The Role of Competency Questions in Enterprise Engineering

Michael Grüninger and Mark S. Fox *

Department of Industrial Engineering, University of Toronto,
4 Taddle Creek Road, Toronto, Ontario M5S 1A4
{gruninger, msf }@ie.utoronto.ca

Abstract

We present a logical framework for representing activities, states, time, and cost in an enterprise integration architecture. We define ontologies for these concepts in first-order logic and consider the problems of temporal projection and reasoning about the occurrence of actions. We characterize the ontology with the use of competency questions. The ontology must contain a necessary and sufficient set of axioms to represent and solve these questions. These questions not only characterize existing ontologies for enterprise engineering, but also drive the development of new ontologies that are required to solve the competency questions.

1.0 Introduction

Market competition is forcing firms to reconsider how they are organized to compete. As a basis for change, they are exploring a variety of concepts, including Benchmarking, Time-based Competition, Quality Function Deployment, Activity-Based Costing, Quality Circles, Continuous Improvement, Process Innovation, and Business Process Re-Engineering. Regrettably, most of the concepts are descriptive, if not ad hoc, and lack a formal model which would enable their consistent application across firms. Consider business process re-engineering [Davenport 93], [Hammer & Champy 93]. It is very much in the "guild" mold of application; management consultants are the "masters" and they impart their knowledge through "apprenticeship" to other consultants. The knowledge of business process re-engineering has yet to be formalized and reduced to engineering practice.

The goal of the Enterprise Engineering Project at the University of Toronto is to:

• Formalize the knowledge found in Enterprise Engineering perspectives such as Benchmarking, Time-based Competition, Quality Function Deployment, Activity-Based Costing, Quality Circles, Continuous Improvement, Process Innovation, and Business Process Re-Engineering. By formalize, we mean the identification, formal representation and computer implementation of the concepts, methods and heuristics which comprise a particular perspec-

*. This research is supported, in part, by the Natural Science and Engineering Research Council, Digital Equipment Corp., Micro Electronics and Computer Research Corp., and Spar Aerospace.

tive. This not only enables a precise formulation of the intuitions implicit in practice, but it is also a step towards automating the execution of certain tasks involved in enterprise engineering.

• Integrate the knowledge into a software tool that will support the enterprise engineering function by exploring alternative organization models spanning organization structure and behaviour. The Enterprise Engineering system allows for the exploration of a variety of enterprise designs. The process of exploration is one of design, analysis and re-design, where the system not only provides a comparative analysis of enterprise design alternatives, but can also provide guidance to the designer.These ideas are formalized in the notion of advisors (cf. [Grüninger & Fox 94]) that are able to analyze, guide, and make decisions about the current enterprise and possible alternatives.

• Provide a means for visualizing the enterprise from many of the perspectives mentioned above. The process of design is performed through the creation, analysis and modification of the enterprise from within each of the perspective visualizations.

Enterprise modelling is an essential step in defining the tasks and functionality of the various components of an enterprise.The goal is to create generic, reusable representations of Enterprise Knowledge that can be applied across a variety of enterprises. Towards this end, the TOVE (Toronto Virtual Enterprise) ontology [Fox et al 93] has been developed and applied to enterprise engineering [Fox et al 94], enterprise integration, and integrated supply chain management. An ontology 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. The TOVE ontology currently spans knowledge of activity, time, and causality, resources, and more enterprise oriented knowledge such as cost, quality and organization structure. 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 ontology. We also present a set of tasks that arise in enterprise engineering and the requirements on any ontology that is used to represent the tasks and their solution. These requirements, which we call competency questions, are the basis for a rigorous characterization of the problems that the enterprise model is able to solve. The enterprise model must be able to represent the tasks specified by the competency questions and their solution. The questions are also those tasks for which the enterprise model finds all and only the correct solutions. Tasks such as these can serve to drive the development of new theories and representations and also to justify and characterize the capabilities of existing theories for enterprise modelling.

2.0 Common Sense Enterprise Modelling

The basic entities in the TOVE model are represented as objects with specific properties and relations. Objects are structured into taxonomies and the definitions of objects, attributes and relations are specified in first-order logic. An ontology is defined in the following way. We first identify the objects in our domain of discourse; these will be represented by constants and variables in our language. We then identify the properties of these objects and the relations that exist over these objects; these will be represented by predicates in our language.

We next define a set of axioms in first-order logic to represent the constraints over the objects and predicates in the ontology. This set of axioms constitutes a microtheory ([Lenat & Guha 90]) and provides a declarative specification for the various tasks we wish to model. Further, we need to prove results about the properties of our microtheories in order to provide a characterization and justification for our approach; this enables us to understand the scope and limitations of the approach. We use a set of problems, which we call competency questions, that serve to characterize the various ontologies and microtheories in our enterprise model. The microtheories must contain a necessary and sufficient set of axioms to represent and solve these questions, thus providing a declarative semantics for the system. It is in this sense that we can claim to have an adequate microtheory appropriate for a given task, and it is this rigour that is lacking in previous approaches to enterprise engineering.

The competency questions are generated by requiring that the ontologies and microtheories be necessary and sufficient to represent the tasks and their solutions for the various components of the system. Within enterprise engineering, these include:

• Temporal projection -- Given a set of actions that occur at different points in the future, what are the properties of resources and activities at arbitrary points in time? This includes the management of resources and activity-based costing (where we are assigning costs to resources and activities).To solve this problem, we 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 will use this task to characterize the ontologies in this paper.

• Planning and scheduling -- what sequence of activities must be completed to achieve some goal? At what times must these activities be initiated and terminated?

• Benchmarking -- Can activities from one enterprise be used in another while still satisfying the constraints that exist within the enterprise's environment and achieving the goals of the enterprise?

• Hypothetical reasoning -- what will happen if we move one task ahead of schedule and another task behind schedule? What are the effects on orders if we buy another machine?

• Execution monitoring and external events -- What are the effects on the enterprise model of the occurrence of external and unexpected events (such as machine breakdown or the unavailability of resources)?

• Time-based competition -- we want to design an enterprise that minimizes the cycle time for a product [Blackburn 91]. This is essentially the task of finding a minimum duration plan that minimizes action occurrence and maximizes concurrency of activities.

Claiming that any ontologies are adequate for enterprise modelling requires proving that the ontologies can represent and solve these competency questions.

3.0 Ontologies and Microtheories

In this section we present the ontologies and microtheories in TOVE for time, activity, and cost. These ontologies will then be used to specify the tasks addressed by the components of the enterprise engineering system; the final section of the paper will present the competency questions that serve to characterize the ontologies and microtheories.

3.1 Time and Action

The problem of benchmarking requires that an enterprise understand its processes and the processes of another enterprise in order to determine whether there are any comparable processes that can be adopted. This requires an adequate representation for processes. In the following sections, we present the TOVE ontology for activity, state, causality, and time, and define the semantics of the constructs in the ontology using the situation calculus.

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; the function $do(a,\sigma)$ is the name of the situation that results from performing action a in situation σ. There is a predicate $Poss(a,\sigma)$ that is true whenever an action a can be performed in situation σ. The structure of situations is that of a tree; two different sequences of actions lead to different situations. The tree structure of the situation calculus shows all possible ways in which the events of the future can unfold. Thus, each branch that starts in the initial situation can be understood as a hypothetical future.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 $actual(\sigma)$.

Situations are assigned different durations by defining the predicate $start(s,t)$. Each situation has a unique start time; these times begin at 0 in σ_0 and increase monotonically away from the initial situation.Time is represented as a continuous line on any branch in the tree of situations; 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'. Using this relation, we can define the temporal relations of [Allen 84] over intervals.

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 to represent is the occurrence of actions at points in time. To represent this we introduce two predicates: $occurs(a,\sigma)$ (action a occurs in situation σ), and $occurs_T(a,t)$ (action a occurs at time t) defined as follows:

$$occurs(a,\sigma) \equiv actual(do(a,\sigma)) \qquad \text{(EQ 1)}$$

$$occurs_T(a,t) \equiv occurs(a,\sigma) \wedge start(do(a,\sigma),t) \qquad \text{(EQ 2)}$$

3.2 Activities and States

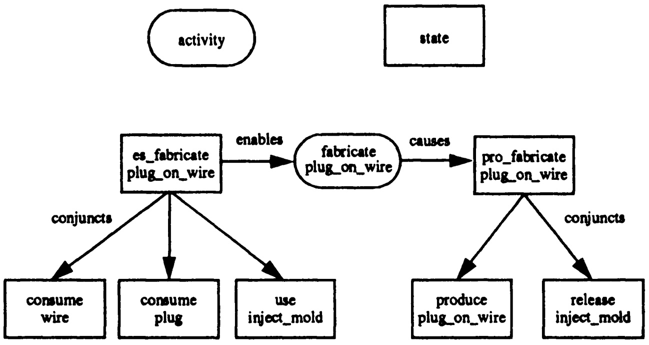
At the heart of the TOVE Enterprise Model lies the representation of an *activity* and its corresponding enabling and caused *states* ([Sathi et al. 85], [Fox et al 93]). In this section we examine the notion of states and define how properties of activities are defined in terms of these states. An activity 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 (see Figure 1).

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***.

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.

FIGURE 1 Activity-State Cluster



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.

3.3 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. For example, we present two of the successor axioms in the microtheory:

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:

$$(\forall\, s,a,e,\sigma)\ holds(status(s,a,committed), do(e,\sigma)) \equiv (e = commit(s,a) \wedge holds(status(s,a,possible),\sigma)) \vee \neg(e = enable(s,a)) \wedge holds(status(s,a,committed),\sigma) \qquad \text{(EQ 3)}$$

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:

$$(\forall\, s,a,e,\sigma)\ holds(status(s,a,enabled), do(e,\sigma)) \equiv (e = enable(s,a) \wedge holds(status(s,a,committed),\sigma)) \vee \neg[(e = complete(s,a) \vee e = disenable(s,a)) \wedge holds(status(s,a,enabled),\sigma)] \qquad \text{(EQ 4)}$$

Using the successor state axioms, we can derive occurrence axioms that make the relationship between the occurrence of the actions that change the status of a state and the preconditions for these actions:

$$(\forall\, s,a,\sigma)\ occurs(commit(s,a),\sigma) \supset holds(status(s,a,possible),\sigma) \qquad \text{(EQ 5)}$$

$$(\forall\, s,a,\sigma)\ occurs(enable(s,a),\sigma) \supset holds(status(s,a,committed),\sigma) \qquad \text{(EQ 6)}$$

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 sentence can be derived from the axioms:

$$(\forall\, s,a,\sigma)\ occurs(enable(s,a),\sigma) \supset (\exists\sigma')\ occurs(commit(s,a),\sigma') \qquad \text{(EQ 7)}$$

Similarly, the precondition for the *commit* action is that the state be *possible*. In [Fadel et al. 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.

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

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

$$(\forall\, s,s_1,\ldots,s_n,a,\sigma)\ disjunctive(s,a) \wedge substate(s_1,s) \wedge \ldots \wedge substate(s_n,s) \supset occurs(enable(s,a),\sigma) \blacksquare occurs(enable(s_1,a),\sigma) \vee \ldots \vee occurs(enable(s_n,a),\sigma) \qquad \text{(EQ 8)}$$

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:

$$(\forall\, s,s_1,\ldots,s_n,a,\sigma)\ conjunctive(s,a) \wedge substate(s_1,s) \wedge \ldots \wedge substate(s_n,s) \supset occurs(enable(s,a),\sigma) \blacksquare occurs(enable(s_1,a),\sigma) \wedge \ldots \wedge occurs(enable(s_n,a),\sigma) \qquad \text{(EQ 9)}$$

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.

3.5 Ontology of Cost

The ontology for activity-based costing is a formal specification of the assignment of costs to activities based on costs for the resources utilized by these activities [Tham et al. 94]. Each resource is assigned a unique cost depending on the status of its terminal state; these are represented by the predicates *committed_res_cost_unit(a,r,q,v), enabled_res_cost_unit(a,r,q,v), disenabled_res_cost_unit(a,r,q,v), reenabled_cost_unit(a,r,q,v)*, for some activity *a* and resource *r*. The parameter *v* represents the cost metric for a unit *q* of the resource. It is assumed that the values for these costs are completely known and that they are unique. Based on the duration of a particular status value, the axioms in the ontology of cost assign a unique cost for the state at a point in time. The cost assigned to an activity at a point in time is the aggregation of the costs for the states of the activity at that point. In this sense, the task addressed by the ontology of activity-based costing is a special case of temporal projection. We thus use successor state axioms similar to those in earlier sections. For example, we have the following successor axiom for computing the cost associated with the enabled status of a terminal state, where t,t' are the endpoints of the interval over which the state is enabled:

$$(\forall a,r,s,t,t',c,c')\ \mathit{holds}(\mathit{enabled\_res\_cost}(s,r,a,c'), \mathit{do}(e,s)) \equiv (e = \mathit{disenable}(s,a) \vee e = \mathit{complete}(s,a)) \wedge \mathit{enabled\_res\_cost\_unit}(r,a,q,v) \wedge \mathit{holds}(\mathit{enabled\_res\_cost}(s,r,a,c), \sigma) \wedge c' = c + vq(t'-t) \vee \neg[(e = \mathit{complete}(s,a) \vee e = \mathit{disenable}(s,a)) \wedge \mathit{holds}(\mathit{enabled\_res\_cost}(s,r,a,c'), \sigma)] \tag{EQ 10}$$

Given the costs computed for each status of a state, the resource cost point (represented by the predicate *cpr*) is computed by summing the costs for each status value of the state:

$$(\forall a,r,s,t,c,c_1,c_2,c_3,c_4)\ \mathit{holds}_T(\mathit{cpr}(s,a,r,c), t) \equiv \mathit{holds}_T(\mathit{committed\_res\_cost}(s,a,r,c1) \wedge \mathit{holds}_T(\mathit{enabled\_res\_cost}(s,a,r,c1) \wedge \mathit{holds}_T(\mathit{disenabled\_res\_cost}(s,a,r,c1) \wedge \mathit{holds}_T(\mathit{reenabled\_res\_cost}(s,a,r,c1) \wedge c = c1+c2+c3+c4 \tag{EQ 11}$$

The cost for an activity at a point in time is the sum of the costs for each of its resources; this is represented by the predicate $\mathit{cpa}(a,c)$.

The ontology for activity-based costing therefore consists of resource cost units, successor state axioms, and axioms defining the aggregation of costs for resources, activities, and orders.

3.6 Competency Questions for Ontologies

In this section we rigorously specify several of the tasks that the various advisors must solve, and claim that the ontologies and microtheories presented earlier in this paper are necessary and sufficient to represent these tasks and their solutions. We can express these as the following theorems; let T_{succ} be the set of successor axioms and let $T_{occurrence}$ be a complete specification of action occurrences and the times at which the actions occurred.

Theorem 1: At any time point *t*, state *s*, and activity *a* there exists a status value *X* such that

$$T_{succ} \cup T_{occurrence} \models \mathit{holds}_T(\mathit{status}(s,a,X), t)$$

In other words, the status of a state is completely determined at any point in time.

Let T_{cost} be the set of successor axioms for cost and the complete set of resource cost units for every resource, activity, and status value.

Theorem 2: At any time point *t*, state *s*, resource *r* and activity *a* there exists a cost *c* and a cost c′ such that

$$T_{succ} \cup T_{occurrence} \cup T_{cost} \models \mathit{holds}_T(\mathit{cpr}(s,a,r,c), t) \wedge \mathit{holds}_T(\mathit{cpa}(a,c'), t)$$

Thus the costs assigned to a resource and activity are completely determined at any point in time.

We can further show that the axioms are necessary and sufficient to prove these theorems in the sense that if any of the axioms are removed then we can no longer prove the theorem. Thus these temporal projection problems serve as benchmarks for any theories of processes and activity-based costing.

Competency questions can also serve to drive the development of appropriate microtheories. For example, the goal of time-based competition is to find the enterprise model with the minimum cycle time. Within the ontology of activity, this is equivalent to finding the ordering of activities with the minimum duration. The first step in solving this task is to define the conditions under which a set of activities may be completely assigned a unique minimum duration; this competency question serves a characterization for any theory of time-based competition. In order to do this, we must also define the conditions for the existence of bottlenecks and other limitations of concurrency within an enterprise model, such as computing the maximum number of activities that may be supported by a resource. This in turn provides a competency question for the ontology of resources in [Fadel et al. 94].

4.0 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 enterprise engineering. To this end, we have defined the TOVE ontologies for activities, states, time, and cost 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 aggregate activities, and the role of temporal structure in this aggregation, is accomplished through axioms that allow us to reason about the occurrence of actions.

Competency questions are used to characterize each of the ontologies and microtheories; these questions present tasks such that the microtheories are a necessary and sufficient set of axioms for representing and solving these tasks. Furthermore, the use of competency questions serves two roles -- they characterize the ontologies and microtheories that have been designed for each task and they also provide direction for the development of new ontologies and microtheories.

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.

5.0 References

[Allen 83] Allen, J.F. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*. 26:832-843, 1983.

[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.

[Fadel et al. 94] Fadel, F. , Fox, M.S., Grüninger, M. A generic enterprise resource ontology.*Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Morgantown, West Virginia, 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 94]Fox, M. S., Grüninger, M., Zhan, Y.. Enterprise engineering: An information systems perspective. *Proceedings of the Industrial Engineering Research Conference 1994*).

[Grüninger & Fox 94] Grüninger, M. and Fox, M.S. An advisor-based architecture for enterprise engineering. *Workshop on Artificial Intelligence in Business Process Reengineering*, AAAI 94, Seattle.

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

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

[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., Grüninger, M. A cost ontology for enterprise modelling. *Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Morgantown, West Virginia, 1994.