Appeared in: Proceedings of the third IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, April 1994, Morgantown, West Virginia (WET ICE '94)

A GENERIC ENTERPRISE RESOURCE ONTOLOGY

Fadi George Fadel, Mark S. Fox, Michael Gruninger
Department of Industrial Engineering, University of Toronto
4 Taddle Creek Road, Toronto, Ontario M5S 1A4 Canada
tel:+1-416-978-6823 fax:+1-416-978-3453 internet: {fadel, msf, gruninger} @ie.utoronto.ca

Abstract*: The complexity of planning and scheduling is determined by the degree to which activities contend for resources. Accordingly this requires any application to have the ability to reason about the nature of the resource and its availability. This paper presents a generic enterprise resource ontology. The ontology described in this paper is one of many that is being created by the* ***TOVE*** *project in the Enterprise Integration Laboratory at the University of Toronto.*

Content Area*: Enterprise Modelling, Enterprise Integration Frameworks*

Keywords: Enterprise Modelling, Enterprise Ontology

1. Introduction

The complexity of planning and scheduling is determined by the degree to which activities contend for resources. Accordingly this requires any application to have the ability to reason about the nature of the resource and its availability. This paper presents the ontology for modelling resources.

Most information systems that support enterprise functions were created independently. This leads to three problems. Firstly, the functions do not share the same representations (i.e. different representations of the same enterprise knowledge); hence, they are unable to share knowledge. Secondly, the representations were defined without an adequate specification of what the terminology means (aka semantics); hence, the interpretations and uses of the knowledge are inconsistent. Thirdly, the representations lack any deductive capability; as a result, users are forced to spend significant effort on programming each new report or function that is required.

The key to integrating different enterprise functions, is to have the knowledge expressed with minimum assumptions of the application. The main aim of an ontology is to enable the coupling of enterprise functions and their respective knowledge and tools. In other words, the ontology acts as protocols for input, output and communication [Fox & Tenenbaum 90] [Gruber 91], creating efficient coordination and communication between different organizational units. Accordingly, the ontology provides a base representation for resources and a standard language of communication.

The goal of the research is to create a *Resource Ontology* for a manufacturing enterprise. Manufacturing functions such as production planning and scheduling depend on the ability to reason about various facets of resources, such as their amounts, capacity and availability. In order to reason about these facets, a clear and precise representation must exist.

The objectives of the research are:

1. Define the span of the model by a set of competency questions.
2. Create an ontology for resources.
3. Implement the *F*irst *O*rder *L*ogic definition an constraints as axioms in Prolog.

The resource ontology described in this paper is one of many that is being created by the TOVE project in the Enterprise Integration Laboratory at the University of Toronto. The TOVE Project includes two major undertakings: the development of an Enterprise Ontology, and a testbed.

The TOVE Enterprise Ontology provides a generic, reusable ontology for modelling enterprises. An ontology is comprised of a reference data model composed of objects, attributes and relations (also called the terms of

the ontology), and formal definitions of terms and their constraints in First Order Logic. The TOVE ontology currently spans knowledge of activity, state, time, causality, resources, cost and quality. The ontology's data model is implemented on top of C++ using the Carnegie Group's ROCK^{TM} knowledge representation tool and the axioms are implemented in Quintus^{TM} Prolog. The Prolog implementation provides a deductive mechanism (aka deductive database) which can be used to answer *simple* common sense questions without additional programming.

The TOVE Testbed provides an environment for analyzing enterprise ontologies. The Testbed provides a model of an enterprise - a lamp manufacturing plant -, and tools for browsing, visualization, simulation and deductive queries.

2. Evaluation Criteria

Though there exists a number of efforts seeking to create sharable representation of enterprise knowledge - CIM-OSA [Esprit 90] , ICAM [Martin 83] , IWI [Scheer 89] , PERA [Williams 91] - little comparative analysis has been performed. Recently, two sets of evaluation criteria have been proposed. Fox & Tenenbaum have proposed the following criteria as a basis for evaluating an ontology: *generality, efficiency, perspicuity, transformability, extensibility, granularity, scalability* and *competence* [Fox & Tenenbaum 90] [Fox et al 93]. Gruber has proposed: clarity, coherence, extensibility, minimal encoding and minimal ontological commitment [Gruber 93].

The criterion we have chosen to evaluate our work is *competence*. The competence of a representation defines the types of tasks that the representation can be used in. Moreover, competency questions represent the starting point of the ontology development; the competency questions set the requirements for the ontology which in turn results in the modification of the competency questions. The obvious way to demonstrate competence is to define a set of questions that can be answered by the ontology. Given a representation and an accompanying theorem prover (such as Prolog), questions can be posed in the form of queries to be answered by the theorem prover. The creation of competency questions and an ontology is an iterative process; from the competency questions the ontology is derived which in turn results in a modification of the competency questions.

For example, in a study performed, over 30 managers from three different Westinghouse Corporation plants were asked to record the questions frequently asked [Fox 83]:

- What is the effect of the current state on the design and material specifications?
- What is the safe inventory level?
- What is the effect of machine X breaking down on the product Z production?
- When is the expected finish time of the job j?
- If order O is shipped then inform the client.
- What are the required specifications that must be met by part X suppliers.

These questions can serve as a set of competency questions for an enterprise model. Following are a subset of questions we have considered in the creation of the **TOVE** model.

- **Divisibility:** Can the resource be divided and still be usable?
- **Quantity:** What is the stock level at time t?
- **Location:** Where is resource R?
- **Consumption:** Is the resource consumed by the activity? If so, how much?
- **Commitment:** What activities is the resource committed to at time *t*?
- **Structure:** What are the subparts of resource R?
- **Capacity:** Can the resource be shared with other activities?
- **Trend:** What is the capacity trend of a resource based on the machine usage history?

3. Resource Ontology

We view that "*being a resource*" is not innate property of an object, but is a property that is derived from the role an object plays with respect to an activity. That is to say that properties of a resources determined by the activity. Accordingly, resources could be: *Machines* such as milling machines when associated with milling activities**,** *Electricity* or *Raw materials* consumed by an activity, *Tools/Equipment* such as fixtures, cranes, chairs etc., *Capital* needed to perform an activity, *Human skill* when needed to perform an activity, *Information* required to start an activity. In our analysis, we have strived to iden-

tify the primitive resource properties on which more complex properties, such as capacity recognition, are defined. Following are some examples of resource ontology* †.

- **Resource-known**: This is the most basic term in the ontology. It specifies *knowledge of a resource* as opposed to its physical existence. The importance of this definition lies in the fact that the ability to reason about a resource depends on it being known.

```
rknown(R).                                    (PRO 1)
```

- **Resource role**: In TOVE, a resource has a role with respect to an activity. These roles are: raw material, product, facility, tool, operator.

```
role(R, A, Role).                             (PRO 2)
```

This entails that when a resource is defined as having a role with respect to an activity, then the resource can not have any other role with respect to the same activity.

$$\forall(r, a, role_1, role_2)\ role(r, a, role_1) \wedge role_1 \neq role_2 \supset \neg\ role(r, a, role_2) \qquad \textbf{(FOL 1)}$$

- **Division of**: This term specifies that a resource could be divided. There are two types of divisions: physical, and functional. "Physical division of" specifies a division that is neither mental, moral or imaginary but is related to the division of the physical body of an object; "functional division of" specifies a division affecting the function and not structure. A "crank shaft" is a physical and a functional *division of* a "motor".

```
physical_division_of(R2, R).                  (PRO 3)
functional_division_of(R2, R).                (PRO 4)
```

- **Divisibility of a resource**: This term specifies the property of a resource as being divisible with respect to an activity without affecting the role of the resource with respect to an activity. There are three types of divisibility: physical, functional and temporal divisibility.

A resource is physically divisible if the act of physically dividing the resource does not affect its role in the activity. In other words, the resource is physically divisible if each division can be used or consumed by the same activity‡. That property is useful for planner/scheduler when deciding whether a portion of resource could support an activity. Functional divisibility of a resource, with respect to an activity, specifies that each division of the resource affects the functionality and not the structure of the resource. A parking lot is an example of such resources as the lot has a number of functional divisions that are used by parking activities. A "motor" is neither *physically* and *functionally divisible* when associated with "driving a car" activity. Finally, a resource is said to be temporal divisible if the use of a resource over time does not affect the future usability of the resource by the same activity.

"A resource is physically divisible with respect to an activity if each physical division of the resource has the same role".

$$\forall(r, a)\ physical\_divisible(r, a) \equiv \forall(r_1, ro_1)\ physical\_division\_of(r_1, r) \wedge role(r_1, a, ro_1) \supset role(r, a, ro_1) \qquad \textbf{(FOL 2)}$$

"A resource is functionally divisible with respect to an activity if each functional division of the resource has the same role".

$$\forall(r, a)\ functional\_divisible(r, a) \equiv \forall(r_1, ro1)\ functional\_division\_of(r_1, r) \wedge role(r_1, a, ro_1) \supset role(r, a, ro_1) \qquad \textbf{(FOL 3)}$$

"A resource is temporally divisible with respect to an activity A1 if there exists a time period in which two activities, including A1, were executing with the condition that the first activity (A1) was either suspended or completed and the resource had the same role with both activities. Moreover, both activities were not executing simultaneously (i.e overlapping constraint)."

$$\forall(r, a)\ temporal\_divisible(r, a) \equiv \exists\ (ti, ti_1, ti_2, tp_1, tp_2, a_1, a_2, s_1, s_2, role_1)\ (uses(s_1, r) \vee consumes(s_1, r)) \wedge (uses(s_2, r) \vee consumes(s_2, r)) \wedge is\_related(a_1, s_1)^{**} \wedge is\_related(a_2, s_2) \wedge time\_bound(s_1, ti_1)^{\dagger\dagger} \wedge time\_bound(s_2, ti_2) \wedge$$

*. For more examples and details please refer to [Fadel 94].

†. The first order logic formulation are tagged as **(FOL #)** while the Prolog implementation are tagged as **(PRO #)**.

‡. i.e each division shares the same role of the original resource.

**. is a term defined in the activity-state ontology specifies an activity is linked (related) to a state.

$$activity(a_1,executing,tp_1)^{*} \wedge period\_contains(ti_1, tp_1)^{\dagger} \wedge ((activity(a_1,suspended,tp\_end) \wedge tp\_end = EP(ti_1)^{\ddagger}) \vee ((activity(a_1, completed,tp\_end) \wedge tp\_end = EP(ti_1)) \wedge activity(a_2, executing, tp_2) \wedge period\_contains(ti_2, tp_2) \wedge \neg overlaps(ti_1,ti_2) \wedge contains(ti, ti1) \wedge contains(ti, ti2) \wedge role(r, a_1, role_1) \wedge role(r, a_2, role_1)$$ **(FOL 4)**

- **Unit of measurement**: This predicate specifies a default measurement unit for a resource, when associated with activity. Accordingly, resource quantity or capacity is to be measured using the specified *unit* of measurement. This term is used for specifying both the qualitative and quantitative units of measurement. Qualitative units of measurement consist of an ordered set such as {large, medium small}. Qualitative units can be also used as a measure of quality: {good, bad}. Quantitative units are used to specify attributes such as weight, length, capacity.

`unit_of_measurement(R,Unit_ID,Unit,A).` **(PRO 5)**

- **Measured by**: "Measured by" defines the objects by which a resource is measured with respect to an activity. This term acts as a constraint on the "unit of measurement" term. Each unit of measure must have a corresponding "measured by" assertion.

`measured_by(R, Unit_id, A).` **(PRO 6)**

As mentioned, a constraint on the "unit of measurement" is that there should be a corresponding measured by assertion

$$(\forall r)(\exists unit\_id, a)\ measured\_by(r, unit\_id, a) \supset (\exists u)\ unit\_of\_measurement(r, unit\_id, u, a)$$ **(FOL 5)**

††. time_bound(s,ti) specifies that ***ti*** is the time of interval of the state.

*. activity(a1, executing, tp1) specifies that activity ***a1*** has to be executing at time point ***tp1***.

†. period_contains(ti, tp) specifies that time point ***tp*** is contained by ***ti*** time period.

‡. EP(ti) is function that returns the end time point of an interval.

- **Component of**: "Component of" specifies a resource as being a part of another resource implying that a resource consists of one or more sub-resources (i.e sub components). A resource can be a physical or functional component of another resource with respect to an activity and each does not share the same role with the original resource. This term is used for example in the bills of material explosion of parts.

"A resource *R2* is a physical component of resource *R1* if *R2* is a physical division of the *R1* and both resources do not share the same role with respect to an activity".

$$\forall(r_1, r_2)\ physical\_component\_of(r_2, r_1) \equiv \forall(a, r, ro_1)\ physical\_division\_of(r_2, r_1) \wedge role(r_2, a, ro_1) \supset \neg role(r_1, a, ro_1)$$ **(FOL 6)**

"A resource *R2* is a functional component of resource *R1* if *R2* is a functional division of the *R1* and both resources do not share the same role with respect to an activity".

$$\forall(r_1, r_2)\ functional\_component\_of(r_2, r_1) \equiv \forall(a, r, ro_1)\ functional\_division\_of(r_2, r_1) \wedge role(r_2, a, ro_1) \supset \neg role(r_1, a, ro_1)$$ **(FOL 7)**

- **Quantity**: A *resource point* (***rp***) specifies a resource's quantity at a some time and unit of measure.

figure 1 3-D resource point - in terms of quantity, time and location

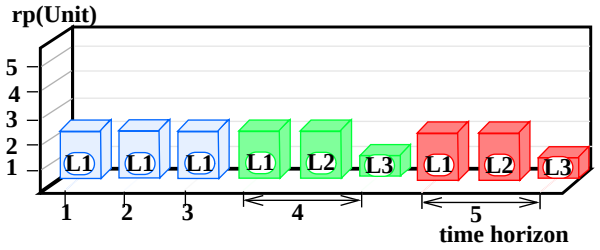
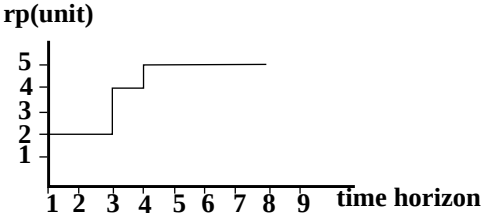


figure 2 2-D resource point - in terms of quantity and time



We have defined two resource point terms - 3-D and 2-D definitions. The 3-D definition (figure 1) is asserted as a ground term and is defined in terms of time, location and quantity* while the 2-D definition (figure 2) represents resource amounts aggregated across all locations and is calculated using 3-D rp assertions†.

```
rpl(plug, 100, tp90, ss12, unit).(PRO 7)
```

For example the above assertion, 3-D *rpl* definition, specifies the existence of a resource point for resource *plug* at time point *tp90*, with quantity of *100* units at location *ss12*.

```
?- rp(resource, Q, time, unit). (PRO 8)
```

The 2-D resource point definition returns the summation of all resource point quantities for *resource*, over all locations at a specific time point. Besides that ability to represent physical resource quantities, resource point is also used for the identification of capacity of reusable resources such as in the example of ftp site where *rp* would denote to the unused accessed lines.

- **Application specification**: *The predicates in this section are defined for notational convenience.* The *consumption*, *use* and *produce specifications* specify the amounts of the resource that is to be consumed, used or produced respectively over a time interval as well as the unit of measurement. The information included in these specifications‡ are already defined in the activity state ontology. There are three application specification: consumption, use and production specifications.

```
consumption_spec(R,A,Ti,Q,Rate,Unit).(PRO 9)
use_spec(R,A,Ti,Q,Rate,Unit).        (PRO 10)
produce_spec(R,A,Q,Rate,Ti,Unit).(PRO 11)
```

where ***Q*** represent the total number consumed/produced by an activity or the portion of the resource used by an activity.

The application specification, consumption specification for example, concatenates the specification into one term as defined in the following FOL formulation:

*. rpl(resource, quantity, time_point, locations, unit).

†. rp(resource, quantity, time_point, unit).

‡. i.e the arguments of each term

$$(\exists\, a, q, ti, u)\ (\forall r)\ consumption\_spec(r, a, q, ti, u) \equiv (\exists\, s, s_2, unit\_id)\ enabling(s, a) \wedge is\_related(a, s_2,) \wedge consumes(s_2, r)^{**} \wedge quantity(s_2, q)^{\dagger\dagger} \wedge time\_bound(s_2, ti)^{\ddagger\ddagger} \wedge unit\_of\_measurement(r, unit\_id, u, a) \wedge measured\_by(r, unit\_id, a) \quad \textbf{(FOL 8)}$$

"The consumption specification term entails that the resource amount will be decremented after the completion of the activity"*** †††.

$$(\forall r, a, s, ti, q', rate, unit)\ consumption\_spec(r, a, ti, q', rate, unit) \equiv (\forall\, s, r, a, q, ti, tp, tp')\ rp(r, q, tp, unit\_id) \wedge ((is\_related(a, s) \wedge consumes(s, r)) \wedge tp = SP(ti) \wedge tp' = EP(ti) \wedge enabling\_state(s, tp, enabled) \supset rp(r, q - q', tp', unit) \quad \textbf{(FOL 9)}$$

"The use specification term entails that the resource amount will remain constant, if the resource is being used"

$$(\forall r, a, s, ti, q', rate, unit)\ use\_spec(r, a, s, ti, q', rate, unit) \equiv (\forall\, s, r, a, q, ti, tp, tp')\ rp(r, q, tp, unit\_id) \wedge (is\_related(a, s) \wedge uses(s, r)) \wedge tp = SP(ti) \wedge tp' = EP(ti) \wedge enabling\_state(s, tp, enabled) \supset rp(r, q, tp', unit) \quad \textbf{(FOL 10)}$$

"The produce specification term entails that the resource amount will increase by a constant, if the resource is being used".

$$(\forall r, a, s, ti, q', rate, unit)\ produce\_spec(r, a, s, ti, q', rate, unit) \equiv (\forall\, s,r,a,q,ti,tp,tp')\ rp(r, q, tp, unit\_id) \wedge (is\_related(a, s) \wedge produces(s, r)) \wedge tp = SP(ti) \wedge tp' = EP(ti) \wedge enabling\_state(s, tp, enabled) \supset rp(r, q + q', tp', unit) \quad \textbf{(FOL 11)}$$

. consumes(s2,r) specifies that state **s2 consumes resource **r**.

††. quantity(s2, q) specifies that state **s2** requires **q** of the resource.

‡‡. time_bound(s2, ti) specifies that **ti** is the time interval of **s2** state

***. Reiter discusses effect axioms in [Reiter 91]. In TOVE, there are number of effect axioms and the committed-related ones are some examples [TOVE 92] [Gruninger & Fox 94].

†††. *SP(ti)* represents that starting point of time period *ti*, while *EP(ti)* represents the end point of time interval *ti*.

- **Continuous vs. discrete resources**: A continuous resource indicates a resource that is uncountable. These resources are marked by uninterrupted extension in volume. Discrete resources on the other hand specify that a resource is countable. These terms are defined relative to an activity.

$$(\forall\ r, a)\ [continuous(r, a) \equiv physical\_divisible(r, a)] \quad \textbf{(FOL 12)}$$

$$(\forall\ r, a)\ [discrete(r, a) \equiv \neg\ continuous(r, a)] \quad \textbf{(FOL 13)}$$

The implication of the above is if the resource is discrete and the consumption or the use specification is defined in terms of integer amounts*.

$$\forall (r, a, q, ti, u)\ (consumption\_spec(r, a, q, rate, ti, u) \vee use\_spec(r, a, q, rate, ti, u)) \wedge discrete(r, a) \supset integer(q) \quad \textbf{(FOL 14)}$$

- **Usage Mode:** Usage mode axiom returns whether a resource supports an activity on a discrete or continuous basis. The term does not imply that the activity is discrete or continuous. The mode of usage is depended on the activity that uses/consumes the resource.

"The mode of usage is achieved through checking the use/consume/produce specification term. If the of the quantity term *(Q)* is equal to the rate term *(Rate)* in the specification, then the process is discrete otherwise the process is continuous with a rate which is equal to the *rate* parameter"

$$(\forall\ r, a)\ continuous\_mode(r, a) \equiv (\exists\ q, unit, rate, ti)\ ((use\_spec(r, a, ti, q, rate, unit) \vee consumption\_spec(r, a, ti, q, rate, unit) \vee produce\_spec(r, a, ti, q, rate, unit)) \wedge q \neq rate \quad \textbf{(FOL 15)}$$

$$(\forall\ r, a)\ discrete\_mode(r, a) \equiv (\exists\ q, u, rate, unit)\ ((use\_spec(r, a, ti, q, q, unit) \vee consumption\_spec(r, a, ti, q, q, unit) \vee produce\_spec(r, a, ti, q, q, unit)) \quad \textbf{(FOL 16)}$$

If a usage mode of a resource is continuous with respect to an activity, that implies that the resource is continuous.

$$(\forall r,a)continuous\_mode(r,a) \equiv continuous(r,a) \quad \textbf{(FOL 17)}$$

*. integer(q) is used specifying that q is an integer

- **Simultaneous Use Restriction**: Simultaneous use restriction prohibits the use/consumption of a resource by two activities simultaneously. For example, this term is used to specify that two activities can not use the resource "oven" simultaneously as both activities require different operating temperature†.

The predicate is defined as ground term with the following parameters:

  - **A1**: activity ID of the first activity using a resource.
  - **A2**: activity ID of the second activity using a resource.
  - **R**: ID of the resource to be used.

`simultaneous_use_restriction(A1, A2, R).` **(PRO 12)**

This term specifies that activities *A1* and *A2* can not be supported by resource *R* at the same time. This entails that both activities can not commit the resource over two overlapping intervals.

$$(\forall\ a1, a2, r)\ simultaneous\_use\_restriction(a1, a2, r) \equiv (\forall\ s, s2, r, a, a2)\ (\neg\ \exists t)\ use(s, a) \wedge uses(s, r) \wedge use(s2, a2) \wedge uses(s2, r) \supset enabling\_state(s, tp, enabled) \wedge enabling\_state(s_2, tp, enabled) \quad \textbf{(FOL 18)}$$

- **Committed to**: This predicate to specifies the commitment of a resource to an activity thereby making the resource partly/fully unusable/inconsumable by any other activity. A resource is committed to an activity as a result of a scheduling activity.

Committed_to is defined as a ground term with the following parameters:

  - **R**: ID of the resource to be used/consumed.
  - **A**: ID of the activity to use/consume the resource.
  - **S**: the ID of the state that is satisfied by the assertion of the committed term.
  - **Ti**: the ID of the time interval of the activity.
  - **Amount**: amount of the resource that is committed to an activity.

†. this term is also use case one activity negatively interacts with the other

- **Unit**: unit of measurement.

`committed_to(R,A,S,Ti,Amount,Unit).` **(PRO 13)**

"A constraint on the *committed to* term is that the time interval of commitment should either be equal or greater than that is defined in the specification".

$$(\forall r, a, s, ti, ti_2, q, q', rate, unit)\ committed\_to(r,a,s,ti,q',u) \wedge (consumption\_spec(r, a, ti_2, q, rate, unit) \vee use\_spec(r, a, ti_2, q, rate, unit)) \supset (contains(ti, ti_2) \vee equal(ti, ti_2)) \quad \textbf{(FOL 19)}$$

- **Total Committed**: This predicate specifies the total amount committed of a resource to all activities at a specified time point. *The total commitment* of a resource is defined to be the summation of all amount committed of resources to all activities. The first order logic would be in the form of:

$$(\forall r, tp, u)\ (\exists TQ)\ total\_committed(r, TQ, tp, u) \equiv (\exists pd1, pd2 \ldots pdn, a_1, a_2 \ldots a_n, q_1, q_2 \ldots q_n)\ committed\_to(r,a_1,pd1,q1,u) \wedge period\_contains(ti, pd1) \wedge (committed\_to(r, a_2, pd2, q2, u) \wedge \ldots\ldots \wedge ommitted\_to(r, a_n, pd2, q_n, u) \wedge TQ = q_1 + q_2 + \ldots + q_n \quad \textbf{(FOL 20)}$$

"An effect of a resource being committed is that after the completion of the activity the total amount committed will be decremented"*.

$$(\forall s, r, a, q, q', ti, tp, tp', u)\ ((use(s, a) \wedge uses(s, r)) \vee (consume(s, a) \wedge consumes(s, r))) \wedge total\_committed(r, q', tp, u) \wedge enabling\_state(s, tp, possible) \wedge (tp = SP(ti)) \wedge (tp' = EP(ti)) \supset total\_committed(r, q - q', tp', u) \quad \textbf{(FOL 21)}$$

- **Capacity**: Capacity is defined to be the maximum set of activities that can simultaneously use/consume a resource at a specific time. In the case where the resource is physically indivisible then the capacity denotes an activity that could use/consume the resource. On the other hand if the resource is physically divisible†, capacity represents the number of activities that a resource can support simultaneously.

The complexity of the process of determining the capacity of a resource depends on the activities requiring the resource and the activities already supported by the resource. Capacity determination problem is reducible to the scheduling problem. The capacity recognition process is solvable in polynomial time in the case where the activities using/consuming the resource are homogenous. Homogeneity implies that activities require equivalent amounts of the resource and processing time or integral multiple thereof. Accordingly the output of the capacity recognition process is reducible to a number which represents the number of activities that the resource could be allocated to. The process becomes complex in the case where the activities requiring the resource are heterogenous. Finding the maximum set of activities is reducible to a single machine scheduling problem which is NP-hard. If the resource is functionally divisible, then the process becomes NP-hard in the strongest sense as the resource can support multiple activities simultaneously. What is required is a sequencing heuristic for a number of activities that are to use or consume a resource in a predetermined time window. The sequencing heuristic is to be defined for a certain objective such as to minimize the number of tardy activities.

One of the terms defined for the capacity recognition process is *available for.* A resource is available for an activity if the resource's quantity can support the activity and there is no *simultaneous use restriction* between the activity requiring the resource and the ones already supported by the resource. Following is a logical specification of this definition:

$$(\forall r)\ (\exists a, ti,)\ available\_for(r, [a], ti) \equiv (\forall tp \in ti)\ (\exists unit\_id, u, amt\_required, tq, q, amount_1, amount_2 \ldots amount_n, rate_1, rate_2 \ldots rate_n)\ (consumption\_spec(r, a_1, ti, amount_1, rate_1, unit) \vee use\_spec(r, a_1, ti, amount_1, rate_1, unit)) \wedge (consumption\_spec(r, a_2, ti, amount_2, rate_2, unit) \vee use\_spec(r, a_2, ti, amount_2, rate_2, unit)) \wedge \ldots\ldots \wedge (consumption\_spec(r, a_n, ti, amount_n, rate_n, unit) \vee use\_spec(r, a, ti, amount_n, rate_n, unit)) \wedge amt\_required = amount_1 + amount_2 + \ldots\ldots + amount_n \wedge (period\_contains(ti, tp) \wedge$$

*. In TOVE, this axiom is called an effect axioms which links the resource ontology with the causal theory of activity [Fadel et al 94].

†. implying having the ability being shared by multiple activities.

$$(total\_committed(r, tq, tp, unit)) \wedge unit\_of\_measurement(r, unit\_id, u, a) \wedge measured\_by(r, unit\_id, a) \wedge rp(r, q, tp, unit) \wedge (amt\_required \geq q\text{-}tq) \wedge ((\forall a_x \in a)\ no\_restricition(r, a, a_x, ti))$$ **(FOL 22)**

$$no\_restricition(r,a,a_x,ti) \equiv (committed\_to(r,a_x,s,ti_2,q,u) \wedge (a \neq ax) \wedge period\_overlaps(ti, ti2) \wedge \neg(simultaneous\_use\_restriction(a, a_x, r)))$$ **(FOL 23)**

- **Activity history**: This predicate specifies the history of usage or consumption of a resource before a specified time point. A list of activities that were supported by the resource will be returned.

  "An activity will be included in the list of activities, that were supported by the resource, if the resource was committed to the activity for a time period with end time less or equal than the specified time point".

$$(\forall r)\ (\exists act\_list, tp)\ activity\_history(r, act\_list, tp) \equiv (\forall a \in act\_list)(\exists ti, q, u, a, s) (committed\_to(r, a, s, ti, q, u) \wedge period\_before(ti, tp) \wedge enabling\_state(s, tp, completed)$$ **(FOL 24)**

- **Resource configuration**: This term specifies the configuration of a resource with respect to an activity. This term implies that for the activity to the resource must have the specified configuration. Moreover, after the completion of the activity, **C** is going to be the configuration of the resource unless changed.

  This term is divided as a ground term with three arguments:

  - **R**: resource ID
  - **C**: ID specifying the configuration of the resource
  - **A**: activity with which the resource has **C** configuration

  `resource_configuration(R, C, A).` **(PRO 14)**

  This implies that if activities a_1 and a_2 require the same resource but both activities require different configuration, then that implies that the resource can not support both activities simultaneously (i.e there is a simultaneous use restriction constraint).

$$(\forall a_1, a_2, r)\ (\exists q_1, q_2, ti_1, ti_2, rate_1, rate_2, c_1) (use\_spec(r, a_1, ti_1, q_1, rate_1, u) \vee consumption\_spec(r, a_1, ti_1, q_1, rate_1, u)) \wedge (use\_spec(r, a_2, ti_2, q_1, rate_2, u) \vee consumption\_spec(r, a_2, ti_2, q_1, rate_2, u)) \wedge resource\_configuration(r, c_1, a_1) \wedge \neg resource\_configuration(r, c_1, a_2) \supset simultaneous\_use\_restriction(a_1, r, a_2)$$ **(FOL 25)**

- **Set up constraint**: Set-up term specifies the duration required to set-up a resource for usage by an activity. The set time is defined with regards to pair of activities. The set-up time includes configuration and location dependent time. In the configuration set-up time, the specified duration as a function of the time required to change the resource's configuration state, as result of the last activity supported, to that implied by the activity requiring the resource. As for the location set-up time, it specifies the duration required for the resource to transport or be transported from one location to another.

  "Set up time is equal to the time required to change the resource's configuration. If the resource needs to relocated, then the set up time also includes the time of transportation"

$$(\forall r, a_2, l_2, dur, u)\ set\_up(r, a_2, l_2, dur, u) \equiv (\exists a_1, ti, q, c, tp_1, l_1, ct, lt, u_2, s_1) committed\_to(r, a_1, s_1, ti, q, u) \wedge tp = EP(ti) \wedge resource\_configuration(r, c, a) \wedge rpl(r, q, tp_1, l_1, u) \wedge (config\_set\_up(r, a_1, a_2, ct, u_2) \wedge loc\_set\_up(r, l_1, l_2, lt, u_2) \wedge dur = ct + lt)$$ **(FOL 26)**

  The set-up duration is the summation of the configuration and location dependent set-up times. If the resource is not to be moved from a location*, then the set up time id only the time needed to re-configure the resource.

  Configuration set-up time (config_set_up) is defined as a ground term with these arguments:

  - **R**: resource ID
  - **A1**: the activity that caused the last configuration change of the resource

*. i.e already in the location required by the activity

- **A2**: activity requiring the change in the configuration of the resource
- **CT**: configuration dependent set-up time
- **Unit**: temporal unit of measurement

```
config_set_up(R, A1, A2, CT,
Unit).
```
(PRO 15)

Location set-up time is defined as a ground term with five arguments:

- **R**: resource ID
- **L1**: the location from which the resource is to be relocated
- **L2**: destination of the resource
- **LT**: location dependent set-up time
- **Unit**: temporal unit of measurement

```
loc_set_up(R, L1, L2, LT, Unit).
```
(PRO 16)

- **Alternative resource**: This term specifies an alternative resource(s) to be used or consumed by an activity. This is useful in the case when an alternative resource is required because of a machine breakdown or unavailability of a resource.

  "There exist an alternative resource for an activity, if there exists a disjunct non-terminal enabling state"

$$(\forall\ r, a)\ (\exists\ list)\ alternative\_resource(r, a, list) \equiv (\exists\ s, s_2, disjunct\_state)\ uses(s_2, r) \wedge is\_related(s_2, s) \wedge subclass\_of(s, disjunct\_state)$$ **(FOL 27)**

- **Relation between resource ontology and activity-state ontology**: A *state* in TOVE represents what has to be true in the world in order for an activity to be performed, or what is true in the world after the completion of an activity. The status of a state, and any activity, is dependent on the status of resources that the activity uses or consumes. One of the status that a state could have is *possible*. A consume state* could be possible if:
  - the resource is available for the activity and
  - the resource has not been committed yet to the resource and
  - the activity is not executing.

$$(\forall\ state\_id)\ (\exists\ tp)\ possible(state\_id, tp) \equiv (\exists\ r, a, ti, unit)\ (consume(state\_id, a) \vee consumes(state\_id, r)) \vee (use(state\_id, a) \vee uses(state\_id, r)) \wedge available\_for(r, a, ti) \wedge \neg committed\_to(r, a, state\_id, ti, amount, unit) \wedge period\_contains(ti, tp) \wedge \neg activity(a, executing, tp)^{\dagger}$$ **(FOL 28)**

Moreover a consume state is completed if:

- the activity is in the activity history of the resource.

$$(\forall\ state\_id)\ (\exists\ tp)\ completed(state\_id, tp) \equiv (\exists\ r, a, act\_list)\ ((consume(state\_id, a) \wedge consumes(state\_id, r)) \vee (use(state\_id, a) \wedge uses(state\_id, r))) \wedge activity\_history(r, act\_list, tp) \wedge member\_of(a, act\_list)$$ **(FOL 29)**

4. Examples of competency questions:

Recall that the competency questions are the basis (starting point) of ontology development. Similar to the approach in defining the ontology, competency questions are defined starting with the most simple questions‡. What is presented in this section is some examples of competency questions and their Prolog implementation**; it shows how the ability to answer a query depends on the availability of number of ontologies.

☞ ☞ *Is there enough capacity for the performance of "fabricate_short_shaft_1" and "fabricate_short_shaft_2" activities on the "mold1" resource in the next hour?* (Given a number of activities requiring the resource ***mold1***, what is the ***capacity*** of the resource during a time period?)

```
|?- available_for(mold1, [fabricate_-
short_shaft_1, fabricate_short_-
shaft_2], pd3, Result)††.
```

*. i.e the state of an activity consuming a resource.

†. activity(a, executing, tp) specifies that activity **a** is *executing* at time point **tp**.

‡. i.e the competency are stratified.

**. A query in Prolog is proceeded by "|?-". A variable is represented as capital letters.

††. Figures 3 and 4 present a taxonomy of ontology and axioms required to answer the query.

figure 3 ”available for” requires the following ontology

Requires

```
|?- functional_division_of(R, mold1)
no.
```

division_of is usedto reason about the sharability of the resource resource.(i.e used indivisible term)

```
|?- physical_division_of(R2, mold1).
R2 = rectangle_1;
no
|?-unit_of_measurement(mold1,Unit_ID
   U, fabricate_short_shaft_1).
Unit_ID = rectangle_1
U = rectangle_1;
no
|?- functional_divisible(mold1,
    fabricate_short_shaft_1).
yes
|?- use_specification(mold1,
   fabricate_short_shaft_1,pd1,30,30,
   recatngle_1).
Q = 30;
no

|?- total_committed(mold1, TQ, tp,
    rectangle_1).
TQ = 30;
no

|?- rpl(mold1, Q, tp2, rectangle_1).
Q = 150;
no

|?- simultaneous_use_restriction(
    fabricate_shirt_shaft_1,
A2, mold1).
no.
```

simultaneous_use_restriction is used so that no two confliction activities can use resource simultaneously

continued next figure

figure 4 ”available for” requires the following ontology -cont’d

continuation of last figure

```
|?- has_current_activity(mold1,List,
.   tp).
```

has_current_activity is used to to find out if the activities requiring and supported by the resource are homogenous/ heterogenous

Requires

```
|?-committed_to(mold1,A,S,_,tp,
     rectangle_1).
   A = fabricate_short_shaft_2
   S = es_fabricate_short_shaft_2;
   no

|?-enabling_state(
    fabricate_short_shaft_2,tp,
    Status).
  Status = enabled;
  no

List = assemble_clip_reading_lamp,
     assemble_hand;
     no


temporal ontology (e.g before, after)


Result = [fabricate_short_shaft_1];
no
```

the resource has the capacity to only support fabricate_short_shaft_1 activity

A smooth production requires the availability of resources at the time of productions. Resources should be replenished before the stock reaches a certain limit and that is checked through using the following query.

☞☞ Is the stock for `short_shaft` in danger of being depleted between now and the end of the week?*

```
|?- rp(short_shaft, Q, Tp), Tp > now,
Tp < tp333, Q < 20.
```

*. i.e will the resource point fall below then a certain value

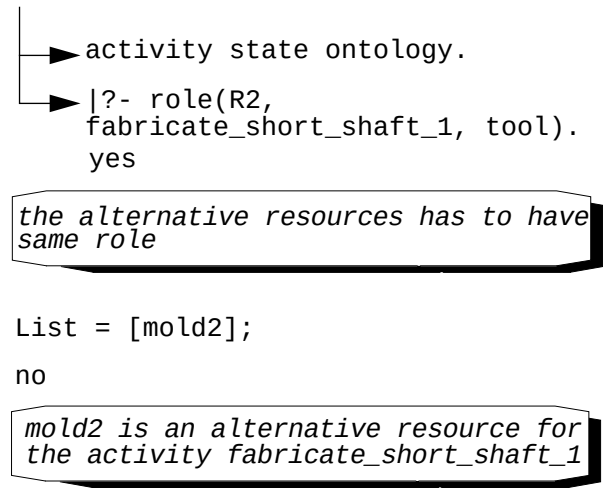
```
Q = 15
Tp = tp23;
Q = 10
Tp = tp200;
no
```

Assignment of resources (or alternative resources) is a crucial function in a manufacturing environment. This is used in case of resource unavailability, for example, due to a machine breakdown or delay in a shipment.

☞☞ *Can resource* `mold2` *be* ***used instead*** *of resource* `mold1`? (Is mold2 an ***alternative resource*** of mold1 for activity `fabricate_short_shaft`?)

```
|?- alternative_resource(mold1, fab-
ricate_short_shaft, List), member_-
of(mold2, List).
```

figure 5 "alternative resource" requires

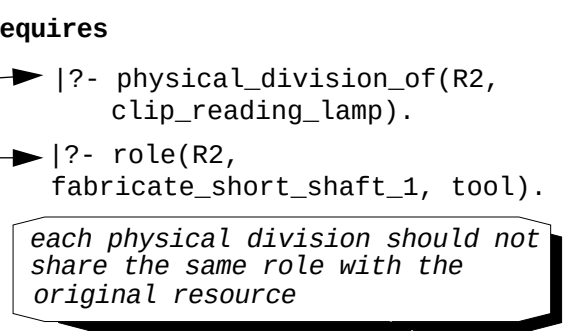


Another important function in planning is to be able to explode the bill of material of a resource.

☞☞ *What are the components of resource* `clip_reading_lamp`*?* (What are the ***physical component of*** the resource?)

```
|?- physical_component_of(R,
clip_reading_lamp, A, Type).
```

figure 6 "physical component of" requires




```
R = clip_base
A = assemble_clip_reading_lamp
Type = physical;
R = short_arm
A = assemble_clip_reading_lamp
Type = physical;
R = small_head
A = assemble_clip_reading_lamp
Type = physical;
no
```

5. Conclusion

In this paper we have presented an ontology for resources in a manufacturing enterprise environment. This ontology has the characteristics of being generic and reusable over a wide variety of applications. The ontology provides the capability of deductively answering common sense questions about the enterprise knowledge. In particular it us to reason about how properties of resources change as the result of activities, and also to reason about the allocation of resources in a scheduling task through capacity recognition. This work lays the foundation for further research in additional planning, scheduling and modelling tasks in enterprise engineering.

6. Acknowledgments

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

7. References

[Esprit 90]ESPRIT-AMICE. CIM-OSA - A Vendor Independent CIM Architecture. *Proceedings of CINCOM 90*, pages 177-196. National Institute for Standards and Technology, 1990.

[Fadel 94]Fadel, Fadi George, *Microtheory of resources*, M.A.Sc thesis, University of Toronto, to be published 1994.

[Fox 83] Fox, M.S.,*The Intelligent Management System: An Overview*, Processes and Tools for Decision Support, North-Holland Publishing Company, 1983.

[Fox & Tenenbaum 90] Fox, M.S., and Tenenbaum, J.M., (1991), *Proceedings of the DARPA Knowledge Sharing Workshop*, Santa Barbara Ca.

[Fox et al 93]Fox, Mark S., Chionglo, John F., Fadel, Fadi G., *Towards Common Sense Modelling of an Enterprise*, Proceeding of the Second Industrial Engineering Research Conference, 1993.

[Fox & Gruninger 94] Fox, Mark S., Gruninger, Michael, *Ontology for activity for Enterprise Engineering*, submitted to Twelfth National Conference on Artificial Intelligence (AAAI-94).

[Gruber 91]Gruber, Thomas R., *The Role of Standard Knowledge Representation for Sharing Knowledge-Based Technology*, To appear in: Allen, J. A., Fikes, R., and Sandewell, E. (Eds) Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, 1991.

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

[Martin 83]Martin, C., and Smith, S. *Integrated Computer-aided Manufacturing (ICAM) Architecture Part III/Volume IV: Composite Information Model of "Design Product" (DES1).* Technical Report AFWAL-TR-82-4063 Volume IV, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, 1983.

[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.

[Scheer 89]Scheer, A-W. *Enterprise-Wide Data Modelling: Information Systems in Industry.* Springer-Verlag, 1989.

[TOVE 92]Fox, Mark S., Chionglo, John, Fadel, Fadi George, *The **TO**ronto **V**irtual **E**nterprise model*, Enterprise Integration Laboratory, University of Toronto, 1993.

[Williams 91]Williams, T.J., and the Members, Industry-Purdue University Consortium for CIM. *The PURDUE Enterprise Reference Architecture.* Technical Report Number 154, Purdue Laboratory for Applied Industrial Control, Purdue University, West Lafayette, IN 47907, 1991.