Dynamic customisation, validation and integration of product data models using semantic web tools

Sylvere Krima* and Allison Barnard Feeney

National Institute of Standards & Technology,
100 Bureau Drive, MS 8260,
Gaithersburg, Maryland 20899-8260, USA
E-mail: sylvere.krima@nist.gov
E-mail: allison.barnardfeeney@nist.gov
*Corresponding author

Sebti Foufou

Department of Computer Science and Engineering,
Qatar University,
P.O. Box 2713, Doha, Qatar
E-mail: sfoufou@qu.edu.qa

Abstract: Standard product data models enable information exchange across different organisations, actors, processes and stages in the product lifecycle. 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 models; 2) extend them to support new requirements over time; 3) implement the resulting gigantic information models. ISO 10303, the reference standard for PLM-related data models provides mechanisms to enable specialisation of generic product data to address some of these challenges. In this paper, we introduce the need for dynamic product data models, detail the ISO method and identify its limitations. We present enhancements to that methodology using ontologies and the SPARQL Inference Notation (SPIN) for validating product data. To conclude, we show how these ontologies can be leveraged to ease and strengthen PLM data integration through the use of Linked Data.

Keywords: Web Ontology Language; OWL; Linked Data; product data integration; product lifecycle management; PLM; product data ontology.

Reference to this paper should be made as follows: Krima, S., Feeney, A.B. and Foufou, S. (2014) 'Dynamic customisation, validation and integration of product data models using semantic web tools', *Int. J. Product Lifecycle Management*, Vol. 7, No. 1, pp.38–53.

Biographical notes: Sylvere Krima has been working for the Engineering Lab (EL) at the National Institute of Standards and Technology (NIST) for 6+ years where his research includes product data sharing, exchange and integration using innovative approaches based on the web semantic. He received his PhD and MSc in Computer Science from the University of Burgundy, France.

Allison Barnard Feeney is the leader of the Smart Manufacturing Operations Planning and Control Programme in the Engineering Lab (EL) at the National Institute of Standards and Technology (NIST). This programme addresses

national problems related to measurements and standards supporting research in advanced manufacturing. She has worked in the areas of manufacturing standards implementation, conformance testing, product data standards, and systems integration for 25 years. She has been a key participant in the development of the STEP product data standard (STEP – Standard for the Exchange of Product Model Data, ISO 10303). She was awarded the Department of Commerce Silver Medal in 2005. She was the 2006 winner of PDES, Inc.'s Bryan K. Martin Technical Excellence Award.

Sebti Foufou received his PhD in Computer Science in 1997 from the University Claude Bernard Lyon 1, France, for a dissertation on parametric surfaces intersections. He worked as a Teaching and Research Assistant at the University Lyon 1 from 1996 to 1998. He was with the Department of Computer Science at the University of Burgundy, France from 1998 to 2009 as Associate Professor then as Full Professor. Between 2004 and 2006, he worked as a Temporary Guest Researcher at the National Institute of Standards and Technology (NIST), MD, USA. He joined Qatar University in September 2009. He is currently the Head of the Computer Science and Engineering Department. His research activities include data representation using semantic web technologies for product lifecycle management, geometric constraint solving with applications to curves and surfaces design, geometric modelling and shape representation. He has published more than 90 papers in peer reviewed journals and international conferences and supervised a dozen of PhD students.

This paper is a revised and expanded version of a paper entitled 'Dynamic customization and validation of product data models using semantic web tools' presented at the 9th International Conference on Product Lifecycle Management, Montreal, Canada, 9–11 July 2012.

1 Introduction

We live in the information age. Data has become an essential asset for most every-day situations and business interactions. The need to share data, to generate information, and create new information from that data is common to all fields of research and all economic activity. To manage data well, we must understand that it has a lifecycle composed of several steps including definition, instantiation, transformation, validation, integration and archive. When not properly defined and integrated (Lenzerini, 2002), data might become incomplete, inconsistent or, even worse, unusable. Data requirements evolve and we must define and manage data over its entire lifecycle. Evolving data requirements is an important issue and a technological challenge because it is not possible to define, in advance, data structures that meet requirements you do not yet know.

Specifying data requirements is particularly challenging in domains such as product lifecycle management[1] (PLM) 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 model 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 PLM that we have identified with these characteristics is manufacturing.

Manufacturing 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 data models are a static view of a domain of discourse (Spyns et al., 2002), extending them require numerous updates of the initial 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 (Pratt, 2001), informally known as STEP, is the reference standard among product-related data models, and is often considered as common model in PLM approaches (Mehta et al., 2009). ISO 10303 provides two mechanisms that enable specialisation of generic product data to address some of these issues. The first goal of this paper is to demonstrate how ontologies and semantically rich models can enable dynamic customisation of product models, using STEP as an example. The second goal of this paper is to demonstrate the role and the importance of closed world validation, and how to achieve it when dealing with ontologies. Finally, we will demonstrate benefits and limitations of semantically rich product data models, to simplify PLM data integration in a disparate and heterogeneous environment.

2 Technical background and foundation

2.1 ISO 10303 for interoperability

ISO 10303, most commonly known as the Standard for Exchange of Product (STEP) model data, 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 standardised, and is simply referred to as Part 21 (ISO 10303-21, 2002).

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 customisation. STEP provides two mechanisms that enable customisation 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 ability 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. We will focus on the external classification approach.

The STEP external classification approach defines added semantics 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', an external concept formally-defined in the external library whose identifier is 'automotive-library'.

```
#1 = PRODUCT($, $, 'Car Assembly',());
#2 = APPLIED_CLASSIFICATION_ASSIGNMENT(#3, $, (#1));
#3 = EXTERNALLY_DEFINED_CLASS('Car', $, $, #4);
#4 = EXTERNAL_CLASS_LIBRARY('automotive-library');
```

2.2 The OWL for reference data

Since STEP does not provide any recommendation on the formalism to use, in this paper we choose to represent the external classification, also known as reference data, using an Web Ontology Language (OWL) ontology (W3C, 2004a). Because "an ontology is an explicit specification of a conceptualization" (Gruber, 1995) and provides "a shared, formal, explicit and partial account of [that] conceptualization" (Uschold and Gruninger, 1996), it is an appropriate candidate to represent reference data. OWL is also recommended by the Organization for the Advancement of Structured Information Standards (OASIS) Product Lifecycle Support Technical Committee (OASIS, 2010) for implementing ISO 10303-239 (2005). Using ontologies for reference data allows us to use unique resource identifiers (URIs) to refer to the external class, as follows:

```
#1 = PRODUCT($, $, 'Car Assembly', ());
#2 = APPLIED_CLASSIFICATION_ASSIGNMENT(#3, $, (#1));
#3 = EXTERNALLY_DEFINED_CLASS('http://nist.gov/rdl#Car', $, $, #4);
#4 = EXTERNAL_CLASS_LIBRARY('automotive-library');
```

2.3 Linked data for integration

PLM data is often spread across a network of systems that produce different product information (e.g., requirements, design, manufacturing, maintenance, logistics), in different representations, in different physical locations. One challenge is to integrate the information in the varied representations together to obtain a global and homogeneous view of the product information, independent from location. Thanks to the internet, data can be relatively easily exchanged and shared across the globe. When properly represented and linked together, disparate information can build one integrated view of the product information.

Berners-Lee (2006) introduced the concept of Linked Data, as part of the W3C semantic web activity, and described it as a way of publishing and linking structured and independent data together, over the web, to enrich its meaning. This concept of Linked Data is based on four principles defined by Tim Berners-Lee, and that appear in the W3C Linked Data platform definition (W3C, 2013):

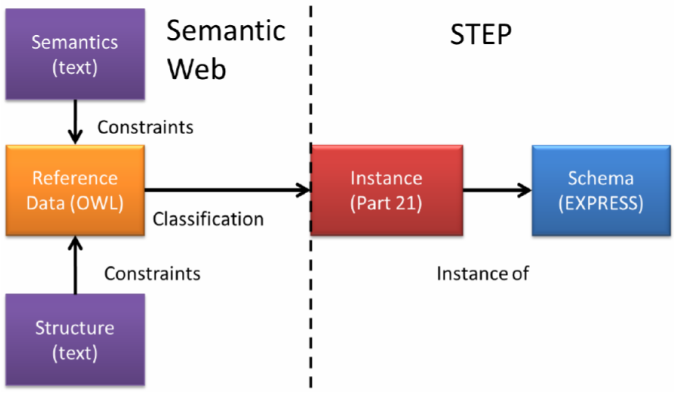
- use URIs as name for things
- use HTTP URIs so that people and user agents can look up those names
- provide useful information, in standard form (RDF*, SPARQL)
- include links to other URIs to enable further discovery

Using globally unique identifiers to name things, and representing information about those things using standards, allows people and software to discover related information, formally and without ambiguity, over the internet. Consequently, identical things (and their related information) will be represented with different identifiers when published by different sources or in different datasets. The connection between things and related information and identification of the identical things through different datasets enable building a rich, meaningful and structured network of information.

3 Dynamic customisation: using OWL for semantic STEP

With OWL, extensions using *External_class* do have well-defined semantics, but present their own set of problems because of the heterogeneous architecture (see Figure 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.

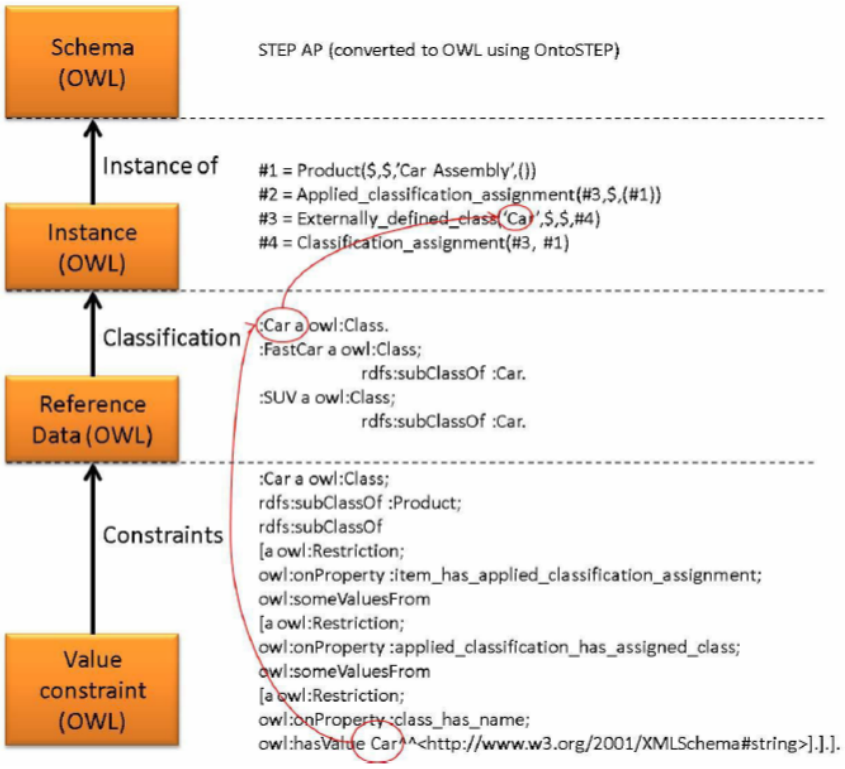
Figure 1 STEP external classification heterogeneous architecture (see online version for colours)



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 is a language for implementing this mechanism and has been used, in OntoSTEP[2] (Krima et al., 2009) 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. Figure 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 Figure 2). After classification, the Product instance #1 is not only an instance of Product but also an instance of Car. In Figure 2 instances are shown as Part 21 for ease of readability only. Reference data and the OWL constraint are expressed using the N3 notation[3].

Figure 2 Example of external classification using STEP and OWL in a homogeneous architecture (see online version for colours)



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.

4 Using OWL for STEP validation

In the previous section, 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 constraint 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) (Elçi et al., 2008). 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 (Motik et al., 2007; Sirin and Tao, 2009) that provide solutions for validation of integrity constraints when using OWL. SPARQL Inference Notation (SPIN) (Knublauch et al., 2011) 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[4]) 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, $);
#7 = PRODUCT($, $, 'Wheel', ());
#8 = PRODUCT_DEFINITION_FORMATION($, '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 Figure 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[5] {
   ?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_definition
?pd.
   ?nauo
   :product_definition_relationship_has_relating_product_definition
?pdw.
   ?pdw :product_definition_has_formation ?pdfw.
   ?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.

5 Implementing Linked Data with OWL

In the previous sections of this paper we discussed using ontologies, classification and query mechanisms provided by semantic web languages and tools to enable easy customisation and validation of data. The power of semantic product data modelling does not stop there. This section will demonstrate how to leverage semantic models to ease data unification across PLM, by following the Linked Data principles.

5.1 Using owl:sameAs for Linked Data

We previously introduced Linked Data as a way of connecting sets of disparate data. These datasets are generally represented using RDF and OWL, which propose properties to connect pieces of information together. Based on the four principles of the Linked Data, two properties are of interest and used:

1 the rdfs:seeAlso[6] property specifies a resource that might provide additional information about the subject resource

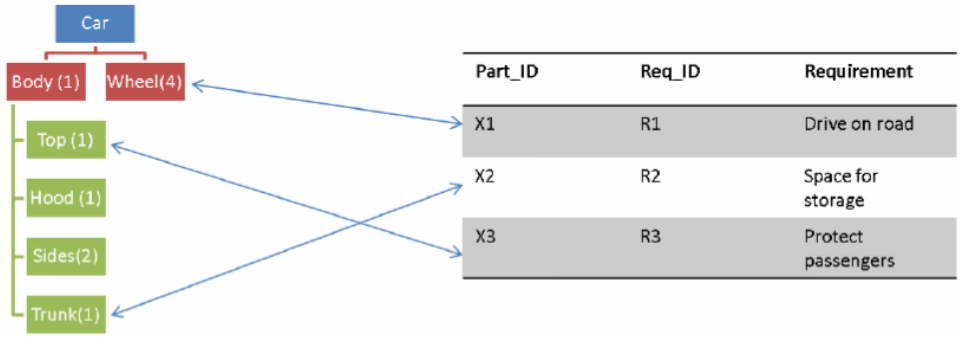
2 the owl:sameAs[7] property to build links between different pieces of information about the identical subject resource, indicating a possible overlap between datasets.

In this paper, we will focus on using the owl:sameAs property to consolidate PLM information from different datasets. The owl:sameAs construct links identical individuals together, having them to share properties, or as specified by the OWL specification: "an owl:sameAs statement indicates that two URI references actually refer to the same thing". When one has, on one side, a set of parts with their requirements and, on the other side, a set of parts with the assembly structure, if the same part appears in both sets, its representations can be linked together using owl:sameAs. This results in a new and more complete dataset that has both requirements and assembly structure. The use of owl:sameAs is made possible because the same part is present in both datasets. Figure 3 shows data we want to link together (the arrows represent owl:sameAs links) based on the part:

1 the graph on the left side is a minimalist assembly structure of a car

2 the table on the right side is a minimalist requirements table of a car.

Figure 3 Linking the assembly structure to the requirements (see online version for colours)



In this use case, data schema and instances are implemented with RDF/OWL as presented in Table 1 and Table 2.

Table 1 Assembly structure and requirements ontologies

Assembly structure model	*Requirement model*

`:Part a owl:Class;`
`rdfs:label`
`"Part"^^xsd:string.`
`:Part_ID a rdf:property;`
`rdfs:domain :Part;`
`rdfs:range xsd:string.`
`:Assembly a owl:Class;`
`rdfs:label`
`"Assembly"^^xsd:string.`
`:Assembly_ID a`
`rdf:property;`
`rdfs:domain :Assembly;`
`rdfs:range xsd:string.`
`:hasSubAssembly a`
`rdf:property;`
`rdfs:domain :Assembly;`
`rdfs:range :Assembly.`
`:HasPart a rdf:property;`
`rdfs:domain :Assembly;`
`rdfs:range :Part.`
	`:Part a owl:Class;`
	`rdfs:label`
	`"Part"^^xsd:string.`
	`:Part_ID a rdf:property;`
	`rdfs:domain :Part;`
	`rdfs:range xsd:string.`
	`:Name a rdf:property;`
	`rdfs:domain :Part;`
	`rdfs:range xsd:string.`
	`:hasRequirement a rdf:property;`
	`rdfs:domain :Part;`
	`rdfs:range :Requirement.`
	`:Requirement a owl:Class;`
	`rdfs:label`
	`"Requirement"^^xsd:string.`
	`:Req_ID a rdf:property;`
	`rdfs:domain`
	`:Requirement;`
	`rdfs:range xsd:string.`
	`:Requirement_text a rdf:property;`
	`rdfs:domain`
	`:Requirement;`
	`rdfs:range xsd:string.`

Table 2 Assembly structure and Requirement instances

Assembly structure instances	*Requirement instances*
:Car a :Assembly; :Assembly_ID "A1"^^xsd:string; :hasSubAssembly :A2; :hasPart [a :Part; :Part_ID "X1"^^xsd:string]. :A2 a :Assembly; :Assembly_ID "A2"^^xsd:string; :hasPart [a :Part; :Part_ID "X2"^^xsd:string]. :hasPart [a :Part; :Part_ID "X3"^^xsd:string].	:Wheel a :Part; :Name "Wheel"^^xsd:string; :Part_ID "X1"^^xsd:string; :hasRequirement [a :Requirement; :Req_ID "R1"^^xsd:string; :Requirement_text "Drive on road".]. :Trunk a :Part; :Name "Trunk"^^xsd:string; :Part_ID "X2"^^xsd:string; :hasRequirement [a :Requirement; :Req_ID "R2"^^xsd:string; :Requirement_text "Space for storage".]. :Top a :Part; :Name "Top"^^xsd:string; :Part_ID "X3"^^xsd:string; :hasRequirement [a :Requirement; :Req_ID "R3"^^xsd:string; :Requirement_text "Protect passengers".].

Because of the use of the owl:sameAs, the two datasets are virtually integrated together into a single model, shown in Table 3. The two datasets are only integrated together when a reasoner is ran and duplicates all the objects of the owl:sameAs predicates, and their properties, into the subject dataset, creating the inferred view. This inferred view, which corresponds to the integrated model, provides new information that can be queried: one can now seamlessly query the model and find out what requirement is attached to the part X1.

Table 3 Assembly structure – requirement integrated model (see online version for colours)

Assembly_Id	*Has_Sub_Assembly*	*Has_Part*		*Part_ID*	*Name*	*Req_ID*	*Requirement*
A1	A2	X1	↔	X1	Wheel	R1	Drive on road
A2	N/A	X2	↔	X2	Trunk	R2	Space for storage
A2	N/A	X3	↔	X3	Top	R3	Protect passengers

Leveraging Linked Data allows to:

1 easily infer new information, using the owl:sameAs, that can be read or queried

2 discover up to date information

3 easily navigate through a trusted network of information.

But because most often this network is open and one can easily enrich it by linking in new datasets, the volume of available information can rapidly impacts performances.

5.2 Limitations and drawbacks of using owl:sameAs

5.2.1 (Too) Strong semantics

The owl:sameAs property is known for its strong implication: its subject and object are declared identical and can be seen as one unique individual, or as formally expressed by "The identity of indiscernables" (Forrest, 2012), $\forall x \forall y(\forall P.\ (P(x) \leftrightarrow P(y) \rightarrow x = y))$. In Halpin et al. (2010), the authors state that the owl:sameAs is being abused and present four contexts in which the identity principle is not respected. We have identified the three following contexts to be of interest to us.

- *Same thing as but referentially opaque:* this happens when two declared identical things violate the principle of substitution, which means the substitution of one by the other will alter the truth of the statement. A famous example used in the literature is based on the adventure of Superman. Clark Kent is Superman, Lois Lane believes Superman can fly, but she believes Clark Kent cannot fly. Since Clark is Superman we should be able, by substitution, to infer that Lois believes Clark can fly, which is not a true statement. Here the substitution would raise an inconsistency.
- *Same thing as but different context:* this situation happens when two things are defined to be identical but are used in different contexts. The same assembly can be a product, P1, in one context and part, P2, in another, and the associated data in the different contexts will not be the same. Inference based on owl:sameAs will combine all properties together while it is not necessarily needed. Inferred properties do not necessarily make sense outside of their original context.
- *Very similar to:* in this situation two things that are not identical (as defined by the previous 'Identity of indiscernables') but very similar (the majority of their properties are the same), are still linked with a owl:sameAs property, because this property is the best way OWL has to express a very strong similarity. If two very similar parts, P1 and P2, are linked through the owl:sameAs property, the implied principle of substitution means that, for example, during a maintenance or repair

operation, a defaulting P1 could be replaced by a new P2. Unfortunately, even given a close-to-perfect similarity between P1 and P2, a difference in a physical property could render P2 unsuitable as a substitute.

These three issues are direct consequences of the strong semantics of the owl:sameAs property. Such issues highlight the need for a more flexible property to represent different levels and conditions of similarity and identity. One of the W3C standards, Simple Knowledge Organization System (SKOS) (W3C, 2004b), aims at providing more flexibility in connecting information together. Initially designed to represent systems such as taxonomies, SKOS provides properties with different levels of similarity to represent close match (skos:closeMatch), exact match (skos:exactMatch), broad match (skos:broadMatch), narrow match (skos:narrowMatch) or related match (skos:relatedMatch). Correctly using these properties reduces the chance of facing some of the issues mentioned earlier.

In the context of the very similar parts P1 and P2, replacing the owl:sameAs property between these parts with a skos:closeMatch would avoid violating the principle of substitution. Not only because these parts are not defined as identical anymore, but also because due to the fuzziness introduced by some of the SKOS properties, no reasoner is able to infer information. This lack of reasoning capabilities raises the importance of a careful choice in the way one connects data where a tradeoff is to be made between the degree of semantics and the reasoning capabilities.

5.2.2 Volume of data

Besides the semantics issue described previously, computational performance is an important drawback in a PLM context due to the important volume of data managed through a product lifecycle. To build a semantically rich network of information based on the owl:sameAs, most reasoners will physically duplicate remote information to the source they are run on. Based on the data provided in Table 1, Table 2 and Table 3, enriched with owl:sameAs links between the Parts of each model, if one runs a reasoner on the Assembly structure dataset, requirements data will be physically moved as shown in Table 4. This new information model, inferred by the reasoner, is physically located where the reasoner has been run. In this example, the data volume and complexity are very low, thus the inference is a seamless operation. In the context of PLM, one expects a high volume of information and a complex network leading to possible scalability and computational issues. Unless the 'cartography' of the network is known, as well as the volume of information, it is impossible to predict the time of computation needed by the reasoner, and the amount of data that will be duplicated to the source.

Table 4 Assembly structure and requirement inferred model

Inferred model: assembly structure + requirements					
Initial model: assembly structure model			*Initial model: requirements model*		
Assembly_Id	*Has_Sub_Assembly*	*Has_Part*	*Name*	*Req_ID*	*Requirement*
A1	A2	Y1	Wheel	R1	Drive on road
A2	N/A	Y2	Trunk	R2	Space for storage
A2	N/A	Y3	Top	R3	Protect passengers

6 Conclusions

In this paper, we introduced the benefits of ontology-based product modelling:

1 to support the need for dynamic information models to support changing data requirements during the product lifecycle

2 as a seamless approach to PLM data integration and consolidation using Linked Data.

Based on the ISO 10303 work on dynamic customisation 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 (Gruber, 1995). We highlighted the 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.

Using the W3C work and recommendations for Linked Data principles and usage we demonstrated how ontological product data can be integrated together with the help of a reasoner. Although Linked Data enables seamless data integration and consolidation, we have identified, with the help of Halpin et al. (2010), semantics issues related to the abuse and strong semantics of the standard owl:sameAs property widely used to implement Linked Data. While we have seen that SKOS properties can be used to reduce this abuse by providing different levels of similarity and identity, the fuzziness of these properties does not allow typical reasoning tools to infer the same information.

Though we demonstrated in this paper the clear benefits of semantic product modelling for data representation, validation and integration, some limits have been highlighted, especially in a data integration context. Some directions for future work include:

1 overcoming the reasoning capabilities limitations due to the use of SKOS

2 evaluating the potential benefits of semantic product modelling to long term product data retention

3 ease the use of semantic technologies in a PLM environment for a wider adoption by its engineering community.

References

Berners-Lee, T. (2006) *Linked Data Design Issues* [online] http://www.w3.org/DesignIssues/LinkedData.html (accessed May 2013).

Elçi, A., Rahnama, B. and Kamran, S. (2008) '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.

Forrest, P. (2012) 'The identity of indiscernables', *The Stanford Encyclopaedia of Philosophy*.

Gruber, T.R. (1995) 'Toward principles for the design of ontologies used for knowledge sharing', *Int. J. Hum. Comput. Stud.*, Vol. 43, Nos. 5–6, pp.907–928.

Halpin, H., Jayes, P.J., McCusker, J.P., McGuinness, D.L. and Thompson, H.S. (2010) 'When owl:sameAs isn't the same: an analysis of identity in Linked Data', *9th International Semantic Web Conference ISWC 2010*, pp.305–320.

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*, International Organization for Standardization.

ISO 10303-239 (2005) *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 239: Application Protocol: Product Life Cycle Support*, International Organization for Standardization.

Knublauch, H., Hendler, J.A. and Idehen, K. (2011) *SPIN – Overview and Motivation* [online] http://www.w3.org/Submission/spin-overview/ (accessed May 2013).

Krima, S., Barbau, R., Fiorentini, X., Rachuri, S., Foufou, S. and Sriram, R.D. (2009) 'OntoSTEP: OWL-DL ontology for STEP', *Proceedings of the International Conference on Product Lifecycle Management PLM'09*, pp.770–780.

Lenzerini, M. (2002) 'Data integration: a theoretical perspective', *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp.233–246.

Mehta, C., Patil, L. and Dutta, D. (2009) 'STEP in the context of PLM', *Advanced Design and Manufacturing Based on STEP*, pp.383–397, Springer, London.

Motik, B., Horrocks, I. and Sattler, U. (2007) 'Adding integrity constraints to OWL', *Proceedings of the 3rd International Workshop on OWL: Experiences and Directions*.

OASIS (2010) *Reference Data* [online] http://www.plcs-resources.org/plcs/dexlib/help/dex/techdes_refdata.htm (accessed May 2013).

Pratt, M.J. (2001) 'Introduction to ISO 10303 – the STEP Standard for Product Data Exchange', *J. Comput. Inf. Sci. Eng.*, March, Vol. 1, No. 1, p.102.

Sirin, E. and Tao, J. (2009) 'Towards integrity constraints in OWL', *Proceedings of the 6th International Workshop on OWL: Experiences and Directions*.

Spyns, P., Meersman, R. and Jarrar, M. (2002) 'Data modelling versus ontology engineering', *ACM SIGMOD Record*, Vol. 31, No. 4, p.12.

Uschold, M. and Gruninger, M. (1996) 'Ontologies: principles, methods and applications', *Knowl. Eng. Rev.*, Vol. 11, No. 2, pp.93–136.

W3C (2004a) *OWL Web Ontology Language* [online] http://www.w3.org/TR/owl-ref/ (accessed May 2013).

W3C (2004b) *SKOS – Simple Knowledge Organization System* [online] http://www.w3.org/2004/02/skos/ (accessed May 2013).

W3C (2013) *Linked Data Platform 1.0* [online] http://www.w3.org/TR/ldp/ (accessed May 2013).

Notes

1 In this paper, we refer to the following definition from the *International Journal of Product Lifecycle Management* (*IJPLM*): "Product lifecycle management is defined as a strategic business approach for effective management and use of corporate intellectual capital. PLM systems are gaining acceptance for managing all information about products throughout their whole lifecycle, from conceptualisation to operations/disposal".

2 OntoSTEP plugin for Protégé is available at: http://www.nist.gov/el/msid/ontostep.cfm (accessed May 2013).

3 http://www.w3.org/TeamSubmission/n3/ (accessed May 2013).

4 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*.

5 In this SPIN rule:

- terms prefaced by question marks represent variables bindings
- instructions are delimited by a '.'
- the ':' sign is used to represent the namespace of a class/property.

6 rdfs:seeAlso definition available at http://www.w3.org/TR/rdf-schema/#ch_seealso (accessed May 2013).

7 owl:sameAs definition available at http://www.w3.org/TR/owl-ref/#sameAs-def (accessed May 2013).