial organization strategy." (2) "Invest to increase the strategic impact of people and information on the bottom line." (3) "Use the virtual organization strategy as a dynamic structure both inside and outside the enterprise." (4) "Adopt a value-based strategy to configure your products and services into solutions for your customers."

To remain competitive, enterprises must produce products and services that are "of consistently high quality throughout the product / service's life, customized to local market needs, open in that they may be integrated with other products / services, environmentally benign, and technically advanced."

The entrepreneurial and virtual nature of the agile organization, coupled with the need for people and information to have a strategic impact, entails a greater degree of communication, coordination, and cooperation within and among enterprises. In other words, the agile organization must be integrated. By *integrated,* we mean the structural, behavioral, and informational integration of the enterprise.

Business-process reengineering addresses the issue of structural integration by reorganizing enterprises along critical business processes, such as the supply chain and the product life cycle, that span traditional organizational units such as sales and manufacturing (Davenport 1993; Hammer and Champy 1993).

Just as important as integrating the structure of the enterprise along process lines is the behavioral and informational integration of the enterprise. Hansen (1991) defined five principles of behavioral and informational integration:

First is "when people understand the vision, or larger task, of an enterprise and are given the right information, the resources, and the responsibility, they will 'do the right thing'."

Second is "empowered people—and with good leadership, empowered groups—will have not only the ability but also the desire to participate in the decision process."

Third is "the existence of a comprehensive and effective communications network.… This network must distribute knowledge and information widely, embracing the openness and trust that allow the individual to feel empowered to affect the 'real' problems."

Fourth is "the democratization and dissemination of information throughout the network in all directions irrespective of organizational position…ensures that the Integrated Enterprise is truly integrated."

Fifth is that "information freely shared with empowered people who are motivated to make decisions will naturally distribute the decision-making process throughout the entire organization."

Achieving integration requires more than principles; it also requires the development of an information infrastructure that supports the communication of information and knowledge, the making of decisions, and the coordination of actions. At the heart of this infrastructure lies a model of the enterprise.

Over the last 30 years, the role of enterprise models in the design and operation of enterprises has reached the point that few organizations of significant size can operate without them. For example, manufacturing requirements planning (MRP) systems have at their core a data model of the organization, spanning resources, activities, and products. They use this model to plan and control operations. Today, MRP systems have evolved into enterprise requirements planning systems, where the enterprise model is viewed as a major component.[1] Similarly, business-process reengineering tools, such as FIRSTSTEP from Interfacing Technologies, BONAPART from UBIS GmbH, and RETHINK from Gensym, have at their core an enterprise model.[2] It would not be overly general to say that most information systems in use within an enterprise incorporate a model of some aspect of the enterprise's structure, operations, or knowledge.

The problem that we face today is that the legacy systems that support enterprise functions were created independently and, consequently, do not share the same enterprise models. We call this the *correspondence problem*. Although each enterprise model might represent the same concept, for example, activity, they will have a different name, for example, operation versus task. Consequently, communication among functions is not possible without at least some translation, but no matter how rational the idea of renaming them is, organizational barriers impede it. Further, these representations lack an adequate specification of what the objects (terminology) mean (that is, semantics). Instead, concepts are poorly defined, and their interpretations overlap, leading to inconsistent interpretations and uses of the knowledge. Finally, the cost of designing, building, and maintaining a model of the enterprise is large. Each tends to be unique to the enterprise; objects are enterprise specific.

As a solution to this problem, there has been an increasing interest in GEMs. A *GEM* is an object library that defines the classes of objects that are generic across a type of enterprise, such as manufacturing or banking, and can be used (that is, instantiated) in defining a specific enterprise. A GEM is composed of the following: (1) a set of object classes structured as a taxonomy (that is, each object is linked to one or more other objects by a subclass-superclass relationship plus a definition of how a class refines its superclass); (2) for each object class, a set of relations linking it to other object classes plus a definition of the intended meaning of each relation; (3) for each object class, a set of attributes plus a definition of the intended meaning of each attribute.

The benefits of employing a GEM at the outset when creating an enterprise model are as follows:

Predefined object library: Most database

engineers often start from scratch when creating an enterprise model. Defining the correct set of object classes is a daunting and time-consuming task. A GEM provides the object classes, allowing the engineer to quickly move on to model instantiation.

Path for growth: Many enterprise modelers do not know what they have left out until it is too late. By incorporating a GEM, many of the concepts that they might not have anticipated needing are already there; the modeler has benefited from the experience of others.

Shared conceptualization: By adopting a GEM, other parts of the organization stand a greater chance of understanding what is represented in the enterprise model.

Ultimately, these benefits affect the bottom line. Both time and costs are reduced.

Commonsense Enterprise Models

The usefulness of an instantiated GEM is determined by the functions it can support, for example, scheduling, forecasting, and accounting. Because the interface to an enterprise model is through the query language provided by the underlying database, the functions that a GEM can support are determined by the categories of queries that the GEM can provide answers to (if properly instantiated). However, the queries that a GEM can answer are not just determined by the object library and its instantiations but by additional processing that might be provided.

Where does the GEM end and inference begin? If no inference capability is to be assumed, then question answering is strictly reducible to looking up an answer that is represented explicitly in the model. In contrast, current object models have assumed at least inheritance as a deduction mechanism; answers can be provided that assume properties of the class apply to an instance. In defining an enterprise model, key questions are as follows: Should we be restricted to just an object library? Should the objects assume an inheritance mechanism or some type of theorem-proving capability, as provided, say, in a logic programming language with axioms restricted to Horn clauses (that is, PROLOG)? In other words, what is the deductive capability that is to be assumed by a GEM?

We introduce three types of query: (1) factual, (2) expert, and (3) common sense. Consider a relational database system. Such databases support factual queries by the direct retrieval of information represented explicitly in the model (that is, surface-level processing). Consider a model with an SQL interface. Information is explicitly represented if it can be retrieved using a simple select command. For example, if the model contains a works-for relation, and it is explicitly represented that Joe works-for Fred, then the database can return the answer *Fred* in response to a query of "who does Joe works-for".[3]

Expert queries require that the information system have extensive knowledge and reasoning capabilities (that is, *deep-level processing*). Expert systems provide deep-level processing (Fox 1990). By deep level, we mean that a significant amount of knowledge or search, that is, deductions, has to be performed to provide a response to a query. To answer a query regarding the cause of a machine malfunction, the expert system might have to reason about the structure and behavior of the machine. It must have a detailed model of the domain, and it can be unique to the specific enterprise. Such systems tend to be costly to build and maintain and are narrow in scope.

Commonsense queries require that the information system be able to deduce answers to questions that one would normally assume can be answered if one has a commonsense understanding of the enterprise. Such an understanding often represents knowledge about the enterprise acquired over a relatively short period of time, for example, three to nine months, and does not denote knowledge of an expert nature. That is, the knowledge should be broad and not deep and must support a tractable subclass of queries. For example, knowledge of an organization's structure, roles, goals, and resources would enable the deduction of what resources a person might allocate based on his/her role in the organization.[4] It could be argued that the majority of queries posed to a database are in this third category: common sense. If GEMs were designed to support commonsense queries, a significant portion of the management information system (MIS) backlog could be done away with.

Commonsense queries require that the information system be able to deduce answers to questions that one would normally assume can be answered if one has a commonsense understanding of the enterprise.

Commonsense query processing assumes a third level of processing that we refer to as *shallow-level processing*. By shallow level, we mean retrieval that requires a small number of deductions to answer the query. For an enterprise model to support commonsense query processing, it must provide a set of rules of deduction, that is, axioms. For the works-for example, we would require an axiom stating that works-for is transitive:

x works-for y AND y works-for z IMPLIES x works-for z . (1)

We distinguish between (1) an enterprise model that includes axioms that support deduction

and (2) a model without axioms where deductions are specified by the query. In the first case, the model would be able to deduce that Joe works-for John in response to a query asking "who does Joe work for?" In the second case, the user would have to specify a complex query that would include as many join commands as necessary to travel along the works-for relation. Because the user does not know at the outset the depth of the works-for path, he/she might not get the information he/she is looking for. We call a GEM that includes axioms and a deduction engine (that is, theorem prover or a deductive database) a DEM. We say a DEM possesses common sense if its axioms define the meaning of the relations and attributes in the object library.

In summary, the design, creation, and maintenance of software is fast becoming the dominant cost of automation. A significant portion of these costs is for software that provides answers deduced from the contents of the enterprise model.[5] Many of these questions could be answered automatically if the enterprise model had the common sense to answer them!

Enterprise-modeling ontologies *are distinguished by their scope and the central role of integrating multiple ontologies. The ontologies must be able to represent concepts in the domains of activity, time, resource, product, service, organization, goal, and policy.*

Enterprise Modeling: Current Approaches

Enterprise-modeling ontologies are distinguished by their scope and the central role of integrating multiple ontologies. The ontologies must be able to represent concepts in the domains of activity, time, resource, product, service, organization, goal, and policy. Further, these ontologies must be integrated to support reasoning that requires the use of multiple ontologies and support interoperability among tools using different ontologies. For example, the notion of manufacturability requires reasoning about the product properties, preconditions, and effects of activities and the capabilities of resources.

All ontologies consist of a vocabulary along with some specification of the meaning or semantics of the terminology within the vocabulary. The various ontologies can also be distinguished by their degree of formality in the specification of meaning. With informal ontologies, the definitions are expressed loosely in natural language. Semiformal ontologies provide weak axiomatizations, such as taxonomies, of the terminology. These ontologies can serve as a framework for shared understanding among people but is often insufficient to support interoperability, and ambiguity can hinder integration. Formal ontologies define the language of the ontology, a set of intended interpretations of the terminology, and a set of axioms that are sound and complete with respect to the intended interpretations; that is, every consistent interpretation of the axioms is an intended interpretation, and every intended interpretation is consistent with the axioms.

ICAM

Attempts to create industrywide standards began to appear with the United States Air Force integrated computer-aided manufacturing (ICAM) effort in the early 1980s. A GEM for the aerospace industry was developed (Martin and Smith 1983; Martin et al. 1983). The following list is an example of the part object's relations from a design perspective, as defined in the ICAM model; each relation is followed by other objects in the ICAM GEM that the relation links the part object to: (1) Is Changed By—part change (also shown as "is modified by"); (2) Appears As—next assembly usage item (also shown as "is referenced as"); (3) Has—replacement part; (4) Has Subtype (Is)—parts list item, replacement part; (5) Is Used As—next assembly usage, advance material notice item part, configuration list item; (6) Is Totally Defined By—drawing; (7) Is Listed By (Lists)—configuration list; (8) Is Used In—effectivity; and (9) Is Fabricated From—authorized material.

The following are the basic relations and objects they are linked to for a part from a manufacturing perspective in the ICAM GEM (Smith, Ruegsegger, and St. John 1983): (1) Has—N.C. program, material issue, component part, alternative part, part-process specification use, material receipt, work package, part tool requirement, part requirement for material, standard routing use, image part, part drawing; (2) Is Assigned To (Has Assigned To It)—index; (3) Is Defined By (Defines)— released engineering drawing; (4) Is Subject Of—quote request, supplier quote; (5) Is Transported By—approved part carrier; (6) Is Received By—Supplier del lot. (7) Appears As—part lot, ordered part, serialized part instance, scheduled part, requested purchase part; (8) Conforms To—part specification; (9) Is Inverse—component part, alternate part, section, end item, configured item, image part; (10) Is Used As—component part callout, process plan material callout; (11) Is Supplied By—approved part source: (12) Manufacture Is Described By—process plan; (13) Satisfies—end-item requirement for part; (14) Is Requested By—manufacturing request; (15) Is Stored At—Stock location use for part; and (16) Is Specified By—bill of materials (BOM) item.

Toronto Virtual Enterprise Ontology

The goal of the TOVE Project (Fox, Chionglo, and Fadel 1993) is to create an ontology that has the following characteristics: (1) it provides a shared terminology for the enterprise that every application can jointly understand and use; (2) it defines the meaning (semantics) of each term in a precise and an unambiguous as possible manner using first-order logic; (3) it implements the semantics in a set of PROLOG axioms that enable TOVE to automatically deduce the answer to many commonsense questions about the enterprise; and (4) it defines a symbology for depicting a term, or the concept constructed thereof, in a graphic context.

The TOVE DEM currently spans knowledge of activity, time, and causality (Gruninger and Pinto 1995; Gruninger and Fox 1994), resources (Fadel 1994; Fadel, Fox, and Gruninger 1994), cost (Tham, Fox, and Gruninger 1994), quality (Kim and Fox 1994; Kim, Barbuceanu, and Gruninger 1995), organization structure (Fox, Barbuceanu, and Gruninger 1995), product (Lin, Fox, and Bilgic 1996), and agility (Atefi 1997). The TOVE test bed provides an environment for analyzing enterprise ontologies; it provides a model of an enterprise and tools for browsing, visualization, simulation, and deductive queries.

Enterprise Ontology

The ENTERPRISE Project at the University of Edinburgh (Uschold et al. 1997) aims to provide an environment for integrating methods and tools for capturing and analyzing key aspects of an enterprise based on an ontology for enterprise modeling. This ontology (ENTERPRISE) is semiformal; it provides a glossary of terms expressed in a restricted and structured form of natural language supplemented with a few formal axioms.

The ENTERPRISE ontology has five top-level classes for integrating the various aspects of an enterprise: (1) metaontology: entity, relationship, role, actor, and state of affairs; (2) activities and processes: activity, resource, plan, and capability; (3) organization: organizational unit, legal entity, management, and ownership; (4) strategy: purpose, strategy, help to achieve, and assumption; (5) marketing: sale, product, vendor, customer, and market.

IDEF Ontologies

The ontologies developed at KBSI are intended to provide a rigorous foundation for the reuse and integration of enterprise models (Fillion et al. 1995). Thus, the ontologies play two important roles: (1) providing a neutral medium for integrating modeling tools within a software environment, and a framework within which to interpret individual enterprise models, to draw logical connections between models and (2) detecting inconsistencies when integrating models.

This emphasis on semantic integration requires a formal axiomatization of the classes and relations within an ENTERPRISE model. The ontology is a first-order theory consisting of a set of foundational theories, along with a set of ENTERPRISE models that are extensions to these theories, for some specific set of logical constants that specify the entities within the ENTERPRISE model. The approach has been used to provide axiomatizations of IDEF0 and IDEF1X.

Process-Interchange Format

The goal of the Process-Interchange Format (PIF) Project is to develop an interchange format to support the automatic exchange of process descriptions among a wide variety of business-process modeling and support systems, such as work-flow tools, process simulation systems, business-process reengineering tools, and process repositories. Rather than develop ad hoc translators for each pair of process descriptions, PIF serves as the common format for all systems to support interoperability.

PIF is a formal ontology that is structured as a core ontology plus a set of extensions known as *partially shared views* (PSVs). The intuition is that all systems agree on the definitions of terms within the core but agree on the definitions for other terms if they are defined in common PSVs.

The PIF core ontology consists of the classes and relations used to describe the basic elements of any process. The top-level of the ontology defines the classes' activity, object, agent, and time point along with the relations *performs* (over agents and activities), *uses, creates, modifies* (all three over activities and objects), *before* (over time points), and *successor* (over activities).

NIST Process-Specification Language

The goal of the Process-Specification Language (PSL) Project (Schlenoff et al. 1996) at the National Institute of Standards and Technology (NIST) is to create a process-specification language to facilitate complete and correct exchange of process information among manufacturing applications. Included in these applications are scheduling, process planning, simulation, project management, work flow, business-process reengineering, and product

realization process modeling. Although primarily an ontology for processes, it lays the foundations for the integration of ontologies required for enterprise modeling by adopting the following structure:

The *core* is the most basic, essential requirement inherent to all processes. Although all processes contain core requirements, the core requirements provide the basis for representing only the simplest of processes (for example, time, resource, activity).

The *outer core* is the pervasive but not essential requirement for describing processes common to most applications (for example, temporal constraints, resource grouping, alternative tasks).

Extensions are the groupings of related requirements, common to some, but not all, applications that together provide an added function (for example, goals, intentions, organization constraints, products).

Application-specific requirements are requirements that are only relevant within specific applications (for example, dynamic rescheduling for the production-scheduling application).

CIMOSA

The objective of CIMOSA (COMPUTER-INTEGRATED MANUFACTURING—OPEN-SYSTEM ARCHITECTURE) (AMICE 93) (Bernus, Nemes, and Williams 1996) is the appropriate integration of enterprise operations by means of efficient information exchange within the enterprise with the help of information technology; further, it defines an integrated methodology to support all phases of a CIM system life cycle from requirements specification through system design, implementation, operation, and maintenance. This description is used to control the enterprise operation and to plan, design, and optimize updates of the real operation environment. To fully model specific aspects of the enterprise, CIMOSA defines four different views concerned with the enterprise: (1) function, (2) information, (3) resources, and (4) organization.

The *function view* describes the functional structure required to satisfy the objectives of the enterprise and the related control structure, that is, the rules that define the control sequences, or the flows of actions within the enterprise and the principles of the underlying business processes.

The *information view* describes the information required by each function.

The *resource view* describes the resources and their relationship to functional and control structures and organizational structures.

The *organization view* is the description of the enterprise organizational structures, that is, the responsibilities assigned to individuals for functional and control structures, information, and resources.

CIMOSA considers an *enterprise function* as a unified construct of the business user's view of what tasks are required to achieve a particular enterprise objective. The enterprise function consists of three major sections: (1) the *functional part* (which captures the objectives and constraints as well as the relationship between input and output), (2) the *behavior part* (which captures the dynamic section of the enterprise function such as procedural rules for flow of control), and (3) the *structural part* (which specifies the relationships among different levels of decomposition within a given enterprise function).

PERA

The PURDUE REFERENCE ARCHITECTURE (PERA) was developed as an endeavor in enterprise modeling for a computer-integrated manufacturing (CIM) factory by the Purdue Laboratory for Applied Industrial Control at Purdue University (Bernus, Nemes, and Williams 1996; Williams 1991).

The functional descriptions of the tasks and functions of the enterprise are divided into two major streams: (1) *information* (including decision, control, and information) and (2) *manufacturing,* or *customer service*. The information stream is initiated by the planning, scheduling, control, and data management requirements of the enterprise, whereas the manufacturing stream is initiated by the physical production requirements of the enterprise. On implementation, the two functional streams are rearranged into three implementation sets of tasks and functions: (1) human activities that are information and manufacturing or customer-service related, (2) information stream activities not carried out by humans, and (3) manufacturing and customer-service activities not carried out by humans.

GERAM

GERAM (GENERIC ENTERPRISE REFERENCE ARCHITECTURE AND METHODOLOGY) is about those methods, models, and tools that are needed to build an integrated enterprise (Bernus, Nemes, and Williams et al. 1996). The coverage of the framework spans products, enterprises, enterprise integration, and strategic enterprise management, with the emphasis on the middle two.

Enterprisewide Data Modeling

The *enterprisewide data-modeling* contribution by Scheer (1989) was initiated in Germany in the mid-1980s and undertakes to construct

data structures for typical functional areas (departments), such as production, engineering, purchasing, human resources, sales and marketing, accountancy, and office administration, that are generally encountered in an enterprise with the aim of supporting planning, analysis, and traditional accounting systems in general.

Scheer uses the entity-relationship model to systematically develop the data structures for the enterprise in terms of entities (something that can be identified in the users' work environment), attributes (characteristics-properties of an entity), and relationships (the association of entities with one another). The temporal (time) aspects of data structures are recognized in a rudimentary fashion through the identification of key attributes of entities as calendar time related, such as date, year, and time-table-number. The temporally dependent processes, which are typically transaction data, are defined by their links with the special entity type *time*.

Enterprisewide data modeling is then achieved by the transformation of the data structures into relational and network data models to represent the data relationships between the various functional areas of the enterprise. An integrated database for the modeled enterprise is formed by combining the data structures, and this database is embedded within the framework of an MIS.

Evaluating Enterprise Models

Although all the efforts described earlier seek to create a sharable representation of the enterprise, there has never been a well-defined set of criteria that these efforts should satisfy. Put another way, how can you determine which DEM is right for your task?[5] We believe there are six characteristics that should be used to evaluate a DEM:

First is *functional completeness:* Can the DEM represent the information necessary for a function to perform its task?

Second is *generality.* To what degree is the DEM shared between diverse activities such as engineering design and manufacturing or design and marketing? Is the DEM specific to a sector, such as manufacturing, or applicable to other sectors, such as retailing and finance?

Third is *efficiency.* Does the DEM support efficient reasoning, that is, space and time, or does it require some type of transformation?

Fourth is *perspicuity*. Is the DEM easily understood by the users so that it can be applied consistently and interpreted across the enterprise? Does the representation document itself?

Fifth is *precision granularity.* Is there a core set of ontological primitives that are partitionable, or do they overlap in meaning? Does the representation support reasoning at various levels of abstraction and detail?

Sixth is *minimality.* Does the DEM contain the minimum number of objects (that is, terms or vocabulary) necessary (Gruber 1993).

The problem is, How are these criteria made operational? We introduce the concept of a DEM's competence (Gruninger and Fox 1994). Given a properly instantiated model of an enterprise and an accompanying theorem prover (perhaps PROLOG or a deductive database), the *competence* of a DEM is the set of queries that it can answer. Ideally, the competency questions should be defined in a stratified manner, with higher-level questions requiring the solution of lower-level questions. Another view of competency is that it evaluates the expressiveness of the ontology that is required to represent the competency questions and characterize their solutions.[6]

Using the concept of competency, we can determine how well a particular DEM satisfies them, as follows:

The *functional completeness* of a DEM is determined by its competency, that is, the set of queries it can answer with a properly instantiated model. Given a particular function (application), its enterprise-modeling needs can be specified as a set of queries. If these queries can be reduced to the set of competency questions specified for the chosen DEM, then the DEM is sufficient to meet the modeling needs of the application.[7]

The generality of a DEM can be determined by evaluating whether the union of queries from a broad set of functions, perhaps drawn from different sectors, is reducible to a DEM's competency.

Given that a theorem prover is the deduction mechanism used to answer questions, the *efficiency* of a representation can be defined by the number of logical inferences per second required to answer a query. However, experience has demonstrated that there is more than one way to represent the same knowledge, and each representation does not have the same complexity when answering a specific class of questions. Furthermore, the deductive capability provided with the DEM affects the store versus compute trade-off. If the deduction mechanisms are taken advantage of, certain concepts can be computed on demand rather than stored explicitly. By computing the average complexity of the competency questions of the DEM, we can estimate its efficiency.

The *perspicuity* of a DEM is enhanced by its axiomatization. That is, by providing formal

definitions of objects and their relations and attributes, we make it possible for users to understand their intended meaning, although the definitions do not guarantee that programs that access a DEM will interpret the results correctly.

The *precision* of a DEM refers to what extent the definitions of concepts are *distinct* (Sowa 1995). Given that we have formal definitions, we can determine whether a concept subsumes another or what concepts lie at their intersection or union (generalization). *Granularity* refers to the capability of representing concepts at differing levels of abstraction. Whether a DEM is precise enough or allows for the capability of abstraction should be determined from the competency questions of the DEM. Conversely, an application specifies its precision and granularity requirements in the form of queries. If these queries can be reduced to the DEM's competency questions, then the DEM is sufficient.

Finally, the *minimality* of a DEM can be determined, using the axioms, by proving that for every object in the DEM, there is no other object that is logically equivalent.

The Toronto Virtual Enterprise Deductive Enterprise Model

In this section, we describe a portion of the TOVE DEM being developed at the University of Toronto. Our approach to engineering the TOVE DEM begins with defining its competency, which is in the form of questions that the DEM must be able to answer. The second step is to define its terminology: its objects, attributes, and relations. The third step is to specify the definitions and constraints on the terminology, where possible. The specifications are represented in first-order logic and implemented in PROLOG. Finally, we test the competency of the DEM by proving the competency questions with the PROLOG axioms.

Figure 1 shows the subsets of the TOVE DEM, that is, ontologies. In this section, we give an example of a small ontology for resource spoilage. We first describe a portion of TOVE's activity-state ontology and the semantic foundation on which it is built, followed by the competency of a resource spoilage ontology, its terminology, and axioms.

Activity-State-Time: Foundational Theory

At the heart of the TOVE activity ontology lies the representation of a primitive activity and its corresponding enabling and caused states (Fox, Chionglo, and Fadel 1993; Sathi, Fox, and Greenberg 1985). An *activity* is the basic transformational action primitive with which processes and operations with duration can be represented. *Enabling states* define the preconditions for the activity (what has to be true of the world for the activity to be performed) and cause *state-define effects* (what is true of the world once the activity has been completed).

An activity, along with its enabling and caused states, is called an *activity cluster* (figure 2). The state tree linked by an *enables relation* to an activity specifies what has to be true for the activity to be performed. The state tree linked to an activity by a causes relation defines what is true of the world once the activity has been completed.

TOVE primitive activities are resource based; that is, their preconditions and effects are parameterized by resource terms. We capture this relationship with the following relations: *uses*(*r*, *a*), *consumes*(*r*, *a*), *modifies*(*r*, *a*), and *produces*(*r, a*). Intuitively, a resource is used and released by an activity if none of the properties of a resource are changed when the activity is successfully terminated, and the resource is released. A resource is consumed or produced if some property of the resource is changed after termination of the activity, including the existence and quantity of the resource or some arbitrary property such as color. Thus, *consume*(*r, a*) signifies that a resource is to be used up by the activity and will not exist once the activity is completed, and *produce*(*r, a*) signifies that a resource, which did not exist prior to the performance of the activity, has been created by the activity.

Different classes of aggregate activity are defined using different constraints on the occurrence of the subactivities, including sequences of actions, conditional actions (which are triggered or performed only under certain conditions), and iterated actions.

Activity-State-Time: Ontology

Within TOVE, we have adopted the situation calculus in Reiter (1991) as the foundational theory to provide a semantics to the ontologies of activity, state, and time. In this role, the foundational theory provides the framework in which we can compare different intuitions and alternative formalizations of these intuitions. Also, the axioms in the foundational theory serve as the nucleus of any implementation of the deductive queries that are supported by the ontologies.

The situation calculus has predicates for actions, situations, time points, and fluents. The intuition behind the situation calculus is

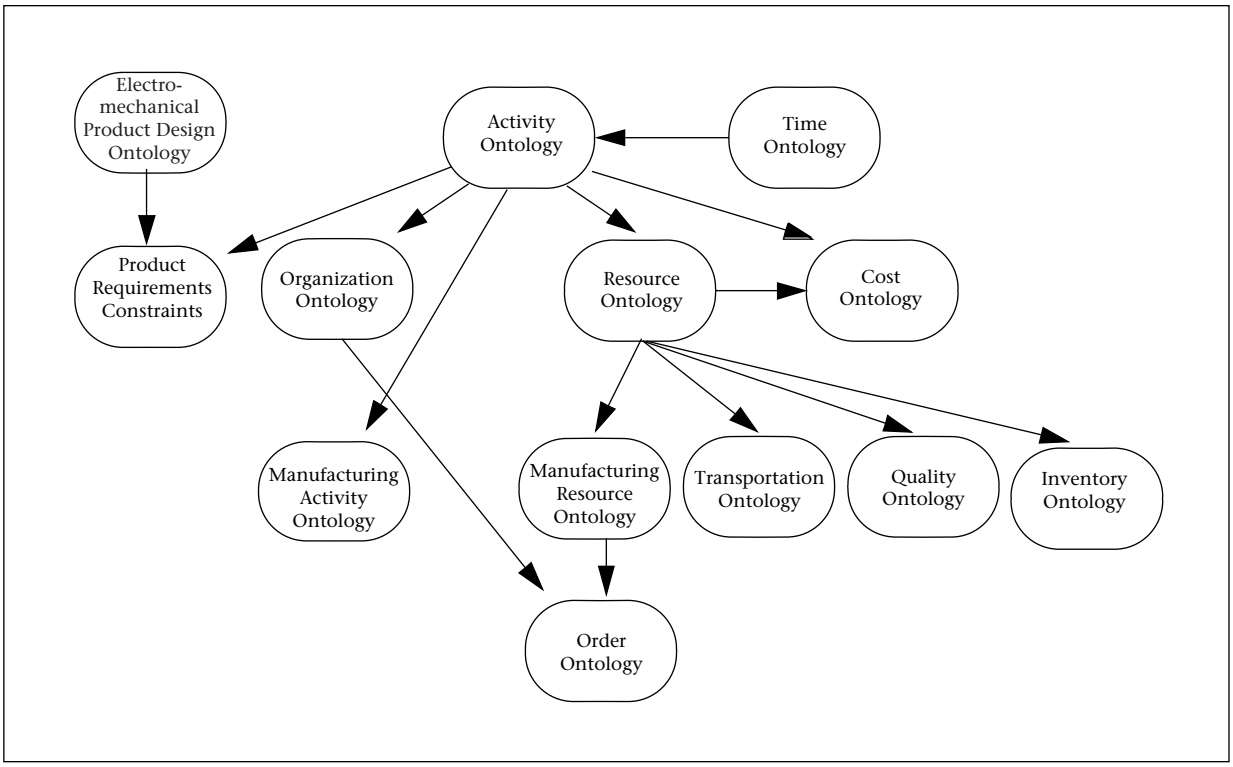


Figure 1. Toronto Virtual Enterprise Ontologies.

that there is an initial situation (denoted by the constant S_0) and that the world changes from one situation to another when actions are performed. There is a function *do*(*a, s*), which is the name of situation that results from performing action *a* in situation *s*, and there is a predicate *poss*(*a, s*) that is true whenever an action *a* can be performed in situation *s*. The structure of situations is that of a tree; two different sequences of actions lead to different situations. Thus, each branch that starts in the initial situation can be understood as a hypothetical future. The tree structure of the situation calculus shows all possible ways in which the events in the world can unfold. Therefore, any arbitrary sequence of actions identifies a branch in the tree of situations. For example, in figure 3, the actions A_1, A_2, A_3, and A_4 occur along one branch, and the actions A_2, A_4, and A_5 occur along another branch. In this way, the notion of hypothetical branches allows us to formalize the analysis of what-if scenarios.

A *fluent* is a relation or function whose value might change between situations. To define the evaluation of the truth value of a sentence in a situation, the predicate *holds*(*f, s*) is used to represent the fact that some fluent *f* is true in situation *s*. For example, to represent that the substate *consume wire* is true in a situation *s*, we would write

holds(*consumes*(*wire, fabricate_plug_on_wire*), *s*) . (2)

One important property that must be formalized in the activity ontology is the notion of *causality,* that is, the specification of what holds in the world after performing some action. As part of the logical specification of the activity ontology, we define successor state axioms that specify how actions change the value of a fluent. These axioms provide a complete characterization of the value of a fluent after performing any action. Thus, if we are given a set of action occurrences, we can determine the value of a fluent at any time point (that is, temporal projection) by first finding the situation containing the time point and then using the successor state axioms to evaluate the fluent in this situation.

The foundational theory for TOVE also includes the extension of the situation calculus in Pinto and Reiter (1993) in which one branch of the situation tree is selected to describe the evolution of the world as it actually unfolds, and time points are associated with the start and end of each situation in a branch. The predicate *actual* specifies those situations that are in the actual branch, and the predicate *occurs*(*a, s*) is defined to represent actions performed along the actual branch, and *occursT*(*a, t*) represents that action *a* occurs

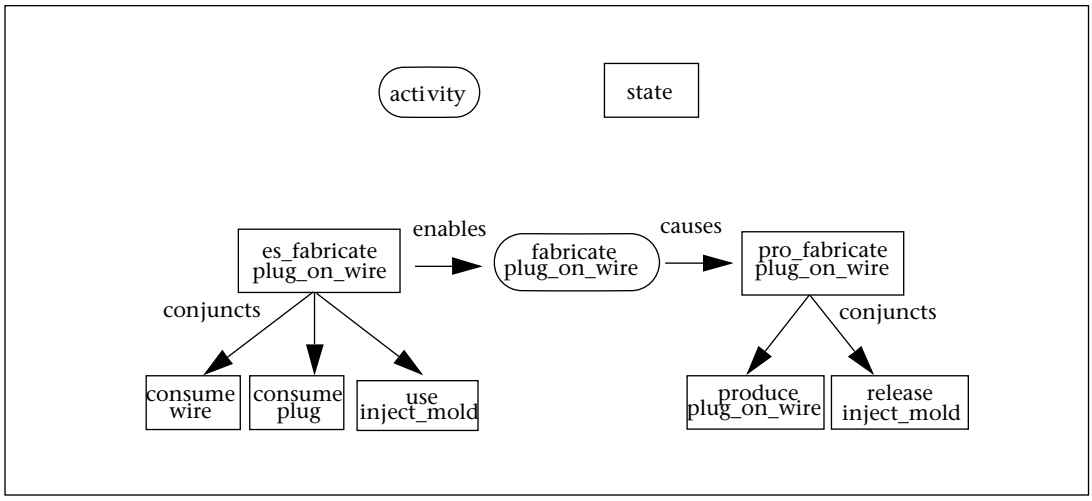


Figure 2. Activity Cluster.

at time t on the actual branch. For example, in figure 3, the actions A_1, A_2, A_3, and A_4 occur on the actual branch.

Example Ontology: Resource Spoilage

Consider an ontology that supports reasoning about the spoilage of perishable products. Given products with code age (that is, shelf life) and spoilage limits, we informally define the competency of the ontology to include, Will shipment-10 of oranges spoil if they are not shipped before Friday? Is any milk spoiled by Wednesday? If so, how much?

We introduce the following terminology: First is *spoiled*(*r*)—Resource *r* is spoiled. Second is *shelf_life*(*r,d*)—Resource *r* has *d* time units before it is spoiled. Third is *spoilage*(*r*)—An action creates spoilage in resource *r.*

Spoilage is intuitively characterized by the following properties: An event occurs at some (possibly fixed) time after production of a resource. The effects of this event change some fluents for the resource such that some action that required the resource and that was formerly possible is no longer possible. This event is preventable; that is, we can perform other actions whose effects falsify the preconditions for the spoilage event and, thus, prevent it from happening. The effects of the event are not repairable; that is, there are no actions that can reachieve the fluents that are established by the spoilage event. These intuitions informally capture the design decisions for representing spoilage; the axioms in the DEM must capture all these intuitions.

Using the notion of an actual line, we can reason about hypothetical branches where we allow constraints to be violated but enforce these constraints on the actual line, so that branches that violate the constraints cannot be actual. For example, suppose we want to represent the constraint that no spoiled food is allowed. We cannot represent this constraint as a state constraint, which must be satisfied in all situations, for example,

$$(\forall\, r, s)\ \neg holds(spoiled(r), s)\ , \qquad (3)$$

because situations can exist where spoilage occurs; however, we do not want spoilage to occur. Using the notion of actual branch, we can represent this constraint as

$$(\forall\, r, s)\ actual(s) \supset \neg holds(spoiled(r), s)\ , \qquad (4)$$

which allows spoilage on hypothetical branches but does not allow spoilage on actual branches. This example uses the foundational theory of an ontology (in this case, situation calculus) to express alternative formalizations and then decide which one is consistent with our intuitions.

We can represent the occurrence of spoilage as

$$occurs_T(terminate(a), t) \wedge produces(a, r) \wedge shelf\_life(r, d) \supset occursT(spoilage(r), t + d) \qquad (5)$$

along with the axiom for the fluent *spoiled*, which states that *spoiled*(*r*) is only achieved by the action *spoilage*(*r*), and there is no action that can falsify *spoiled*(*r*) once it has been achieved:

$$(\forall a, r, s)\ holds(spoiled(r), do(a, s)) \equiv (\neg holds(spoiled(r), s) \wedge a = spoilage(r)) \vee holds(spoiled(r), s)\ . \qquad (6)$$

In addition, we have the following axiom for the preconditions of all activities that consume some resource *r*, which states that the resource cannot be spoiled if it is consumed by the activity:

$$(\forall a, r, s)\ Poss(a(r), s) \supset \neg holds(spoiled(r), s)\ . \qquad (7)$$

If we were to use the occurrence axiom in equa-

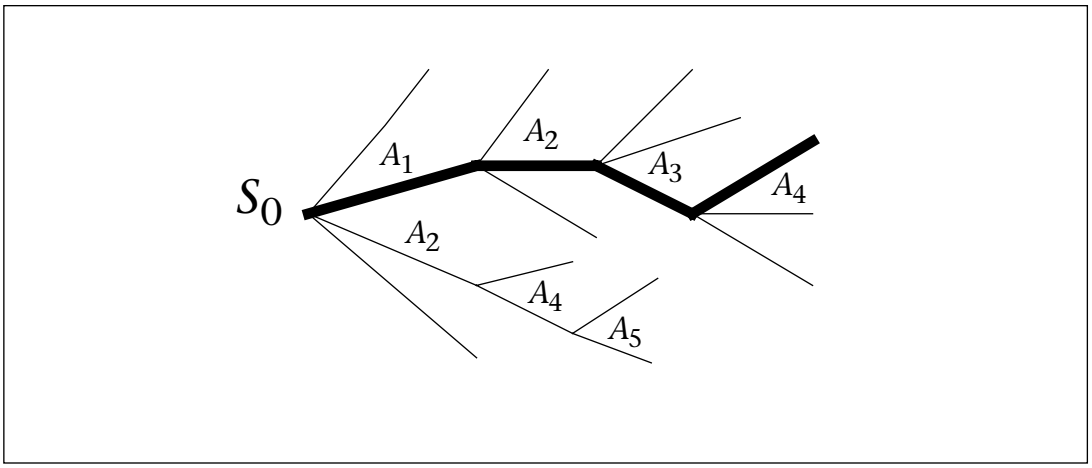


Figure 3. Situation Trees.

tion 7, the action *spoilage*(*r*)would not be preventable. To axiomatize preventable occurrences, we need to use the following sentence:

$$occursT(terminate(a), t) \wedge produces(a, r) \wedge shelf\_life(r, d) \wedge PossT(spoilage(r), t + D) \supset occursT(spoilage(r), t + d) \quad . \quad (8)$$

We would then include precondition axioms for *spoilage*(*r*), such as

$$Poss(spoilage(r), s) \equiv holds(rp(r, q), s) \wedge q > 0 \wedge \neg holds(frozen(r), s) \quad , \quad (9)$$

so that spoilage is possible iff there is a nonzero amount of resource that exists, and it is not frozen. In this case, the only way to prevent spoilage is to either consume the resource or freeze it.

Example Queries

Using our foundational theory (that is, the situation calculus), our activity-time ontology, and our theory of spoilage, we can answer the following competency questions for spoilage: The queries would be typed using a natural language interface (NLI). The equations are the translated form of the queries that the NLI would output to the DEM.

Given some initial state, which resources are perishable?

$$(\exists r, s)\ S_0 < s \wedge poss(spoilage(r), s) \quad . \quad (10)$$

Given a schedule, will any resources spoil? That is, does a situation exist during the performance of A in which spoilage occurs for some resource?

$$(\forall s_1, s_2)\ Do(A, s_1, s_2) \supset (\exists r, s)\ s_1 < s < s_2 \wedge occurs(spoilage(r), s) \quad . \quad (11)$$

Given some scenario of events, when will a particular resource spoil?

$$(\forall s_1, s_2)\ Do(A, s_1, s_2) \supset (\exists s, t)\ s_1 < s < s_2 \wedge occurs(spoilage(R), s) \wedge start(s, t) \quad . \quad (12)$$

Is it possible to prevent the spoilage of some resource? That is, if we are given a scenario in which spoilage might occur, does an alternative scenario exist in which spoilage does not occur?

$$((\forall s_1, s_2)\ Do(A, s_1, s_2) \supset (\exists s)\ s_1 < s < s_2 \wedge occurs(spoilage(R), s)) \supset ((\exists a)\ subaction(a, A) \wedge ((\forall s_1, s_2)\ Do(A, s_1, s_2) \supset \neg(\exists s)\ s_1 < do(spoilage(R), s) < s_2) \quad . \quad (13)$$

Conclusions

The drive for more agile enterprises requires a degree of integration that is not possible without the use of a sophisticated information infrastructure. At the core of this infrastructure lies an enterprise model. Efforts are under way to create GEMs, whose use significantly reduces the time to design and implement enterprise models. More recently, DEMs have been developed that play a more active role in the support of enterprise operations by deducing answers to commonly asked questions. DEMs have the potential for substantially reducing the MIS backlog.

Acknowledgments

This research is supported in part by the Natural Science and Engineering Research Council of Canada, Carnegie Group Inc., BHP Ltd., Digital Equipment Corp., IBM, Information Technology Research Corp., Manufacturing Research Corporation of Ontario, Micro Electronics and Computer Research Center, MITEL Corp., Numetrix Ltd., Quintus Corp., Spar Aerospace, and Toyo Engineering.

Notes

1. In fact, the importance of enterprise modeling was recognized by industry's decision to create a parallel

organization to the Object Management Group to focus on the definition of standard business objects.

2. See Spurr et al. (1994) for a view of reengineering tools.

3. Given a relation *works-for*(*Supervisor, Supervisee*), then the SQL query would be (SELECT Supervisor FROM works-for WHERE Supervisee = Joe).

4. A relational database cannot support commonsense queries directly. Instead, the commonsense query would have to be specified as a series of factual queries containing one or more JOINs combined with SELECTs, which is equivalent to performing deduction. The lack of a commonsense deductive capability forces users to spend significant resources on programming each new report or function that is required.

5. Because a generic enterprise model (GEM) is a subset of a deductive enterprise model (DEM), it does not contain the axioms; we only refer to DEMs from here on.

6. These competency questions do not generate ontological commitments; rather, they are used to evaluate the ontological commitments that have been made.

7. By *reducible*, we mean the questions can be rewritten using the objects provided by the chosen deductive enterprise model.

References

AMICE. 1993. *CIM-OSA Open System Architecture for CIM*. 2d rev. ed. Berlin: Springer-Verlag.

Atefi, K. 1997. Formalization of Business-Process Reengineering Heuristics. M.A.Sc. thesis, Mechanical and Industrial Engineering Department, University of Toronto.

Bernus, P.; Nemes, L.; and Williams, T. J. 1996. *Architectures for Enterprise Integration*. London: Chapman and Hall.

Campbell, A. E., and Schapiro, S. C. 1995. Ontologic Mediation: An Overview. In *Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*. Menlo Park Calif.: AAAI Press.

Davenport, T. H. 1993. *Process Innovation: Reengineering Work through Information Technology*. Cambridge, Mass.: Harvard Business School Press.

DoD. 1993. The DoD Enterprise Model, Office of the Secretary of Defense, Washington D.C.

Fadel, F. 1994. Resource Ontology for Enterprise Modeling. M.A.Sc. thesis, Department of Industrial Engineering, University of Toronto.

Fadel, F.; Fox, M. S.; and Gruninger, M. 1994. A Resource Ontology for Enterprise Modeling. Paper presented at the Third Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises 17–19 April, Morgantown, West Virginia.

Fillion, F.; Menzel, C.; Blinn, T.; and Mayer, R. 1995. An Ontology-Based Environment for Enterprise Model Integration. Paper presented at the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, 19–20 August, Montreal, Quebec, Canada.

Fox, M. S. 1990. Artificial Intelligence and Expert Systems: Myths, Legends, and Facts. *IEEE Expert* 5(1): 8–22.

Fox, M.; Chionglo, J. F.; and Fadel, F. G. 1993. A Commonsense Model of the Enterprise. In Proceedings of the Second Industrial Engineering Research Conference, 425-429. Norcross Ga.: Institute for Industrial Engineers.

Fox, M. S.; Barbuceanu, M.; Gruninger, M. 1995. An Organization Ontology for Enterprise Modeling: Preliminary Concepts for Linking Structure and Behavior. *Computers in Industry* 29:123–134.

Gruber, T. R. 1993. Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Technical report, KSL-93-04, Knowledge Systems Laboratory, Stanford University.

Grüninger, M., and Fox, M. S. 1995. Methodology for the Design and Evaluation of Ontologies. In *Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*. Menlo Park Calif.: AAAI Press.

Grüninger, M., and Fox, M. S. 1994. The Role of Competency Questions in Enterprise Engineering. In *Benchmarking—Theory and Practice,* ed. A. Rolstadas, 83–95. London: Chapman and Hall.

Gruninger, M., and Pinto, J. A. 1995. A Theory of Complex Actions for Enterprise Modeling. Paper presented at the AAAI Spring Symposium Series 1995—Extending Theories of Action: Formal Theory and Practical Applications, 27–29 March, Stanford, California.

Hammer, M., and Champy J. 1993. *Reengineering the Corporation*. New York: Harper Business.

Hansen, W. C. 1991. The Integrated Enterprise. In Foundations of World-Class Manufacturing Systems: Symposium Papers, National Academy of Engineering, Washington, D.C.

Kim, H., and Fox, M. S. 1995. An Ontology of Quality for Enterprise Modeling. Paper presented at the Fourth Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises, 17–19 April, Morgantown, West Virginia.

Kim, H., and Fox, M. S. 1994. Formal Models of Quality and ISO9000 Compliance: An Information Systems Approach. In Proceedings of the Forty-Eighth Annual Quality Congress, 17–23. Milwaukee Wisc.: American Society for Quality Control.

Lee, J.; Grunninger, M.; Jin, Y.; Malone, T.; Tate, A.; Yost, G., and Other Members of the PIF Working Group. 1996. The PIF Process-Interchange Format and Framework, Version 1.1. Paper presented at the Workshop on Ontological Engineering, ECAI '96, 11–16 August, Budapest, Hungary.

Lenat, D., and Guha, R. V. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Reading, Mass.: Addison Wesley.

Lin, J.; Fox, M. S.; and Bilgic, T. 1996. A Requirements Ontology for Concurrent Engineering. *Concurrent Engineering: Research and Applications* 4(4): 279–291.

Martin, C., and Smith, S. 1983. Integrated Computer-Aided Manufacturing (ICAM) Architecture, Part III/Volume IV: Composite Information Model of "Design Product" (DES1), Technical report AFWAL-TR-82-4063 Volume IV, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base,

Ohio.

Martin, C.; Nowlin, A.; St. John, W.; Smith, S.; Ruegsegger, T.; and Small, A. 1983. Integrated Computer-Aided Manufacturing (ICAM) Architecture, Part III/Volume VI: Composite Information Model of "Manufacture Product" (MFG1), Technical Report AFWAL-TR-82-4063 Volume VI, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio.

Nagel, R. N. 1993. Understanding Agile Competition. *Computer-Assisted Logistics Systems,* Winter 1993, pp. 75–85.

Nagel, R. N., and Dove, R. 1991. Twenty-First Century Manufacturing Enterprise Strategy: An Industry-Led View, Technical report, Iacocca Institute, Lehigh Univ.

Pinto, J., and Reiter, R. 1993. Temporal Reasoning in Logic Programming: A Case for the Situation Calculus. In Proceedings of the Tenth International Conference on Logic Programming, 21–28 June, Budapest, Hungary.

Reiter, R. 1991. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy.* San Diego, Calif.: Academic.

Sathi, A.; Fox, M. S.; and Greenberg, M. 1985. Representation of Activity Knowledge for Project Management. *IEEE Transactions on Pattern Analysis and Machine Intelligence 7:531–552.*

Scheer, A-W. 1989. Enterprise-Wide Data Modeling: Information Systems in Industry. New York: Springer-Verlag.

Schlenoff, C.; Knutilla, A.; Ray, S. 1996. Unified Process Specification Language: Requirements for Modeling Process, Interagency Report 5910, National Institute of Standards and Technology, Gaithersburg, Maryland.

Smith, S.; Ruegsegger, T.; and St. John, W. 1983. Integrated Computer-Aided Manufacturing (ICAM) Architecture—Part III/Volume V: Composite Function Model of "Manufacture Product" (MFG0), Technical Report AFWAL-TR-82-4063 Volume V, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio.

Sowa, J. F. 1995. Distinctions, Combinations, and Constraints. In *Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*. Menlo Park, Calif.: AAAI Press.

Spurr, K.; Layzell, P.; Jennison, L.; and Richards, N. 1994. *Software Assistance for Business Reengineering*. London: Wiley.

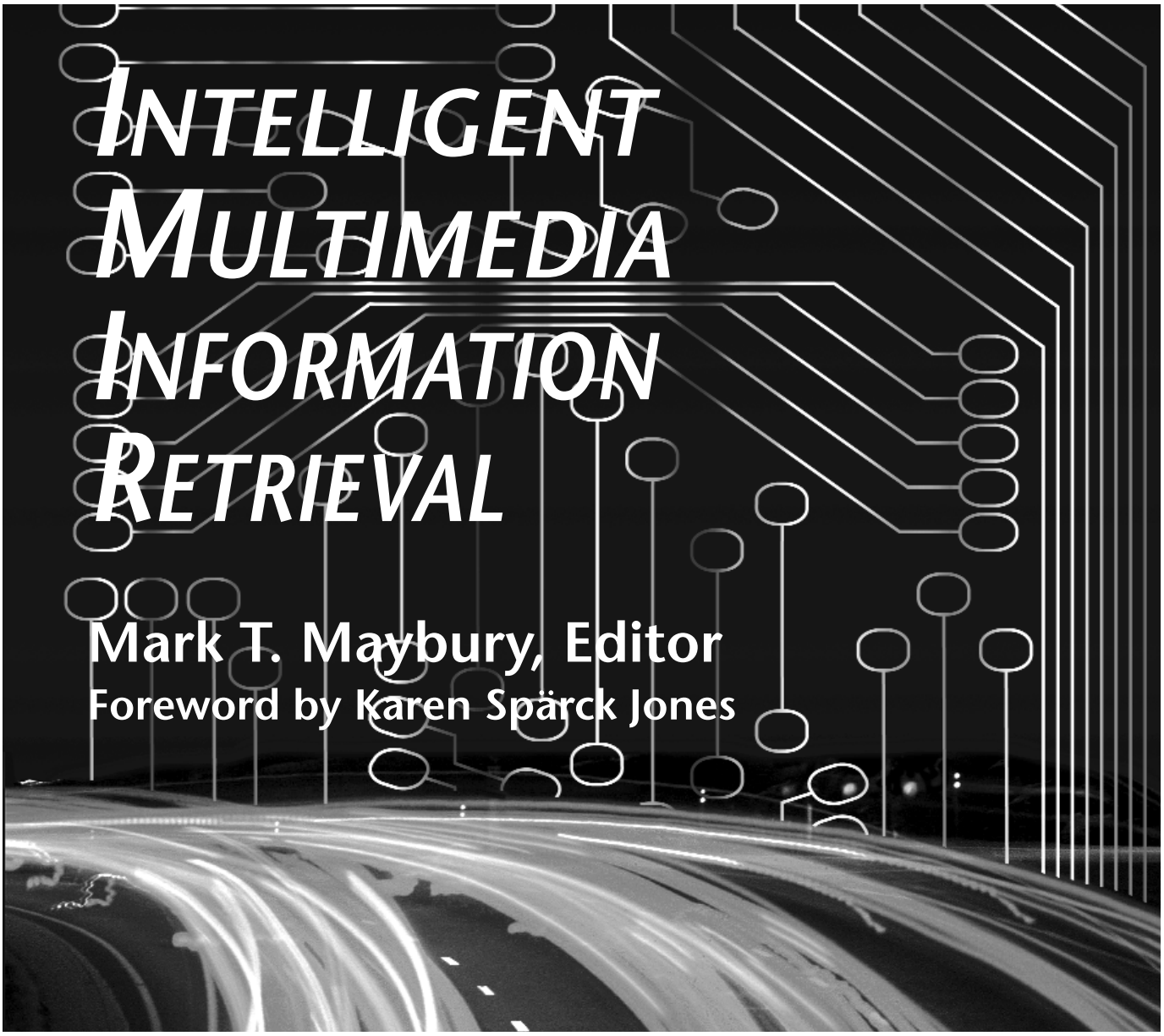
Tham, D.; Fox, M. S.; and Gruninger, M. 1994. A Cost Ontology for Enterprise Modeling. Paper presented at the Third Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises, 17–19 April, Morgantown, West Virginia.

Uschold, M.; King, M.; Moralee, S.; and Zorgios, Y. 1997. The Enterprise Ontology. *Knowledge Engineering Review* 13:71–88.

Williams, T. J., and the Members of the Industry–Purdue University Consortium for CIM. 1991. The PURDUE Enterprise Reference Architecture, Technical report, 154, Purdue Laboratory for Applied Industrial Control, Purdue University.

Mark Fox is a professor of industrial engineering with cross-appointments in the Department of Computer Science and the Faculty of Management Science at the University of Toronto. Fox is head of the Enterprise Integration Laboratory and director of the graduate program in integrated manufacturing. He is holder of the NSERC Industrial Research Chair in Enterprise Integration. He received his B.Sc. in computer science from the University of Toronto in 1975 and his Ph.D. in computer science from Carnegie Mellon University in 1983. Over the last decade, his research has pioneered the application of AI to factory planning and scheduling problems, project management, and material design. His research in constraint-directed reasoning as applied to job-shop scheduling explored the problem of reasoning in the presence of constraints, determining their applicability, and selectively relaxing those that cannot be satisfied. His research resulted in the creation of the new field of knowledge-based scheduling. In 1991, Fox was elected a fellow of the American Association for Artificial Intelligence (AAAI), was a AAAI councilor, and is a cofounder of the AAAI Special Interest Group in Manufacturing. He was also elected a joint fellow of the Canadian Institute for Advance Research and PRECARN. Fox's current research focuses on theories of enterprise engineering, constrained-directed reasoning, a unified theory of scheduling, enterprise modeling, and coordination theory.

Michael Gruninger has been a research scientist in the Enterprise Integration Laboratory at the University of Toronto since 1993 and is project manager of the Enterprise Engineering Project. Gruninger received his B.Sc. in computer science from the University of Alberta in 1987 and his M.Sc. from the University of Toronto in 1989. His doctoral work at the University of Toronto has been in the area of logic and object recognition in computer vision, constructing ontologies to support two-dimensional object recognition in scenes with occlusion. Gruninger currently supervises the development of the Toronto virtual enterprise within the Enterprise Integration Laboratory. His work focuses on the design and evaluation of ontologies for reasoning about processes and time-based competition.



Intelligent multimedia information retrieval lies at the intersection of artificial intelligence, information retrieval, human-computer interaction, and multimedia computing. Its systems enable users to create, process, summarize, present, interact with, and organize information within and across different media such as text, speech, graphics, imagery, and video. These systems go beyond traditional hypermedia and hypertext environments to analyze and generate media, and support intelligent interaction with or via multiple media.

The chapters in this volume span a broad range of topics. The book is organized into seven sections: Content-Based Retrieval of Imagery, Content-Based Retrieval of Graphics and Audio, Content-Based Retrieval of Video, Speech and Language Processing for Video Retrieval, Architectures and Tools, Intelligent Hypermedia Retrieval, and Empirical Evaluations.

Published by the AAAI Press / The MIT Press
500 pp., $40.00 ISBN 0-262-63179-2 *Prices higher outside the U.S. and subject to change without notice.*

To order call 800-356-0343 (US and Canada) or (617) 625-8569.
Distributed by The MIT Press, 5 Cambridge Center, Cambridge, MA 02142