A network of ontologies for the integration of planning and scheduling activities in batch process industries

Marcela Vegetti [1], Gabriela Henning[2]

[1] INGAR(CONICET/UTN), Avellaneda 3657, Santa Fe 3000, Argentinta

mvegetti@santafe-conicet.gov.ar

[2] INTEC (CONICET,UNL), Ruta Nacional 168, km 461,5, Santa Fe 3000, Argentina

ghenning@intec.unl.edu.ar

Abstract. In the last decades, the integration of informatics applications supporting planning, scheduling and control has been a serious concern of the industrial community. Many standards have been developed to tackle this issue by addressing the exchange of data between the scheduling function and its immediate lower and upper levels in the planning pyramid. However, a more comprehensive approach is required to tackle integration problems, since this matter entails much more than data exchange. So, this article presents an ontological framework that provides the foundations to reach an effective interoperability among the various applications linked to scheduling activities.

1 Introduction

Despite over twenty years of research in batch scheduling, advanced scheduling support systems are not very common in the chemical industry yet. In addition, most of the available commercial systems are not based on the solution methodologies that academia has developed. One of the reasons why academic approaches are not adopted in industry is the fact that decision support tools do not integrate with the enterprise and manufacturing applications because they rely upon quite different knowledge representations. In the last decades, the integration of informatics applications supporting planning, scheduling and control has been a serious concern of the industrial community. The ISA-88 [1] and ISA-95 [2] standards have been developed to tackle this issue by addressing the exchange of data between the scheduling function and its immediate lower and upper levels in the planning pyramid. However, a more comprehensive approach is required to address integration problems, since this matter entails much more than data exchange. In last decade, ontologies have been considered an effective solution to interoperability problems in many domains. Therefore, this article presents an ontological framework that provides the foundations to reach an effective interoperability among the various applications linked to scheduling activities, focusing on one of the ontologies of such framework.

The paper is organized as follows. Section 2 points out some of the problems of dealing with multiple knowledge representations in the scheduling domain. Section 3 introduces the ontological approach that is proposed to address integration problems.

By means of example, Section 4 describes the ontology that was developed to formalize the Resource Task Network (RTN) [3] model, which is one of the components of the ontology network that is proposed in this contribution. The proposed formalization is a definitional extension of the Process Specification Language (PSL) [4]. Finally, section 5 presents some concluding remarks.

2 Different knowledge representations in the scheduling field

Many academic approaches addressing scheduling problems resort to intermediate representations, like the state-task (STN) or resource-task (RTN) networks (See Figs. 1 and 2 of [5]), before developing the mathematical model that indeed solves the problem. However, this representation does not have a direct mapping to the master data that is usually employed in industry. Thus, the human scheduler has to manually create the STN/RTN graphical models from data that is spread in different tables of the enterprise databases.

On the other hand, in the industrial domain, the most important input for the scheduling problem is the ISA-88 master recipe, which provides the set of data that uniquely defines the production requirements of a specific product batch. Recipes are recursive structures containing five components: Header, Formula, Procedure, Equipment requirements and Other Information. See for example Fig. 1, which depicts the master recipes of P1 and P2, which are the final products of the STN shown in Fig. 2 of [5]. Fig. 1 shows that the procedure component of the P1 master recipe is defined in terms of the T2, T1 and T3 operations. The INT3 recipe entity is also shown in Fig. 1 and the ones corresponding to the other intermediates can be found in [6]. An analysis of Fig. 1, the material in [6], along with Figs. 1 and 2 of [5] shows that these recipe representations are quite different and that a direct mapping is not possible. On top of the recipe information, the scheduling problem needs additional input data, such as demands to be fulfilled, plant topology with unit features, etc. This last type of information is not represented neither in the STN/RTN graphs nor in the ISA-88 recipe model. Usually, industry specifies it in the so called physical model proposed by ISA-88, while in academia it is handled in an informal way that is not machine procesable and can lead to misinterpretations.

Having all this input information, a mathematical model is built and then solved. The resulting production schedule needs to be communicated to the adjacent levels in the planning pyramid: (i) to the lower process control layer to materialize batch execution and (ii) to the upper production planning and control (PPC) level for plan management activities. In practice, the results included in the solver output file need to be translated into the control recipe (the one that according to ISA-88 standard is employed by the control system to perform batch execution) and into the operations schedule, an explicit representation of the schedule according to the ISA-95 standard.
As a result, it can be seen that in order to perform scheduling activities and to articulate them within the planning pyramid, several knowledge representations and models need to interplay. For instance, to transform a given master recipe into a control one, the procedure that is conceptualized in Fig. 2 would need to be executed (actually, such an approach was never made explicit by researchers devoted to the industrial scheduling field and it is presented in this contribution). Currently, the translation

from one representation to another is manually done, in the few cases it is indeed carried out. In fact, none of the academic proposals take into account the automatic translation of the solver output file into representations that are useful from an industrial point of view, i.e. control recipes and explicit schedule representations. In addition to the manual translation workload, these heterogeneous data models employ distinct terms to refer to the same concepts. Besides these semantic issues, since the models are not formal, they cannot be interpreted by a computer and can also be ambiguous. Moreover, an analysis of the ISA-88 and ISA-95 standards [7] reveals some overlappings on the information and activities handled by them (e.g. product definition vs. recipe specification; equipment capability vs. physical model), which discloses some collision points. The problems pointed out in the previous paragraphs are some of the difficulties preventing the integration of scheduling activities within the planning pyramid and, more specifically, precluding the adoption of advanced scheduling approaches in industry.

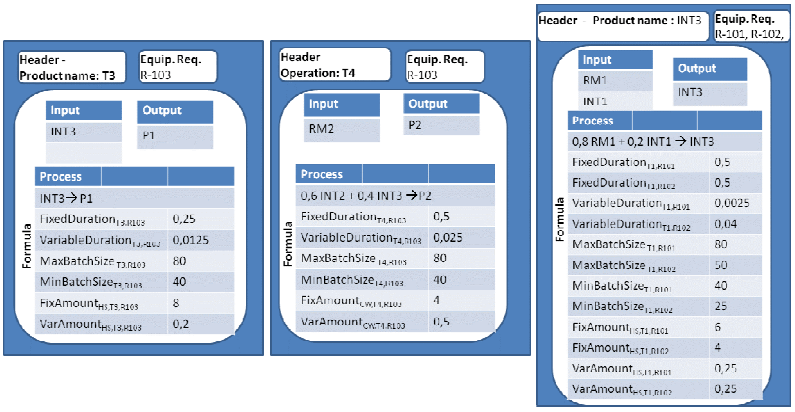

Fig. 1. Recipe entities for products P1, P2 and INT3

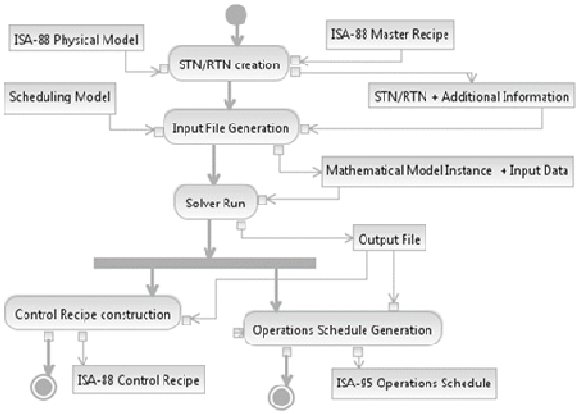

Fig. 2. Data manipulation and model translation associated with scheduling

3 Ontological approach that supports scheduling activities

In order to tackle the problems posed in the previous section an ontological approach is addressed in this contribution. Following the interlingua approach [8], this contribution proposes the construction of a network of ontologies (see Fig. 3) that has the Schedule Reference Ontology (SRO) as a common vocabulary. This network also contains local ontologies that formalize the different sources of information supporting scheduling activities. Specifically, it contains the ontologies of the ISA-88 and ISA-95 standards, the STN/RTN representations, the mathematical programming models (whenever MILP/MINLP-based solution approaches are pursued), etc. The implementation of this ontology network requires the formalization (triangles in Fig. 3), with a previous conceptualization in certain cases (diamonds in Fig.3), of the various information sources. Moreover, the development of ontology alignment agents (circles in Fig. 3), which map concepts of the local ontologies to their definitions in the SRO common vocabulary, is also required. This core ontology plays a central role in the network, acting as a bridge between the different ontologies in the net. The development of the proposed network of ontologies is the starting point for the automatic construction and translation of models that are employed during the scheduling activities, as well as for the correct exchange of information among the scheduling applications and the other manufacturing applications with which they interplay. Some work has been done by the authors in relation to the formalization of ISA-88 standard [7,9]. In this article, the conceptualization and formalization of the RTN is addressed.

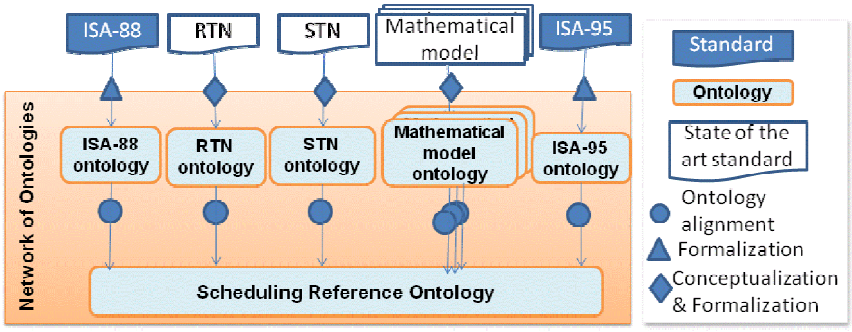

Fig. 3. The proposed network of ontologies

4 RTN conceptualization and formalization

The approach that has been followed to formalize the RTN model is to develop a definitional extension of PSL (Process Specification Language)[4]. The aim of PSL is to create a process specification language to facilitate the complete and correct exchange of process information among manufacturing applications. PSL is structured in two main layers: core theories and definitional extensions. Fig. 4 shows the PSL theories and extensions that are required for the RTN extension proposed in this con-

tribution. The *Core Theory* is the kernel of PSL since it defines its basic concepts, which are *Activity*, *Object* and *Timepoint*. These three concepts are used in almost all theories and/or definitional PSL extensions. The *Outer Core* theory groups the theories related to *Subactivities*, *Occurrence Trees*, *Activity Occurrence*, *Discrete State*, *Atomic* and *Complex Activities.* All of them require the concepts that are defined in the *Core* theory. The *Resource Requirements* theory uses the *Core* and the *Outer Core* theories to define the set of axioms that constrains the definition of the resource requirements of an activity. All these theories become the foundation for the definition of several extensions. In particular, the *Processor Activity* [10] and *Resource Roles* [11] definitional extensions are required for the specification of the RTN extension to be presented in the next two sections.

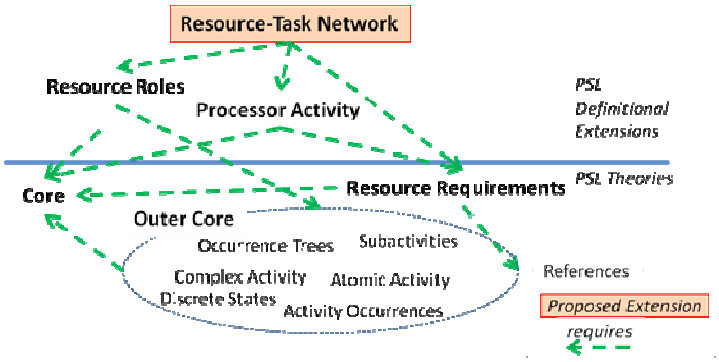

Fig. 4. Parts of PSL that are required by the RTN extension

The RTN model is based on two fundamental concepts: resources and tasks. A task is an operation with a given duration that can consume and generate resources [12]. According to Castro et al. [3], the concept of resource is entirely general and includes all entities that are involved in the process steps, such as materials, processing and storage equipment, personnel, as well as utilities (steam, cooling water, etc.).

The proposed extension specifies the definition of the Task and Resource concepts as well as some axioms that constrain these definitions in the domain of RTN. In this proposal the Common Logic implementation [13] of PSL was chosen and therefore, the proposed extension has been implemented in this language.

4.1 Resource definition

The Resource Roles extension (RRe) [11] axiomatizes the fundamental intuitions about resources. According to this extension an Object is a resource only with respect to the role that it plays in some activity that requires the Object. Therefore, this extension does not axiomatize any other properties of resources [14].

In RTN the concept of resource includes all the entities that are involved in the process steps [3]. In this sense this definition is similar to the resource concept that is specified in PSL. However, in RTN, resources can be specialized in materials (raw materials, intermediate ones, and final products), processing and storage equipment,

manpower and utilities. Therefore, the proposed formalization specifies this taxonomy. The proposed RTN uses the RRe definitions to specify the different resources that are involved in the definition of an RTN model.

Definition 1: Every resource ?r is either a material, a utility, a storage_unit, a processing_unit or personnel.

```
(forall (?r) (iff (resource ?r)
             (or (material ?r)
              (storage_unit ?r)
              (processing_unit ?r)
              (utility ?r)
              (personnel ?r) )))
```

Definition 2: A resource ?m is a material if and only if ?m is an input material or output material of at least one task ?t. Both, input_material and output_material are definitions belonging to Processor Activity extension [10]. This extension states that an object ?m is an input material for an activity ?t if and only if ?t is a processor activity which consumes or modifies ?m. Similarly, an object ?m is an output material for an activity ?t if and only if ?t is a processor activity which produces or modifies ?m.

```
(forall (?m) (iff (material ?m)
  (exists (?t) (and (task ?t) (or
                    (input_material ?m ?t)
                    (output_material ?m ?t))))))
```

Definition 3: A material ?m is a raw material if ?m participates as an input material in at least one task ?t1 and there is no task in which ?m is an output material

```
(forall (?m)(iff (raw_material ?m) (and
    (material ?m)
    (exists (?t1) (and (task ?t1)
                       (input_material ?m ?t1)))
    (not (exists (?t2) (and (task ?t2)
                       (output_material ?m ?t2)))))))
```

Definition 4: A material ?m is an intermediate material if ?m participates both as an input_material in at least one task ?t1 and as an output_material in at least another task ?t2.

```
(forall (?m) (iff (intermediate_material ?m)
(and (material ?m)
     (exists (?t1 ?t2) (and (task ?t1) (task ?2)
                            (input_material ?m ?t1)
                            (output_material ?m ?t2)
                            (<> ?t1 ?t2) )))))
```

Definition 5: A material ?m is a final product if ?m does not participate as input_material in any task and is an output_material in at least one task.

```
(forall (?m) (iff (final_product ?m)
 (and (material ?m)
  (exists (?t1)(task ?t1)(output_material ?m ?t1))
 (not (exists (?t2)(task ?t2)(input_material ?m ?t2))))))
```

Definition 6: A dedicated_storage_unit is a storage_unit that can only store a unique material.

```
(forall (?dsu ?m1 ?m2)
  (iff (dedicated_storage_unit ?dsu)
     (and (material ?m1) (material ?m2)
          (can_store ?dsu ?m1)(can_store ?dsu ?m2)
           (= ?m1 ?m2))))
```

Definition 7: A shared storage unit is a storage_unit that can store different materials

```
(forall (?ssu)
(iff (shared_storage_unit ?ssu) (exists (?m1 ?m2)
  (if (and (can_store ?ssu ?m1)(can_store ?ssu ?m2)
           (<> ?m1 ?m2))))))
```

Definition 8: A processing_unit is a reusable or possible_reusable resource that can perform at least one task.

```
(forall (?pu) (iff (processing_unit ?pu)
 (exist (?t) (and (or (reusable ?pu ?t)
                      (possibly_reusable ?pu ?t))
                (can_perform ?pu ?t)))))
```

The definitions of reusable or possible_reusable resources are stated in RRe [11]. A resource ?r is reusable (possible_reusable) by an activity ?a if any other activity that also requires ?r is still possible to be performed after ?a completes its occurrence, in every (some) possible future.
An example of a reusable resource is a processing unit that does not require set-up/changeover between activities. As soon as one activity occurs, it is always possible to execute the next activity. In contrast, a possibly reusable resource is a processing unit that requires some setup/changeover between different activities. After the first activity occurs, it is available for the other activity, but only if the setup/activity activity occurs first.

Definition 9: A utility is a reusable resource that acts as a service or commodity having a limited capacity

```
(forall (?ut) (iff (?utility ?ut)
        (exists (?t ?q)(and (reusable ?ut ?t)
        (max_capacity ?ut ?q)))))
```

4.2 Task definition

The other main concept of RTN model is task. This section deals with the definition of this important concept and with the specification of a set of axioms that constrains it.

Definitions 10-12: a task is a processor activity that has at least one input_material and at least one output_material, which are consumed and created, respectively, in given quantities.

```
(forall (?t) (iff (task ?t)(exists (?m ?m2 ?q1 ?q2)
        (and  (processor_activity ?t)
              (input_material_demand ?m ?t ?q1)
              (output_material_generation ?m2 ?t ?q2)))))

(forall  (?m ?t ?q1)(iff (input_material_demand ?m ?t ?q1)
    (and (material ?m) (task ?t) (input_material ?m ?t))))

(forall  (?m ?t ?q1)
  (iff (output_material_generation ?m ?t ?q1)
    (and (material ?m) (task ?t)(output_material ?m ?t))))
```

The Processor Activity extension of PSL [10] defines a processor_activity as "an activity which uses some set of resources, consumes or modifies some other set of resources, and produces or modifies a set of objects."

A set of axioms is also developed to represent constraints on the RTN ontology. These axioms make coherent the concepts in the considered engineering field.

Axiom 1: Every task ?t is related to one or more processing units in which a task occurrence can be performed.

```
(forall (?t) (if (task ?t)(exists (?pu)
    (and (processing_unit ?pu)
         (can_perform ?pu ?t)))))
```

Axiom 2: Every task has bounds in the size of the batch that it can handled in each processing unit. These limits are indicated by the parameters min and max batch size. Both parameters should be expressed in the same unit of measure, but due space limits the specification of this constraint, which is indicated by the legal_batch_size expression, is not shown.

```
 (forall (?t) (if (task ?t) (exists (?pu ?maxbs ?minbs)
    (and (processing_unit ?pu) (can_perform ?pu ?t)
         (max_batch_size ?t ?pu ?maxbs)
         (min_batch_size ?t ?pu ?minbs)
         (legal_batch_size ?maxbs ?minbs)))))
```

Axiom 3: The duration of each task ?t depends on the processing unit ?pu in which it will be executed. There are two alternative ways of specifying the task duration. On the one hand, the duration is represented by a unique parameter (?fixd). On the other

hand, two parameters are used: a fixed duration (?fixd) and a variable duration (?vard), which depends on the batch size. When using two parameters for describing the task duration, some constraints are imposed in the unit of measures of both parameters. These constraints are specified in the definition of the legal_duration expression. However, due to the lack of space, this specification is not shown in this article.

```
(forall (?t) (if (task ?t)
   (or (exists (?pu ?fixd)(fixed_durantion ?t ?pu ?fixd))
    (exists (?pu ?fixd ?vard)
      (and (fixed_durantion ?t ?pu ?fixd)
           (variable_duration ?t ?pu ?vard)
           (legal_duration ?fixd ?vard))))))
```

Axiom 4: The utility requirement of a task in a given processing unit is expressed using two parameters: fixed and variable amount. The unit of measure of both parameters should be consistent. This restriction is specified in the definition of legal_amount expression, which is not shown in this article for space reasons.

```
(forall (?t ?ut)
  (if(and (task ?t)(utility ?ut)(require ?t ?ut))
     (exists (?fixa ?vara)
       (and (fixed_amount ?t ?pu ?fixa)
            (variable_amount ?t ?pu ?vara)
            (legal_amount ?fixa ?vara)  ))))
```

Axiom 5: A task occurrence (an activity occurrence) ?tocc has to be executed in a processing unit ?pu that is related to its corresponding task ?t and its size in this specific ?pu has to be between the max and min batch size specified for ?t

```
(forall (?tocc)
  (iff (occurrence_of ?tocc ?t)
    (exist (?t ?pu ?q ?maxbs ?minbs)  (and
       (task ?t)  (processing_unit ?pu)
       (can_perform ?t ?pu) (executed_in ?tocc ?pu)
       (size ?tocc ?pu ?q)
       (max_batch_size ?t ?pu ?maxbs)
       (min_batch_size ?t ?pu ?minbs)
       (and (>= ?q ?minbs)(<= ?q ?maxbs))))))
```

5 Conclusions

Ontologies support interoperability by providing semantic terminology in a computer understandable format. This article proposes the definition of a network of ontologies to solve the interoperability problems associated with the execution of scheduling activities and their interplay with other functions within the Planning pyramid. This ontology network, whose architecture is presented in this contribution, is the starting

point for the automatic construction and translation of models that are employed during the scheduling activities. In addition, one of the ontologies of this network is introduced along with its implementation in common logic as a definitional extension of PSL.

6 Acknowledgments

The authors acknowledge the financial support received from CONICET (PIP 11220110100906 and PIP 11220110101145), ANPCyT (PICT-2013 1310), UNL (PI CAI+D 2011) and UTN (PID 25-O156).

7 References

1. ANSI/ISA–88.00.01: Batch Control Part 1: Models and Terminology, 2010.
2. ANSI/ISA-95.00.01-2000: Enterprise-Control System Integration. Part 1: Models and terminology, 2000.
3. Castro, P.M., Barbosa-Póvoa, A.P., Matos, H.A. (2003). Optimal Periodic Scheduling of Batch Plants Using RTN-Based Discrete and Continuous-Time Formulations: A Case Study Approach, Ind. Eng. Chem. Res., 42, 3346-3360.
4. ISO 18629-1 (2004) Process specification language -- Part 1: Overview and basic principles. ISO/TC 184/SC 4
5. Giménez, D. Henning, G., Marevelias, C. (2009). A novel network-based continuous-time representation for process scheduling: Part I. Main concepts and mathematical formulation. Computers and Chemical Engineering, 33, 1511–1528
6. VH (2014) SRO Data Repository https://sites.google.com/site/sropse2015/Last access: 30th November 2014.
7. Vegetti, M. and Henning, G. (2014). ISA-88 formalization. A step towards its integration with the ISA-95 standard, FOMI 2014 Proceedings, Brasil.
8. Ciocoui, M., Gruninger, M., Nau, D. (2000). Ontologies for integrating Engineering Applications, Journal of Computing and Information Science and Engineering, 1, 12-22.
9. Vegetti, M. and Henning, G. (2015). An ontological approach to integration of planning and scheduling activities in batch process industries, PSE/ESCAPE 2015 Proceedings, Denmark.
10. National Institute of Standards and Technologies (2008). PSL ontology - Processor Activities extension. Available on-line: http://www.mel.nist.gov/psl/psl-ontology/part46/processor.html
11. National Institute of Standards and Technologies (2008). PSL ontology - Resource Roles extension. Available on-line: http://www.mel.nist.gov/psl/psl-ontology/part44/res_role.html
12. Wassick, J.M, Ferrio J. (2011). Extending the resource task network for industrial applications, Computers and Chemical Engineering 35, 2124– 2140
13. ISO/IEC 24707 (2007). Common Logic (CL): a framework for a family of logic-based languages.
14. Schlenoff, C., Gruninger, M., Lee, J., (2000). The Essence of the Process Specification Language. Transaction of the Society for Computer Simulation International, Vol. 16, N4.