ISA-88 Formalization. A Step Towards its Integration with the ISA-95 Standard

Marcela Vegetti
INGAR (CONICET-UTN)
mvegetti@santafe-conicet.gov.ar

Gabriela Henning
INTEC (CONICET-UTN)
ghenning@intec.unl.edu.ar

Abstract.

ANSI/ISA-88 and ANSI/ISA-95 are two well accepted standards in the industrial domain that provide a set of models considered as best engineering practices for industrial information systems in charge of manufacturing execution and business logistics. The main goal of ANSI/ISA-88 is the control of batch processes, whereas the one of the ANSI/ISA-95 standard is the development of an automated interface between enterprise and control systems. In consequence, both standards should interoperate. However, there are gaps and overlappings between their corresponding terminologies. Moreover, there are additional problems, such as semantic inconsistencies within each of the standards, as well as the use of an informal graphical representations in one of the ANSI/ISA-88 models. This work presents an ontological approach that aims at formalizing the ISA-88 standard as a first step towards its integration with a formal representation of the ANSI/ISA-95 one. Additionally, methodological aspects of the ontology development process are presented.

1 Introduction

Industrial organizations address their planning activities within the context of the enterprise hierarchical planning pyramid. This pyramid (see Figure 1) includes activities performed at different time frames, handles information having distinct granularities, and involves scheduling interplaying with the Production Planning and Control (PPC) and Plant Control (PC) functions. The difficulties associated with these interactions were pointed out almost a decade ago [Sho02] and this topic has recently gained renewed attention. To tackle the integration of PPC and scheduling, researchers have proposed various solution strategies [Mar09]. In addition, a few authors have pointed out the requirements that apply to the data exchange in order to support such integration [Kre06]. Similarly, regarding integration between scheduling and plant control, researchers have started to draw the attention to data exchange problems [Mun10], [Har09]. Moreover, the standards ANSI/ISA-88 [Ans10] and ANSI/ISA-95 [Ans00] (referred as ISA-88 and ISA-95 for simplicity reasons) have been developed to tackle issues related to planning, scheduling and control activities and data. Whereas both standards address the exchange of data between the scheduling function and its immediate upper and lower levels in the planning pyramid, a more comprehensive approach is required to address integration problems, since this matter entails much more than data exchange.

The aforementioned problems are related with semantic issues, which constitute challenges that should be addressed to reach a semantic integration of enterprise applications. For example, the *batch* concept has three definitions in ISA-88. This term is used as an abstract contraction of the words "the production of a batch"[Ans10]. Therefore, according to the context, batch means both the material made by and during the process and also an entity that represents the production of that material. In a similar way, the terms *procedure* and *recipe component* have different meanings within the ISA-88 standard. The opposite situation, more than one term to represent a given concept, also occurs between the ISA-88 and ISA-95 standards. Both use different terms to define concepts such as "amount of material", "equipment", "group of products scheduled to be manufactured" and "how to make a product". Moreover, ISA-88 lacks a formalism to represent the procedural part of a recipe, since it employs an informal graphical model

In: Alexandre Rademaker and Vinay K. Chaudhri (eds.): Proceedings of the 6th Workshop on Formal Ontologies meet Industry, Rio de Janeiro, Brazil, 22-SEP-2014, published at http://ceur-ws.org

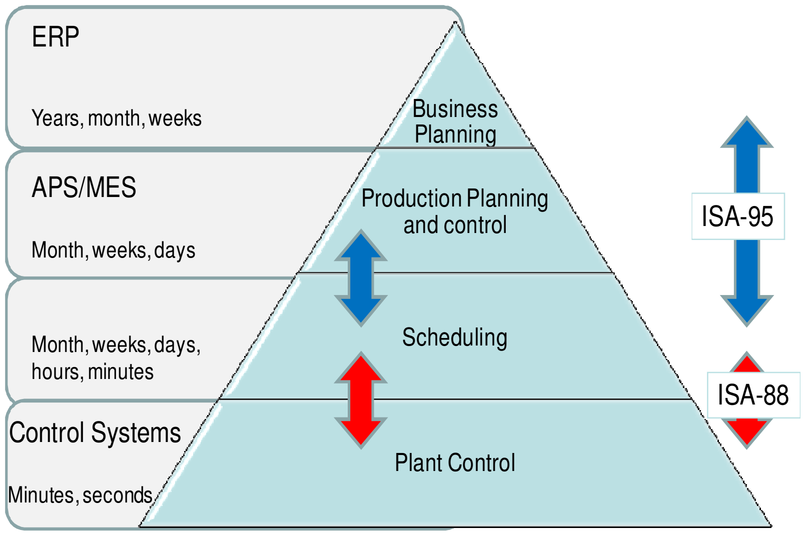

Figure 1. Scheduling function in the enterprise hierarchical planning pyramid

To overcome these semantic challenges, this work proposes the formalization of the ISA-88 [Ans10] standard as a first step towards its semantic integration with the ISA-95 [Ans00] one. This article is organized as follows: section 2 briefly introduces the ISA-88 standard. Section 3 presents general methodological considerations for the development of the proposal, describes the main concepts of the ontology and its application to a case study. Conclusions and future work are drawn in Section 4.

2 ISA-88 standard

The ISA-88 standard originally addressed batch process control issues, and was later extended to tackle discrete manufacturing and continuous processes. It organizes knowledge along three different perspectives: the physical model, the process model, and the procedural control one. All these representations are hierarchical ones.

The *physical model* hierarchically organizes the enterprise into sites, areas, process cells, units, as well as equipment and control modules. The three higher levels are more precisely defined in the ISA-95 standard, as they are often beyond the scope of batch control. The criteria for establishing these boundaries are not well defined neither in the ISA-88 nor in the ISA-95 standard. The *process model* is a multi-level hierarchical model for the high level representation of a batch process, and it is the basis for defining equipment independent recipe procedures. The ISA-88 standard divides a batch process hierarchically into process stages, process operations and, finally, process actions. The *procedural control model* is a hierarchical representation that depicts the orchestration of procedural elements to carry out process oriented tasks.

According to ISA-88 a recipe provides the necessary set of information that uniquely defines the production requirements for a specific product. Recipes are defined at different abstraction levels: General, Site, Master and Control. Distinct recipes (at the same abstraction level) may exist for different sets of raw materials that can be used to manufacture the same product. This work focuses on the Master and Control recipes since they are the ones that take part on integration issues. A Master recipe includes cell specific information, which is based upon equipment types and classes. Finally, the Control recipe is obtained from the Master one by incorporating batch specific information, such as batch size, the allocation of process equipment, and the definition of quantities of raw materials by scaling the recipe parameters according to the adopted batch size. Thus, the master recipe acts as a template for the derivation of control recipes. Typically, the information comprised in a recipe includes: (i) the header, with the recipe ID, version, status, date, etc., (ii) the formula, containing process inputs (data about the materials, energy and other resources that are required), expected outputs (products, by-products and waste products), and process parameters (temperatures, flowrates, processing times, etc.), (iii) equipment requirements, (iv) the recipe procedure, which defines the sequential and parallel actions needed to produce a batch of a certain product, and (v) safety and compliance information.

The recipe procedure, which is the core part of a recipe, can be hierarchically decomposed into recipe unit procedures, recipe operations and recipe phases, which define the procedural control model. Each unit procedure consists of an ordered set of operations, which consist of an ordered set of phases. The ISA-88 standard proposes a graphical model, referred as the Procedure Function Chart (PFC), in order to represent the recipe procedure for both, the Master recipe and the Control one. Although there are several accepted ontologies for process representation, like PSL (Process Specification Language) [ISO04], the model included in the standard is one adopted by batch industries for the specification of their production processes. Therefore, in this article a formalization of PFC is done. A PFC is defined by a set of symbols representing recipe procedural elements, begin and end points, resource allocations, synchronization elements, recipe transitions and directed links, as well as selection and simultaneous sequences. Figure 2 illustrates the PFC of a recipe for Mint Swirled Toothpaste production and partial details of the low level PFC associated with this recipe [Par00]. According to the control model of ISA-88, the recipe of a Mint Swirled toothpaste batch is related to a procedure called *Mint Swirled Toothpaste Production* in the example. Figure 2 shows three encapsulated unit procedures, which correspond to the production of mint and gel toothpaste and the mixing of both components. Figure 2 also shows the details of the *Make Gel Toothpaste* unit procedure, which encapsulates a sequence of three operations: *Add Ingredients, React* and *Prepare to Transfer*. The first operation combines various ingredients to form a new intermediate material. The second one mixes, heats, and then cools the new mixture to create toothpaste. Finally, the last operation gets the vessel ready to transfer its contents to the blender/extruder unit. The last two operations are linked by an explicit transition, which implies that the condition *Approved by lab* has to be satisfied before the *Prepare to Transfer* step operation starts.

An expansion of the *Add Ingredients* operation is included in the right part of Figure 2. Five phases are comprised in this operation. Each phase performs a unique and independent function. The phase *Add Water* precedes the following three parallel phases: *Add Filler*, *Add Flavorings* and *Add Stabilizers*. The three phases have to be completed before the *Add Sodium Fluoride* phase starts. The two pairs of horizontal lines represent the begin and end of the parallel sequence.

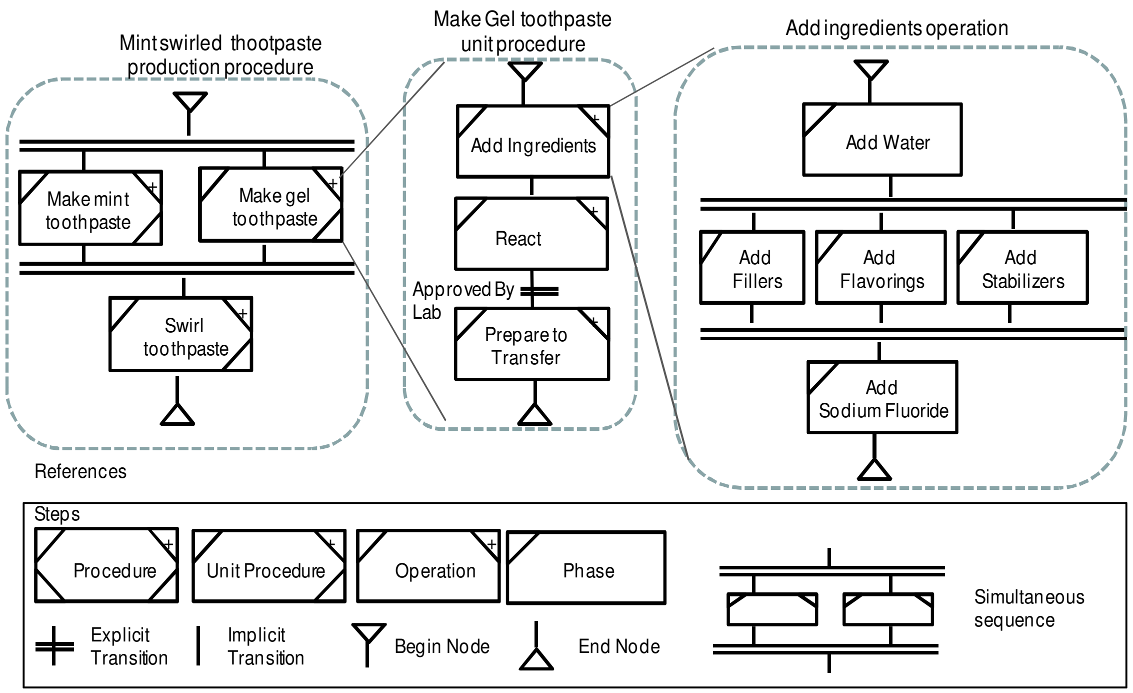

Figure 2. PFC example

As it can be seen in Figure 2 any procedural element above the level of a phase may represent an encapsulation of other procedural elements pertaining to the subsequent lower level in the procedural control hierarchy. Moreover, the level of the control model to which a step belongs, is represented by 1 to 4 diagonal lines located at the corners of the step rectangle. Therefore, the different types of steps are identifiable only by a person that analyzes the diagram. Only such person is capable of detecting which steps (unit procedures, operations, phases) are executed in parallel, which is the predecessor or the successor of a given step, or which are alternative steps. In consequence, the relation between a procedural element and the components that it encapsulates is not understandable without a human eye. Therefore, the implicit nature of this graphical representation makes it impossible for a computer to infer knowledge from it. In addition, the graphical representation of a PFC lacks the capacity to automatically validate the correctness of a diagram. The formalization of the PFC that is presented in this work helps to overcome this drawback.

3 ISA-88 Formalization. A First Step towards its Integration with ISA-95

In order to deal with the semantic challenges introduced in section 1, this work proposes a formalization of the ISA-88 and ISA-95 standards by defining an ontology per each standard and then an ontology to integrate them. This approach avoids defining just one big ontology, which would be very complex and rigid, by sticking to a "divide and conquer" strategy.

The development of each ontology has been carried out separately. The formalization of ISA-88 has been done first because it is the one that has the major semantic inconsistencies in its term definitions. The models proposed within ISA-88 are taken into account to divide the proposed ontology into small ontology modules. The same approach was considered in the formalization of the other standard. Due to space limitations, only a part of the formalization of the ISA-88 ontology is presented in this paper. In particular, the main concepts involved in its Procedural Control Module, which is the one of the most advanced modules in this project's implementation, are introduced in the following paragraphs.

For the development of the Procedure Control ontology, an ad-hoc methodology based on well accepted principles has been proposed. It has the following four stages:

- Requirements specification: identifies the scope and purpose of the ontology.
- Conceptualization stage: organizes and converts an informally perceived view of the domain into a semi-formal specification using UML diagrams.
- Implementation stage: codifies the ontology using a formal language.
- Evaluation stage: allows making a technical judgment of the ontology quality and usefulness with respect to the requirements specification, competency questions and/or the real world.

It should be mentioned that these stages were not truly sequential; indeed, any ontology development is an iterative and incremental process. If some need/weakness was detected during the execution of a given stage, it was possible to return to any of the previous ones to make modifications and/or refinements. Moreover, as it is proposed by Ontology Summit 2013 Communiqué [Neu13], some evaluation activities have been done in all the stages of the proposed development process. Some highlights of these methodological stages are given in the remaining of this section.

3.1 Requirement Specification

Competency questions, which were proposed by Uschold and Gruninger [Usc96] in their methodology, helped at this stage to identify the requirements. The gathered competency questions were classified in groups related to the levels defined in the control model. A small sample of the competency questions that were stated, are the following:

- Which are the procedural elements associated with a given Master Recipe?
- Which are the conditions that should be fulfilled to start a given Operation?
- Which are the operations comprised in a given Unit Procedure?
- Which are the phases that need the conclusion of a given phase P in order to start their execution?
- Which phases are executed in parallel with a given phase P?

3.2 Conceptualization

The second main step in the development process required the identification and capture of the domain concepts and their relationships, trying to fulfill the previous requirements. To support this activity, UML (Unified Modelling Language) [OMG13] was adopted. In addition to class diagrams, constraints about the objects in the model and invariants on classes have been added using OCL (Object Constraint Language) [OMG14]. The complete results of this stage are not described due to lack of space. However, a partial model will be described in this section.

As it was mentioned in Section 1.1, the ISA-88 procedural control model describes recipe procedures. Therefore, the main concepts in the corresponding ontology are related to recipes and their components. Figure 3 introduces the main concepts associated with recipe definition.

The *RecipeEntity* is the combination of a *ProceduralElement* with associated recipe information, which includes a *Header*, *EquipmentRequirements* and a *Formula*.

Each *Recipe Entity* is built up of lower-level recipe entities. But these levels should be hierarchically organized according the procedural control model. Therefore, the concepts *ProcedureRecipeEntity*, *UnitProcedureRecipeEntity*, *OperationRecipeEntity* and *PhaseRecipeEntity* have been added to the ontology in order to represent recipe entities at *Procedure*, *UnitProcedure*, *Operation* and *Phase* Level respectively. Eq. 1 shows how the instances of *ProcedureRecipeEntity* are inferred. Similar expression have been defined to compute recipe entities at the other levels. In addition, to guarantee consistency in this hierarchy, constraints on the *composedOf* relationship have been added to the proposed ontology. All *RecipeEntities,* except the ones at *Phase* level, which does not have components, should include *composedOf* recipe entities belonging to the immediate lower level. Eq. 2 states this constraint for the *ProcedureRecipeEntity*.

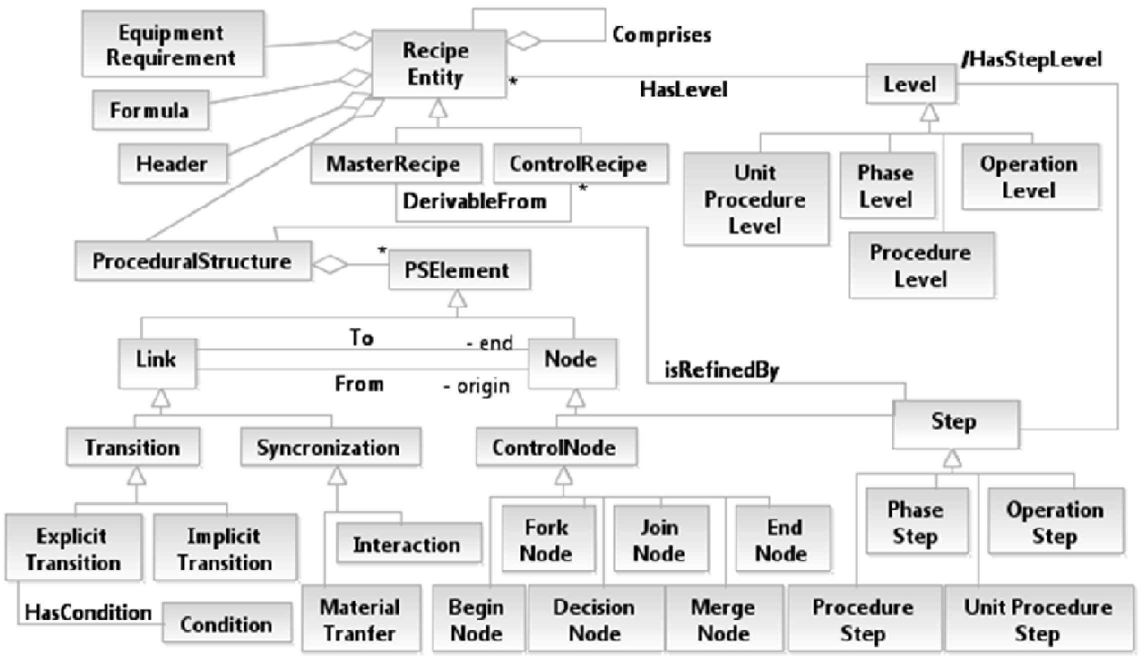

Figure 3. Recipe entity and its procedural structure definition

```
context ProcedureRecipeEntity::allInstances():Set(ProcedureRecipeEntity)                (1)
body: RecipeEntity.allInstances()->Select(r| r.level.oclIstypeOf(ProcedureLevel))
context ProcedureRecipeEntity inv ProcedureComposedOfUnitProcedure:                      (2)
self.component->forall(r:RecipeEntity | r.oclIsTypeOf(UnitProcedureRecipeEntity))
```

A *Recipe* is a *RecipeEntity*, which is defined at the highest level of the procedural control model. In particular, a *MasterRecipe* is a *RecipeEntity* at the *ProcedureLevel,* which is the highest one (Eq. 3). Similarly, a *ControlRecipe* is also a *RecipeEntity* that is derivable from a *MasterRecipe*.

```
context MasterRecipe::allInstances():Set(MasterRecipe)                                   (3)
body: RecipeEntity.allInstances()->Select(r:RecipeEntity|
r.level.oclIstypeOf(ProcedureLevel))
```

Figure 3 also shows the relation of a recipe entity with its components: *Header*, *EquipmentRequirement*, *ProceduralStructure* and *Formula*. As it was indicated in Section 2, ISA-88 suggests the use of Procedural Function Charts (PFC) to describe the procedural structure of a recipe. The proposed ontology formalizes the elements that this type of diagram includes. A *ProceduralStructure* is a set of procedgural elements (*PSElement* in Figure 3) that depicts the procedural logic for all levels of the recipe: recipe procedure, recipe unit procedure, and recipe operation.

A Procedural Element may be a *Link* or a *Node*. A *Node* is a procedural element that represents an action (procedure, unit procedure, operation or phase) or a symbol that controls the transition between steps. A *Link* defines a relation between two nodes. Different types of links and nodes are defined in order to formalize all PFC symbols. Some of the proposed links and nodes are shown in Table 1.

Valid diagrams have to follow consistent rules for threads of execution. The formalization of the procedural structure allows the definition of constraints that helps building valid procedural structures. For example, the specialization of the *Step* concept to represent steps at different levels of the Procedural Control Model allows defining restrictions that avoid the construction of a PFC in which *UnitProcedures*, *Operations* or *Phases* could be mingled in the same diagram. For lack of space, the restriction are not shown in this article.

Table 1: Different types of links and nodes

Type of Link	
Transition	A direct link between nodes.
Implicit Transition	A transition whose only condition is that the directly preceding step has to finish its execution.
Explicit Transition	A transition having a condition that has to evaluate to true in order to activate the following step in the transition.
Synchronization	A link that relates recipe elements among which there is a certain form of synchronization.
Material Transfer	A link that represents material transfer from a step to another one.
Interaction	A kind of synchronization that does not involve movement of material.
Type of Node	
Step	A node that represents a recipe procedural element: procedure recipe entity, unit procedure recipe entity, operation recipe entity or phase recipe entity. According to the level of the procedure control model associated to the PFC in which the step is defined, this concept can be specialized in *ProcedureStep*, *UnitProcedureStep*, *OperationStep*, and *PhaseStep.*
Control Node	A node that controls de intended thread of execution of the recipe procedural elements.
Begin Node	A node that identifies the start of each procedural structure and/or each subordinate structure.
End Node	A node that indicates the end of a procedural structure and/or a subordinate structure.
Fork Node	A node that defines the start of independent threads of execution of certain recipe elements, which are executed in parallel.
Join Node	A node that indicates the end of independent threads of execution.
Decision Node	A node that specifies the beginning of alternative threads of execution.
Merge Node	A node that shows the joining of alternative threads of execution.

3.3 Implementation and Evaluation

The main goal of the ontology implementation activity is to create a computable model deployed in an ontology language from the conceptual model created during the conceptualization phase. Based on its ample acceptance, the OWL 2 language [14], developed by the W3C (World Wide Web Consortium), was chosen to implement the ontology and the Protegé 4.3 editor has been selected to support the ontology development and implementation.

The development process has been guided by the principles of coherence, conciseness, intelligibility, adaptability, minimal ontological commitment and efficiency. Some of these principles are conflicting among themselves. Due to such incompatibilities, a suitable balance between the clashing principles was sought. Nowadays it is widely accepted that there is a lack of a formal methodology that considers all these criteria, which could be applied to evaluate domain ontologies. According to Gómez-Pérez [Gom96] , the ontology evaluation phase comprises three aspects: (i) ontology validation, (ii) ontology verification, and (iii) ontology assessment. Validation and verification activities are associated with a technical judgment of the content of the ontology with respect to a frame of reference, which can be requirement specifications, competency questions, or the real world. In turn, assessment focuses on judging the ontology content from the user's point of view. The results of using the proposed ontology in the development of different types of applications in various contexts are required to be analyzed in order to assess the ontology.

Although there is no formal methodology that considers all the aforementioned evaluation criteria, many authors have reached an agreement on assessment needs during all the ontology lifecycle stages [Neu13]. Having them in mind, partial evaluation through competency questions has been done during the implementation phase. To do so, the ontology has been applied to represent recipe information of several literature case studies [Hai11] [Fis90], including the PFC that is shown in Figure 2 [Par00]. In particular, the results of executing some of the competency questions of this case study are introduced in Table 2. For the sake of simplicity, prefixes are omitted in Table 2. While the namespace corresponding to the proposed ontology is represented with the p0 prefix, the *rdf* one identifies the namespace of the RDF (Resource Description Framework) language.

The formal competency question in the first row of the table asks about explicit transition links (?lk rdf:type p0:ExplicitTransition) having a condition (?lk p0:HasCondition ?cnd.) that points to a step (?lk p0:To ?stp.) at the operation level (?stp p0:HasStepLevel p0:Operation.) and, from this set of links, the query filters the operation step *Prepare2TransferStep* (**FILTER** (?stp = p0:Prepare2TransferStep). The execution of this SPARQL query gives as a result the *ApprovedByLab* condition which according to the PFC shown in Figure 2 is necessary to be fulfilled in order to start the mentioned operation.

The second SPARQL query in Table 2 selects all the recipe entities (?rcp rdf:type p0:RecipeEntity.) at the *UnitProcedure* level (?rcp p0:HasLevel p0:UnitProcedure.) whose procedural structure has some step as element (?rcp p0:HasProceduralSTR ?up,
?up p0:HasElement ?op and ?op rdf:type p0:Step triples). From this set of results, the query filters the unit procedure that is of interest to the competency question (FILTER (?rcp = p0:MakeGelRcp)).

The third question searches whether there is a phase that needs the end of the *AddWater* phase to start its execution. This query is more complex because it should consider different situations in which the mentioned phase is located in the PFC: (i) before a *DecisionNode*; (ii) before a *ForkNode*; (iii) in a simple sequence, (iv) after a *JoinNode*; or (v) after a *MergeNode*.

Table 2. SPARQL queries used to formalize some competency questions

Informal CQ	SPARQL query	Results
Which are the conditions that should be fulfilled to start the *Prepare2Transer* operation?	**SELECT** ?cnd **WHERE** {?lk rdf:type p0:ExplicitTransition. ?lk p0:HasCondition ?cnd. ?lk p0:To ?stp. ?stp p0:HasStepLevel p0:Operation. **FILTER** (?stp = p0:Prepare2TransferStep)}	|cnd| SampleAprovedByLab
Which phases are executed in parallel with the *AddFillers* phase?	**SELECT** ?ph **WHERE** { ?frk rdf:type p0:ForkNode. ?lk p0:From ?frk. ?lk p0:To ?ph. ?ph rdf:type p0:Step. ?lk2 p0:From ?frk. ?lk2 p0:To p0:AddFillersStep. **FILTER** (?ph != p0:AddFillersStep) }	|ph| AddFlavoringStep AddStabilizersStep
Which are the operations comprised in the *MakeGelToothpaste* UnitProcedure?	**SELECT** ?op **WHERE** { ?rcp rdf:type p0:RecipeEntity. ?rcp p0:HasLevel p0:UnitProcedure. ?rcp p0:HasProceduralSTR ?up. ?up p0:HasElement ?op. ?op rdf:type p0:Step. **FILTER** (?rcp = p0:MakeGelRcp)}	|op| AddIngredientStep ReactStep Prepare2TransferStep

Which are the phases that need the conclusion of *AddWater* phase in order to start their execution?	**SELECT** ?op2 **WHERE** { {?lk p0:From ?op. ?lk p0:To ?cnd. ?cnd rdf:type p0:DecisionNode. ?lk2 p0:From ?cnd. ?lk2 p0:To ?op2. ?op2 rdf:type p0:Step. **FILTER** (?op = p0:AddWaterStep) } **UNION**
	{ ?lk p0:From ?op. ?lk p0:To ?cnd. ?cnd rdf:type p0:ForkNode. ?lk2 p0:From ?cnd. ?lk2 p0:To ?op2. ?op2 rdf:type p0:Step. **FILTER** (?op= p0:AddWaterStep) } **UNION** { ?op2 rdf:type p0:Step. ?op rdf:type p0:Step. ?lk p0:To ?op2. ?lk p0:From ?op. ?op2 rdf:type p0:Step **FILTER** (?op = p0:AddWaterStep)} **UNION**
	{ ?lk p0:From ?op. ?lk p0:To ?cnd. ?cnd rdf:type p0:JoinNode. ?lk2 p0:From ?cnd. ?lk2 p0:To ?op2. ?op rdf:type p0:Step. **FILTER** (?op = p0:AddWaterStep)} **UNION** { ?lk p0:From ?op. ?lk p0:To ?cnd. ?cnd rdf:type p0:MergeNode. ?lk2 p0:From ?cnd. ?lk2 p0:To ?op2. ?op rdf:type p0:Step **FILTER** (?op = p0:AddWaterStep)} }
		|op2|
		AddFlavoringsStep
		AddFillersStep
		AddStabilizersStep

The last question presented in Table 2 looks for phases that should be executed in parallel with the *AddFillers* one. In order to do that, the query searches if there is any step that follows the *ForkNode* to which the *AddFillers* phase is linked to. If the mentioned phase does not follow a *ForkNode*, the query gives an empty result because *AddFillers* is not in a parallel sequence.

The implementation of several cases studies, as well as the formalization and execution of the whole set of competency questions have allowed the identification of missing concepts and properties that have to be added to the ontology. Among others, these concepts are related to the detailed description of equipment units and their relations to recipes and the different levels of the procedural control model.

4 Conclusions and Future Work

This contribution describes an ontology based approach that can play a central role in solving nowadays integration problems that appear in the enterprise hierarchical planning pyramid when adopting the ISA-88 and ISA-99 standards. The definition of three ontologies is suggested: one per each standard formalization and another ontology to integrate the previous ones. In particular, the article focuses on the formalization of PFCs, the graphical representation of the recipe procedures proposed by ISA-88. Since knowledge is explicitly and formally expressed, the ontology supports inference processes and an analysis of the construction of valid PFCs. In addition, some methodological aspects of the proposed ontology development process are also shown.

It is necessary to perform more activities to evaluate and validate the proposed formalization. In parallel with these tasks, the formalization of ISA-95 and the definition of the integration ontology are being carried out.

Acknowledgements.

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

References

[Ans00] ANSI/ISA-95.00.01-2000: Enterprise-Control System Integration. Part 1: Models and terminology, 2000.

[Ans10] ANSI/ISA–88.00.01: Batch Control Part 1: Models and Terminology, 2010.

[Fis90] Fisher, T. Batch Control Systems: Design, Application and Implementation. Instrument Society of America, North Carolina, 1990.

[Gom96] Gómez-Pérez, A.: A Framework to Verify Knowledge Sharing. Experts Systems with Application 11, 519–529, 1996.

[Hai11] Hai, R., M. Theiβen, W. Marquardt: An Ontology Based Approach for Operational Process Modeling. Advanced Engineering Informatics, 25, 748-759, 2011.

[Har09] Harjunkoski, I., Nyström, R., Horch, A.: Integration of Scheduling and Control – Theory or Practice? Computers and Chemical Engineering, 33, 1909-1918, 2009.

[ISO04] ISO 18629 - Process specification language. Part 1: Overview and basic principles, 2004.

[Kre06] Kreipl, S., Dickersback, J.T., Pinedo, M.: Coordination Issues in Supply Chain Planning and Scheduling. In: Herrmann, J. (ed), Handbook of Production Scheduling, 177-212. Springer ,2006.

[Mar09] Maravelias, C., Sung C., Integration of Production Planning and Scheduling: Overview, challenges and opportunities, Computers and Chemical Engineering, 33, 1919-1930, 2009.

[Mun10] Muñoz, E., Espuña, A., Puigjaner, L.: Towards an Ontological Infrastructure for Chemical Batch Process Management, Computers and Chemical Engineering, 34, 668-682, 2010.

[Neu13] Neuhaus, F., Vizedom, A., Baclawski, K., Bennett, M., Dean, M., Denny, M., Grüninger, M., Hashemi, A., Longstreth, T., Obrst, L., Ray, S., Sriram, R., Schneider, T., Vegetti, M., West, M., Yim, P. Towards Ontology Evaluation Across the Life Cycle. Journal of Applied Ontology, 8, 179-194, 2013.

[OMG13] Object Management Group. Unified Modeling Language (UML) Version 2.5 - Beta 2 http://www.omg.org/spec/UML/2.5/Beta2/. 2013

[OMG14] Object Management Group. Object Constraint Language (OCL) Version 2.4. http://www.omg.org/spec/OCL/. 2014

[Par00] Parshall, J., L. Lamb. Applying S-88: Batch Control from a User's Perspective. Instrument Society of America, North Carolina, 2000.

[Sho02] Shobrys, D.E, White, D.C.: Planning, Scheduling and Control Systems: Why Cannot they Work Together? Computers and Chemical Engineering 26, 149-160, 2002.

[Usc96] Uschold, M. and M. Gruninger. Ontologies: Principles, Methods and Applications. Knowledge Engineering Review 11, 93-155, 1996.

[W3c04] W3C OWL Working Group, OWL 2 Web Ontology Language Document Overview. Technical Report. http://www.w3.org/TR/owl2-overview/. 2004