Gruninger, M., and Fox, M.S., (1996), “The Logic of Enterprise Modelling”, *Modelling and Methodologies for Enterprise Integration*, P. Bernus & L. Nemes (Eds.), Cornwall, Great Britain: Chapman and Hall.

THE LOGIC OF ENTERPRISE MODELLING

M. Gruninger and M.S. Fox

Department of Industrial Engineering, University of Toronto

4 Taddle Creek Road, Toronto, Ontario M5S 3G9 CANADA

tel:+1-416-978-6823 fax:+1-416-971-2479 email:{msf, gruninger}@ie.utoronto.ca

1.0 INTRODUCTION

An Enterprise Model is a computational representation of the structure, processes, information, resources, goals and constraints of a business, government activity, other organisational system. It can be both definitional and descriptive - spanning what should be and what is. The role of an enterprise model is to achieve model-driven enterprise design, analysis and operation.

A number of issues exist concerning the design of Enterprise Models. *Reusability* is concerned with the large cost of building enterprise-wide data models. Is there such a thing as a generic, reusable enterprise model whose use will significantly reduce the cost of information system building? A second issue is the *Consistent Usage* of the model: Given the set of possible applications of the model, can the model's contents be precisely and rigorously defined so that its use is consistent across the, enterprise? A third issue is model *Accessibility*. Given the need for people and other agents to access information relevant to their role, can the model be defined so that it supports query processing, both surface and shallow[1]. Lastly, there is the *Selection* issue: How do I know which is the right Enterprise Model for my application?

A many enterprise models have been proposed over the last fifteen years, including those by:

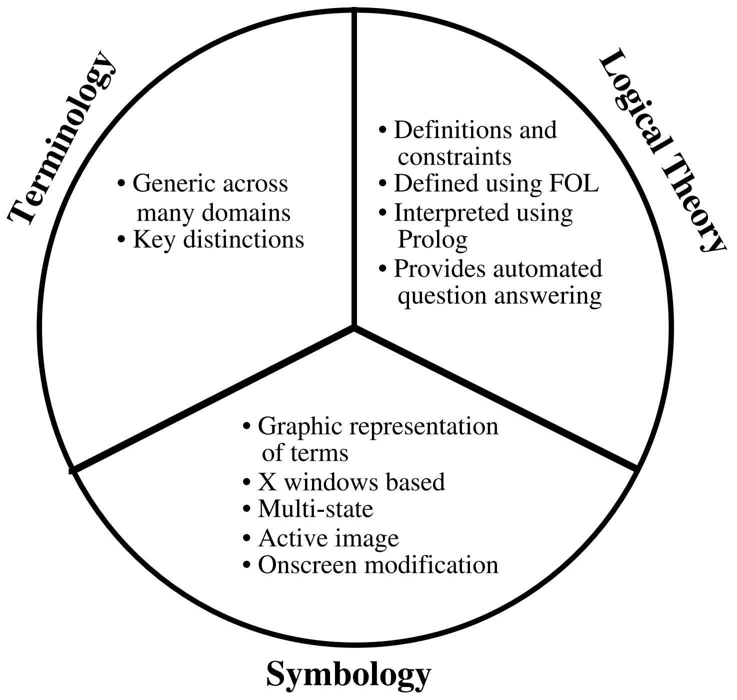
- **CAMI:** A US-based non-profit group of industrial organizations for creating manufacturing software and modelling standards.
- **CIM-OSA:** A reference model developed by the European Esprit project: AMICE [Jorysz & Vernadat 90a; 90b] [Klittich 90].
- **ICAM:** A project run by the Materials Lab. of the US Air Force [Davis et al. 83] [Martin et al. 83] [Martin & Smith 83].
- **IWI:** A reference model developed at the Institut fur Wirtschaftsinformatk Universitat des Saarlandes, Germany [Scheer 89].

The primary focus of these models has been on reusability; they all provide a data dictionary of object classes which can be reused, mostly for manufacturing organisations, Consistency of usage depends upon the clarity of the written documentation. Accessibility is limited to surface access provided by typical database systems. Lastly, the appropriateness of the model, i.e., selection, is

1. By surface, level processing we mean the direct retrieval of data that is represented explicitly in the model. By shallow level processing we mean retrieval that requires a small number of deductions, i.e., 1-100, in order to answer the query.

left up to the user.

It is our belief that the issues of reusability, consistent usage, accessibility and selection can best be addressed by taking a more formal approach to enterprise modelling. By formal, we are not referring to analytical models as found in Operations Research, but to logical models as found in Computer Science. Towards this end, the T'OVE ontology [Fox et al. 93] has been developed. An ontology [Gruber 93] is a formal description of entities and their properties; it forms a shared terminology for the objects of interest in the domain, along with definitions for the meaning of each of the terms.



Terminology
• Generic across many domains
• Key distinctions
Logical Theory
• Definitions and constraints
• Defined using FOL
• Interpreted using Prolog
• Provides automated question answering
• Graphic representation of terms
• X windows based
• Multi-state
• Active image
• Onscreen modification
Symbology

The goal of the TOVE (TOronto Virtual Enterprise) project is to create an enterprise ontology that has the following characteristics: 1) provides a shared terminology for the enterprise that every application can jointly understand and use, 2) defines the meaning (semantics) Of each term in a precise and as unambiguous manner as possible using First Order Logic, 3) implements the semantics in a set of Prolog axioms that enable TOVE to automatically deduce the answer to many “common sense” questions about the enterprise, and 4) defines a symbology for depicting a term or the concept constructed thereof in a graphical context.

The TOVE ontology currently spans knowledge of activity [Gruninger & Fox 94], time, and causality, resources [Fadel 94] [Fadel et al. 94], and more enterprise oriented knowledge such as cost [Tham et al. 94], quality [Kim & Fox 94] [Kim et al. 95] and organization Structure [Fox et al. 95] and agility [Atefi 95]. The TOVE Testbed provides an environment for analyzing enterprise ontologies; it provides a model of an enterprise and tools for browsing, visualization, simulation, and deductive queries.

In this paper we present a logical framework for the TOVE Enterprise Model. We first review our process for engineering an ontology. We then describe our logical framework which is based on Reiter's solution of the frame problem [Reiter 91] and Pinto's formalization of occurrence and the incorporation of time within the situation calculus [Pinto & Reiter 93]. We then provide an ontology for activity and extensions for resource spoilage. This is followed by example queries that the ontology supports.

2.0 ONTOLOGY ENGINEERING

For any given ontology, the goal is to agree upon a shared terminology and set of constraints on the objects in the ontology. We must agree on the purpose and ultimate use of our ontologies, We must therefore provide a mechanism guiding the design of ontologies, as well as providing a framework for evaluating the adequacy of these ontologies. Such a framework allows a more precise evaluation of different proposals for an ontology, by demonstrating the competency of each proposal with respect to the set of questions that arise from the applications. These justify the existence and properties of the objects with the ontology.

The process of engineering an ontology that we have developed, starts from scenarios describing the applications which the model is to support. Next, we define a set of informal competency questions [Gruninger & Fox 94] that represent the types of the information the model is to provide to each application. Next, an initial terminology composed of objects, attributes and relations is designed. The terminology and questions are then iteratively refined until a precise and unambiguous set of terms are created

Motivating Scenarios: The development of ontologies is motivated by scenarios that arise in the applications. In particular, such scenarios may be presented by industrial partners as problems which they encounter in their enterprises. The motivating scenarios often have the form of story problems or examples which are not adequately addressed by existing ontologies. A motivating scenario also provides a set of intuitively possible solutions to the scenario problems. These solutions provide a first idea of the informal intended semantics for the objects and relations that will later be included in the ontology.

- **Informal Competency Questions:** Given the motivating scenario, a set of queries will arise which place demands on an underlying ontology. We can consider these queries to be requirements that are in the form of questions that an ontology must be able to answer. These are the informal competency questions, since they are not yet expressed in the formal language of the ontology.

  Ideally, the competency questions should be defined in a stratified manner, with higher level questions requiring the solution of lower level questions. It is not a well-designed ontology if all competency questions have the form of simple lookup queries; there should be questions that use the solutions to such simple queries.

  These competency questions do not generate ontological commitments; rather, they are used to evaluate the ontological commitments that have been made. They evaluate the expressiveness of the ontology that is required to represent the comptency questions and to characterize their solutions.

- **Specification in First-order Logic -- Terminology:** Once informal competency questions have been posed for the proposed new or extended ontology, the terminology of the ontology must then be specified using first-order logic.

Recall that an ontology is a formal description of objects, properties of objects, and relations among objects. This provides the language that will be used to express the definitions and constraints in the axioms. This language must provide the necessary terminology to restate the informal competency questions.

The first step in specifying the terminology of the ontology is to identify the objects in the domain of discourse. These will be represented by constants and variables in the language. Attributes of objects are then defined by unary predicates: relations among objects defined using n-ary predicates.

- **Specification in First-Order Logic -- Axioms:** The axioms in the ontology specify the definitions of terms in the ontology and constraints on their interpretation; they are defined as first-order sentences using the predicates of the ontology. It is important to understand the significance of using axioms to define the terms and constraints for objects in the ontology. Simply proposing a set of objects alone, or proposing a set of ground terms in first-order logic, does not constitute an ontology. Axioms must be provided to define the semantics, or meaning, of these terms.

  It is also important to realize that this is not the implementation of the ontology; it is the specification of the ontology. However, the implementation of the ontology should be translatable into KIF.

  The process of defining axioms is perhaps the most difficult aspect of defining ontologies. However, this process is guided by the formal competency questions. As with the informal competency questions, the axioms in the ontology must be necessary and sufficient to express the competency questions and to characterize their solutions; without the axioms we cannot express the question or its solution, and with the axioms we can express the question and its solutions. Further, any solution to a comptency question must be entailed by or consistent with the axioms in the ontology alone. If the proposed axioms are insufficient to represent the formal competency questions and characterize the solutions to the questions, then additional objects or axioms must be added to the ontology until it is sufficient. This development of axioms for the ontology with respect to the competency questions is therefore an iterative process.

  There may be many different ways to axiomatize an ontology, but the formal competency questions are not generating these axioms. Rather, we use them to evaluate the completeness of the sets of axioms in any particular axiomatization. In this sense, we can compare the expressiveness of different sets of axioms using the competency questions. If there is a competency question that one set of axioms can represent and another cannot, then the first set is more expressive. If two different axiomatizations can represent a competency question and characterize its solutions, then they are equivalent with respect to the question, and any comparison must use other criteria.

- **Completeness Theorems:** Once the competency questions have been formally stated, we must define the conditions under which the solutions to the questions are complete. This forms the basis for completeness theorems for the ontology.

The TOVE Ontology Engineering process addresses the issues raised earlier. Reusability is addressed by defining a terminology that is generic across many domains. Consistent Usage is addressed by providing semantics for the terminology in the form of axioms in first-order logic. Accessibility is addressed by implementing the axioms in Prolog thereby providing a deductive query processing facility. Lastly, the Selection issue is addressed by providing a way of characterizing the span of an ontology by what we call competency questions. The ontology must contain a necessary and sufficient set of axioms to represent and solve these questions. It is in this sense that we can claim to have, an adequate ontology appropriate for a given task, and it is this rigor that is lacking in previous approaches to enterprise modelling.

3.0 LOGIC FRAMEWORK: SITUATION CALCULUS

At the core of the TOVE ontology lies a representation of action composed of activity, state, time and causality. The competency of this ontology is focused primarily on projecting what will be true in time. That is, given a set of actions that occur at different points in the future, what are the properties of resources and activities at other points in time. This is also called Temporal Projection and induces the following set of requirements on the ontologies:

- Temporal projection requires the evaluation of the truth value of a proposition at some point in time in the future. We therefore need to define axioms that express how the truth of a proposition changes over time. In particular, we need to address the frame problem and express the properties and relations that change or do not change as the result of an activity.
- We must define the notion of a state of the world, that is, define what is true of the world before and after performing different activities. This is necessary to express the causal relationship between the preconditions and effects of an activity.
- The time period over which the state has a certain status is bounded by the times at which the appropriate actions that change status occur. This period defines the duration of a state if the status is enabled. This is essential for the construction of schedules.
- We want a uniform hierarchical representation for activities (aggregation). Plans and processes are constructed by combining activities. We must precisely define how activities are combined to form new ones. The representation of these combined activities should be the same as the representation of the subactivities. Thus aggregate activities (sets of activities or processes) should themselves be represented as activities.
- The causal and temporal structure of states and subactivities of an activity should be explicit in the representation of the activity.

Within the TOVE project, we have adopted the situation calculus to provide a semantics to our ontology of activity and state. The intuition behind the situation calculus is that there is an initial situation, and that the world changes from one situation to another when actions are performed. There is a predicate $Poss(a,\sigma)$ that is true whenever an action a can be performed in a situation σ.

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, we impose a structure over situations that is isomorphic to the natural numbers by introducing the notion of successor situation [Reiter 91]. The function $do(a,\sigma)$ is the name of situation that results from performing action *a* in situation σ. We also define an initial situation denoted by the constant σ_0.

To define the evaluation of the truth value of a sentence in a situation, we will use the predicate *holds(f,*σ*)* to represent the fact that some ground literal *f* is true in situation σ. A fluent is a predicate or function whose value may change between situations. Using this predicate we can define state constraints, which are sentences that must be satisfied in all situations.

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 status of the state in that situation.

Another important notion is to represent the occurrence of actions. The work of [Pinto & Reiter 93] extends the situation calculus by selecting one branch of the situation tree to describe the evolution of the world as it actually unfolds. This is done using the predicate *actual* defined by the following axioms:

The initial situation is always actual:

$$actual(\sigma_0) \tag{EQ 1}$$

If a situation is actual, then its immediate predecessor must also be actual:

$$(\forall\, a,\sigma)\; actual(do(a,\sigma)) \supset actual(\sigma) \wedge Poss(a,\sigma) \tag{EQ 2}$$

An actual situation has at most one actual successor situation.

$$(\forall\, a_1,a_2,\sigma)\; actual(do(a_1,\sigma)) \wedge actual(do(a_2,\sigma)) \supset a_1 = a_2 \tag{EQ 3}$$

Thus, actual defines a line within the situation tree.

To represent occurrences, we then introduce the predicate *occurs(a,*σ*)* defined as actions performed along the actual line:

$$occurs(a,\sigma) \equiv actual(do(a,\sigma)) \tag{EQ 4}$$

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.

$$(\forall\ r,\sigma)\ holds(spoiled(r),\sigma) \tag{EQ 5}$$

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

$$(\forall\ r,\sigma)\ actual(\sigma) \supset holds(spoiled(r),\ \sigma) \tag{EQ 6}$$

We can thus represent maintenance and prevention as actual line constraints with universal quantifiers (we will call these universal actual line constraints), and represent goals and deadlines as actual line constraints with existential quantifiers (which we will call existential actual line constraints). For example, we can represent the goal of producing 50 bolts as the following actual line constraint:

$$(\exists\ r,q,\sigma)\ actual(\sigma) \wedge holds(rp(bolt,50),\sigma) \tag{EQ 7}$$

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.

4.0 TIME AND ACTION

We represent time as a continuous line; on this line we define time points and time periods (intervals) as the domain of discourse. We define a relation < over time points with the intended interpretation that $t < t'$ iff t is earlier than t'. One important property that must be represented is the intuition that some action a occurs and then some action b occurs, and that there is no intervening event between a and b. Furthermore, we want to define what holds in the world after performing some action in order to capture the notion of causality. How do we express these notions if we have a continuous time line? Since situations have duration, they can be defined as a set of distinguished intervals on the time line; they will be denoted by the letters σ. The following axioms establish the properties of the relation < over situations:

$$(\forall\ a,\ \sigma_1,\sigma_2)\ \sigma_1 < do(a,\sigma_2) \equiv \sigma_1 \leq \sigma_2 \tag{EQ 8}$$

$$(\forall\ a_1,a_2,\ \sigma_1)\ do(a_1,\sigma_1) = do(a_2,\sigma_1) \supset a_1 = a_2 \tag{EQ 9}$$

$$(\forall\ \sigma_1,\sigma_2)\ \sigma_1 < \sigma_2 \supset \neg\ \sigma_2 < \sigma_1 \tag{EQ 10}$$

$$(\forall\ \varphi)[\varphi(\sigma_0) \wedge (\forall\ \sigma,\mathrm{a})\ (\varphi(\sigma) \supset \varphi(do(a,\ \sigma))] \supset (\forall\ \sigma)\ \varphi(\sigma) \tag{EQ 11}$$

This enables us to define the intuition of no intervening events, that is, there is no situation between a situation and its successor, which is a consequence of the axioms:

$$(\forall\ \alpha,\sigma,\sigma')\ \neg\ (\sigma < \sigma' < do(a,\sigma)) \tag{EQ 12}$$

Situations are assigned different durations by defining the predicate *start(s,t)* [Pinto & Reiter 93]. Each situation has a unique start time; these times begin at 0 in σ_0 and increase monotonically away from the initial situation.

$$(\forall\ \sigma)\ (\exists\ t)\ start(\sigma,t) \tag{EQ 13}$$

$$start(\sigma_0, 0) \tag{EQ 14}$$

$$(\forall\ \sigma, t,t')\ start(\sigma,t) \wedge start(\sigma,t') \supset t = t' \tag{EQ 15}$$

$$(\forall\ a.\ \sigma, t,t')\ start(\sigma,t) \wedge start(do(a,\sigma),t') \supset t < t' \tag{EQ 16}$$

To define the evaluation of the truth value of a sentence at some point in time, we will use the predicate $holds(f,\sigma)$ to represent the fact that some ground literal f is true in situation σ. Using the assignment of time to situations, we define the predicate $holds_T(f, t)$ to represent the fact that some ground literal f is true at time t. A fluent is a predicate or function whose value may change with time.

Another important notion is that actions occur at points in time. To represent this we introduce two predicates, $occurs(a,\sigma)$ and $occurs_T(a,t)$, defined as follows:

$$occurs(a,\sigma) \equiv \sigma_0 < do(a,\sigma) \tag{EQ 17}$$

$$occurs_T(a,t) \equiv occurs(a,\sigma) \wedge start(do(a, \sigma), t) \tag{EQ 18}$$

We will now apply this formalism to the representation of activities in an enterprise.

5.0 ACTIVITIES AND STATES

At the heart of the TOVE Enterprise Model lies the representation of an *activity* and its corresponding enabling and caused *states* ([Sathi et al. 85], [Fox et al. 93]). In this section we examine the notion of states and define how properties of activities are defined in terms of these states. An activity is the basic transformational action primitive with which processes and operations can be represented; it specifies how the world is changed. An enabling state defines what has to be true of the world in order for the activity to be performed. A caused state defines 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*. The state tree linked by an *enables* relation to an activity specifies what has to be true in order 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. Intermediate states of an activity can be defined by elaborating the aggregate activity into an activity network.

There are two types of states: *terminal* and *non-terminal*. In Figure 1, *es_fabricate_plug_on_wire* is the nonterminal enabling state for the activity *fabricate_plug_on_wire* and *pro_fabricate_plug_on_wire* is the caused state for the activity. The terminal conjunct substates of *es_fabricate_plug_on_wire* are *consume_wire, consume_plug*, and *use_inject_mold* since all three resources must be present for the activity to occur; the terminal states of *pro_fabricate_plug_on_wire* are *produce_plug_on_wire* and *release_inject_mold*. The activity *assemble2 wire_switch* is enabled by the consumption of plug_on_wire (*consume plug_on_wire*) and the use of an assembly area (*use asmbly_area*); this is represented by the nonterminal state *es2_assemble_wire_switch*. This activity causes the production of wire_switch (*produce wire_switch*) and the release of the used resource (*release asmbly_area*); this is represented by the nonterminal state *pro2_assemble_wire_switch*.

In TOVE there are four terminal states represented by the following predicates:*use(s,a)*, *consume(s,a)*, *release(s,a), produce(s,a)*. These predicates relate the state with the resource required by the activity. 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(s,a)* signifies that a resource is to be used up by the activity and will not exist once the activity is completed, and *produce(s,a)* signifies that a resource, that did not exist prior to the performance of the activity, has been created by the activity. We define use and consume states to be enabling states since the preconditions for activities refer to the properties of these states, while we define release and produce states to be caused states, since their properties are the result of the activity.

Terminal states are also used to represent the amount of a resource that is required for a state to be enabled. For this purpose, the predicate *quantity(s,r,q)* is introduced, where *s* is a state, *r* is the associated resource, and *q* is the amount of resource r that is required. Thus if *s* is a consume state, then *q* is the amount of resource consumed by the activity, if *s* is a use state, then *q* is the amount of resource used by the activity, and if *s* is a produce state, then *q* is the amount of resource produced.

In this section, we formalize the relationship between states and activities. First we examine the notion that an activity specifies a transformation on the world; this requires that we introduce fluents for states and activities, and the actions that change these fluents. The axioms presented adequate for solving the temporal projection problem for these properties of states and activities.

To formalize the notions of nonterminal states and aggregate activities, we introduce occurrence axioms for a set of actions.

5.1 Successor Axioms for Status of Terminal States

The primary fluents we will consider are the values assigned to states to capture the notion of the status of a state. We define a new sort for the domain of the status with the following set of constants:{*possible, committed, enabled, completed, disenabled, reenabled*}. The status of a state is changed by one of the following actions:*commit(s,a), enable(s,a), complete(s,a), disenable(s,a), reenable(s,a)*. Note that these actions are parametrized by the state and the associated activity.

The next step is to define the successor axioms that specify how the above actions change the status of a state. 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 point in time) by first finding the situation containing that time point, and then using the successor axioms to evaluate the status of the state in that situation.

The status of a state is committed in a situation iff either a commit action occurred in the preceding situation, or the state was already committed and an enable action did not occur.

$$(\forall\, s,a,e,\sigma)\ holds(status(s,a,\ committed),\ do(e,\sigma)) \equiv (e = commit(s,a) \wedge holds(status(s,a,possible),\ \sigma)) \vee \neg(e = enable(s,a)) \wedge holds(status(s,a,\ committed),\ \sigma) \qquad \text{(EQ 19)}$$

The status of a state is enabled in a situation iff either an enable action occurred in the preceding situation, or the state was already committed and a complete action or disenable action did not occur.

$$(\forall\ s,a,e,\ \sigma)\ holds(status(s,a,\ enabled),\ do(e,\ \sigma)) \equiv (e= enable(s,a) \wedge holds(status(s,a,committed),\ \sigma)) \vee \neg[(e=complete(s,a) \vee e=disenable(s,a)) \wedge holds(status(s,a,\ enabled),\ \sigma)] \qquad \text{(EQ 20)}$$

The status of a state is completed in a situation iff either a complete action occurred in the preceding situation, or the state was already completed.

$$(\forall\ s,a,e,\ \sigma)\ holds(status(s,a,\ completed),\ do(e,\ \sigma)) \equiv [e= complete(s,a) \wedge (holds(status(s,a,enabled),\ \sigma) \vee holds(status\ (s,a,reenabled),\sigma))] \vee holds(status(s,a,\ completed),\ \sigma) \qquad \text{(EQ 21)}$$

The status of a state is disenabled in a situation iff either a disenable action occurred in the preceding situation, or the state was already disenabled and a reenable action did not occur.

$$(\forall\ s,a,e,\ \sigma)\ holds(status(s,a,\ disenabled),\ do(e,\ \sigma)) \equiv [e= disenable(s,a) \wedge (holds(status(s,a,enabled),\ \sigma) \vee holds(status\ (s,a,reenabled),\sigma))] \vee \neg\ (e=reenable(s,a)) \wedge holds(status(s,a,\ disenabled),\ \sigma) \qquad \text{(EQ 22)}$$

The status of a state is reenabled in a situation iff either a reenable action occurred in the preceding situation, or the state was already reenabled and a complete action or disenable action did not occur.

$$(\forall\ s,a,e,\ \sigma)\ holds(status(s,a,\ reeenabled),\ do(e,\ \sigma)) \equiv (e= reeenable(s,a) \wedge holds(status(s,a,disenabled),\ \sigma)) \vee \neg(e=complete(s,a) \vee e=disenable(s,a)) \wedge holds(status(s,a,\ reenabled),\ \sigma) \qquad \text{(EQ 23)}$$

Note that in each of these axioms we also specify the precondition for the action. These preconditions can equivalently be expressed as the following occurrence axioms:

$$(\forall\ s,a,\ \sigma)\ occurs(commit(s,a),\ \sigma) \supset holds(status(s,a,possible),\ \sigma) \qquad \text{(EQ 24)}$$

$$(\forall\ s,a,\ \sigma)\ occurs(enable(s,a),\ \sigma) \supset holds(status(s,a,committed),\ \sigma) \qquad \text{(EQ 25)}$$

$$(\forall\ s,a,\ \sigma)\ occurs(complete(s,a),\ \sigma) \supset (holds(status(s,a,enabled),\ \sigma) \vee holds(status(s,a,reenabled),\ \sigma)) \qquad \text{(EQ 26)}$$

$$(\forall\ s,a,\ \sigma)\ occurs(disenable(s,a),\ \sigma) \supset (holds(status(s,a,enabled),\ \sigma) \vee holds(status(s,a,reenabled),\ \sigma)) \qquad \text{(EQ 27)}$$

$$(\forall\ s,a,\ \sigma)\ occurs(reenable(s,a),\ \sigma) \supset holds(status(s,a,disenabled),\ \sigma) \qquad \text{(EQ 28)}$$

How are these incorporated into the activity-state clusters, which only represent the causal relationships among states and activities? The occurrence of a commit action is not explicitly given in the specification of an activity. However, since the status fluents can only be changed by the above set of actions, the following sentences can be derived from the axioms:

$$(\forall\ s,a,\ \sigma)\ occurs(enable(s,a),\ \sigma) \supset (\exists\sigma')\ occurs(commit(s,a),\ \sigma') \qquad \text{(EQ 29)}$$

$$(\forall s,a,\sigma) occurs(complete(s,a),\sigma) \supset (\exists\sigma')(occurs(enable(s,a),\sigma') \vee occurs(reenable(s,a),\sigma')) \qquad \text{(EQ 30)}$$

$$(\forall\ s,a,\ \sigma)\ occurs(disenable(s,a),\ \sigma) \supset (\exists\sigma')\ (occurs(enable(s,a),\ \sigma') \vee occurs(reenable(s,a),\ \sigma')) \qquad \text{(EQ 31)}$$

$$(\forall\ s,a,\ \sigma)\ occurs(reenable(s,a),\ \sigma) \supset (\exists\sigma')\ occurs(disenable(s,a),\ \sigma') \qquad \text{(EQ 32)}$$

Similarly, the precondition for the *commit* action is that the state be *possible*. In [Fadel 94] it is shown how the *possible* status is defined in terms of the availability of a resource for the activity. This includes the configuration or setup of a resource as well as capacity constraints for the concurrent execution of activities with a shared resource. Axioms similar to those above would be used to express the occurrence of the appropriate setup activities for some activity. This is necessary for formalizing time-based competition, where the occurrence of setup activities is minimized.

5.2 Status of Non-Terminal States

In TOVE, non-terminal states enable the boolean combination of states. We will consider four non-terminal states: *conjunctive, disjunctive, exclusive, not*. What precisely does it mean for a non-terminal state to be a boolean combination of states? For example, how do we define the status of a non-terminal state given the status of each substate? To define this notion, we must refer to the occurrence of the actions that change the status of the states.

Disjunctive states are used to formalize the intuition of a resource pool. We may have a set of resources, such as machines or operators, that can possibly be used by an activity. The activity only requires one of these resources, so the activity only needs to nondeterministically choose one of the alternative resources in the pool. Thus, the status of the disjunctive state changes if one of the resources has been selected and its status has been changed. For example, we have

$$(\forall\ s,s_1,\ldots,s_n,a,\ \sigma)\ disjunctive(s,a) \wedge substate(s_1,s) \wedge \ldots \wedge substate(s_n,s) \supset occurs(enable(s,a),\ \sigma) \equiv occurs(enable(s_1,a),\ \sigma) \vee \ldots \vee occurs(enable(s_n,a),\ \sigma) \qquad \text{(EQ 33)}$$

The successor axioms for the other values of status are defined in the same way. In other words, the occurrence of an action for a disjunctive state is equivalent to a disjunctive sentence of occurrence literals for each disjunct substate.

Similarly, we have the following constraints on conjunctive states:

$$(\forall\ s,s_1,\ldots,s_n,a,\ \sigma)\ conjunctive(s,a) \wedge substate(s_1,s) \wedge \ldots \wedge substate(s_n,s) \supset occurs(enable(s,a),\ \sigma) \equiv occurs(enable(s_1,a),\ \sigma) \wedge \ldots \wedge occurs(enable(s_n,a),\ \sigma) \qquad \text{(EQ 34)}$$

The occurrence of an action for a conjunctive state is equivalent to a conjunctive sentence of occurrence literals for each conjunct substate. Note that we make the assumption that all conjunct substates change their status at the same time.

For exclusive states we have constraints of the form

$$(\forall\ s,s_1,\ldots,s_n,a,\ \sigma)\ exclusive(s,a) \wedge substate(s_1,s) \wedge \ldots \wedge substate(s_n,s) \supset occurs(enable(s,a),\ \sigma) \equiv occurs(enable(s_1,a),\sigma) \vee \ldots \vee occurs(enable(s_n,a),\ \sigma) \qquad \text{(EQ 35)}$$

$$(\forall\ s,s_i,s_j,a,\ \sigma)\ exclusive(s,a) \wedge substate(s_i,s) \wedge substate(s_j,s) \wedge occurs(enable(s,a),\ \sigma) \supset (occurs(enable(s_i,a),\ \sigma) \equiv \neg occurs(enable(s_j,a),\ \sigma) \qquad \text{(EQ 36)}$$

so that the occurrence of an action for an exclusive state is equivalent to the occurrence of the action for exactly one of the substates.

For not states we have the constraint that the action for the substate does not occur when the action for the nonterminal state occurs:

$$(\forall\ s,s_1,a,\ \sigma)\ not(s,a) \wedge substate(s_1,s) \supset occurs(enable(s,a),\ \sigma) \equiv \neg\ occurs(enable(s_1,a),\ \sigma) \qquad \text{(EQ 37)}$$

In this way we can define arbitrary nonterminal states as occurrence axioms.

5.3 Status of Activities

Just as status was defined for states, we can define the status of an activity. We define a new sort for the domain of the status of an activity with the following set of constants:{ *dormant, executing, suspended, reExecuting, terminated* }. The status of an activity is determined by the status of its enabling and caused states.

An activity is dormant iff its enabling state is committed. In this case, the resources associated with the state are committed but not yet enabled.

$$(\forall\ a,s,\ \sigma)\ enabling(s,a) \supset holds(status(a,\ dormant),\ \sigma) \equiv holds(status(s,a,committed),\ \sigma) \qquad \text{(EQ 38)}$$

An activity is executing iff either its enabling or caused state is enabled.

$$(\forall\ a,\ \sigma)\ holds(status(a,\ executing),\ \sigma) \equiv (\exists\ s)\ (enabling(s,a) \vee caused(s,a)) \wedge holds(status(s,a,\ enabled),\ \sigma) \qquad \text{(EQ 39)}$$

An activity is suspended iff its enabling and caused states are disenabled.

$$(\forall\ a,s,\ \sigma)\ (enabling(s,a) \vee caused(s,a)) \supset holds(status(a,\ suspended),\ \sigma) \equiv holds(status(s,a,\ disenabled),\ \sigma) \qquad \text{(EQ 40)}$$

An activity is reexecuting iff its enabling and caused states are reenabled.

$$(\forall\ a,s,\ \sigma)\ (enabling(s,a) \vee caused(s,a)) \supset holds(status(a,\ reExecuting),\ \sigma) \equiv holds(status(s,a,\ reenabled),\ \sigma) \qquad \text{(EQ 41)}$$

An activity is terminated iff its enabling and caused states are completed.

$$(\forall\ a,s,\ \sigma)\ (enabling(s,a) \vee caused(s,a)) \supset holds(status(a,\ terminated),\ \sigma) \equiv holds(status(s,a,completed),\ \sigma) \qquad \text{(EQ 42)}$$

5.4 Duration

By combining the ontology of time with the ontology of states of activities, we arrive at the notion of duration, which is essential for scheduling and the analysis of activities in time-based competition. The duration of a state is defined as the time period beginning at the time that the state is enabled and ending at the time that the state is completed. Similarly, the duration of an activity is defined as the time period beginning at the time that activity begins the status of executing and ending at the time that the activity begins the status of terminated. The duration of a state is represented by the predicate *state_duration(s,d)*, while the duration of an activity is represented by the predicate *activity_duration(a,d)*. In the representation of an activity, the duration of a state satisfies the occurrence axiom

$$(\forall a,s,t,t',d)\, state\_duration(s,d) \equiv occurs_T(enable(s,a),t) \wedge occurs_T(complete(s,a),t') \wedge d = t - t' \quad \text{(EQ 43)}$$

We can also define intervals for the remaining status values, such as committed:

$$(\forall a,s,t,t',d)\ committed\_duration(s,d) \equiv occurs_T(commit(s,a), t) \wedge occurs_T(complete(s,a), t') \wedge d = t - t' \quad \text{(EQ 44)}$$

In [Fadel 94], the committed duration is necessary to schedule the availability of a resource for a set of activities over some time interval. The resource must have sufficient capacity to support each activity at every time point in the interval and the resource may not be available to one activity if it is committed to other activities .

5.5 Aggregation of Activities

An important requirement for an ontology for activities is the ability to aggregate a set of activities to form a new activity. Activity clusters may be also aggregated to form multiple levels of abstraction. An activity is elaborated to an aggregate activity (an activity network), which then has activities [Sathi, et al. 85]. These activities are subactivities of the aggregate activity. We introduce the predicate *subactivity(a,a′)* to denote that activity *a′* is a subactivity of activity *a*. For example, consider the activity clusters in Figure 1; the activities *fabricate plug_on_wire* and *assemble2 wire_switch* are sub-activities of **a***ssemble_ws aggregation.* The states *es_fabricate plug_on_wire* and *es2_assemble wire_switch* are substates of *enable_ws aggregation*, and the states *pro_fabricate plug_on_wire* and *pro_assemble wire_switch* are substates of *pro_ws_aggregation.*

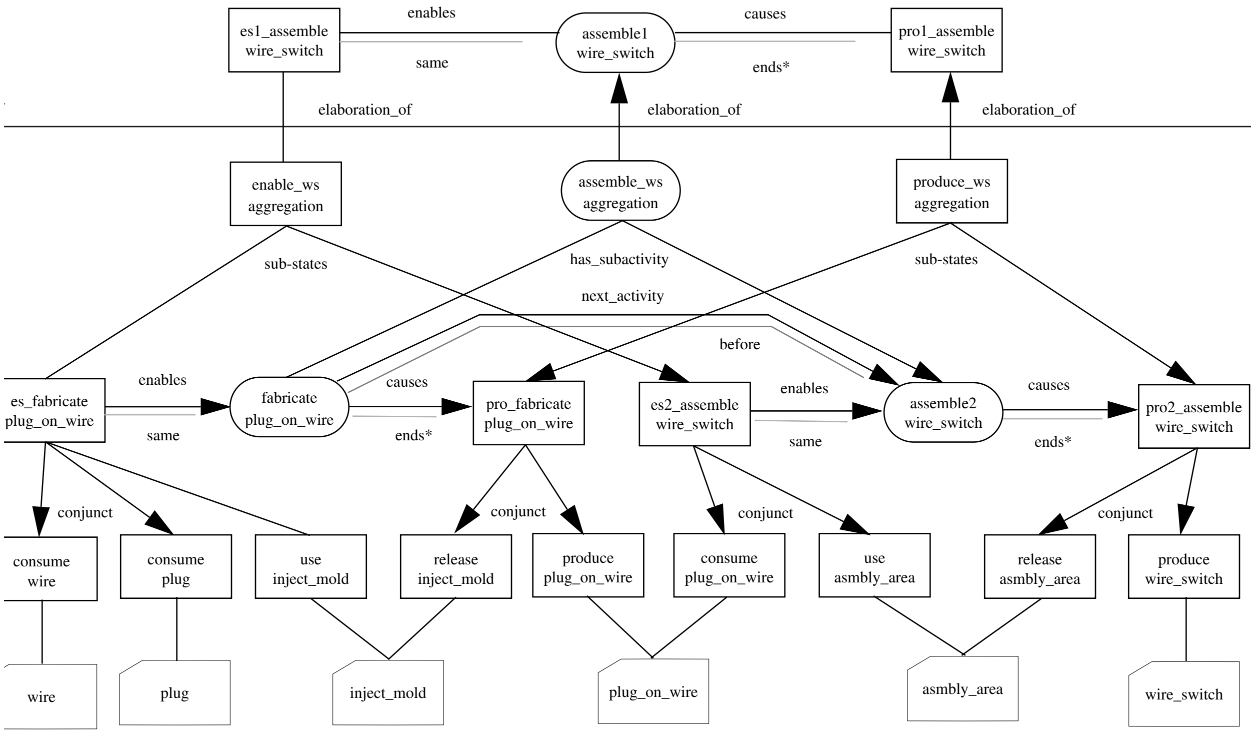
To completely specify an aggregate activity, we must define the temporal relations over its subactivities and the states of the subactivities. Indeed, the definition of status for activities allows to represent the temporal structure of an aggregate activity in terms of the enabling and caused states of each subactivity; we do this using occurrence axioms.

Essentially, the representation of an activity consists of a sequence of actions that commit, enable, and complete states. These actions may be partially ordered; once the actions in an activity have been totally ordered, we then assign times to the situations in which the actions occur. Thus activities will be represented by an occurrence axiom of the form:

$$(\forall a, s, s_1,\ldots,s_n,\sigma)\ enabling(s,a) \wedge substate(s_1,s) \wedge \ldots \wedge substate(s_n,s) \supset [occurs(enable(s,a),\sigma) \supset (\exists \sigma_1,\ldots,\sigma_n,t_1,\ldots t_n)\ occurs(enable(s_1,a), \sigma_1) \wedge \ldots \wedge occurs(enable(s_k,a), \sigma_i) \wedge occurs(complete(s_1,a), \sigma_{i+1}) \wedge \ldots \wedge occurs(complete(s_k,a), \sigma_n) \wedge occurs_T(enable(s_1,a), t_1) \wedge \ldots \wedge occurs_T(enable(s_k,a), t_i) \wedge occurs_T(complete(s_1,a), t_{i+1}) \wedge \ldots \wedge occurs_T(complete(s_n,a), t_n)] \quad \text{(EQ 45)}$$

Note that this specification of the activity does not place any constraints on the ordering over the situations and times in which the actions occur. The complete specification of the aggregation of a set of arbitrary activities would impose a total ordering over the occurrences. In general, this is a scheduling problem which remains for future work.

FIGURE 1. Activity Cluster



ends* - All time relations in this diagram illustrates how a state's time interval is related to that of an activity's time interval. Therefore, <activity> does not end <state>, <state> ends <activity>.

temporal relation — activity-state relation

6.0 Spoilage Axioms

In preparation for the example queries in the next section, we introduce an additional set of axioms that define simple theory of resource spoilage, The representation and reasoning about spoilage is important in food domains where inventory must be moved prior to its spoilage date. We begin by introducing the notion that once a resource is produced, and given a limited shelf life, the resource will be spoiled unless somthing else makes that impossible:

$$\text{OccursT}(\text{terminate}(a),t) \wedge \text{produces}(a,r) \wedge \text{shelf\_life}(r,d) \wedge \text{PossT}(\text{spoilage}(r), t+d) \supset \text{OccursT}(\text{spoilage}(r),t+d) \quad \text{(EQ 46)}$$

The possibility that the resource can spoil is determined by the precondition axiom that states that as long as the quantity is greater than 0, e.g., it has not been shipped to a customer, then it can spoil:

$$\text{poss}(\text{spoilage}(r), s) \equiv \text{holds}(\text{rp}(r,q),s) \wedge q>0 \quad \text{(EQ 47)}$$

The following are additional axioms to complete the set. Successor axiom for the fluent spoiled:

$$(\forall\, a, r, s)\ \text{holds}(\text{spoiled}(r), \text{do}(a,s)) \equiv ((\neg\text{holds}(\text{spoiled}(r), s) \wedge a=\text{spoilage}(r)) \vee \text{holds}(\text{spoiled}(r), s)) \quad \text{(EQ 48)}$$

$$\text{quantity}(s,r,q) \wedge \text{enables}(s,a) \supset (\text{Poss}(a, s) \supset \neg\text{holds}(\text{spoiled}(r), s)) \quad \text{(EQ 49)}$$

7.0 Example Queries

The following are examples of queries whose answers are deduced using an initial set of assertions, the axioms presented above and other axioms, such as activity-based costing axioms, that are not presented here. The english version of the query is given in italics, followed by its prolog specification and the answered returned.

What is the status of the state "pro_clip_base" at times 13 and 17?

```
:- HoldsT(status(pro_clip_base, X), 13).

X = enabled.

:- HoldsT(status(assemble_clip_base, X), 17).

X = completed.
```

Determine how much of round_nut1 exists at time 7.

```
:- holdsT(rp(round_nut1, Q), 7).

Q = 20.
```

rp, which stands for resource point, specifies the quantity of a resource that exists.

What is the available capacity of clip_base resource at time tp1?

```
:- available_capacity(clip_base, tp1, Amount, Unit).

Amount = 12

Unit = object;
```

How many clip_base's are committed at time point tp1?

```
:- total_committed(clip_base, Amount, tp1, object).

Amount = 2;
```

total_committed specifies the quantity of a resource that is committed to activities.

Will shipment10 of oranges spoil if they are not shipped before Friday?

```
:- holdsT(spoiled(shipment10), T), T < friday.

no;
```

Is any milk spoiled by Wednesday?

```
:- holdsT(spoiled(milk), wednesday).

yes;
```

Is there any time at which the stock level for bolts at the Scarborough factory reaches the minimum safety level?

```
:- holdsT(stock_status(bolt1, Scarborough, minimum), T).

no;
```

What is the total cost for order O1?

```
:- holdsT(cpo(O1,C), 25).
C = 15;
```

cpo is a function that returns the cost of an order.

How much of the cost for assemble_clip_reading_lamp is due to machine breakdown?

```
:-holdsT(suspend_act_cost(assemble_clip_reading_lamp, C), 17).
C = 5;
```

susspend_act_cost returns the portion of the cost of an activity due to its being suspended.

How much of the cost for assemble_clip_reading_lamp associated with the resource bolt is due to idleness before the activity begins executing?

```
:- holdsT(committed_res_cost(bolt1, assemble_clip_reading_lamp,C), 10).
C = 15;
```

commited_res_cost returns the cost of committing a resource to an activity.

8.0 SUMMARY

In this paper, we presented a logical formalization of the TOVE ontology of activity and time that has been designed to specify the tasks that arise in integrated supply chain management and enterprise engineering. To this end, we have defined the TOVE ontologies for activities, states, and time within first-order logic. This formalization allows deduction of properties of activities and states at different points in time by formalizing how these properties do or do not change as the result of an activity (temporal projection). The representation of aggregatete activities, and the role of temporal structure in this aggregation, is accomplished through axioms that allow us to reason about the occurrence of actions. We have also shown an extension of the ontology to cover spoilage and given examples of the ontologies deductive query support when implemented in prolog.

The ontologies for activities, states, and time defined in this paper have been implemented on top of C++ using the ROCK knowledge representation tool from Carnegie Group. The successor state axioms and occurrence axioms have been implemented using Quintus Prolog. The formalization of activities is being extended to handle concurrent activities, reasoning about the availability and capacity of resources, and activity-based costing.

The benefits of our approach to enterprise modelling include:

- Providing a shareable, reusable representation
- Providing a deductive database able to deduce anwers to common sense questions thereby reducing the need for ad hoc report generators and interfaces
- Providing a standard for visualizing enterprise knowledge
- Providing a formal representation of meaning!

9.0 ACKNOWLEDGEMENTS

This research is supported, in part, by the Natural Science and Engineering Research Council of Canada, Carnegie Group Inc., Digital Equipment Corp., Micro Electronics and Computer Research Center, Quintus Corp., and Spar Aerospace. An earlier draft of this paper appeared in the proceedings of the IFIP WG 5.12 workshop on Enterprise Modelling and the Proceedings of the British Computer Society's 1995 Expert Systems Conference.

10.0 REFERENCES

[Blackburn 91] Blackburn J. *Time-based Competition*. Business One Irwin, 1991.

[Davenport 93] Davenport, T.H. *Process Innovation: Reengineering Work through Information Technology.* Harvard Business School Press, 1993.

[Davis et al. 83] Davis, B.R., Smith, S., Davies, M., and St. John, W. Integrated Computer-aided Manufacturing (ICAM) Architecture Part III/Volume III: Composite Function Model of "Design Product" (DES0). Technical Report AFWAL-TR-82-4063 Volume III, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, 1983.

[Fadel 94] Fadel, F. *Resource Ontology for Enterprise Modelling*. M.A.Sc. thesis, Department of Industrial Engineering, University of Toronto.

[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), pp. 117-128.

[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,* pp. 425-429.

[Fox et al. 94] Fox, M. S., Gruninger, M., Zhan, Y.. Enterprise engineering: An information systems perspective*Proceedings of the Industrial Engineering Research Conference 1994,* pp. 461-466.

[Fox et al. 95] Fox, M.S., Barbuceanu, M., Gruninger, M. An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour, *Fourth Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises*,(West Virginia University 1995).

[Gruber 93] Gruber, Thomas R., *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, KSL 93-4, Computer Science Department, Stanford University, 1993.

[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.

[Hammer & Champy 93] Hammer, M. and Champy J. *Reengineering the Corporation.* Harper Business, 1993.

[Jorysz & Vernadat 90a] Jorysz, H.R. and Vernadat, F.B. , CIM-OSA Part 1: total enterprise modelling and function view, *International Journal of Computer Integrated Manufacturing*, 1990, Vol. 3, Nos. 3 and 4, pp. 144 - 156.

[Jorysz & Vernadat 90b]Jorysz, H.R. and Vernadat , CIM-OSA Part 2: information view, *International Journal of Computer Integrated Manufacturing*, 1990, Vol. 3, Nos. 3 and 4, pp. 157 - 167.

[Klittich 90] Klittich, M., CIM-OSA Part 3: CIM-OSA integrating infrastructure - the operational basis for integrated manufacturing systems, *International Journal of Computer Integrated Manufacturing*, 1990, Vol. 3, Nos. 3 and 4, pp. 168 - 180.

[Kim & Fox 94] Kim, H., and Fox, M.S., "Formal Models of Quality and ISO9000 Compliance: An Information Systems Approach", *Proceedings of the 48th Annual Quality Congress*, Milwaukee WI: American Society for Quality Control, pp. 17-23, 1994.

[Kim & Fox 95] Kim, H. and Fox, M.S. An Ontology of Quality for Enterprise Modelling, *Fourth Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises*, (West Virginia University 1995).

[Lenat & Guha 90] Lenat, D. and Guha, R.V. *Building Large Knowledge-based Systems: Representation and Inference in the CYC Project.* Addison Wesley, 1990.

[Martin & Smith 83] Martin, C., and Smith, S. *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 45433, 1983.

[Martin et al. 83] Martin, C., Nowlin, A., St. John, W., Smith, S., Ruegsegger, T., and Small, A. Integrated Computer-aided Manufacturing (ICAM) Architecture Part III/Volume VI: Composite Information Model of "Manufacture Product" (MFG1). Technical Report AFWAL-TR-82-4063 Voluem VI, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, 1983.

[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.

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

PostScript error (--nostringval--, findresource)