Dynamic customization and validation of product data models using semantic web tools

Sylvere Krima[1, 3], Allison Barnard Feeney[1], Sebti Foufou[2, 3]

[1]National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA
{sylvere.krima, abf}@nist.gov
[2] Qatar University, Doha, Qatar
sfoufou@qu.edu.qa
[3] Le2i Lab UMR 6306, University de Bourgogne, France

Abstract. Product Lifecycle Management (PLM) has always required robust solutions for representing product data models. Product data models enable information exchange across different organizations, actors, processes and stages in the product lifecycle. In this context, standardization of models plays a key role, since it ensures interoperability between the different systems that support information exchange. These standard models need to support diverse domain-specific requirements from the multitude of disciplines involved during a product's lifecycle. Due to this diversity, challenges are to (1) develop multidisciplinary reusable models, (2) extend them to support new requirements over time (new products, new regulations, new materials, new processes) and (3) implement the resulting gigantic information models. ISO 10303, the reference standard for PLM-related data models provides two mechanisms that enable specialization of generic product data to address some of these challenges. In this paper we introduce the need for dynamic PLM-related information models, detail the existing ISO 10303 method and identify its limitations. We then present a methodology for enhancing that method using the Web Ontology Language (OWL) and ontologies for representing product data models and the SPARQL Inference Notation (SPIN), a new Semantic Web technology, for validating product data and overcoming OWL limitations.

1 Introduction

We live in the information age. Data has become an essential asset for most everyday situations and business interactions. The need to share data, to generate information, and create new knowledge from that data is common to all fields of research and all economic activity. Whether it is financial data, product data, health data, or disaster data, managing that data is a critical, and sometimes costly, process. To manage data well, we must understand that it has a lifecycle composed of several steps including definition, instantiation, transformation, validation and archive. When not proper-

ly defined, data might become incomplete, inconsistent or, even worse, unusable. Requirements for data evolve and we must define new data or update existing data over the entire data lifecycle. Evolving data requirements is an important issue and a technological challenge as it is not possible to define, in advance, information structures that meet requirements you do not yet know.

Specifying information requirements is particularly challenging in domains such as manufacturing where information exchange involves many actors and sharing across multiple functions and software applications. In these situations, each function has its own needs and each application has its own input/output requirements. As a result, it becomes hard to find a common information structure for representing data. The challenge is even bigger when a temporal aspect has to be considered since it requires the ability to extend the information structure dynamically over time. One area within the manufacturing domain that we have identified with these characteristics is Product Lifecycle Management (PLM). PLM involves many global actors using a myriad of software applications that perform a series of product management functions that can last from weeks to decades.

Because the mechanism to extend models is static by its nature, requiring numerous updates of the initial information model, this operation is expensive in cost and time. It requires an understanding of the entire initial model to ensure correct extensions are developed. Software components may need to be updated so they can exchange, understand, and use the information in the new model. Finding an alternative is crucial when dealing with complex products and multiple requirements typical of PLM.

ISO 10303 [1], informally known as STEP, is the reference standard among PLM-related data models [2]. ISO 10303 provides two mechanisms that enable specialization of generic product data to address some of these issues.

2 Presentation of ISO 10303

2.1 Introduction

ISO 10303, most commonly known as the Standard for Exchange of Product model data (STEP), is an international standard designed to exchange digital information, enabling an ever-widening range of engineering software systems to interoperate. STEP is divided into parts, to ease its use and implementation. The parts of STEP that are designed for implementation are called Application Protocols (APs). APs contain information models developed using a standard language, called EXPRESS. The most common exchange structure for EXPRESS information models is also standardized, and is simply referred to as Part 21 [3].

STEP has a broad scope and new capabilities are continually being added to cover emerging user needs. However, the standards-development timeline is quite long, and a more responsive approach was sought for certain types of schema customization. STEP provides two mechanisms that enable customization for domain-specific needs. First, users can define and add new attributes to existing concepts. Second, users can classify STEP instances with an externally controlled vocabulary - this is called external classification. Although user-defined attributes give users the possibility to add new properties to instances, those properties have no formally-defined semantics. Due to their implementation as independent key-value pairs, they are only human interpretable properties. This paper focuses on the external classification approach.

2.2 External classification

The STEP external classification approach defines added semantics formally with an external resource - such as a taxonomy or controlled vocabulary and uses it to classify instances so each instance will contain a link to its formal definition.

To establish links between an instance and its external definition, STEP uses three EXPRESS entities: *Applied_classification_assignment*, *Externally_defined_class* and *External_class_library*. *External_class_library* represents an external classification, *Externally_defined_class* represents a classifier formally defined in the external classification and *Applied_classification_assignment* is the way to apply the external classifier to an instance. The following Part 21 code shows an example of classification where an instance of the *Product* EXPRESS entity is classified as a 'Car', 'http://myontology.org/Car' being an external concept formally-defined in the external library whose identifier is 'http://myontology.org'.

```
#1 = PRODUCT($,$,'Car Assembly',());
#2 = APPLIED_CLASSIFICATION_ASSIGNMENT(#3,$,(#1));
#3 = EXTERNALLY_DEFINED_CLASS('Car',$,$,#4);
#4 = EXTERNAL_CLASS_LIBRARY('http://myontology.org');
```

STEP does not provide any restriction on the formalism to use, so in this example we choose to represent the external classification, also known as Reference Data, using an ontology implemented in **Web Ontology Language (OWL)** [7]. OWL is recommended by the OASIS Product Lifecycle Support Technical Committee [4] for implementing ISO 10303-239 [5]. This ontology formally defines "Car" and a few other concepts. Any of these concepts can be used to classify STEP instances.

2.3 Toward a semantic STEP

Extensions using *External_class* do have well-defined semantics, but present their own set of problems because of the heterogeneous architecture (See **Fig. 1.**) where the classifiers and the instances require integrating two different implementation technologies - OWL and Part 21, which increases the complexity for developers to implement a mechanism for classification of instances.

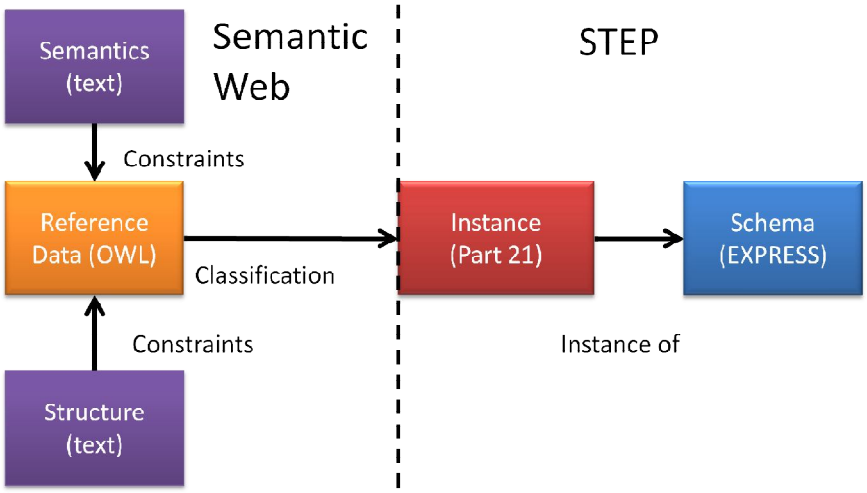

Fig. 1. STEP external classification heterogeneous architecture

One needs to convert both the classifiers and the instances to a common implementation technology that allows dynamic classification of instances so that the type of an instance can change through its lifecycle. A technology that enables such dynamic classifications is the ontologies where classification of instances is driven by constraints. OWL[6] is a language for implementing this mechanism and has been used, in OntoSTEP[1][7], as a destination language to translate STEP APs and instances originally in EXPRESS/Part 21. Once STEP APs and instances are transformed into OWL and combined with an external classification in OWL, one can achieve an automatic classification of instances by enriching the external classification with OWL restrictions. **Fig. 2.** shows an example where an instance of *Product* is classified using an ontology where **Car**, **FastCar** and **SUV** are defined. We then enrich this ontology so that any instance of Product is classified as an instance of **Car** when the STEP method for external classification is correctly used (see bottom condition in **Fig. 2.**). After classification, the Product instance #1 is not only an instance of Product but also an instance of **Car**. In **Fig. 2.** instances are shown as Part 21 for ease of

[1] OntoSTEP plugin for Protégé is available at: http://www.nist.gov/el/msid/ontostep.cfm

readability only. Reference data and the OWL constraint are expressed using the N3 notation[2].

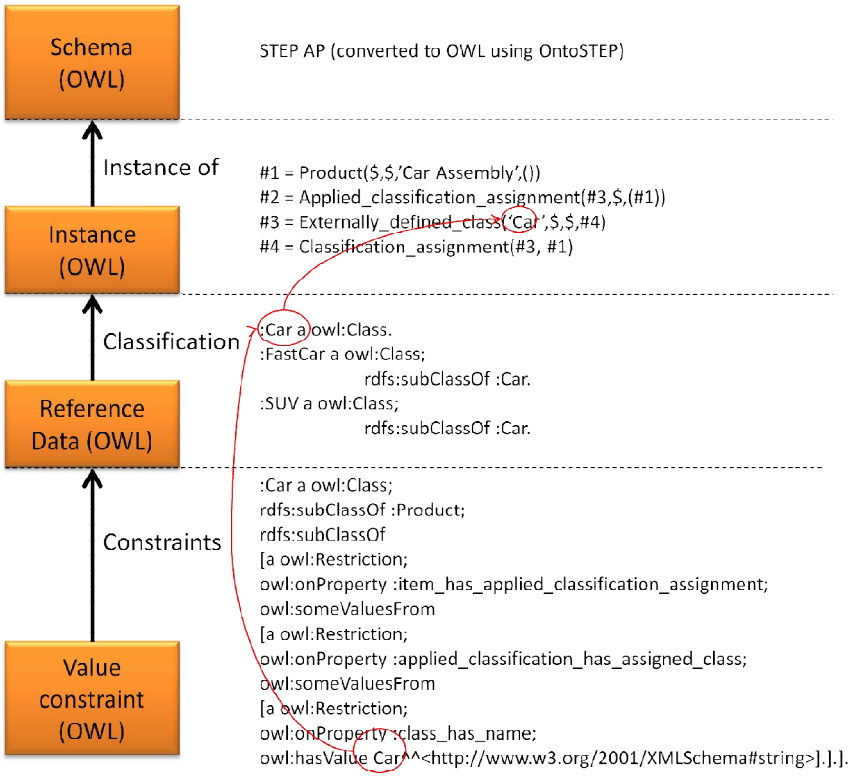

Fig. 2. Example of external classification using STEP and OWL in a homogeneous architecture.

In this section we presented a mechanism, implemented with OWL, which enables automatic classification of instances based on constraints. We converted STEP instances into OWL to perform this classification. However, OWL has limitations that we explain and overcome in the next section.

[2] http://www.w3.org/TeamSubmission/n3/

3 Using OWL for STEP validation

In Section 2, we classified instances based on the string value of an attribute (i.e., the name of the external class used for classification). This classification is purely syntactic and does not ensure that classified instances are semantically correct. To be semantically correct, classified instances need to pass some integrity constraints validation.

As an example of an integrity constraint, consider the following. A car is defined as a product with four wheels (the constraint); any instance of car that does not have four wheels should be seen as an error. Using OWL with the absence of Unique Name Assumption (UNA), where different names refer to different instances, we are essentially dealing with the Open World Assumption (OWA) [8]. In the open world, a car with three wheels cannot be seen as inconsistent with our constraint. This can happen because it is possible that this car has four wheels, but the information about the fourth wheel has not been discovered yet. In other words, open world means that we cannot assume that our knowledge base, used to build our assumptions, is complete. As a result, it is quite complex to use native OWL mechanisms for integrity constraint validation. We need an approach that simulates a closed world.

Research efforts in this domain have yielded some approaches, implementations, and software [9, 10] that provide solutions for validation of integrity constraints when using OWL. SPIN (SPARQL Inferencing Notation) [11] is the solution we have chosen. SPIN is a SPARQL-based rules and constraints language with an object-oriented approach. With SPIN users can define rules and constraints at the class definition level, and then apply them to instances. More importantly for our purpose, implementations of SPIN can produce data validation constraint results as if the world was closed.

Let us consider the following Part 21 instance file (syntactically valid with respect to STEP AP203[3]) that represents five products where one, instance #1 is defined as a car, is an assembly of #6, defined as a body, and three instances of #9, defined as a wheel. The reference data used in #17 is defined using OWL.

```
#1 = PRODUCT($,$,'Car Assembly',());
#2 = PRODUCT_DEFINITION_FORMATION($,'Car assembly',#1);
#3 = PRODUCT($,$,'Body',());
#4 = PRODUCT_DEFINITION_FORMATION($,'Body',#3);
#5 = PRODUCT_DEFINITION($,'Body',#4,$);
#6 = PRODUCT_DEFINITION($,'Car',#2,$);
#7 = PRODUCT($,$,'Wheel',());
```

[3] ISO 10303-203:1994 Industrial automation systems and integration -- Product data representation and exchange -- Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies

```
#8 = PRODUCT_DEFINITION_FORMATION($,'Wheel',#7);
#9 = PRODUCT_DEFINITION($,'Wheel',#8,$);
#10 = NEXT_ASSEMBLY_USAGE_OCCURRENCE($,$,'Body',#6,#5,$);
#11 = NEXT_ASSEMBLY_USAGE_OCCURRENCE($,$,'RF',#6,#9,$);
#12 = NEXT_ASSEMBLY_USAGE_OCCURRENCE($,$,'LF',#6,#9,$);
#13 = NEXT_ASSEMBLY_USAGE_OCCURRENCE($,$,'RR',#6,#9,$);
#22 = APPLIED_CLASSIFICATION_ASSIGNMENT(#19,$,(#1));
#21 = APPLIED_CLASSIFICATION_ASSIGNMENT(#20,$,(#3));
#20 = EXTERNALLY_DEFINED_CLASS('Body',$,$,#17);
#19 = EXTERNALLY_DEFINED_CLASS('Car',$,$,#17);
#18 = EXTERNALLY_DEFINED_CLASS('Wheel',$,$,#17);
#17 = EXTERNAL_CLASS_LIBRARY('http://myOntology/Car');
#16 = APPLIED_CLASSIFICATION_ASSIGNMENT(#18,$,(#7));
```

After applying OntoSTEP and the mechanism described in **Fig. 2.** , we are able to classify instances #1, #6 and #9. Unfortunately, because of the OWA, it is impossible to enrich the reference data, defined in #17, with the following rule: if an instance of *Car* does not have four *Wheels* then the instance is inconsistent.

To overcome the OWA limitations we will use SPIN to enrich the reference data in a way that any instance of Car will raise an inconsistency if it does not have four wheels. First we create an OWL object property called *hasWheel* whose domain is *Car* and range is *Wheel*. We then create a rule, which we attach to the Car class, that instantiates the *hasWheel* object property every time an instance of Wheel is used in the assembly of an instance of a *Car*.

```
CONSTRUCT[4] {
  ?this :hasWheel ?x
}
WHERE {
  ?x rdf:type :Wheel.
  ?pdf :product_definition_formation_has_of_product
  ?this.
  ?pd :product_definition_has_formation :pdf.
  ?nauo
  :product_definition_relationship_has_related_product_de
finition ?pd.
  ?nauo
  :product_definition_relationship_has_relating_product_d
efinition ?pdw.
  ?pdw :product_definition_has_formation ?pdfw.
```

[4] In this SPIN rule: 1) terms prefaced by question marks represent variables bindings 2) instructions are delimited by a "." 3) the ":' sign is used to represent the namespace of a class/property

```
  ?pdfw : product_definition_formation_has_of_product
  ?wheel.
  ?wheel rdf:type :Wheel.
}
```

Now we can enrich the reference data ontology with a SPIN constraint, which we attach to the *Car* class definition that represents an integrity constraint, to raise an inconsistency when an instance of *Car* does not have four wheels; when this instance of *Car* does not have four instances of the *hasWheel* object property we previously defined. Such a constraint can be expressed in SPIN, as follows:

```
ASK WHERE{
  {
    FILTER(spl:objectCount(?this, :hasWheel) <4).
  }UNION{
    FILTER(spl:objectCount(?this, :hasWheel) >4).
  }UNION{
    ?this :hasWheel ?wheel.
    FILTER(!spl:instanceOf(?wheel, :Wheel)).
  }.
}
```

After running the SPIN engine with the enriched reference data and the OntoSTEP result we had, the instance #1 is first classified as an instance of *Car*, but then it is flagged because it only has three wheels. This error could not have been identified by an OWL reasoner due to the OWA. Using SPIN we are able to overcome the OWL's OWA in order to enable integrity constraints validation.

4 Conclusion and Future Work

In this paper, we introduced the need for dynamic information models to support changing data requirements in the Product Lifecycle Management (PLM) area. We then reported on an approach standardized in ISO 10303 (STEP). We acknowledged that although the STEP external classification mechanism shows promise as an effective solution for changing data requirements, its implementation using reference data represented in OWL leads to some issues. The heterogeneous architecture issues that result from the use of different implementation technologies for the STEP data and the external classification are resolved using OntoSTEP. By transforming STEP information models and data to OWL, OntoSTEP enables a homogeneous architecture that takes full advantage of OWL[7].

We highlighted the Open World Assumption (OWA) as an issue when validating classified data. We demonstrated that SPIN, a new semantic web technology, can

overcome validation issues by producing data validation results as if the world was closed. Using SPIN, we are able to maintain consistency despite OWL's OWA.

The scope of this paper is limited to data representation and validation. However, PLM involves numerous and varied data exchanges and sharing[12]. The use of Service Oriented Architecture (SOA) has been identified as an approach for PLM implementations supporting capabilities beyond file-based data exchange[13][14]. Our future work will evaluate methods for enabling our framework to be integrated within SOAs.

References

1. Pratt, M.J.: Introduction to ISO 10303—the STEP Standard for Product Data Exchange. Journal of Computing and Information Science in Engineering. 1, 102 (2001).

2. Mehta, C., Patil, L., Dutta, D.: STEP in the Context of PLM. Advanced Design and Manufacturing Based on STEP. pp. 383-397. Springer London (2009).

3. ISO: 10303-21:2002 Industrial automation systems and integration -- Product data representation and exchange -- Part 21: Implementation methods: Clear text encoding of the exchange structure.

4. OASIS: Reference Data, http://www.plcs-resources.org/plcs/dexlib/help/dex/techdes_refdata.htm.

5. ISO: 10303-239:2005 Industrial automation systems and integration -- Product data representation and exchange -- Part 239: Application protocol: Product life cycle support.

6. W3C: OWL Web Ontology Language, http://www.w3.org/TR/owl-ref/.

7. Krima, S., Barbau, R., Fiorentini, X., Rachuri, S., Foufou, S., Sriram, R.D.: OntoSTEP: OWL-DL ontology for STEP. Proceedings of the International Conference on Product Lifecycle Management PLM'09. pp. 770-780. Inderscience Publishers, Bath, UK (July 6-8, 2009).

8. Elçi, A., Rahnama, B., Kamran, S.: Defining a Strategy to Select Either of Closed/Open World Assumptions on Semantic Robots. 32nd Annual IEEE International Computer Software and Applications Conference. pp. 417-423. IEEE (July 28 - August 1,2008).

9. Motik, B., Horrocks, I., Sattler, U.: Adding Integrity Constraints to OWL. Proceedings of the 3rd International Workshop on OWL: Experiences and Directions. CEUR (June 6-7, 2007).

10. Sirin, E., Tao, J.: Towards Integrity Constraints in OWL. Proceedings of the 6th International Workshop on OWL: Experiences and Directions. CEUR (October 23-24, 2009).

11. Knublauch, H., Hendler, J.A., Idehen, K.: SPIN - Overview and Motivation, http://www.w3.org/Submission/spin-overview/.

12. Srinivasan, V., Lämmer, L., Vettermann, S.: On architecting and implementing a product information sharing service. Journal of Computing and Information Science in Engineering. 8, 110061-1100611 (2008).

13. Lämmer, L., Bugow, R.: PLM Services in Practice. In: Krause, F.L. (ed.) The Future of Product Development Proceedings of the 17th CIRP Design Conference. pp. 503-512. Springer (March 27-28, 2007).

14. Gunpinar, E., Han, S.: Interfacing heterogeneous PDM systems using the PLM Services. Advanced Engineering Informatics. 22, 307-316 (2008).