ELSEVIER Computers in Industry 29 (1996) 123–134

COMPUTERS IN INDUSTRY

An organisation ontology for enterprise modeling: Preliminary concepts for linking structure and behaviour

Mark S. Fox, Mihai Barbuceanu, Michael Gruninger *

Department of Industrial Engineering, University of Toronto, 4 Taddle Creek Road, Toronto, Ontario M5S 1A4, Canada

Abstract

The paper presents our preliminary exploration into an organisation ontology for the TOVE enterprise model. The ontology puts forward a number of conceptualizations for modeling organisations: activities, agents, roles, positions, goals, communication, authority, commitment. Its primary focus has been in linking structure and behaviour through the concept of empowerment. Empowerment is the right of an organisation agent to perform status changing actions. This linkage is critical to the unification of enterprise models and their executability.

Keywords: Activity; Empowerment; Ontology; Organisation; Role

1. Introduction

What is an organisation and how do we model it in an information system? Many disciplines have explored the former and every information system built has created a version of the latter. The purpose of this paper is to explore the latter from the perspective of Artificial Intelligence.

As information systems play a more active role in the management and operations of an enterprise, the demands on these systems have also increased. Departing from their traditional role as simple repositories of data, information systems must now provide more sophisticated support to manual and automated decision making; they must not only answer queries with what is explicitly represented in their Enterprise Model, but must be able to answer queries with what is *implied* by the model. The goal of the TOVE Enterprise Modeling project is to create the next generation Enterprise Model, a *Common Sense Enterprise Model*. By common sense we mean that an Enterprise Model has the ability to deduce answers to queries that require extended but relatively shallow knowledge of the domain.

We are taking what can be viewed as a 'second generation knowledge engineering' approach to constructing our Common Sense Enterprise Model. Rather than extracting rules from experts, we are building models of domains by 'engineering ontologies'. Ontologies are shared views of parts or domains of the world. They provide conceptualizations that are agreed upon by people engaged in collaborative action or the development of various artifacts, including software. The shared nature of these conceptualizations allows people or programs to communicate effectively and supports the development of information systems by building interoperable

* Tel: + 1-416-978-6823; fax: + 1-416-971-2479; e-mail: {msf, mihai, gruninger}@ie.utoronto.ca; http://www.ie.utoronto.ca/EIL/eil.html.

components that view and manipulate information in a unified, clearly defined and consistent manner.

An ontology consists of formal descriptions of entities and their properties, relationships, constraints, behaviours. Our approach to engineering ontologies begins with defining an ontology's requirements; this is in the form of questions and issues that an ontology must be able to address. We call this the *competency* of the ontology. The second step is to define the terminology of the ontology – its objects, attributes, and relations. The third step is to specify the definitions and constraints on the terminology, where possible. The specifications are represented in First Order Logic and implemented in Prolog. Lastly, we test the competency of the ontology by 'proving' the competency questions with the Prolog axioms.

Our initial efforts have focused on ontologies to support reasoning in industrial environments. The tasks that we have targeted to support are in 'supply chain management' which extends MRP (Manufacturing Requirements Planning) to include logistics/distribution and 'Concurrent Engineering' which looks at issues of coordination of engineering design. Much of our effort has been in creating representations of organisation behaviour: activity, state, causality and time, and the objects they manipulate: resources [7], inventory, orders and products. We also have efforts underway in formalising knowledge of ISO 9000 quality [13], activity-based costing [21] and organisation agility.

This paper describes the *organisation* ontology being developed as part of the TOVE Project. In particular it focuses on organisation structure, roles, authority and empowerment.

2. What is an organisation?

We consider an organisation to be a set of constraints on the activities performed by a set of collaborating agents. This view follows that of Weber [22] who views the process of bureaucratization as a shift from management based on self-interest and personalities to one based on rules and procedures.

Mintzberg [16] provides an early (and informal) analysis of organization structure distinguishing among five basic parts of an organization and five distinct organization configurations that are encountered in practice. This 'ontology' includes several mechanisms that together achieve coordination, like goals, work processes, authority, positions and communication. The various parts of an organization are distinguished by the specific roles they play in achieving coordination with the above means.

The 'language/action perspective' [23] on cooperative work in organizations provides an ontology that emphasizes the social activity by which 'agents' generate the space of cooperative actions in which they work, rather than the mental state of individuals. The basic idea is that social activity is carried out by language and communication. The pragmatic nature of communication as the way of creating commitments among participants is exploited in the Coordinator system [8].

In the same vein, Auramaki [1] presents a method for modeling offices as systems of communicative action through which people engage in actions by creating, modifying and deleting commitments that bind their current and future behaviors.

The work of Lee [14] looks at language acts in the bureaucratic office, viewing language not as a mechanism for information transfer but as a mechanism for social interaction and control. He presents a logic-based representation of deontic notions – authorization, permission, prohibition and the like - and shows how this can be used to model cooperative work in the office.

More recently, Yu and Mylopoulos [24] have proposed a framework for modeling organizations as being made of social actors that are intentional, having motivations, wants and beliefs and strategic, evaluating their opportunities and vulnerabilities with respect to each other. This formal model is used to explore alternative process designs in business reengineering.

3. Ontology competence

A problem in the engineering of ontologies is their evaluation. A number of criteria have been proposed including [9,10]:

- *Generality.* To what degree is the representation shared between diverse activities such as design and trouble-shooting, or even design and marketing?
- *Competence.* How well does it support problem solving? That is, what questions can the representation answer or what tasks can it support?
- *Perspicuity.* Is the representation easily understood by the users? Does the representation 'document itself?'
- *Transformability.* Can the representation be easily transformed into another more appropriate for a particular problem?
- *Extensibility.* Can the representation be extended to encompass new concepts?
- *Granularity.* Does the representation support reasoning at various levels of abstraction and detail?
- *Scalability.* Does the representation scale to support large applications?
- *Minimality.* Is there a core set of ontological primitives that are partitionable or do they overlap in denotation? A minimal set of terms should be in the ontology.

But the criterion we have found most useful is *competence*. For any task in which the ontology is to be employed, the task imposes a set of requirements on the ontology. These requirements can best be specified as a set of queries that the ontology should be able to answer, if it contains the relevant information. These requirements, which we call competency questions, are the basis for a rigorous characterization of the information that the ontology is able to provide to the task. Competency questions are benchmarks in the sense that the ontology is necessary and sufficient to satisfy the task requirements specified by the competency questions. They are also those questions for which the ontology finds all and only the correct solutions. Tasks that specify their requirements as competency questions can serve to drive the development of new ontologies and also to justify and characterize the capabilities of existing ontologies.

The basic entities in the TOVE ontology 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 provides a declarative specification for the various definitions and constraints on the terminology. Further, we need to prove the competency of the ontology. The ontology 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 a competent ontology, and it is this rigor that is lacking in previous approaches to ontology engineering.

The competency questions are generated by requiring that the ontologies be necessary and sufficient to support the various tasks in which it is employed. Within our applications, 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 [4]. This is essentially the task of finding a minimum duration plan that minimizes action occurrence and maximizes concurrency of activities.

4. Activity / Time Ontology

In this section we define the ontology of time and action that is used to represent the behaviour of the organisation. An important component of representing behaviour is the ability to temporally project, that is, to determine the possible set of future states given a current state. Temporal projection 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.

4.1. Situation calculus specification

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 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? The extended situation calculus of [17] allows us to incorporate the notions of situations and a time line by assigning durations to situations.

The intuition behind the situation calculus is that there is an initial situation, and that the world changes from one situation to another when actions are performed. There is a predicate $\mathrm{Poss}(a, \sigma)$ that is true whenever an action a can be performed in a situation σ.

The structure of situations is that of a tree; two different sequences of actions lead to different situations. Thus, each branch that starts in the initial situation can be understood as a hypothetical future. The tree structure of the situation calculus shows all possible ways in which the events in the world can unfold. Therefore, any arbitrary sequence of actions identifies a branch in the tree of situations.

Further, we impose a structure over situations that is isomorphic to the natural numbers by introducing the notion of successor situation [18]. The function $do(a, \sigma)$ is the name of situation that results from performing action a in situation σ. We also define an initial situation denoted by the constant σ_0.

Situations are assigned different durations by defining the predicate $starts(s, t)$ [17]. 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 $holds(f, \sigma)$ to represent the fact that some ground literal f is true in situation σ. Using the assignment of time to situations, we define the predicate $holds_T(f, t)$ to represent the fact that some ground literal f is true at time t. A fluent is a predicate or function whose value may change with time.

Another important notion is that actions occur at points in time. The work of Pinto and Reiter [17] 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*. To represent occurrences, we then introduce two predicates, $occurs(a, \sigma)$ and $occurs_T(a, t)$, defined as follows:

$$occurs(a, \sigma) \equiv actual(do(a, \sigma)), \qquad (1)$$

$$occurs_T(a, t) \equiv occurs(a, \sigma) \land start(do(a, \sigma), t). \qquad (2)$$

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

4.2. Terminology

At the heart of the TOVE Enterprise Model lies the representation of an *activity* and its corresponding enabling and caused *states* ([19,9]). 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.

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.

A state may have a status whose value is one of the following 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.

Similarly, activities have a status whose value is one of the following constants: *{dormant, executing, suspended, completed}*. The status of an activity is changed by one of the following actions: *execute(a)*, *suspend(a)*, *complete(a)*.

As part of our logical specification of the activity ontology, we 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 [18]. 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 one of the successor axioms in the ontology:

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.

$$\begin{aligned}&(\forall s, a, e, \sigma)\, holds(status(s, a, committed),\\ &\quad do(e, \sigma)) \equiv (e = commit(s, a)\\ &\quad \wedge\, holds(status(s, a, possible), \sigma))\\ &\quad \vee\, \neg(e = enable(s, a))\\ &\quad \wedge\, holds(status(s, a, committed), \sigma). \qquad (3)\end{aligned}$$

A more complete specification can be found in [11].

5. Competency

In linking the structure of an organisation with the behaviour of agents within the organisation, we must define how the organisation ontology is integrated with the activity ontology.

If we consider organisation to be a set of constraints on the activities performed by agents, then the competency questions for the organisation ontology are extensions of the temporal projection and plan existence problems to incorporate the abilities and obligations of agents. The temporal projection problem is used to characterize the constraints that agents must satisfy to be able to perform activities, and plan existence characterizes the set of achievable goals. We can then propose the following set of competency questions for the organisation ontology.

5.1. Structure

- What is the structure of the organisation? How is the organisation decomposed into units?
- What are the members of a particular unit of the organisation?
- What positions exist in the unit?
- What position does person X occupy?
- Who must person X communicate with?
- What kinds of information does person X communicate?
- Who does X report to?

5.2. Behaviour

- What are the goals of the unit?
- What are the goals of the position?
- What are the goals of person X?
- What activities must a particular position perform?
- What activities must person X perform?
- Is it possible for an agent to perform an activity in some situation? That is, does the agent have the ability to perform the activity?

5.3. Authority, empowerment and commitment

- What resources does the person have authority to assign?
- What activities may a person execute without explicit permission?
- In order to perform a particular activity, whose permission is needed?
- Is an agent allowed to perform an activity in some situation?
- What goals is person X committed to achieving?
- Is a goal achievable by an agent given its current commitments and the commitments of other agents?
- If a goal is unachievable for a given set of agents, how can they be empowered to be capable of performing the activities to achieve the goal? That is, how can the constraints defining empowerment for the agents be modified so as to be able to achieve the goal?
- What authority constraints are necessary among a set of agents in order to achieve a goal?

5.4. Goal achievement

- What goals are solitarily unachievable for a given agent? That is, what goals are unachievable using a plan that contains only activities that the agent is capable of performing? Such goals require the assistance of other agents to achieve them.
- What goals are achievable by an agent given the effects of activities that other agents are capable of performing?
- If a goal is solitarily unachievable for a given agent, what agents are required to assist the agent in achieving the goal?

6. Description logic specification

It is important to be as precise as possible when describing ontologies. For this reason, logic is a natural choice as an ontology specification language. In the previous sections we have talked about activity in a temporal projection framework using situation calculus. In the next sections we describe the organization ontology using a more structured notation for logic (known as description logics [6,5,15,2]) that allows more concise specifications of structured concepts and a general object oriented organization.

The main entities of the language are concepts, roles and instances. Concepts are equivalent to unary predicates describing a class of individuals and are generally specified as conjunctions of descriptions. Defined concepts contain necessary and sufficient conditions for an individual to belong to the respective concept. Primitive concepts contain only necessary conditions. Primitive concepts can be explicitly used as components of other concepts and sets of them can be declared as disjoint. Roles describe binary relations between a domain and a range concept. Roles can be composed by conjunction from other roles. Instances are by definition disjoint and represent particular individuals. Table 1 shows some of the description types we use. The interpretation I(d) of a description d is the set of individuals represented by d, r(x,y) states that x and y are related through the r role and C(x) states that x belongs to concept C.

Based on the descriptors in Table 1, we also use (:some r C) which is a more perspicuous notation for (:and (:all r C)(:atleast 1 r)) and (:the r C) which is a notation for (:and (:all r C)(:atleast 1 r)(:atmost 1 r)). The descriptor (:oneof f1 f2 .. fn) denotes a set of fillers from which only one will actually fill a given role in an instance.

7. Organisation terminology

7.1. Organisation

To begin, an organization consists of a set of Organisation-Agents (said to be members of the organisation), a set of Organisation-Units (recursive subcomponents having a structure similar to organisations) and an Organisation-Goal tree that specifies the goal (and its decomposition into subgoals) the members try to achieve. Using the description logic concept language, the concept of organisation can be specified as:

```
(concept organization :is(:and org-entity
(:some org-goal organisation-goal)
(:some org-unit organisation-unit)
(:some member organisation-agent)))
```

The Organisation-Unit recursively describes the sub-organizations that compose an organisation:

```
(concept organisation-unit :is (:and org-entity
(:the member-of organisation)
(:some unit-goal organisation-goal)
(:some unit-member organisation-agent)
(:all unit organisation-unit))).
```

For example, the Department of Industrial Engineering (IE)can be modeled as an organisation having: a number of goals related e.g. to education and research, component units like the Enterprise Integration Laboratory (EIL), the Human-Computer Interfaces Laboratory, etc. and a number of organisation agents consisting of faculty, research staff, students, etc. Equally, the Department of Industrial Engineering can be seen as an organisation-unit member of the Faculty of Applied Science and Engineering (ASE) which at its turn is an organisation unit member of the University of Toronto (UofT).

If O is a particular organisation, we represent this as an assertion organisation(O). For example, we can have

organisation(IE),
organisation(ASE)
organisation(UofT).

To represent the fact that Industrial Engineering (IE) is a member of Applied Science and Engineering (ASE) we use a binary assertion:

member-of(IE, ASE).

7.2. Organisation agent

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

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

In the concept language, Organisation-Agent is described as:

(concept organisation-agent :is (:and org-entity
(:some org-membership (or organization organisation-unit))
(:some agent-position organisation-position)
(:some agent-empowerment empowerment)
(:all agent-communication-link communication-link))).

For example, to represent the fact that MB and MG are organisation agents of IE, we write:
organisation-agent(MB)
organisation-agent(MG)
org-membership(MB, IE)
org-membership(MG. IE).

7.3. Organisation-role

An *Organisation-Role* defines a prototypical function of an agent in an organisation. A particular agent can assume several roles at the same time. For an individual agent, examples of organisational roles include 'project manager', 'reviewer', 'troubleshooter', etc. Once an agent is assigned to a role, that creates a commitment (more on commitments later) on the agent's part to act to achieve the goal(s) of the role.

We define Organisation-Roles as follows:

(concept org-role :is (:and org-entity
(:some role-goal organisation-goal)
(:all role-skill skill)
(:all role-process organisation-activity)
(:all role-policy policy)
(:all role-communication-link communication-link))).

Each Organisation-Role has:

- *Goals*: one or several goals which the agent playing the role is responsible for. Example: role-goal(project-manager, manage-cost).
- *Skills*: one or more skills required to achieve the goals. Example: role-skill(project-manager, cost-estimation).
- *Processes*: activity networks that have been defined to achieve the goals. Example: role-process(project-manager, hire-personnel).
- *Policies*: constraints on the performance of the role's processes. These constraints are unique to the organisation role. Example: role-policy(project-manager, equal-opportunity-hiring).
- *Communication-Link*: these are communication links to other agents in specified roles. Communication consists of exchanging speech acts according to specific conversation structures that are also formally represented [3].

Example:
role-communication-link(project-manager, comm-link-to-technology-VP).

7.4. Organisation position

An *Organisation-Position* defines a formal position that can be filled by an OA in the organisation. Examples of positions include 'president', 'laboratory director', 'senior researcher', 'sales-representative', etc. Any position essentially consists of a set of roles the agent filling it will have to carry out. Thus, positions are the means to relate agents to roles and goals. For example, the 'laboratory director' position would include roles for project supervision, securing financing, liaison with sponsors, etc.

Second, positions define certain authority relations with other positions in the organization. The 'laboratory director' position for example also implies authority over any 'senior researcher' position in the respective organisation unit.

We define the Organisation-Position concept as:

(concept organisation-position :is(:and org-entity
(:some position-role organisation-role)
(:all authority-over organisation-position)
(:the filled-by organisation-agent)))

Examples:
organisation-position (laboratory-director)
position-role (laboratory-director project-supervision)
position-role (laboratory-director securing-financing)
filled-by (laboratory-director, MSF).

With respect to the last example above, in general we assume that positions are filled by individual agents. Note however that a group agent may also fill a position.

7.5. Organisation goal

Organisation agents play roles that assume goals to achieve. Our ontology models organisation goals that can be decomposed into and-or subgoal trees, can be achieved by executing activity clusters and have dependency relations amongst them. A goal G1 is said to depend on a goal G2 if achieving G1 depends on having achieved G2 in a previous situation. If the dependency is *weak*, G1 could still be achieved even if G2 is not, although more difficulty. If the dependency is *strong*, G1 can not be achieved unless G2 has been achieved previously.

```
(concept organisation-goal :is (:and org-entity
(:the goal-description string)
(:all goal-activity activity)
(:all dependency goal-dependency)
(:some held-by-agent org-agent)))

(concept and-goal :is (:and organisation-goal
(:some conjunct organisation-goal)))

(concept or-goal :is (:and organisation-goal
(:some disjunct organisation-goal)))

(concept goal-dependency :is (:and org-entity
(:the depender organisation-goal)
(:the dependee organisation-goal)
(:the dependency-strength (:oneof strong weak))))
```

7.6. Communication-link

Communication-Links are established among organisational agents in various roles. We distinguish between two forms of communication links. *Information-Links* capture the notion of benevolent communication in which agents regarding each other as peers volunteer information that they believe relevant to other agents. This exchange does not create obligations for any agent. *Communication-with-Authority* links are used to send/receive communication that creates obligations according to the established authority relations in the organisation. They are used for example to request agents to commit to given goals or to report on the execution status of activities.

The *Information-Link* is a unidirectional link used to communicate information from one agent to another. It describes, for an agent in a given organisational role, the information it is interested in receiving and the information it can benevolently distribute to others.

For example, an agent in the 'C + + programmer' role may distribute information about the state of the file server to other programmers, alerting them each time the server is down.

The *Communication-with-Authority* link, used when communication is intended to create obligations, specifies the two agents, one in the authority position, among which communication takes place. Because we model communication as exchange of speech-acts, authority of an agent appears as the set of speech-acts this agent can use in order to create obligations for the other agent. For example, an agent may have authority to request another agent to perform action A1, but not to perform action A2. In this case, the second agent will have to commit to achieving A1 when requested by the first agent, but not A2.

The Communication-with-Authority concept is defined as follows:

```
(concept communication-with-authority
:is (:and communication-link
(:the committing-agent org-agent)
(:the committing-agent-role org-agent)
(:the committed-agent org-agent)
(:the committed-agent-role org-agent)
(:all authority-for speech-act))).
```

The authority relationship is defined among agents in given organisation roles. An agent in a 'project-manager' role can request an another agent in a 'programmer' role to write a program for a given function, but can not request the second agent to, say, make a coffee. This is because writing programs is a goal of the 'programmer' role, while making coffee is not.

7.7. Authority and commitment

We have introduced the concept of an organisation agent's commitment to achieving a goal. The predicate committed-to(OA, G) signifies that *Organisation-Agent* OA is committed to the achievement of Goal G. Consequently, the totality of activities performed by OA must include the achievement of G. Prioritisation of goals, etc. are not considered here.

We use *authority* to refer to the control relationship that exists between two organisational agents. For OA_1 to have *authority* over OA_2 implies that OA_1 is able to extract a commitment from OA_2 to achieve a goal that is defined as part of OA_2's *organisation-roles*. In order to extract that commitment, OA_1 has to be related directly or indirectly by an *authority-link* relation that is created either as a consequence of the organisational positions of the agents (see the Organisation-Position) or of Communication-with-Authority links among agents:

8. Empowerment: Linking structure and behavior

With the introduction of organisational knowledge, we now have to address the problem of how to specify 'who can do what'. That is, what is the set of activities that an OA is allowed to perform as a member of the Organisation. It would appear that by associating processes with OAs via the *role-process* property, we have solved the problem. That is, an OA can perform any activities specified by its processes. But consider the following situation:

"Jill, in her role as a CNC machine operator, has a process she must perform in order to achieve the goal of producing an order. The process is composed of three activities: 1) machine-setup, 2) machine-run and 3) machine-teardown. But before the machine-run activity can commence, she must receive permission from her supervisor."

The problem is that Jill has a process that specifies a sequence of activities that she must perform. But she cannot perform the second activity, machine-run, without permission. The implication is that within our Activity ontology, she is not allowed to change the state of the machine-run activity to 'execute'.

An obvious way to solve the problem is to insert a fourth activity between machine-setup and machine-run where she seeks approval from her supervisor. If approval is obtained, then she can commence the subsequent machine-run activity. Again we have a problem. Who is allowed to change the status of this new approval activity to completed? If Jill is allowed to make any status changes she wants to the activities in her process, she can change the status of the approval activity regardless of whether she obtained approval or not.

The problem lies with who is allowed to make status changes to states and activities. When Jill goes to her supervisor for permission, is it Jill who changes the status of the approval activity to 'completed' or her supervisor? It is not clear. Therefore the only solution to the problem of permission to perform an action lies in precisely stating who is allowed to change the status of the activity, e.g., from 'dormant' to 'executing'.

We introduce the concept of Empowerment as a means of specifying the status changing rights of an OA. *Empowerment is the right of an OA to perform status changing actions*, such as commit, enable, suspend, etc. Empowerment naturally falls into two classes: state and activity empowerment.

State empowerment specifies the range of stati through which an OA may take a state by performing the appropriate actions, such as commit. State empowerment not only specifies allowable status changes but may be used to restrict the set of resources an OA is empowered to commit to a use/consume state. An OA may be empowered for any type of resource, including other OAs. The implication being the first OA may commit the second to a state.

Activity empowerment specifies the range of stati through which an OA may take an activity by performing the appropriate actions, such as execute and suspend. Even though an activity may be enabled, the OA whose role contains the plan which contains the activity may not be empowered to start its execution.

With the addition of empowerment, a second type of authority arises. That is, the supervising agent may alter what a supervisee is empowered to do.

We now present a number of axioms meant to clarify the meaning of empowerment.

First, for any activity a that requires that the agent be empowered, the status changing actions for the activity require $holds(activity\_empowered(agent, a), \sigma)$ as a precondition.

Similarly, for any state s that requires that the agent be empowered, the status changing actions for the activity require $holds(state\_empowered(agent, s), \sigma)$ as a precondition.

1. It is possible to for one agent to empower another agent for an activity if the first agent supervises the second, and the supervisor is empowered for that activity.

$$Poss(activity\_empowers(agent, agent', a), \sigma) \equiv holds(supervises(agent, agent'), \sigma) \wedge holds(activity\_empowered(agent, a), \sigma). \quad (4)$$

2. It is possible to for one agent to disempower another agent for an activity if the first agent supervises the second, and the supervisor is empowered for that activity.

$$Poss(activity\_disempowers(agent, agent', a), \sigma) \equiv holds(supervises(agent, agent'), \sigma) \wedge holds(activity\_empowered(agent, a), \sigma). \quad (5)$$

3. An agent is empowered for an activity only as a result of the action *activity_empowers*, and is no longer empowered only as a result of the action *activity_disempowers*.

$$holds(activity\_empowered(agent, a), do(a', \sigma)) \equiv (\exists agent')a' = activity\_empowers(agent', agent, a) \vee holds(activity_{empowered}(agent, a), \sigma) \wedge \neg(\exists agent')a' = activity\_disempowers(agent', agent', a). \quad (6)$$

4. It is possible to for one agent to empower another agent for changing the status of states if the first agent supervises the second, and the supervisor is empowered for changing the status of that state.

$$Poss(state\_empowers(agent, agent', s), \sigma) \equiv holds(supervises(agent, agent'), \sigma) \wedge holds(state\_empowered(agent, s), \sigma). \quad (7)$$

5. It is possible to for one agent to disempower another agent for changing the status of that state if the first agent supervises the second, and the supervisor is empowered to change the status of that state.

$$Poss(state\_disempowers(agent, agent', s), \sigma) \equiv holds(supervises(agent, agent'), \sigma) \wedge holds(state\_empowered(agent, s), \sigma). \quad (8)$$

6. An agent is empowered for changing the status of a state only as a result of the action *state_empowers*, and is no longer empowered only as a result of the action *state_disempowers*.

$$holds(state\_empowered(agent, a), do(a', \sigma)) \equiv (\exists agent')a' = activity\_empowers(agent', agent, a) \vee holds(state\_empowered(agent, a), \sigma) \wedge \neg(\exists agent')a' = state\_disempowers(agent', agent', a). \quad (9)$$

9. Conclusions

The paper presents our preliminary exploration into an organisation ontology for the TOVE enterprise model. The ontology views organisations as composed of agents playing roles in which they are acting to achieve specific goals according to various constraints defining the 'rules of the game'. A primary focus has been in linking structure and behaviour through the concept of empowerment. Empowerment is the right of an organisation agent to perform status changing actions. This linkage is critical to the unification of enterprise models and their executability.

The ontology is currently being used by members of our group to model activity-based costing [21], ISO-9000 quality [13] and is also at the basis of an advisor that suggests ways to reengineer an organization to increase its responsiveness to changing market demands. Even so, much work still remains to be done in the development of our ontology and especially its axiomatisation.

Acknowledgements

This research is supported, in part, by the Natural Science and Engineering Research Council, Manufacturing Research Corporation of Ontario, Digital Equipment Corp., Micro Electronics and Computer Research Corp., Spar Aerospace, Carnegie Group and Quintus Corp.

References

[1] E. Auramaki, E. Lehtinen and K. Lyytinen, "A speech-act-based office modelling approach", *ACM Transactions on Office Information Systems* 6(2) (1988), 126–152.

[2] M. Barbuceanu, "Models: Toward integrated knowledge modeling environments", *Knowledge Acquisition* 5 (1993) 245–304.

[3] M. Barbuceanu and M.S. Fox, "COOL: A language for describing coordination in multiagent systems", *First International Conference on Multiagent Systems*, San Francisco, 12–14 June 1995.

[4] J. Blackburn, "Time-based competition". *Business One Irwin* (1991).

[5] A. Borgida, R.J. Brachman, D.L. McGuiness, L.A. Resnick, "CLASSIC: A structural data model for objects", *Proc. 1989 ACM SIGMOD International Conference on Management of Data* (June 1989) 59–67.

[6] R.J. Brachman and J.G. Schmolze, "An overview of the KLONE knowledge representation system", *Cognitive Science* 9(2) (1985) 171–216.

[7] F. Fadel, M.S. Fox and M. Gruninger, "A resource ontology for enterprise modelling", *Third Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises*, West Virginia University, Morgantown, WV, 1994.

[8] F. Flores, M. Graves, B. Hartfield and T. Wionograd, "Computer systems and the design of organizational interaction", *ACM Transactions on Office Information Systems* 6(2) (1988) 153–172.

[9] M.S. Fox, J. Chionglo and F.A. Fadel, "Common-sense model of the enterprise", *Proceedings of the Industrial Engineering Research Conference 1993.*

[10] T.R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing", Report KSL 9304, Stanford University, August 1993.

[11] M. Gruninger and M.S. Fox, "The role of competency questions in enterprise engineering", *Proceedings of the IFIP WG5.7 Workshop on Benchmarking – Theory and Practice*, Trondheim, Norway, June 1994.

[12] N.R. Jennings, "Commitments and conventions: The foundation of coordination in multi-agent systems", *The Knowledge Engineering Review* 8(3) (1993) 223–250.

[13] H. Kim and M.S. Fox, "Quality systems modelling: A prospective for enterprise integration", *Fourth Annual Meeting of the Production and Operations Management Society*, 1993.

[14] R.M. Lee, "Bureaucracies as deontic systems", *ACM Transactions on Office Information Systems* 6(2) (1988) 87–108.

[15] R. McGregor and R. Bates, "The LOOM knowledge representation language", ISI-IRS-87-188, USC/ISI Marina Del Rey, CA, 1987.

[16] H. Mintzberg, *Structure in Fives – Designing Effective Organizations*, Prentice Hall Inc., New York, 1983.

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

[18] R. Reiter, "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, CA, 1991.

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

[20] J. Searle, *Speech Acts*, Cambridge University Press, Cambridge, UK, 1969.

[21] D. Tham, M.S. Fox and M. Gruninger, "A cost ontology for enterprise modelling", *Third Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises*, West Virginia University, Morgantown, WV, 1994.

[22] M. Weber, *Economy and Society*, University of California Press, Berkeley, CA, 1987.

[23] T. Winograd, "A language/action perspective on the design of cooperative work", *Human Computer Interaction* 3(1) (1987–1988), 3–30.

[24] E.S.K. Yu and J. Mylopoulos, "From E-R to 'A-R'-modelling strategic actor relationships for business process reengineering", *13th Int. Conf. on the Entity-Relationship Approach*, Dec. 13–16 1994, Manchester, UK.