Submitted to CoopIS-94 AREA: RESEARCH

Ontologies for Enterprise Integration

Mark S. Fox and Michael Gruninger

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:{msf, gruninger}@ie.utoronto.ca

Abstract

We present a logical framework for representing the agents in two different cooperating information systems. We define an architecture for Integrated Supply Chain Management in which the supply chain is managed by a set of intelligent agents responsible for one or more activities. We also define an architecture for an enterprise engineering system that allows the exploration of a variety of enterprise designs. We introduce the notion of an advisor as a formalization of the different perspectives that we have with respect to an enterprise. By representing activities in both architectures as sets of first-order axioms in a microtheory, the tasks for the different agents in the architecture can be represented as finding satisfying interpretations of the constraints.

1.0 Introduction

A necessary first step in the design of cooperative information systems is the precise definition of the tasks performed by different components 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. This requires the development of a formal representation, or ontology, that is adequate for the specification.

In this paper we present two projects in the Enterprise Integration Laboratory at the University of Toronto and the representation necessary to define the tasks of the various components in each project. These projects are Integrated Supply Chain Management ([Fox et al 93b]) and Enterprise Engineering ([Fox et al 93c]). In each case, the execution model is based on the Enterprise Management Network [Roboam & Fox 92], and is called the Enterprise Information Architecture (EIA). The execution environment is composed of a set of functional and information (software)

agents, each of which performs continuously and autonomously. Functional agents perform activities, and information agents manage the distribution and consistency of information. For each of the above projects, we give a formal specification of the agents and their capabilities.

The supply chain is a set of activities which span enterprise functions from the ordering and receipt of raw materials through the manufacturing of products through the distribution and delivery to the customer. We view the supply chain as being managed by a set of intelligent agents, each responsible for one or more activities in the supply chain, and each interacting with other agents in the planing and execution of their responsibilities.

Enterprise engineering is concerned with the design and execution of enterprises. The goal of the enterprise engineering project is to formalize the knowledge required for business process reengineering ([Davenport 93], [Hammer & Champy 93]) and create an environment that facilitates the application of this knowledge to a particular company. First, we must formally represent the knowledge found in enterprise engineering perspectives such as efficiency, activity-based costing, quality, agility, and resource management. We must then integrate the knowledge into a software tool that will support the enterprise engineering function by exploring alternative organization models spanning organization structure and behaviour, analyzing each alternative, provide guidance to the designer, and automatically execute some task. Each perspective is implemented as an agent, which we call an advisor, in the enterprise integration architecture.

Enterprise execution focuses on the implementation of an enterprise design. In particular, it is concerned with both monitoring the performance of enterprise as specified by the model, and executing tasks that can be automated. Given an enterprise design, we should be able to deduce what the organization structure of the enterprise information system should be, the functional agents and their assigned activities, and how information is to be distributed and maintained across the network in order to support the functional agents.

In order to support the integration of supply chain agents and enterprise engineering perspectives, it is necessary for there to exist a shareable representation of knowledge about the enterprise that each agent can jointly understand and use and that minimizes ambiguity in communication. The enterprise model must also support deductive query processing. In this paper, we will first present the ontologies and theories that are necessary to specify the agents in each project, and then examine each project in detail.

2.0 Common Sense Enterprise Modelling

Enterprise modelling is an essential component in defining an enterprise.The goal of our enterprise modelling research is to create a generic, reusable representations of Enterprise Knowledge that can be reused across a variety of enterprises. Towards this end, we have been developing the TOVE enterprise ontology [Fox et al 93]. 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. TOVE provides a rich and precise representation of generic knowledge, such as, activities, processes, resources, time, and causality, and of more enterprise oriented knowledge such as cost, quality and organization structure.

The basic entities in our model are represented as objects with specific properties and relations. Objects are structured into taxonomies. Definitions of objects, attributes and relations are specified in first-order logic, where possible. We then define an ontology 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.

Intuitively, the axioms in the microtheory enable the model to deduce answers to questions that one would normally assume can be answered if one has a “common-sense” understanding of the enterprise. To formalize this intuition we also 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. 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 and integrated supply chain management.

The functional specifications of the agents in the supply chain and advisors in the enterprise engineering environment will serve as competency questions for the ontologies and microtheories which we will be presenting. The ontologies must be able to represent the tasks for the agents and also specify what constitutes a solution for these tasks.

2.1 Activities, States, and Time

The formalization of the notion of process and activity is crucial in any attempt at representing an enterprise or supply chain. Activities are the basic events that specify a transformation on the world [Fox et al. 93]. States specify what must be true for an activity to be performed, and what is true once the activity is completed. Activities are initiated at points in time, and once initiated, they have duration over some interval of time. Further, properties of states hold over the duration of these activities.

There are four kinds of states: *use(s,r,a), consume(s,r,a), release(s,r,a), produce(s,r,a)*. These predicates relate the state *s* with the resource *r* required by the activity *a*. Intuitively, a resource is used by an activity if none of the properties of the 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.

States are assigned a status, defined by the following predicates: *possible(s,r,a), committed(s,r,a), enabled(s,r,a), completed(s,r,a)*. A set of actions is defined that change the status of a state; these actions are parametrized by the state and the activity, and have the form *commit(s,r,a), enable(s,r,a), complete(s,r,a), disenable(s,r,a)*.

We use the extended situation calculus in [Pinto and Reiter 93] to represent change and time. All actions occur in situations, and the start of a situation is assigned a time by the function *start(*σ*)*. We use the predicate *holds(f,* σ*)* to represent that the property *f* is true in situation σ. For each action we have a set of effect axioms that define the changes caused by the actions. We also introduce the predicate *occurs(a,*σ) to denote an action *a* that occurs in situation σ; the predicate $occurs_T(a,t)$ represents an action *a* that occurs at time *t*.

Essentially, 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 existential sentence of the form:

$occurs(a,\sigma) \equiv (\exists\ \sigma_1,...,\sigma_n,t_1,...t_n)\ occurs(enable(s_1,r_1,a),\ \sigma_1) \wedge ... \wedge occurs(enable(s_k,r_m,a),\ \sigma_i) \wedge occurs(complete(s_1,r_1,a),\ \sigma_{i+1}) \wedge ... \wedge occurs(complete(s_k,r_m,a),\ \sigma_n) \wedge occurs_T(enable(s_1,r_1,a),\ t_1) \wedge ... \wedge occurs_T(enable(s_k,r_m,a),\ t_i) \wedge occurs_T(complete(s_1,r_1,a),\ t_{i+1}) \wedge ... \wedge occurs_T(complete(s_k,r_m,a),\ t_n)$

A schedule will consist of a set of activities with an ordering over the situations and times in which the actions occur. Note that the specification of the activity does not place any constraints on this ordering; as we will see in the next section, these constraints are posted by the different agents or advisors in the system.

We can represent the nondeterministic choice of some resource in a set of resources $r_1,...,r_n$ for an activity by the sentence

$$occurs(enable(s_1,r_1,a),\sigma) \vee ... \vee occurs(enable(s_n,r_n,a),\sigma)$$

Using this representation of activities, we can predict what properties of the world must be true at some point in a plan or schedule; this can be used when monitoring the execution of the schedule to determine whether the schedule must be modified in the face of unexpected events.

2.2 Resources

All activities require that some objects be available at the time that the activity is performed; this is the motivation for a theory of resources. The various properties that are axiomatized in the microtheory include resource commitment and the availability of resources [Fadel et al 93].

Resource requirements for activities over some time interval starting at t and ending at t' are represented by the predicates $use\_spec(r,a,q,t,t')$ and $consume\_spec(r,a,q,t,t')$, where q is the quantity of the resource r that is used or consumed by the activity a. It is assumed that for any enterprise, the resource requirements for all activities are completely determined.

The microtheory of resources represents the quantity q of a resource r with the predicate $rp(r,q)$. Thus the quantity of a resource in some situation σ is represented by $holds(rp(r,q),\ \sigma)$. The amount q of a resource r committed to an activity a between times t and t' is represented by the predicate

committed_to(r,a,q,t,t′). The total amount of a resource committed to all activities at time *t* is represented by the predicate *total_committed(r,q,t)*. These predicates are changed as effects of the actions *commit*, *enable*, and *complete*.

One of the central problems in the design of an enterprise or schedule is deciding whether a resource can support multiple activities that must execute over the same time interval, including the interaction between preconditions of activities preventing them from executing concurrently. This is determined by the axioms defining the predicate *available_for(a,r,q,t,t′)* [Fadel et al 93]. These axioms in the microtheory for resources represent additional constraints that must be placed on the ordering of situations and starting times in the definition of an activity.

2.3 Quality

The work in this domain is concerned with creating a terminology that spans quality concepts found in ISO9000, Baldridge Award, etc. [Kim & Fox 93]. In particular, it uses a microtheory of ISO 9003 compliance. 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 as stated by the axioms in the microtheory.

2.4 Activity-based Costing

The goal of this ontology is to formalize the concepts found in activity-based costing. Given a set of activities and the resources required by these activities, we must be able to assign costs based on the usage and consumption of these resources.

For every resource and status of a state associated with the resource, we have complete knowledge of the cost accrued by using or consuming the resource as a function of time. Additional axioms compute the cost associated with an activity by summing the costs for each state and each status over the time in which the activity was executing. If we know the starting times for each action in an activity that changes the status of a state, then the axioms of the cost microtheory uniquely determine the cost assigned to the activity.

3.0 The Architecture of ISCM

The Integrated Supply Chain Management (ISCM) is composed of a set of cooperating agents, where agent performs one or more supply chain management function, and coordinates its decisions with other relevant agents. Each functional agent is responsible for the planning and control of a set of activities in the supply chain. Information agents support other agents by providing information and communication services.

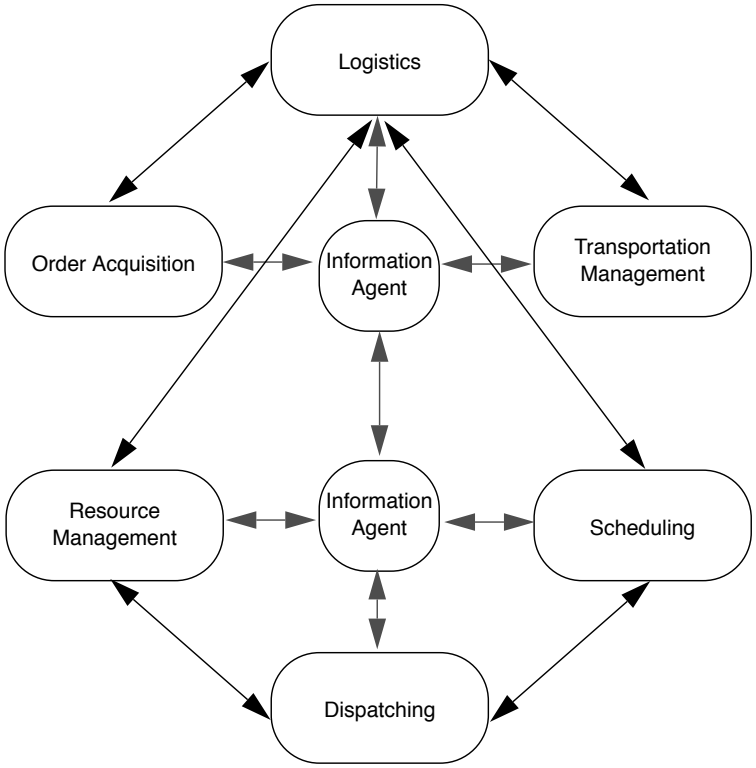
The decomposition of supply chain functions and their allocation to agents represents one of the first tasks in the project. The problem is that existing decompositions of functions, as found in MRP systems today, arose out of organizational constraints, legacy systems, and limitations on algorithms. We are currently working on five functional agents: Logistics, Transportation Management, Resource Management, Scheduling and Dispatching.

Agents are constraint-based problem solvers: given a set of goals and constraints, they search for a solution that optimizes the goals and satisfies the constraints. Agents also have the ability to generate more than one solution, thereby the enabling the consideration of alternatives and tradeoffs by a set of cooperating agents. The goals and constraints for an agent are represented by sentences in the microtheories associated with the agent; satisfying the constraint is equivalent to finding a satisfying interpretation for the microtheory. Agents communicate by posting new constraints that must be satisfied. Coordination occurs when agents not only satisfy their own internal constraints but also the constraints of other agents. Negotiation occurs when constraints that cannot be satisfied are modified by the subset of agents directly concerned.

All agents exist within an Enterprise Information Architecture (EIA) that provides a distributed information environment. Figure 1 shows the relationship among the various functional and information agents in the EIA. We will now examine the functional agents in more detail.

Information agents are responsible for maintaining a consistent form of shared information among agents, aggregating information to produce reports and answer queries, propagating changes in the state of the modeled enterprise over the models of various agents, and resolving inconsistencies that arise during agent interaction.

FIGURE 1. The ISCM agents



3.1 Functional Agents

Given the representation of activities and time in our microtheories, we view the planning/scheduling function as the computational core of the integrated supply chain. In fact, scheduling is done at several levels by the Logistics, Scheduling, Dispatching, and Transportation agents. In this section we examine in detail the tasks of each of these supply chain agents.

Scheduling agent. This agent is responsible for scheduling and rescheduling activities in the factory and exploring hypothetical "what-if" scenarios for potential new orders.

Recall that activities are represented as existential sentences and other sets of axioms that represent additional constraints. A schedule is constructed by combining the sentences that define the activities, and then specifying the ordering over the situations and starting times for actions in the activity definitions so that the goals of the logistics agent are satisfied. In addition, the specification of an activity may include nondeterministic actions, such as selecting a machine from some set of machines to be used by an activity. To the extent that the schedule does not completely determine the sequence of actions in an activity, it provides degrees of freedom in the schedule for

the dispatcher to work with. Thus the scheduling agent is constructing a satisfying interpretation of a set of sentences. This set includes the sentences defining the activities that must be executed in the schedule, the microtheories of activities, states, time, and resources, and any domain-dependent constraints. These additional constraints that the scheduler must satisfy may be deadline constraints such as

$$(\exists\, \sigma)\ holds(rp(desk\_lamp, 100), \sigma) \wedge start(\sigma) < t_1$$

or constraints on the times of actions such as

$$occurs_T(a_1,t_1) \wedge occurs_T(a_2,t_2) \supset t_2 - t_1 > 10$$

The hypothetical reasoning capabilities for the scheduling agent are the focus of current work.

Dispatching agent. Given degrees of freedom in the schedule, the dispatcher makes decisions as to what to do next by adding enough new constraints to the axioms defining the schedule so that there is a unique satisfying interpretation of the axioms. This includes fixing start times of situations and choosing specific resources required by the actions.

Given a schedule, the dispatcher also monitors the status of the factory floor and availability of resources, and communicates deviations in the schedule to the scheduling agent for repair. This involves determining the failure if actions and violated preconditions, as well as violated start times or durations of activities in the schedule. Any of these events serve as new constraints that are given to the scheduling agent to construct a new schedule.

Resource agent. This agent dynamically manages the availability of resources so that the schedule can be executed. It estimates resource demand, determines resource order quantities, generates purchase orders and monitors the delivery of resources.

The resource agent may decide that a schedule is infeasible because there are insufficient resources available; in this case we have

$$(\exists\, \sigma)\ holds(rp(r,q), \sigma) \wedge q < 0$$

The quantity of a resource is completely known at all points during the schedule; this is guaranteed by the effect axioms in the microtheory of resources.

If a schedule is infeasible due to the lack of resources, the resource agent generates purchase orders to obtain the resources from suppliers. The representation of purchase orders is the focus of current research; the problem is that the delivery of the resources from the suppliers is an event external to the schedule whose occurrence is a necessary precondition for activities in the schedule.

Logistics agent. This agent is responsible for coordinating multiple-plants, multiple-supplier, and multiple-distribution centers of the enterprise by generating a global schedule that is given to the scheduling agents in each factory who perform more detailed scheduling. The logistics agent also specifies goals (orders for products) that the scheduling agent must achieve. In this sense, the relationship between the logistics and scheduling agents is analogous to the relationship between the scheduling agent and the dispatcher.

The logistics agent also manages the movement of products or materials across the supply chain from the supplier of raw materials to the customer of finished goods. It thus introduces new constraints of the form

$(\exists\, \sigma)\ holds(rp(desk\_lamp, 100), \sigma) \wedge holds(located(Calgary, desk\_lamp, 100), \sigma) \wedge start(\sigma) < t_1$

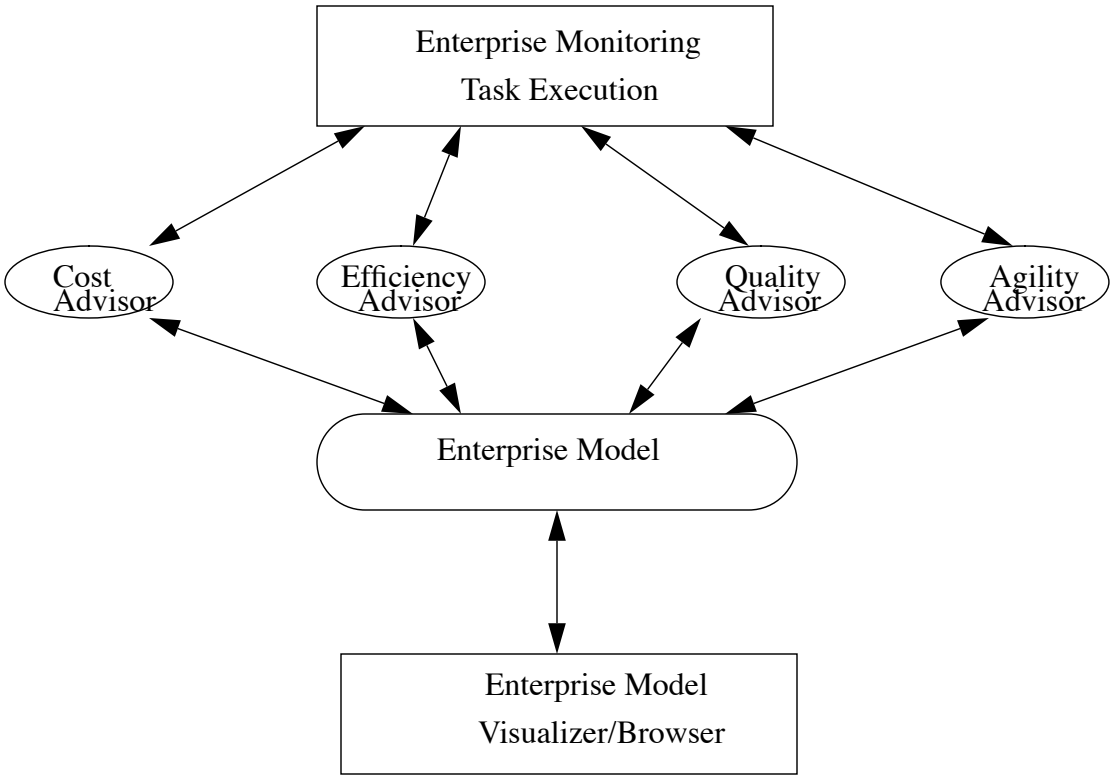
that serve as goals for the transportation agent.

Transportation agent: This agent is responsible for the assignment and scheduling of transportation resources in order to satisfy the goals (inter-plant movement requests) specified by the Logistics Agent.

4.0 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). 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. Within the enterprise integration architecture, these advisors are implemented as agents that operate autonomously on the current enterprise design.

Figure 2: Enterprise Engineering Architecture



4.1 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, 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.

To formalize the intuition of design perspectives, we introduce the notion of advisors. An advisor is an encapsulation of one or more micro-theories. It has the ability to analyse, guide and make design decisions. We are currently constructing advisors for Efficiency, Activity-based Costing, Resource Management, and ISO9000; we anticipate advisors for goals and objectives, and organization structure. In each case, we define the tasks, purpose, and responsibilities of the advisor, and

represent these tasks using the appropriate ontology. As with the competency questions in the first section, each advisor is rigorously characterized by these tasks. This includes specifying what an advisor is analyzing and in what way that they guide (propose different alternatives).

Advisors are not independent of each other. Each of the microtheories implemented in the advisors is closely related to the other microtheories, and this leads to a close interdependency among the different perspectives. The status of activities is dependent on the times at which its resources are available. Resource usage constraints may play an important role in the quality of a product. As we can see from the axioms of the cost microtheory, there is an interaction between cost and the temporal structure of activities in a process, since the cost is dependent on the status of an activity, such as when it is executing and when it has been suspended. All of these relationships provide different perspectives on the processes in an enterprise, as captured in the advisors. As with the agents in the integrated supply chain, we exploit the representation of advisors as sets of axioms in the associated microtheories. Advisors interact through the communication of constraints in each microtheory; the tasks of an advisor that require interaction with other advisors can then be represented as finding a satisfying interpretation of the union of constraints arising from each relevant advisor.

4.1.1 Efficiency 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 efficiency 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.

A fundamental capability of the advisor is to evaluate a set of activities and determine whether or not it satisfies certain integrity constraints. An enterprise should have no “black holes” (resources that are produced but not consumed by other processes or shipped as final products). An enterprise should have no “miracles” (resources produced by a process for which there are no resources consumed). All resources that are used must be released. All resources must be used, consumed, or produced. We can represent these constraints by sentences which must be satisfied by all sets of activities:

$$(\forall a)(\exists\ s,r)\ produce(s,r,a)$$

$$(\forall a)(\exists\ s,r)\ consume(s,r,a)$$

$$(\forall a)(\exists\ s,r)\ consume(s,r,a) \supset (\exists\ s',a')\ release(s',r,a')$$

$$(\forall s,r,a)\ use(s,r,a) \vee consume(s,r,a) \vee produce(s,r,a)$$

The efficiency 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. Such resources would be identified using the axioms for *available* in the microtheory of resources.

4.1.2 Quality Advisor

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.

Recall that 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 as stated by the axioms 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. Thus the advisor determines which activities comply with the ISO 9000 standard, which activities do not comply, and the reason why they do not comply.

4.1.3 Cost Advisor

One task is to perform an activity-based costing analysis of a resource that is produced by some set of activities, including the cost of every resource and activity that are necessary to produce the resource. This requires the ability to recursively determine which activities establish preconditions for the activity that actually produces the resource.

5.0 Current Status

We have developed ontologies for: activities and states, time, resources, quality, and cost. Advisors and their corresponding microtheories are under development for resource management, activity-based costing and ISO9000 quality compliance. A "virtual factory", called TOVE, has been defined using the ontology, and serves as a testbed for research into enterprise integration. It is implemented in C++ using the ROCK knowledge representation tool from Carnegie Group. The axioms in the various microtheories are implemented using Quintus Prolog which is integrated with the knowledge base in ROCK.

We have also developed a distributed simulation environment called TOVESim which oversees the execution of events and maintains time across multiple agents spread across the internet. We are currently working on extending the ontologies, advisors and the ideas for enterprise visualization discussed in this paper.

6.0 Conclusions

In this paper, we have proposed that the knowledge implicit in engineering practice must be formally represented and characterized through ontologies and microtheories. This formalization provides the foundation for the other components of the system.

We have presented an architecture for integrated supply chain management in which the supply chain is managed by a set of intelligent agents responsible for one or more activities. By representing activities as sets of axioms in our microtheory, the tasks for the different agents in the supply chain can be represented as satisfying constraints. Coordination among agents is achieved by communicating constraints.

We have also defined an architecture for an enterprise engineering system that allows the exploration of a variety of enterprise designs. 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. The notion of advisors leads to the architecture presented in Figure 2 and it enables us to automate the execution of certain enterprise engineering tasks. This also allows us to represent workflow monitoring and execution within the same framework.

7.0 References

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

[Fadel et al] Fadel, F., Fox, M.S. , Gruninger M. A Resource Ontology for Enterprise Modelling (submitted).

[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*.

[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.

[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).

[Roboam & Fox 92] Roboam, M. and Fox, M.S. Enterprise Management Network Architecture, *Artificial Intelligence Applications in Manufacturing*. AAAI Press / MIT Press, 1992.