The Role of Competency Questions in Enterprise Engineering[1]

Michael Gruninger and Mark S. Fox [2]

Department of Industrial Engineering, University of Toronto,
4 Taddle Creek Road, Toronto, Ontario M5S 1A4
tel:1-416-978-6823 fax:1-416-971-1373 internet:{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. As such, they serve as benchmarks for the development of ontologies for the various tasks in enterprise engineering. These questions not only characterize existing ontologies but also drive the development of new ontologies that are required to solve the competency questions.

Key Words: Enterprise modelling, process engineering, competency

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 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.

---

1. Submitted to IFIP WG5.7 Workshop on Benchmarking - Theory and Practice, Trondheim, Norway, 1994.

2. 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.

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

- Formalize the knowledge found in Enterprise Engineering perspectives such as 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 perspective. 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.

- 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.

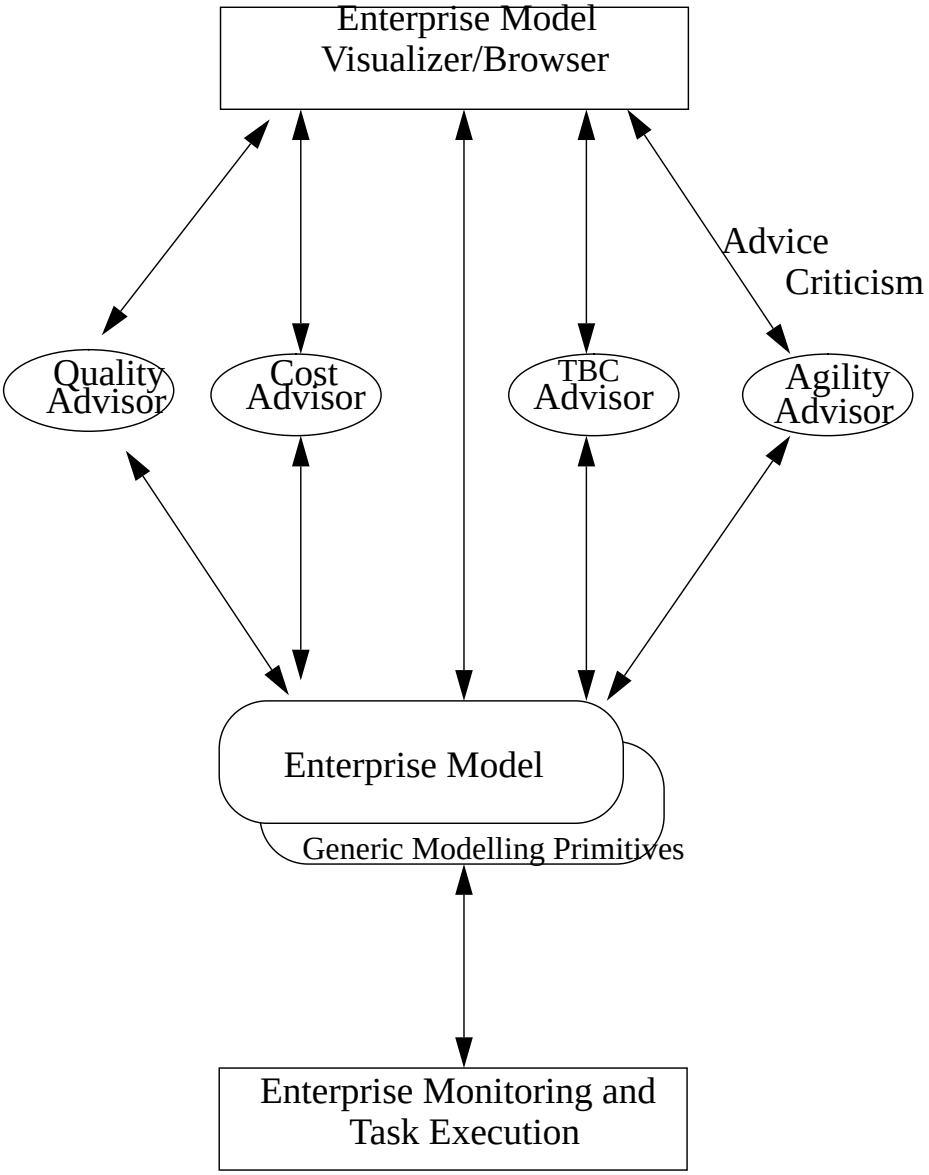
Enterprise modelling is an essential step in defining the tasks and functionality of the various components of an enterprise.The goal is to create generic, reusable representations of Enterprise Knowledge that can be applied across a variety of enterprises. Towards this end, the TOVE (Toronto Virtual 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, providing a new approach to benchmarking as applied to enterprise modelling and business process engineering. Competency questions are the benchmarks in the sense that the enterprise model

is necessary and sufficient to represent the tasks spcified by the competency questions and their solution. They 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 Architecture for Enterprise Engineering

The Enterprise Engineering system is composed of four main components: its common-sense enterprise model, advisors, visualization, and information agents (see Figure 1).



The foundation for the system is the common-sense enterprise model. It provides a set of generic reusable representations of enterprise knowledge. This includes representations for processes, activities, time, causality, resources, quality, and cost. The enterprise model is used by all other components of the system by providing a shared terminology and set of constraints.

Various perspectives exist in an enterprise, such as efficiency, quality, and cost. Any system for enterprise engineering must be capable of representing and managing these different perspectives in a well-defined way. These ideas are formalized in the notion of advisors that are able to analyze, guide, and make decisions about the current enterprise and possible alternatives.

Lastly, there is an execution environment where the portions of the enterprise design may be, i.e., those portions that define databases and machine executable activities, down-loaded for execution by the run-time system.

3.0 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:

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

- 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).

- 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)?

- 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?

- 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.

The primary task which arises in this paper is that of temporal projection in an enterprise, which 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 interval over which the state has a certain status is bounded by the times at which the appropriate actions that change status occur. This interval 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.

4.0 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.

4.1 Time

In this section we define the ontology of time and action that is used throughout the this paper. 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. Further, we impose a structure over these intervals that is isomorphic to the natural numbers by introducing the notion of successor situation [Reiter 91]. The function *do(a,σ)* 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.

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:

$$(\forall\ \alpha,\sigma,\sigma') \neg (\sigma < \sigma' < do(a,\sigma)) \qquad \text{(EQ 1)}$$

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.

To define the evaluation of the truth value of a sentence at some point in time, we will use the predicate *holds(f,σ)* 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,σ)* and $occurs_T(a,t)$, defined as follows:

$$occurs(a,\sigma) \equiv \sigma_0 < do(a,\sigma) \quad \text{(EQ 2)}$$

$$occurs_T(a,t) \equiv occurs(a,\sigma) \wedge start(do(a, \sigma), t) \quad \text{(EQ 3)}$$

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

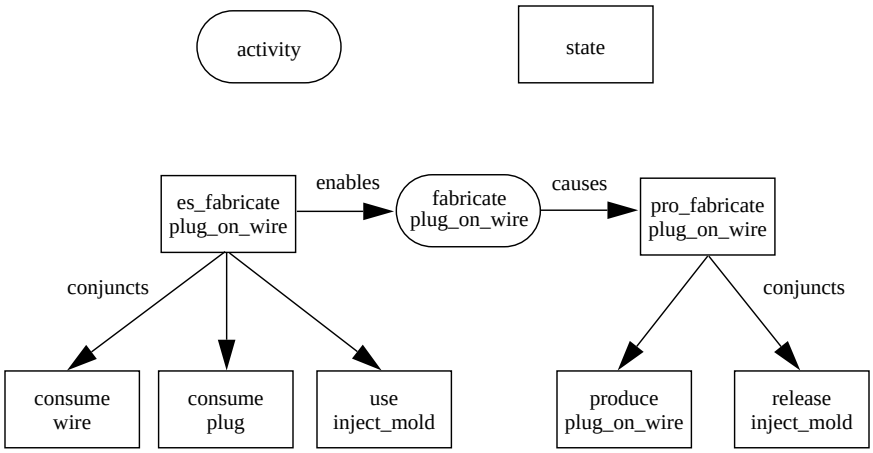
4.2 Activities and States

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*.

FIGURE 1 Activity-State Cluster



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.

4.3 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. 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.

$$(\forall\ s,a,e,\sigma)\ holds(status(s,a,committed), do(e,\sigma)) \equiv (e = commit(s,a) \wedge holds(status(s,a,possible),\sigma)) \vee \neg(e=enable(s,a)) \wedge holds(status(s,a,committed),\sigma) \quad \text{(EQ 4)}$$

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.

$$(\forall\ s,a,e,\sigma)\ holds(status(s,a,enabled), do(e,\sigma)) \equiv (e = enable(s,a) \wedge holds(status(s,a,committed),\sigma)) \vee \neg[(e=complete(s,a) \vee e=disenable(s,a)) \wedge holds(status(s,a,enabled),\sigma)] \quad \text{(EQ 5)}$$

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:

$$(\forall\ s,a,\sigma)\ occurs(commit(s,a),\sigma) \supset holds(status(s,a,possible),\sigma) \quad \text{(EQ 6)}$$

$$(\forall\ s,a,\sigma)\ occurs(enable(s,a),\sigma) \supset holds(status(s,a,committed),\sigma) \quad \text{(EQ 7)}$$

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:

$$(\forall\ s,a,\sigma)\ occurs(enable(s,a),\sigma) \supset (\exists\sigma')\ occurs(commit(s,a),\sigma') \quad \text{(EQ 8)}$$

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.

4.4 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.

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

$$(\forall\ s,s_1,...,s_n,a,\sigma)\ disjunctive(s,a) \wedge substate(s_1,s) \wedge ... \wedge substate(s_n,s) \supset occurs(enable(s,a),\sigma) \equiv$$
$$occurs(enable(s_1,a),\sigma) \vee ... \vee occurs(enable(s_n,a),\sigma) \quad \text{(EQ 9)}$$

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:

$$(\forall\ s,s_1,...,s_n,a,\sigma)\ conjunctive(s,a) \wedge substate(s_1,s) \wedge ... \wedge substate(s_n,s) \supset occurs(enable(s,a),\sigma) \equiv$$
$$occurs(enable(s_1,a),\sigma) \wedge ... \wedge occurs(enable(s_n,a),\sigma) \quad \text{(EQ 10)}$$

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

$$(\forall\ s,s_1,a,\sigma)\ not(s,a) \wedge substate(s_1,s) \supset occurs(enable(s,a),\sigma) \equiv \neg\ occurs(enable(s_1,a),\sigma) \quad \text{(EQ 11)}$$

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

4.5 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

$$(\forall\ a,s,t,t',d)\ state\_duration(s,d) \equiv occurs_T(enable(s,a),\ t) \wedge occurs_T(complete(s,a),\ t') \wedge d = t - t' \quad \text{(EQ 12)}$$

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

$$(\forall\ a,s,t,t',d)\ committed\_duration(s,d) \equiv occurs_T(commit(s,a),\ t) \wedge occurs_T(complete(s,a),\ t') \wedge d = t - t' \quad \text{(EQ 13)}$$

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. In turn, this task plays an important role in time-based competition, since it identifies bottlenecks within an enterprise model and characterizes the limitations of concurrency within the model.

4.6 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 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:

$$(\forall\ a,r,s,t,t',c,c')\ holds(enabled\_res\_cost(s,r,a,c'), do(e,s)) \equiv (e= disenable(s,a) \vee e=complete(s,a)) \wedge enabled\_res\_cost\_unit(r,a,q,v) \wedge holds(enabled\_res\_cost(s,r,a,c)\ , \sigma) \wedge c' = c+vq(t'-t) \vee \neg[(e=complete(s,a) \vee e=disenable(s,a)) \wedge holds(enabled\_res\_cost(s,r,a,c'), \sigma)] \quad \text{(EQ 14)}$$

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:

$$(\forall\ a,r,s,t,c,c_1,c_2,c_3,c_4)\ holds_T(cpr(s,a,r,c),\ t) \equiv holds_T(committed\_res\_cost(s,a,r,c1) \wedge holds_T(enabled\_res\_cost(s,a,r,c1) \wedge holds_T(disenabled\_res\_cost(s,a,r,c1) \wedge holds_T(reenabled\_res\_cost(s,a,r,c1) \wedge c = c1+c2+c3+c4 \quad \text{(EQ 15)}$$

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 *cpa(a,c)*.

5.0 Advisors

The best enterprise design is one that optimises each of the perspectives that exist in the enterprise. Examples of enterprise perspectives include: Quality, Cost, Efficiency, Incentives, and Agility. We are developing for each perspective a theory of design that results in the optimization of the perspective. The theory incorporates the ability to measure a partial/complete design and to guide the designer in the decision making. One issue is whether there exists sufficient knowledge of the process of designing and optimizing business activities/processes to incorporate in knowledge-based tools.

To formalize the intuition of design perspectives, we introduce the notion of advisors. An advisor is an encapsulation of one or more micro-theories. The tasks for each advisor fall into three main groups: evaluation, analysis, and guidance. We are currently constructing advisors for Time-based Competition, Activity-based Costing, Agility, and ISO9000 compliance; we anticipate advisors for goals and objectives, organization structure, incentives, and culture. In each case, we are defining the tasks, purpose, and responsibilities of the advisor, and represent these tasks using the appropriate ontology. As with the competency questions in the preceding section, each advisor is rigorously characterized by these tasks, including a specification of what an advisor is analyzing and in what way that they guide (propose different alternatives). This in turn provides a set of competency questions for the ontologies and microtheories in the previous section..

5.1 Functionality of Advisors

A necessary first step is the precise definition of the tasks performed by different advisors in the system and the ways in which they interact. This specification is independent of the algorithms used to solve the tasks - we are specifying the problem and what constitutes a solution to the problem. In this way we define the functionality of each advisor; this will require the definition of what is the appropriate input to each advisor and what is the correct output. The specifications of these tasks for the advisors will serve as competency questions for the different ontologies and microtheories that are being designed. Thus the design of the advisors may lead to extensions of existing ontologies and microtheories and will also serve to justify the axioms. In this section, we informally present the tasks for the different advisors; in the following section we formally specify some of these tasks and the associated competency questions.

The specification of the analysis tasks for the advisors is complicated by the rich interaction among the ontologies and microtheories of the relevant advisors. To represent this interaction, each advisor is a constraint-based problem solver - given a set of goals and constraints, an advisor searches for a solution that optimizes the goals and satisfies the constraints. Advisors also have the ability to generate more than one solution, thereby the enabling the consideration of alternatives and trade-offs. The goals and constraints for an advisor are represented by sentences in the microtheories associated with the advisor; satisfying the constraint is equivalent to finding a satisfying interpretation for the microtheory. Advisors interact by posting new constraints that must be satisfied.

The specification of evaluation tasks for each advisor is the first step. Evaluation tasks will either be decision tasks (does the enterprise model satisfy some set of requirements, such as ISO 9003 compliance) or property evaluation of an enterprise model (what is the cost associated with some set of activities). This will also require the ability to compare two different enterprise models along some dimension, such as cost or quality. In addition, the distinction must be made between evaluation of the enterprise model (static set of acivities) and the evaluation of a plan or schedule at some point in time.

Analysis tasks involve prediction, monitoring, identification, and explanation with respect to an enterprise model. Prediction is the determination of the value of some proposition at points in the future. Monitoring is the determination of the value of some proposition after executing some set of activities, and comparing this value to the predicted value. Identification is the task of finding objects that satisfy certain properties in an enterprise model. Explanation is the task of determining why a proposition has a certain value at some point in time; this requires deciding what set of event occurred and what propositions hold that entail the value of the proposition in question. For example, we may want to predict the cost associated with some set of activities and then monitor

and compare the cost of the execution of the scheduled activities. We may want to know why a particular product has a given cost, or why the activity took so long to complete; these tasks require some mechanism for explanation. Another analysis task is the identification of the resource bottlenecks within an enterprise model, or the anticipation of resource conflicts.

The explanation tasks illustrate the relationship between evaluation and guidance for advisors. If a particular enterprise model fails to satisfy some property, we may want to know why it fails. This in turn suggests ways in which we may augment the enterprise model so that it does satisfy the property. For example, an enterprise may not be ISO 9003 compliant, a decision task of the quality advisor; the explanation task would recommend that the appropriate quality control processes be included in order to satisfy compliance.

Finally we have the guidance tasks for these advisors, in which the advisor suggests alternatives. Thus the advisor must be able to represent and model the current status of a process and assess potential changes. For static models, this requires the ability to generate different models. This is also related to the evaluation tasks of the advisors; if a process fails to satisfy certain requirements, the advisor should be able to suggest alternative models of the process which do satisfy the requirements. If we are to compare and evaluate the different alternative futures and possibilities for the processes in an enterprise with respect to the execution of plans and schedules, this will require some mechanism for hypothetical reasoning.

Consider the time-based competition advisor. The modelling task provides ontologies that can be used to construct a model of the activities of a process, temporal relations over these activities, and constraints on the usage of resources by the activities. Based on these models the advisor provides tools to design, analyze and evaluate the enterprise from the perspective of optimising efficiency. For example, it can perform a critical path analysis of an activity graph or process, or it may simulate a process if more complex activity behaviours are involved. The advisor must also be able to represent and model the current status of the process and assess potential changes. This is essential if the advisor is required to guide the designer by presenting alternatives. For example, we may need to know if a process would be more efficient given one ordering of activities rather than another. This may entail identifying the resources that prevent activities from being performed concurrently and thus anticipate resource conflicts that lead to bottlenecks.

Now consider the notion of a ISO9000 advisor, which uses a microtheory of ISO 9003 compliance [Kim & Fox 93]. This microtheory introduces axioms to represent the ISO 9003 requirements and axioms defining how an organization can be ISO 9003 compliant. The primary decision-making capability of the quality advisor is therefore determining whether an organization is ISO 9003 compliant.

The ISO 9003 requirements can be divided into those that can be met by just one process (locally compliant requirements) and those that require several or all processes of an enterprise (globally compliant requirements). For example, to satisfy ISO 9003 local compliance, there must exist processes that perform product identification, inspection and testing, identify test status, control nonconformity, and arrange for handling of products; this is an axiom in the microtheory. An advisor can use this axiom in several different ways. It can be used to analyze a process within the enterprise and decide compliance by verifying the existence of the necessary processes. It can also be used by a designer by recommending the appropriate quality control processes that must be included in order to satisfy local compliance.

5.2 Competency Questions for Advisors

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

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

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

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

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 benchmark for any time-based competition advisor. In order to do this, we must also characterize the bottlenecks and other limitations of concurrency within an enterprise model, that is, compute 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 94].

6.0 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.

In order to integrate this knowledge into a software tool that will support enterprise engineering functions, we introduced the notion of an advisor as a formalization of the different perspectives that we have with respect to an enterprise.

Competency questions are used to characterize each of the ontologies and microtheories used in the advisors. These questions present tasks such that the microtheories are a necessary and sufficient set of axioms for representing and solving these tasks. As such, these competency questions serve as benchmarks for any ontology for the task. Furthermore, the use of competency questions serves two roles -- they characterize the ontologies and microtheories that have been designed for each advisor 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. The formalization of activities is being extended to handle concurrent activities and reasoning about the availability and capacity of resources as part of the time-based competition advisor.

7.0 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.

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

[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., Gruninger, M., Zhan, Y.. Enterprise engineering: An information systems perspective (to appear in *Proceedings of the Industrial Engineering Research Conference 1994*).

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

[Kim & Fox 93] Kim, H. and Fox, M.S. Quality Systems Modelling: A Prospective for Enterprise Integration, *Fourth Annual Meeting of the Production and Operations Management Society.* 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 94] Tham, D. A cost ontology for enterprise modelling (submitted).