Integrated Ontologies for Enterprise Modelling

Table of Contents

Michael Gruninger
Department of Industrial Engineering, University of Toronto,
4 Taddle Creek Road, Toronto, Ontario M5S 1A4 CANADA
tel:+1-416-978-6347 *fax:*+1-416-971-2479
*email:*gruninger@ie.utoronto.ca *http:*//www.ie.utoronto.ca/EIL/eil.html

1.0 Defining an Enterprise

An enterprise model is a computational representation of the structure, activities, processes, information, people, behaviour, 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 evaluation.

An integrated enterprise model provides the language used to specify an explicit definition of an enterprise. For reengineering, we need to explore alternative models in the design of enterprises spanning organisation structure and behaviour. In order to reason about alternative designs for enterprises, we need to reason about different possible sets of constraints for enterprises within the language. We need to ask the 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, if we relax one of the policies, how will this affect the quality of products or services provided by the enterprise? If we purchase a new kind of machine, how will this affect the activities that are performed? Will we need to retrain people in the enterprise to give them the skills to use the machine? If we change the activities that are performed, how will this change resource consumption?

A related problem is the use of benchmarking in the reengineering process. In benchmarking, we are

comparing performance between enterprise and then adopting processes and practices from enterprises which are the best performers. However, not all practices can be adopted from other enterprises; the key is to realize that we must identify opportunities for improvement by analyzing the successes and failures of similar enterprises. Herein lies the problem -- what is a similar enterprise? What is compared among enterprises when we use benchmarking? We cannot compare the goals and activities among enterprises unless all constraints and assumptions about the enterprise and its environment are made explicit.

Any enterprise model must have the expressiveness to capture the above sets of constraints. In this paper, we introduce the notion of ontologies and their use in providing rigorous foundations for enterprise modelling.

2.0 Axiomatization of Ontologies

We are taking what can be viewed as a "second generation knowledge engineering" approach to constructing our Common Sense Enterprise Model. Rather than extracting rules from experts, we are "engineering ontologies." An ontology is a formal description of entities and their properties, relationships, constraints, behaviours. In this section, we explore the notion of ontologies, beginning with the role of language and moving through to the semantics and axiomatization of ontologies.

A language is a set of symbols (lexicon) and a specification of how these symbols can be combined to make well-formed formulae (grammar/syntax). The lexicon consists of logical symbols (such as connectives, variables, and quantifiers) and nonlogical symbols. For a process ontology, the nonlogical part of the lexicon consists of expressions (constants, functions, predicates) that refer to everything needed to describe processes. For example, KIF is a language with a proof theory and model theory.

The primary component of the ontology is its terminology for classes of processes and relations for processes and resources, along with definitions of these classes and relations. Such a lexicon of terminology along with some specification of the meaning of terms in the lexicon constitutes the ontology. The model theory of the ontology will provide a rigorous mathematical characterization of the terminology of ontology. The proof theory of ontology provides axioms for the interpretation of terms in the ontology.

The focus of the ontology is not only on the terms themselves, but also on their definitions. We can include an infinite set of terms in our ontology, but they can only be shared if we agree on their definitions. It is the definitions which are being shared, not simply the terms themselves.

The problem is that the meaning of the terminology for many ontologies are in people's heads; we need some framework for making it explicit. Any ideas that are implicit are a possible source of ambiguity and confusion. For a process ontology, the model theory provides a rigorous mathematical characterization of process information and the axioms give precise expression to the basic logical properties of that information in the language.

It is useful to distinguish two types of sentences in this set of axioms:foundational theories and definitions. A foundational theory is a set of distinguished predicates and functions together with some axiomatization. Distinguished predicates are those for which there are no definitions; the intended interpretations of these predicates is defined using the axioms in the foundational theories. All terminology in a generic ontology is defined using classes of sentences in the foundational theories; any terminology that does not have a definition is axiomatized in some foundational theory. So when we speak about a semantics for an ontology, we are referring to the axiomatization of foundational theories and definitions for the ontology's terminology.

Different interpretations for the terminology can be given, but one of these will be the intended interpretation that guides the development of the axioms. The axiomatization allows a characterization of these interpretations. We can reason about the meaning of the terminology of the ontologies using the models of the

axioms in the foundational theories. For example, the situation calculus can be used to provide the set of axioms for a suite of ontologies that support reasoning about actions and change. In such a case, the model theory for those ontologies would be based on the model theory of situation calculus.

A model-theoretic approach also assists in the development of axioms for terminology. One approach is to define intended interpretations using the predicates in the ontology. By determining which of these interpretations are consistent, we are imposing requirements on any axioms defining these predicates. Constructs in an ontology can be evaluated using counterexamples -- to distinguish among different axiomatizations, find falsifying interpretations for each. In this way, we can use model-theoretic arguments to express motivating scenarios, which in turn can be used to intuitively justify axioms and classes of constraints in the ontology.

3.0 TOVE Ontologies

Our approach to engineering ontologies begins with defining an ontology's requirements; this is in the form of questions that an ontology must be able to answer. We call this the *competency* of the ontology. For any task in which the ontology is to be employed, the task imposes a set of requirements on the ontology. These requirements can best be specified as a set of queries that the ontology should be able to answer, if it contains the relevant information. The competency questions are the basis for a rigorous characterization of the information that the ontology is able to provide to the task ([Gruninger and Fox 94]). Competency questions are used to evaluate an ontology in the sense that the ontology must be necessary and sufficient to represent the tasks specified by the competency questions and their solution. These are also the tasks for which the ontology finds all and only the correct solutions. Tasks such as these can serve to drive the development of new ontologies and also to justify and characterize the capabilities of existing ontologies.

The second step in the design of an ontology 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. Lastly, we test the competency of the ontology by "proving" the competency questions with the Prolog axioms.

Figure 1 shows the set of TOVE ontologies. In this section, we give a brief overview of these ontologies, after which we will describe their application to the modelling of enterprises.

3.1 Activity and State Ontology

Within TOVE, we have adopted the situation calculus as the foundational theory to provide a semantics to the ontology of activity, state, and time. The situation calculus is a sorted language with the following sorts:

- actions
- situations
- time points
- fluents

FIGURE 1. TOVE Ontologies

Recall that the axioms of the foundational theories are used to characterize the models of the axioms for the ontology. The intuition behind the situation calculus is 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. 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. Further, the structure over situations is a branch is isomorphic to the natural numbers -- the function *do(a,*s*)* is the name of situation that results from performing action *a* in situation s. There is a predicate *poss(a,*s*)* that is true whenever an action a can be performed in a situation s. The axioms for the situation calculus used as the foundational theory for TOVE can be found in [Pinto and Reiter 93].

FIGURE 2. Situation trees

A fluent is a relation or function whose value may 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.

One important property that must be represented 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 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, so that we can use the solution to the frame problem in [Reiter 91]. Thus if we are given a set of action occurrences, we can solve the temporal projection problem (determining the value of a fluent at any time point) by first finding the situation containing that time point, and then using the successor axioms to evaluate the fluent in that situation.

The foundational theory for TOVE also includes the extension of the situation calculus in [Pinto & Reiter 93] in which one branch of the situation tree is selected to describe the evolution of the world as it actually unfold and time points are associated with the start and end of each situation in a branch. The predicate *actual* specifies those situations which are in the actual branch, and the predicate *occurs(a,*s*)* is defined to represent actions performed along the actual branch. For example, in Figure 3, the actions *A1,A2,A3,* and *A4* occur on the actual branch.

The notion of the actual line and action occurrences plays a crucial role in the representation of enterprises. We need to express the following class of constraints: suppose that a plan exists that violates some constraint, but we do not want to allow plans that violate the constraint. How can we distinguish between this constraint and those that must always be satisfied in order for a plan to exist? Using the notion of actual line, we can reason about hypothetical branches where we allow such 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 as a state constraint which must be satisfied in all situations e.g.

(EQ 1) (forall (?r ?s) (holds (spoiled ?r) ?s)
since there can exist situations where spoilage occurs; however, we do not want spoilage to occur. Using the notion of actual line, we can represent this as

(EQ 2) *(forall (?r ?s) (=> (actual ?s) (holds (spoiled ?r) ?s)))*
We can thus represent maintenance and prevention as actual line constraints with universal quantifiers, and represent goals and deadlines as actual line constraints with existential quantifiers. For example, we can represent the goal of producing 50 bolts as the following actual line constraint:

(EQ 3) *(exists (?r ?q ?s) (and (actual* ?s*) (holds (rp bolt 50)* ?s*)*))

This allows us to reason about hypothetical non-actual situations where the goal is not achieved by the deadline. This also allows us to formally characterize the conditions which must hold in the world and actions that must necessarily occur in order to achieve a goal.

The foundational theory for TOVE also extends the situation calculus to represent complex actions defined by the aggregation of subactions. The predicate *Do(a,s1,s2)* represents the fact that if action *a* is performed in situation *s1*, then *s2* is one of the possible situations reached. A complex action theory is a set of sentences containing *Do* literals. A complex action is defined by a complex action theory which specifies its subactions and constraints on the occurrence of these subactions.

Different classes of actions are defined using different complex action theories. These classes of activities constitute the object libraries of the TOVE activity ontology. At the heart of the TOVE activity ontology lies the representation of a primitive *activity* and its corresponding enabling and caused *states* ([Sathi et al. 85], [Fox et al 93]). 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 in order for the activity to be performed), and caused state define effects (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 parametrized by resource terms. We capture this relationship with
the following relations:*uses(r,a)*, *consumes(r,a)*, *modifies(r,a), 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; this includes 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, that did not exist prior to the performance of the activity, has been created by the activity.

3.2 Organisation Ontology

We consider an organisation to be a set of constraints on the activities performed by agents [Fox et al. 96]. The TOVE organisation ontology focuses on organisation structure, roles, authority and empowerment. In linking the structure of an organisation with the behaviour of agents within the organisation, we must define how the organisation ontology is integrated with the activity ontology.

A concept found in almost all of the literature is that of an agent. An agent performs activities in order to achieve one or more goals. An agent can be a human being, a computer program, or a group of people and/or programs.

Individual-Agent and Group-Agent are subclasses of Organisation-Agent. They represent either individuals, like employees and contractors, or groups like departments, divisions, boards of directors, etc..

Organisation-Agents play various Organisation-Roles, they have Goals to achieve, fill Oganisation-Positions, and communicate with other OAs using Information-links. An Organisation-Role defines a prototypical function of an agent in an organisation. An Organisation-Position defines a formal position that can be filled by an Organisation-Agent in the organisation.

We use *authority* to refer to the control relationship that exists between two organisational agents. For OA*1* to have *authority* over OA*2* implies that OA*1* is able to extract a commitment from OA*2* to achieve a goal that is defined as part of OA*2*'s organisation-roles. In order to extract that commitment, OA*1* has to be related directly or indirectly by an authority link.

3.3 Product Ontology

The primary motivation for the TOVE product ontology has been to support collaborative design. It must therefore be able to represent an evolving and incomplete design for a product, as well as represent the requirements that the product must satisfy. It must capture the design rationale for various features and parameters of the product.

The product ontology is integrated with the other ontologies through the following problem: Given a design for a product, how do we manufacture it? That is, what activities are required to manufacture a product with the properties specified in the design, and what resources and organisational constraints are required to support these activities?

3.4 Service Ontology

Rather than manufacture and distribute products, many enterprises provide services to their customers, and we must also be able to model this enterprises within our framework. This requires us to have an ontology for services, where a service is intuitively an activity performed by agents within the enterprise to either change the properties of some resource (e.g. delivering packages, painting buildings, dry cleaning, repair) or to provide information to customers.

3.5 Resource Ontology

All activities require that some objects be available at the time that the activity is performed; this is one of the primary motivations for a theory of resources [Fadel et al. 94]. In order to support this, the various properties that are axiomatized in the ontology include the distinction between reusable vs. consumable resources, resource commitment, and the availability of resources.

In addition, an ontology of resources is required to support traceability. Given an instance of a product with certain properties, what instances of resources were used or consumed in order to produce the product? What activities were performed to either produce the product or achieve the preconditions (enabling states) of the activities related to the production of the product? These questions form the basis for any capability to perform causal diagnosis within an enterprise model.

4.0 Defining an Enterprise

In this section, we will use the ontologies presented in the previous section to specify the constraints required to define an enterprise.

An enterprise *E* is defined by the following set of constraints represented as sentences in first-order logic:

Definitions of the activities performed by the enterprise:

This consists of the following set of constraints:

- set of successor state axioms for the primitive actions in the enterprise.
- set of precondition axioms for the primitive actions in the enterprise.
- the set of axioms defining the aggregation of actions into complex actions within the enterprise.
- the set of successor state axioms for the knowledge-producing activities in the enterprise.
- the set of knowledge precondition axioms for actions within the enterprise. These define what agents need to know in order to perform their activities, including abilities.

- axioms defining the complete set of activities that can possibly be performed within the enterprise. These activities correspond to the process plans of the enterprise, and the auxiliary activities that support them, such as setup and cleanup activities.

Resource Constraints for the enterprise:

This consists of sentences specifying state and actual line constraints on resource fluents for resources required by activities performed in the enterprise.

- An actual line constraint defining the complete set of resources that can possibly be used or consumed by activities in the enterprise.
- Properties of the resources related to the concurrency constraints by defining the capacities of the resources in the enterprise. These may be actual line or state constraints.
- Universal actual line constraints on the state of resources in the enterprise. This includes constraints such as preventing spoilage and maintaining safety stock levels.

Organization Constraints for the enterprise:

This consists of the set of sentences defining the constraints among organizational roles, positions, and agents within the enterprise. This includes constraints on the behaviour of agents.

Enterprise goals:

This defines the goals to be achieved by the enterprise. This is a set of existential actual line constraints.

Product constraints for the enterprise:

This is the set of sentences defining constraints on product fluents. This includes universal actual line and occurrence constraints for product design requirements, quality constraints, and product standards.

Service constraints for the enterprise:

This is the set of occurrence constraints on product and knowledge fluents that define the activities the enterprise performs as services.

Occurrence constraints on activities in the enterprise:

This includes policies and performance constraints, such as the following examples:

- All deliveries must be made within 15 minutes of placing the order.
- When an order is made, a copy is sent to the regional office.

External constraints for the enterprise:

This is the set of constraints defining the external environment of the enterprise, dealing with customers, markets, suppliers, and competitors. It also includes the definitions of the activities performed by agents external to the enterprise (e.g. suppliers, subcontractors), but whose effects are required by the activities within the enterprise.

Using this framework, we can characterize classes of enterprises by sets of assumptions over their processes, goals, and organization constraints.

5.0 Conclusion

In order to provide a rigorous foundation for enterprise modelling, we have proposed a set of integrated ontologies for representing enterprises spanning activities, states, time, organisation, resources, and products. Using these ontologies, we provide a characterization of the constraints that define enterprises and their role in reengineering. These constraints, in turn, can be used as requirements for the expressiveness of the ontologies.

6.0 References

[Fadel et al 94] Fadel, F., Fox, M.S., and Gruninger, M. A resource ontology for enterprise modelling. *Third Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises*,(West Virginia University 1994).

[Fox et al. 93] Fox, M.S., Chionglo, J., Fadel, F. A Common-Sense Model of the Enterprise, *Proceedings of the Industrial Engineering Research Conference 1993.*

[Fox et al. 96] Fox, M.S., Barbuceanu, M., Gruninger, M. An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour, *Computers in Industry 29:123-134*(1996).

[Gruninger & Fox 94] Gruninger, M., and Fox, M.S., The Role of Competency Questions in Enterprise Engineering *,Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice,* Trondheim, Norway. June 1994.

[Pinto & Reiter 93] Pinto, J. and Reiter, R. Temporal reasoning in logic programming: A case for the situation calculus. In *Proceedings of the Tenth International Conference on Logic Programming* (Budapest, June 1993).

[Reiter 91] Reiter, R. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*. Academic Press, San Diego, 1991.

[Sathi et al 85] Sathi, A., Fox, M.S., and Greenberg, M. Representation of activity knowledge for project management. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. PAMI-7:531-552, September, 1985.