A RESOURCE ONTOLOGY FOR ENTERPRISE MODELLING

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: Planning, scheduling, ordering (etc.) activities are often described in terms of 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 formal model of resources. In particular, it describes the ontology and semantics for modelling resources. An ontology is comprised of data model which is composed of objects, attributes, relations and formal definitions of constraints and terms in the data, defined in first order logic. This permits the sharability and reusability of information between different applications.

INTRODUCTION

This paper presents a formal model of resources. In particular, it describes the ontology for modelling resources. An *ontology* is comprised of data model which is composed of objects, attributes, relations and formal definitions of constraints and terms in the data, defined in first order logic. An ontology permits the sharability and reusability of different domain/application knowledge so that different enterprise departments can share and communicate information.

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 are passive. They do not have the capability to automatically deduce the obvious about what it is representing. For example, if the representation contains a `works-for' relation and it is explicitly represented that Joe `works-for' Fred, and that Fred `works-for' John, then the obvious deduction that Joe `works-for' John (indirectly) cannot be made within the representation system. The lack of a `common-sense' deductive capability forces users to spend significant resources on programming each new report or function that is required.

INSTITUTE OF INDUSTRIAL ENGINEERS
3rd Industrial Engineering Research Conference Proceedings

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 [4], creating an 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 "Common Sense Enterprise Modelling" project in the Enterprise Integration Laboratory at the University of Toronto. The project is creating ontologies for: activities/states, causality, time, products, constraints, cost, quality, etc. The ontologies have the characteristics of being generic, reusable and sharable, allowing different agents to share a knowledge base and at the same time minimizes the ambiguity of the representation. Secondly, the Prolog definitions of terms provide a deductive model (aka deductive database) which can be used to answer *simple* common sense questions without additional programming. In a study performed, over 30 managers from three different Westinghouse Corporation plants were asked to record the questions frequently asked [3] many of which can be answered deductively. Examples of the questions are:

- 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?

The test bed for the resource ontology is **TOVE*** [9]. TOVE is a virtual enterprise which produces desk lamps. The TOVE Enterprise Ontology provides a generic, reusable ontology for modelling enterprises. 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™ knowledge representation tool and the axioms are implemented in Quintus Prolog™.

EVALUATION CRITERIA

Though there exists a number of efforts seeking to create sharable representation of enterprise knowledge - CIM-OSA [1], ICAM [7], IWI [8], PERA [10] - 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* [4] [5]. Gruber has proposed: clarity, coherence, extensibility, minimal encoding and minimal and ontological commitment [6].

*. TOronto Virtual Enterprise.

The criterion we have chosen to evaluate our work is *competence.* Competency is chosen as the main criterion as these questions define the span of the model inaddition to being the starting point for ontology definition by defining what is required to be represented. The competence of a representation defines the types of tasks that the representation can be used in. The obvious way to demonstrate competence is to define a set of questions that can be answered by the ontology. If no inference capability is to be assumed, then question answering is strictly reducible to "looking up" an answer that is represented explicitly. In defining a shared representation, a key question then becomes: should we be restricted to just a terminology? Should the terminology assume an inheritance mechanism? Artificial Intelligence knowledge representations and object-oriented representations assume at least inheritance as a deduction mechanism. Or should we assume that some type of theorem proving capability is provided, say, in a logic programming language with axioms restricted to Horne clauses (i.e., Prolog)? What is the *deductive capability* that is to be assumed by an ontology? We propose that for each category of knowledge, a set of questions be defined that the ontology can answer. Given a representation and an accompanying theorem prover (perhaps Prolog), questions can be posed in the form of queries to be answered by the theorem prover. Given that a theorem prover is the deduction mechanism used to answer questions, the *efficiency* of a representation can be defined by the number of LIPS (Logical Inferences Per Second) required to answer a query.

Competency Requirements

As discussed earlier, the competence criterion focuses on how well the model supports various types of problem solving. We define a model's level of competence by a set of questions it should be able to answer either directly or through deduction. Following are a subset of questions we have considered in the creation of the **TOVE** model.

- **Quantity:** What is the stock level at time *t*? When will I run out of resource *R*?
- **Consumption:** Is the resource consumed by the activity? If so, how much?
- **Divisibility:** Can the resource be divided and still be usable? Can two or more activities use the resource at the same time?
- **Capacity:** Can the resource be allocated to multiple jobs simultaneously?

- **Commitment:** Which activities are currently being supported by the resource?
- **Trend:** What is the capacity trend of a resource based on the machine usage history?

RESOURCE ONTOLOGY

Ontological Engineering is the process by which a domain is analyzed to ascertain the terminological primitives. 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.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. 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. In our analysis, we have strived to identify the primitive resource properties on which more complex properties, such as trend recognition, are defined. Following are some examples of resource ontology* †.

- **Resource role**: According to webster a role is "an identifier attached to an index term to show functional relationships between terms". In TOVE, a resource has a role with respect to an activity. Examples of such roles are: raw material, product, facility, tool, operator.

```
role(R, A, Role).
```
(PRO 1)

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.

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

- **Division of**: This term specifies that a resource could be divided and one of the divisions is R2. 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 body of an object; "functional division of" specifies a division affecting the function and not structure.

```
physical_division_of(R2, R).
```
(PRO 2)

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

- **Divisibility of a resource**: This term specifies the property of a resource of being divisible with respect to an activity without affecting the role of the resource. Divisibility has three types: 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 an 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 "motor car" has a functional and physical division (e.g crank shaft) but the "motor" is neither functionally nor physically divisible with respect to "driving the 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 as in the case of the multiplex lines when associated with *communication activities*.

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

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

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

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

"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)."

*. For more examples and details please refer to [2].

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

$$\forall (r, a)\ temporal\_divisible(r, a) \equiv$$
$$\exists\ (ti, ti_1, ti_2, tp_1, tp_2, a_1, a_2, s_1, s_2, role_1)$$
$$(uses(s_1, r) \vee consumes(s_1, r)) \wedge$$
$$(uses(s_2, r) \vee consumes(s_2, r)) \wedge$$
$$is\_related(a_1, s_1)^{*} \wedge is\_related(a_2, s_2) \wedge$$
$$time\_bound(s_1, ti_1)^{\dagger} \wedge time\_bound(s_2, ti_2) \wedge$$
$$activity(a_1, executing, tp_1)^{\ddagger} \wedge period\_contains(ti_1, tp_1)^{**} \wedge$$
$$((activity(a_1, suspended, tp\_end) \wedge tp\_end = EP(ti_1)^{\dagger\dagger}) \vee$$
$$((activity(a_1, completed, tp\_end) \wedge tp\_end = EP(ti_1)) \wedge$$
$$activity(a_2, executing, tp_2) \wedge period\_contains(ti_2, tp_2) \wedge$$
$$\neg overlaps(ti_1, ti_2) \wedge contains(ti, ti1) \wedge contains(ti, ti2) \wedge$$
$$role(r, a_1, role_1) \wedge role(r, a_2, role_1) \quad \textbf{(FOL 4)}$$

- **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 a countable one. These terms are defined relative to an activity.

$$(\forall\ r, a)\ continuous(r, a) \equiv physical\_divisible(r, a) \quad \textbf{(FOL 5)}$$

$$(\forall\ r, a)\ discrete(r, a) \equiv \neg\ continuous(r, a) \quad \textbf{(FOL 6)}$$

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[‡‡***].

$$(\forall\ r, a, q, ti, u)\ (consumption\_spec(r, a, q, ti, u) \vee use\_spec(r, a, q, ti, u)) \wedge discrete(r, a) \supset integer(q) \quad \textbf{(FOL 7)}$$

- **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 4)**

- **Measured by**: "Measured by" defines the objects by whdich 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 5)**

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

$$(\forall\ r, a, unit\_id)\ measured\_by(r, unit\_id, a) \supset (\exists\ u)\ unit\_of\_measurement(r, unit\_id, u, a) \quad \textbf{(FOL 8)}$$

- **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. 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".

$$\forall (r_1, r_2)\ physical\_component\_of(r_2, r_1) \equiv \forall (a, r, ro_1)\ physical\_division\_of(r_2, r_1, a) \wedge role(r_2, a, ro_1) \supset \neg role(r_1, a, ro_1) \quad \textbf{(FOL 9)}$$

"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".

$$\forall (r_1, r_2)\ functional\_component\_of(r_2, r_1) \equiv \forall (a, r, ro_1)\ functional\_division\_of(r_2, r_1, a) \wedge role(r_2, a, ro_1) \supset \neg role(r_1, a, ro_1) \quad \textbf{(FOL 10)}$$

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

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

†. 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.

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

***. **use and consumption spec** specify the requirement of an activity to use or consume a resource. These terms are defined in the activity-state ontology.

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

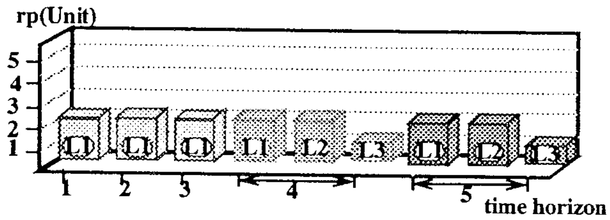
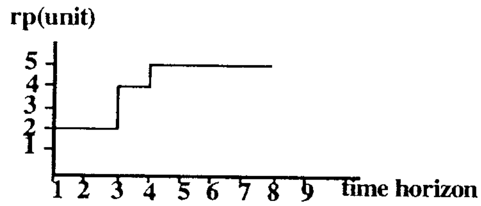


figure 2 2-D resource point - in terms of quantity and time



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

```
rpl(plug, 100, 90, ss12, unit).
```
(PRO 6)

For example the above assertion, 3-D *rpl* definition, specifies the existence of a resource point for resource *plug* at time point '90', with quantity of '100' units at location ss12.

```
?- rp(resource, Q, time, unit).
```
(PRO 7)

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.

- **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/functionally indivisible then the capacity denotes an activity that could use/consume the resource. On the other hand if the resource is physically/ functionally 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. 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 or 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 physically/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:

∀(r, a, time_interval) available_for(r, a, time_interval) ≡
(∃ amount_required, unit)
(consumption_spec(r, a, time_interval,amount_required, unit) ∨
use_spec(r, a, time_interval, amount_required, unit)) ∧
(∀(time_point, tq)(contains(time_interval, time_point) ∧
*(total_committed(r, tq, time_point, unit)***) ∧*
rp(r, q, time_point, unity) ∧
(q - amount_required - tq >= 0) ∧

*. rp(resource, quantity, time_point, location, unit_of_measure).

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

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

**. total_committed(r,tq,time_point,unit) term specifies the total amount committed of a resource at a specific time point

$$(\forall a)\ no\_restriction(a, a_2, r)) \quad \textbf{(FOL 11)}$$

$$no\_restriction(a, a2, r) \equiv (\exists\ time\_interval_2, q,)(committed\_to(r, a_2, time\_interval_2, q, u) \wedge \neg(a = a2) \wedge overlaps(time\_interval, time\_interval_2) \wedge \neg simultaneous\_use\_restriction(a, a_2, r)) \quad \textbf{(FOL 12)}$$

- **Status of activities and its relation with resource activity**: 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 to be consumed.

$$(\forall\ state\_id)\ (\exists\ tp)\ enabled(state\_id, tp) \equiv (\exists\ r, a, act\_list)\ ((consume(state\_id, a) \wedge consumes(state\_id, r)) \vee (use(state\_id, a) \wedge uses(state\_id, r))) \wedge has\_current\_activity(r, act\_list, tp) \wedge member\_of(a, act\_list) \quad \textbf{(FOL 13)}$$

The above specifies that a use state could be enabled at a time point if the activity is currently using the resource (i.e executing).

CONCLUSION

It has become apparent that a competitive and efficient enterprise does not solely depend on capital and management, but also depends on the accessibility of information and the ability to coordinate both decisions and actions. Organizations are striving towards an efficient coordination between organizational units. However this effort is often hampered, especially in large organizations. The problem arises because of the inability of organizational units to share information and coordinate activities. Efficient communication among different organizational enterprise units exists if and only if there exists a common understanding, efficient coordination and controlled accessibility of information. What we have presented in this paper some examples of generic and reusable *resource* modelling objects enabling different application to share and reuse information via off-shelf modelling building blocks.

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.

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

REFERENCES

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

[2] Fadel, Fadi George, *A Resource Ontology for Enterprise Modelling*, M.A.Sc thesis, University of Toronto, to be published 1994

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

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

[5] 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.

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

[7] 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.

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

[9] Fox, Mark S., Chionglo, John, Fadel, Fadi George, Gruninger, Michael, *The TOronto Virtual Enterprise model*, Enterprise Integration Laboratory, University of Toronto, 1994, To appear.

[10] 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.