A generic task ontology for scheduling applications

A Generic Task Ontology for Scheduling Applications

Dnyanesh Rajpathak[1], Enrico Motta[1] and Rajkumar Roy[2]

[1]Knowledge Media Institute
The Open University
Walton Hall, Milton Keynes,
MK7 6AA, UK.

[2]Enterprise Integration
Cranfield University
Cranfield, Bedford,
MK43 0AL, UK.

Abstract

An ontology can be seen as a reference model to describe the entities which exist in an universe of discourse and their properties. These entities may be individuals, classes, relationships, and functions. In sum anything that may be useful to describe specific models. In this paper we present a generic task ontology for scheduling problems. The ontology is generic in the sense that it is both domain and application independent. We refer to it as a 'task ontology' to emphasise that it describes the class of scheduling tasks, independently of the various ways by which these tasks can be solved. The proposed task ontology has been successfully validated to measure its knowledge capturing capability. Our aim is to move beyond current brittle approaches to system development to provide firm theoretical and engineering foundations to various classes of knowledge-based applications.

Keyword: Intelligent Scheduling, Ontologies, Knowledge Modelling, Knowledge Acquisition, Reuse.

1. Introduction

In generic terms the scheduling task can be characterised as *an assignment of time-constrained jobs to time-constrained resources within a pre-defined time framework, which represents the complete time horizon of the schedule. An admissible schedule will have to satisfy a set of hard and/or soft constraints imposed on jobs or resources. The output of a scheduling task is a legal schedule in accordance with a given solution criterion (e.g. complete, admissible)* as compared to [17,21].

If we look at scheduling as a constructive design process, we can say that its main building blocks are time related activities [5]. These activities may differ according to the target-domain and depend on the level of granularity of the application. Unfortunately, this changing nature of the target-domain increases the overall cost and time required for developing the application system. In order to overcome this serious bottleneck a need for reusable components arises in system development [12].

Ideally, we would like to have components that can efficiently be reused across wider domains that can support both the acquisition of the relevant scheduling knowledge and the system development process. Ontology can be viewed as a conceptual information model describing the various entities that exist in a particular domain of discourse such as classes, relationships, and functions [19]. In this paper we present *generic task ontology* for scheduling. We refer to it as a *'task ontology'* to emphasise that it describes the class of scheduling tasks independently of the various ways by which these tasks can be solved. The task ontology is generic in the sense that it is both domain and application independent. In addition to these engineering concerns, the role of the proposed task ontology is also to provide a clear specification of the class of scheduling applications. Although, scheduling has been studied in detail by several authors [13,17,21], and there have been some attempts at developing ontologies for scheduling [1,10,18], these ontologies tend to be fairly coarse-grained and some times are committed to specific domains. The *cost* related issues are not usually expressed, along with the various *preference* criterions in the scheduling domain. More importantly none of the aforementioned approaches provides the desired level of detail and formalisation.

The aim of this paper is therefore to describe our initial work aimed at putting scheduling on firmer ontological foundations. The paper is organised as follows. In the next section we present the scheduling problem. In 2.1 we describe the main concepts in the scheduling task ontology as a class along with its attributes. In 2.2 we define the main axioms in the task ontology. In section 3 we briefly compare our work with other approaches. In section 4 we describe the validation of the task ontology carried out in two domains. Finally, in section 5 we conclude the paper by reiterating the contribution of this work and highlight some issues that require further investigation.

2. A Generic Specification of Scheduling Task

A scheduling task can be represented as a mapping from a seven-dimensional space {J, R, H_C, S_C, Str, P, Cf} to the space of solutions for the schedule {S_{sol1}, …. ,S_{soln}}. The components of the scheduling task are specified as follows.

J = Jobs = a set of jobs that can be assigned to a set of resources = {j_1, …. ,j_n};

R = Resources = a set of available resources to which jobs can be assigned = {r_1, …. ,r_n};

Hc = Hard constraints = a set of hard constraints which must not be violated by the schedule solution = {h_{c1}, …. ,h_{cn}};

Sc = Soft constraints = a set of soft constraints which can be relaxed if necessary to reach the schedule solution = {s_{c1}, …. ,s_{cn}};

Str = Schedule time-range = the complete time horizon of the schedule = {s_{tr1}, …. ,s_{trn}};

P = Preferences = a set of preferences which can be used to define the criterion for choosing among the competing schedule solutions say, S_1 and S_2 = {P_{r1}, …. ,P_{rn}};

Cf = Cost function = is a function that computes a cost to the final schedule solution.

In accordance with the above-mentioned inputs, it is now possible to define the following types of criterions for the schedule solution.

- A schedule, say Si, is *complete c*, if all the jobs are assigned to the available resources by the completion of the schedule.
- A schedule is m*inimally admissible min-a*, if all the sets of hard constraints are satisfied.
- A schedule is m*aximally admissible max-a*, if it is minimally admissible (it satisfy all the hard constraints) and satisfies all the soft constraints as well.
- A schedule is legal, *schedule-solution* (S_{sol}), if it is both *complete* and *maximally admissible*.
- A schedule solution say Ssol-d, is *optimal* $Cf\ (S_{sol\text{-}d}) < Cf\ (S_{sol\text{-}r})$, iff no other schedule has a lower cost than $S_{sol\text{-}d.}$ In other words the cost of the final schedule solution, $S_{sol\text{-}d}$, computed by applying the cost function, is optimal to any other schedule solution $S_{sol\text{-}r.}$

A figure 1 and 2 depicts the generic inputs of the scheduling task ontology with class and relation diagram respectively.

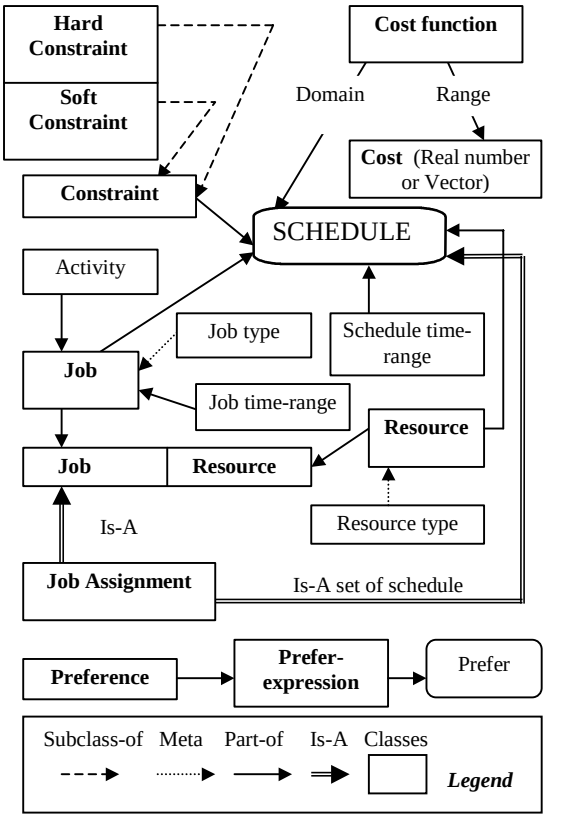

Figure 1. Framework Model of the Scheduling Task Ontology.

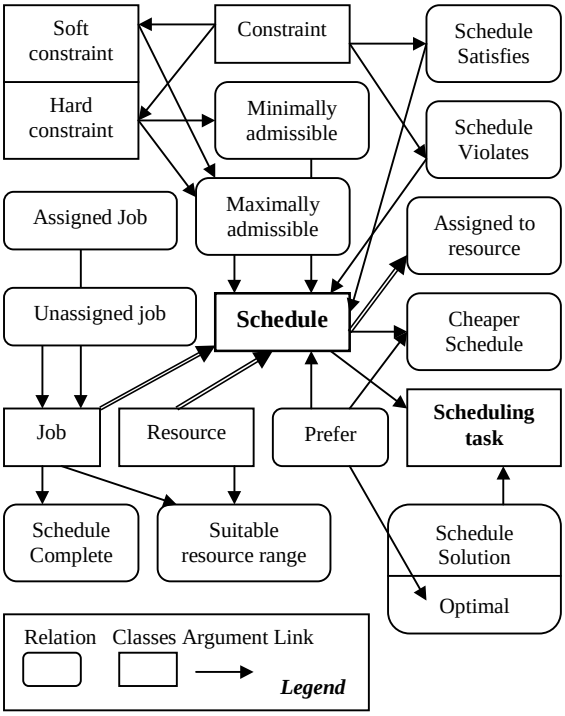

Figure 2. Relation diagram for the classes in the Task Ontology.

The boxes in Figure 1 indicate the classes in the task ontology and the arrows between any two boxes represent the `Subclass-of`, `Meta`, `Part-of` and `Is-A` hierarchy accordingly. The concepts shown in bold in Figure 1 indicate the external viewpoint imposed by the task ontology. In Figure 1 and 2, the boxes with the

rounded corner indicate the relations and the rectangles represent the classes in the task ontology. In figure 2 the arrows between the classes and relations represent the argumentation among them. For example, a class `schedule` is `maximally admissible` if it satisfies every `hard constraint` as well as `soft constraint`. We take the point of view that scheduling is an assignment of jobs to resources but the inverse of this assignment is not always true, i.e. resources may remain unassigned when a scheduling task is completed. Our task ontology is based on a *job centred* point of view [1,9,13,18].

2.1 The Scheduling Task Ontology

In this subsection we describe the major inputs of the task ontology depicted in Figure 1 and 2 from the knowledge modelling perspective by using the following structure. First, we will give the definition of the class, and then its attributes in terms of slots (which represent the binary relation) within those particular classes [3]. The slots are represented as *italics* in the definition of classes. It is important to keep in mind that all these slots are defined separately as a *class* or a *relation* in the task ontology depending upon the requirement. Finally, we will describe the main axioms developed in the task ontology.

This scheduling task ontology comprises about 54 definitions, but the permitted space does not allow us to discuss all these concepts in detail. Here, we will discuss the major modelling decisions taken while developing the task ontology. In doing so, we assume the existence of a *time ontology* and other main *base ontologies* [3]. The base ontologies provide the definitions for a basic modelling concepts such as tasks, relations, methods, roles, numbers, sets etc.

Class JOB

Definition. A job represents the most abstract class that involves various activity-ranges and can be assigned to the resource for its execution.

Activity-Range: this slot represents the fact that every job can have a range of activities that need to be performed in order to accomplish the job (see class activity).

Suitable-Resource-Range: this slot explains that the job has a set of suitable resource ranges on which it can be assigned for its execution.

Time-Range: this slot inherits the values of the class *Job-time-range*, which represents the earliest and latest start and finish times of the job along with the unit of time.

In the task ontology a distinct relation is defined, called `assigned-to-resource`, which actually models the assignment of a job to a resource as an element of the class `schedule` (shown by the double arrows in Figure 2).

Class JOB-TYPE

Definition. All the instances of a class job-type are the subclasses of class `job`. For instance, in the manufacturing environment if the job is *machining* then the job-type is *drilling*, *milling* etc.

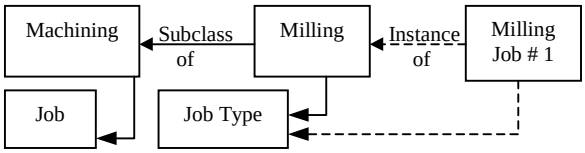

Figure 3. The representation of a job and job-type hierarchy.

Class ACTIVITY

Definition. The activity is something that represents the various sets of operations for any given job. *This offers the scope for the detailed breakdown of job*. It is defined as follows, J = $\{j_1, \ldots, j_n\}$, where ji = $\{a_{i1}, \ldots, a_{in}\}$. For example, if the job is *drilling* then its set of activities could be *the machine set-up, the loading, the actual drilling operation* and *the unloading of the job* etc.

Fixed-Duration: this slot indicates that every activity has a fixed duration for its execution. The cumulative all these duration represents the total duration of job.

Class RESOURCE

Definition. A resource represents the most abstract class to which the jobs can be assigned for their execution.

Handles-Job-Type: the purpose of this slot is twofold. First, this slot explicitly represents the type of job/s that resource can handle. Second, by giving the cardinality value it is capable of handling *n* number of jobs $\{j_1, \ldots, j_n\}$, provided that these jobs adheres to the `resource-availability` axiom (see resource-availability axiom by the end of this section).

Available-Duration: this indicates the duration of the resource for which it can be available to perform the assigned job.

In order to maintain the consistency of this duration, the relation is established between the job duration with that of the available duration of resource. This relation imposes the constraint that the duration of the job must be less than that of the available duration of the resource.

Competence: it is a qualitative measure (*yes/no*) of the resource competence, which shows if the resource is competent to execute the assigned job.

Class RESOURCE-TYPE

Definition All the instances of a class resource-type are the subclasses of class `resource` e.g. type of machine, type of transport vehicle, etc.

Class CONSTRAINT

Definition The class `constraint` has the same definition for both hard and soft constraints. These are modelled as distinctive subclasses of class `constraint`. The hard constraints are the constraints that must not be violated under any circumstances, where the soft constraints have to be satisfied by the completion time of schedule. For example, the due date (soft) constraint often need to be relaxed for the couple of jobs due to limited capacity of the production activity [21].
Both the constraints are applied on a job or a resource through the class `schedule`, which helps to satisfy both minimal as well as the maximal admissibility conditions of a solution.

Has-Expression: such expression has a number of advantages. First, it allows us to reason about the hard and soft constraints and to attach the properties to them. It also specialises the constraints according to specific classes of the scheduling applications. In particular, this expression is parameterised in terms of pairs of `job-resource`, i.e. a job-assignment for a class schedule.

Applicability-Condition: this condition gives us the scope to maintain the truth status of the class schedule. This validates whether the hard and soft constraints associated with jobs or resources are satisfied by the schedule solution.
By using the above-mentioned `has-expression` which states that the job has to satisfy the both constraints is used for satisfying the legal, *schedule-solution* condition.

Class JOB-TIME-RANGE

Definition. This class indicates the complete time range of an individual job. It is specified as a start and end time for a particular job in terms of the following slots, earliest start-time, earliest end-time and latest start-time, latest end-time of the job, along with the unit-of-time. It indicates the unit in which time is expressed.

Earliest-Start-Time: (in terms of a time-point), this shows how early a particular job can start.

Latest-Start-Time: (in terms of a time-point), this shows how late a particular job can start and still not violate the given time-range of the schedule.

Earliest-End-Time: (in terms of a time-point), this shows how early a particular job can finish.

Latest-End-Time: (in terms of a time-point), this shows how late a particular job can finish.

Unit-of-Time: this simply indicates the unit in which the time is specified, e.g. second, hour etc. We take the point of view that if the earliest and latest of the start-time and the end-time are not mentioned explicitly in the problem, then in such cases the *start-time* and *end-time* will be used for representing the allowed time-range of the job.

Class SCHEDULE-TIME-RANGE

Definition. This time-range represents the start and end time of the schedule, which is the total time horizon for which the schedule is constructed. The unit-of-time simply indicates the unit in which the time is specified.

Start-Time: the start time of the schedule.

End-Time: the end time of the schedule.

Unit-of-Time: the unit of time.
In the task ontology a separate relation is defined, called `time-range-between-job-and-schedule`. It is a binary relation between the time-range of a job and a schedule. This relation imposed the constraint which states that the start-time of the first job must be greater than or equal to the start-time of the schedule and the end-time of the last job must be less than or equal to the end-time of the schedule. Hence, it avoids from overshooting a complete time horizon of the schedule. For example, if the schedule starts at (9.00am hour); so the start-time of the first job to be scheduled is either (9.00am hour) or (9.01am hour). Similarly, if a schedule finishes at (6.00pm hour); the end of the last job is either (6.00pm hour) or (5.59pm hour).

Class PREFERENCE

Definition. This class is represented by a `prefer-expression`. Such expression helps

in ranking the various schedule solutions according to some criterion. It is important to keep in mind that the difference between the soft constraint and preferences is rather conceptual than formal. It is expressed by means of a `prefer` relation. It is a binary relation that defines the partial order preference over any two schedules say, S_1 and S_2 depending upon the real life preferences. In our task ontology we choose the o*ptimum* schedule based on the cost specific preferences in scheduling. This allows us to reflect the impact of various real-life preferences from a scheduling environment on the cost of a final schedule such as, missing the deadline, resource usage etc. The preference criterions contribute to the axioms related to the cost while calculating the cost of a final schedule solution.

Class COST-FUNCTION

Definition. This function simply calculates the cost of a schedule in terms of various preferences. It also provides the global criterion for ranking the different schedule solutions.

Domain / Range: As depicted in figure.1, the *domain* of this cost-function is `schedule` and the *range* of a cost-function is `cost`. The cost is represented as a set of either real numbers or vectors.
The cost-function is constructed by subsuming the preferences (see axiom definitions 3 and 4).

Class SCHEDULE

Definition. As indicated in the figure 1, a schedule is represented as a set of job-assignment pairs. The set job-assignment is represented in terms of set membership relation, which is true for any elements of the set job-assignment and false for any other set tuple. The set membership relation is modelled by using the following membership-test.

Membership-Test: The schedule-membership test of a class schedule is a binary relation between a class job and a resource and it is true for pairs of the form, `(?job . ?resource)`, i.e. a job-assignment in a schedule. The domain of a schedule membership relation is a job to be assigned and range is a resource to which it can be assigned. A class job-assignment is used in order to model the pairs of the form `(?job . ?resource)`. Finally, a class job-assignment is used for satisfying the sufficient and necessary (IFF)-condition in a class schedule as indicated in figure 1 by `Is-A set of schedule` arrow.

2.2 Axioms in the Task-Ontology

In the task ontology four axioms have been defined which ensures the legality of the scheduling task specification under any circumstances.

1. *Resource-Availability:* this axiom states that the same resources can not be assigned for two different sets of jobs if their time-ranges are overlapping with each other. As resources are assigned to specific set of jobs in a schedule, then the resources are unavailable for the other sets of jobs and other relevant time periods must be generated and associated with these resources for assigning the next set of jobs.
2. *Constraints-Are-Either-Hard-Or-Soft:* this axiom states that the hard and soft constraints are exhaustive subclasses of a class `constraint`. This gives us the scope to use both of these constraints more efficiently while satisfying the minimal and maximal admissibility schedule solution conditions.
3. *Cost-Subsumes-Preferences:* this axiom states that the cost-function that computes a cost of the schedule, subsumes each of the preferences say, $\{P_{r1}, \ldots, P_{rn}\}$ in order to give the cost of a schedule according to preference specific criterions. In other words the cost-function must be constructed by combining the preference specific cost criterions. It is not much of the knowledge acquisition issue but specifying the preference specific criterions and transforming its effect on the cost of a final schedule.
4. *Cost-Preference-Consistency:* this axiom states that the `cost-function` should not contradict any partial order expressed by the `preference` class. Also, the order over any two schedules for selecting the preferential schedule solution must be consistent with that of the cost-function.

3. Related Work

Here, we compare our work with other three scheduling task ontologies and try to explain the major differences between these and our work.
The OZONE ontology [18] also provides a generic perspective for building scheduling systems. There are some major differences between our work and that of OZONE ontology.
The OZONE ontology takes into account the external environmental factors in scheduling,

such as *Demand, Product* etc., where we are mainly interested in the '*core issues'* involved in building scheduling systems. More importantly, there is no indication about the *cost* and *preference* related issues. Finally, ours is an operational task ontology, which is formerly specified by using the modelling language over the OZONE one.

The CommonKADS [1] ontology and Job Assignment ontology [10] gives the modelling behaviour for the scheduling task. The fundamental difference between these two approaches and ours is the level of granularity. Their approach is characterised at a very abstract level. All the main concepts in these ontologies are informally illustrated at some length, but their definitions are not detailed. For instance, in CommonKADS the job assignment structure is simply characterised as a set of job-assignment tuples. This could obstruct the expressiveness of the user. As in ours it is modelled through the class schedule, which provide the better control over the behaviour of both job and resource. In CommonKADS there is no clear indication about *cost*. As in Job Assignment the *cost as well as* the *preference* related issues are missing. In contrast to all the above three ontologies the main purpose of our ontology is not only to provide the conceptual framework but also the practical reusable resource for modelling the scheduling applications.

4. Validation

To evaluate the strength of our task ontology we have tested it on two different domains: the Office-allocation and the Satellite-scheduling problems.

In the *office-allocation* problem there are number of students (jobs) that need to go in given rooms (resources), which was one of the main hard constraints in the problem. Each student has a number of activities along with the duration of each activity and each room is available for only certain period of time. The important preference used for the usage of the specific rooms is; the research students can share the double-size room if the single-size room is not available. Using the preferences in this fashion gave us more flexibility for using the available resources more efficiently, instead of under-usage of the resources. Also, the students could stay in the room for only a certain period of time without violating the available duration of the room. The final schedule produced by using the task ontology was of the form of a pair {Student, Room}, by maintaining all the constraints, time-ranges and the preferences.

The *Satellite-scheduling* problem was mainly chosen because one application hardly confirms the generality of the task ontology. In this problem there were three satellites that communicates with the available antennas such as low-range-antenna, wide-range-antenna, and metrological-antenna. The main hard-constraints is of the various forms. 1) No two satellites can communicate with the same type of antenna if their time ranges overlap with each other. 2) Every satellite must have at least four communication slots; the gap between any two communications with the same type of antennas must be of two hours. 3) Antennas are of limited resources and can communicate with the assigned satellites for 15 minutes only. The final schedule produced by maintaining all the constraints and the time-ranges is of the form {Satellite . Antenna}, where satellite is a job and the antenna is a resource on which it is assigned.

In the office-allocation problem mainly the *time element* for the availability of the rooms was added, which is not considered by the present ontologies. Additionally, in our task ontology the jobs are broken down into the more detailed level by specifying the number of activities that can be involved in a particular job. As opposed to our approach the jobs are treated as an abstract class without its further breakdown in [1], [10]. In our point of view our task ontology provides the desired level flexibility from the representational perspective. Even though, these two application domains appear on the extreme of the application spectra, as one is from resource-allocation as other is from the space application. They are successfully modelled by using the task ontology.

5. Conclusion and Future Work

The proposed task ontology can now be seen as a knowledge capturing tool in various domains. This satisfies the important reusability aspect discussed in section 1. The reusability was empirically tested as discussed in section 4. The given ontological framework provides a fairly fine-grained structure that is needed to build the scheduling systems. This could help the user in expressing their viewpoint more clearly on the particular scenario. The cost related axioms ensure that an optimal solution is constructed by subsuming the various preferences in scheduling

task specification. The conflict between various jobs for the usage of same resource depending upon their time range overlap is tackled by the `resource-availability` axiom. As discussed in section 1, this task ontology provides firm theoretical and engineering foundations for various classes of knowledge-based applications.

In future we are planning to use this task ontology as a major building block for building the generic problem-solvers for understanding the space of the scheduling behaviour. In order to accomplish the whole process successfully this task ontology can be seen as an initial building block.

References

[1] Breuker, J. and Van de Velde W. CommonKADS Library for Expert Modelling. IOS Press, Amsterdam, The Netherlands, 1994.

[2] Cotter, P. Inside Talligent Technology. Reading, MA: Addison-Wesley. Chapter, The Talligent programming model: framework concepts, 1996.

[3] Domingue, J.; Motta, E. and Corcho Garcia, O. Knowledge Modelling in WebOnto and OCML: A User Guide, URL link: http://kmi.open.ac.uk/projects/webonto, 1999.

[4] Fadel, F. G.; Fox, M. S.; Gruninger, M. A. Generic Enterprise Resource Ontology. Proceedings of the IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, West Virginia (WET ICE 94), 1994.

[5] Friedland, P. and Iwasaki, Y. The Concepts and Implementation of Skeletal Plans. Journal of Automated Reasoning. 1: 161-208, 1985.

[6] Garrido, A.; Salido, M.A.; Barber, F. Scheduling in a Planning Environment. ECAI-2000, Workshop on New Results in Planning, scheduling and design, 2000.

[7] Gruber, T. R. A Translation approach to Portable Ontology Specifications. Knowledge Acquisition. 5 (2): 199-221,1993

[8] Guarino, N. and Giaretta, P. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars (editor), Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. IOS Press, Amsterdam: 25-32, 1995.

[9] Hama, T.; Hori, M.; and Nakamura, Y. Task-Specific Language Constructs for Describing Constraints *in* Job-Assignment Problems. Technical Report IBM Research Report RT0084, 1992.

[10] Hama, T.; Hori, M. & Nakamura, Y. Modelling job assignment problems based on task ontology. Proceedings of the 2nd Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop, Kobe and Hatoyama, Japan. pp. 199-213, 1992.

[11] Hori, M.; Nakamura, Y.; Satoh, H.; Maruyama, K.; Hama, T.; Honda, S.; Takenaka, T. & Sekine, F. Knowledge-level analysis for eliciting composable scheduling knowledge. Artificial Intelligence in Engineering. 9 (4): 253-264, 1995.

[12] Kruger, C. Software reuse. Computing Surveys. 24 (2): 131-183, 1992.

[13] Le Pape, C. Implementation of resource constraints in ilog schedule: A library for the development of constraint-based scheduling systems. Intelligent Systems Engineering, 3 (2): 55-66, 1994.

[14] Lee, J.; Yost, G.; and Group, P.W. The PIFprocess interchange format and framework. Technical Report Working Paper No. 180, MIT Centre for Coordination Science, 1994.

[15] Motta, E. KBS Modelling in OCML. Proceedings of the 5th Workshop on Modelling Languages for KBS, 1995.

[16] Motta, E. Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving. IOS Press, Amsterdam, The Netherlands. ISBN: 1 58603 003 5, 1999.

[17] Saucer, J. Knowledge-Based Systems techniques and Applications in Scheduling. Knowledge-Based Scheduling Techniques in Industry, in: Jain, L., *Intelligent Techniques in Industry*, CRC Press, 1999.

[18] Smith, S. F.; Becker, M. A. An Ontology for Constructing Scheduling Systems. Proceeding of AAAI-97, Spring Symposium on Ontological Engineering, 1997.

[19] Uschold, M. Building Ontologies: Towards a Unified Methodology. Technical Report Technical Report AIAI-TR-197, University of Edinburgh, 1996.

[20] Wielinga, B.; Schreiber, A. & Bruker, J. KADS: a modelling approach to knowledge engineering. Knowledge Acquisition. 4: 5-53, 1992.

[21] Zweben, M.; Fox M. S. (editors) Intelligent Scheduling. Morgan Kaufmann, Palo Alto, 1994.