International Conference on Product Lifecycle Management

OntoSTEP: OWL-DL ontology for STEP

Sylvere Krima[12]
sylvere.krima@nist.gov

Raphael Barbau[12]
raphael.barbau@nist.gov

Xenia Fiorentini[1]
xenia.fiorentini@nist.gov

Sudarsan Rachuri[1]
rachuri.sudarsan@nist.gov

Sebti Foufou[2]
sfoufou@u-bourgogne.fr

Ram D. Sriram[1]
sriram@nist.gov

1: National Institute of Standards and Technology
100 Bureau Drive, Gaithersburg, MD 20899
2: University of Burgundy, LE2I laboratory,
UFR sciences, BP 47870, 21078 Dijon Cedex

Abstract: The Standard for the Exchange of Product model data (STEP) [1] contains product information mainly related to geometry. The modeling language used to develop this standard, EXPRESS, does not have logical formalism that will enable rigorous semantics. In this paper we present an OWL-DL (Web Ontology Language - Description Logic) [2] version of STEP (OntoSTEP) that will allow logic reasoning and inference mechanisms and thus enhancing semantic interoperability. The development of OntoSTEP requires the conversion of EXPRESS schema to OWL-DL, and the classification of EXPRESS instances to OWL individuals. Currently we have considered AP203 [3] - the most widely used Application Protocol (AP) for the exchange of Computer-Aided Design (CAD) files - and STEP Part 21 [4] CAD files - CAD files conformant to the data exchange format defined in Part 21 - for schema level conversion and instance level classification respectively. We have implemented a web application to demonstrate OntoSTEP. We are currently extending OntoSTEP to include information such as function, behavior, and assembly requirements.

Keyword: STEP, OWL, ontology, reasoning, semantics

1 Introduction

Manufacturing organizations spend a considerable amount of resources to understand and apply the Product Lifecycle Management (PLM) approach. The PLM approach enables organizations to manage, in an integrated fashion, the product portfolio from conception to disposal [5]. Representation and management of product information is the key for a successful implementation of PLM.

To enable the exchange of product data through a product lifecycle, the International Organization for Standardization (ISO) has developed the Standard for Exchange of Product model data (STEP) [1] (ISO 10303). Unfortunately the representation of function and behavior is outside the scope of current STEP. We call concepts such as function and behavior as “beyond geometry information” since they are most often related to the geometry of the product.

The STEP APs are defined using the EXPRESS language. EXPRESS (ISO 10303-11) [6] is a data modeling language designed by ISO to model STEP entities. STEP Part 21 [4] defines the syntax for representing data according to a given EXPRESS schema.

There is a limited tool support for EXPRESS. Moreover, since EXPRESS is not based on formal semantics, it is difficult to check the quality of these tools.

Our goal is to overcome these issues by translating STEP in OWL-DL [2] (Web Ontology Language - Description Logic), which allows the application of mechanisms to check models and data validity, to check the consistency of the instances, and to infer new knowledge. Measuring model quality is easier since they are based on a formal semantics. We refer to the translated STEP as OntoSTEP in the remainder of the paper. OntoSTEP could be used to express and semantically enrich product information available in STEP files. In our use-case we translate in OWL the STEP AP 203 data model and Part 21 CAD files. The methodology followed for the use-case is fully applicable to any other STEP AP and any Part 21 file.

In our previous work [7], we created a semantic model including beyond geometry concepts from the NIST (National Institute of Standards and Technology) Core Product Model (CPM) [8] and Open Assembly Model (OAM) [9]. In the future, we plan to combine OntoSTEP with the semantic model of CPM and OAM: the instantiation of such a combined model could be, in part, automatically performed from a Part 21 file.

The paper is organized as follows. We review some works related to OntoSTEP in Section 2. Then we introduce the OntoSTEP mapping rules in Section 3. We present our implementation and the tools used to realize it in Section 4 and, finally, we present our conclusions and future plans in Section 5.

2 Related work

This section discusses two related efforts that aim to develop a translation from EXPRESS to OWL. The approaches taken by these authors and their main contributions are explained below.

Intelligent Self-describing Technical and Environmental Networks (S-TEN) [10] is a project funded by the European Community. This project describes a bi-directional translation between EXPRESS and OWL. S-TEN focuses on translating modules, so the translated parts are used within several APs. Hence in the S-TEN project no AP is covered in full. The STEP modules are also modified, either to take advantage of the use of OWL, or as an improvement. For instance, some entities used to express relationships are directly translated to relationships. Moreover new capabilities are added, such as a better management of the product identifiers. A manual check is performed after the translation of the EXPRESS schemas to ensure that the meaning of the data models is the same in EXPRESS and in OWL. The final ontology is stored in a database.

Zhao and Liu proposed a methodology to represent EXPRESS models in OWL and SWRL [11], a rule-based language for OWL. Zhao and Liu translated procedural code contained in the EXPRESS schemas. The procedural code specifies algorithms which can be used to compute derived attributes or to check the validity of data. Since OWL is not a procedural language, the authors chose to use Jess rules to represent EXPRESS procedures and functions. However it is not clear whether this mapping between procedures and Jess rules could work for all the procedures, especially those from AP 203. Moreover, some aspects of the EXPRESS language are not properly dealt with. For instance, the translation of ordered lists in EXPRESS was not proposed. Automated tools doing the entire translation are planned, but we are not aware of any software released.

3 OntoSTEP

The goal of our work is improving interoperability of product data by defining the semantics of the STEP models in a formal logic. In this paper, we translate the EXPRESS models in OWL 2 [12]. OWL-DL, a sublanguage of OWL based on Description Logic, provides several features we need to add semantics:

- *consistency checking*: this mechanism ensures that no contradictions are present within the model
- *inference*: this capability allows to extract new knowledge through logic reasoning
- *decidability*: this characteristic ensures that the reasoning is performed in finite time.

In this section, we present the rules for translating EXPRESS to OWL. The translation of the instance data resulting from the EXPRESS schemas is also introduced.

3.1 Mapping the main concepts

In our translation, EXPRESS entities and instances map respectively to OWL classes and individuals. Attributes correspond to OWL properties, ObjectProperties link classes together, while DataProperties link classes to data types. The domain of a property defines which classes can have this property. Without restrictions, properties in OWL are aggregations, so an individual can be linked several times to other individuals by using the same property. To define the usage of a property, it is possible to restrict its cardinality through the "ObjectExactCardinality" construct and its values through the "ObjectAllValuesFrom" construct. In the case of an optional attribute, the "ObjectAllValuesFrom" construct is used to link the entity to the union of the attribute type and the class owl:Nothing. This solution is adopted to explicitly express the semantics of the OPTIONAL keyword: a value is not required for this attribute.

An ontology may contain statements related to both classes (TBox) and individuals (ABox). In our translation, a schema is translated into an ontology that contains mainly classes and property definitions [13]. The following table summarizes our proposed translation of the basic concepts from EXPRESS to OWL.

Table 1: Translation of the basic concepts from EXPRESS to OWL

EXPRESS	OWL
Schema	Ontology
Entity	Class
Subtype of	Subclass of
Attribute with an entity type	ObjectProperty. The domain of the property is the class that corresponds to the entity that contains the attribute. This class is restricted to have ObjectExactCardinality equal to 1 and ObjectAllValuesFrom equal to the entity type for that property.
Attribute with a simple data type	DataProperty. The domain of the property is the class that corresponds to the entity that contains the attribute. This class restricted to have ObjectExactCardinality equal to 1 and ObjectAllValuesFrom equal to the data type for that property.
Optional attribute	The range of the property is restricted to have ObjectAllValuesFrom equal to the union of the attribute type and the class Nothing.

Attribute with an aggregation type	The range of the property is restricted to have, for that property, minimum and maximum cardinalities corresponding to the aggregation size.

We also need to redefine the naming conventions for the properties. In EXPRESS attributes are defined to be within the scope of the entity, while in OWL properties have a global scope. We choose to prefix the attributes names with the entity names in order to differentiate attributes that have the same name but that belong to different entities.

3.2 Mapping instances

An EXPRESS schema is instantiated by creating a file as defined in "Clear Text Encoding of the Exchange Structure -10303-21," or Part 21. CAD packages can export data in STEP format that conform to the AP203 schema and conform to STEP Part 21's constraints. In this paper we refer to these export files as "Part 21 files."

The translation to OWL is similar to the process described in the previous section and summarized in Table 1. In STEP the schema and the instances are declared in different files: the related schema is specified in the Part 21 file in the FILE_SCHEMA section. OWL provides a similar mechanism of import. The instance file contains an import statement that relates instances to the schema ontology. This import mechanism allows us to maintain the schema ontology separate from the instance one. By having the final ontology containing both the TBox and the ABox, we are able to check the consistency of the instances against the schema. The namespace of the elements declared in the schema ontology indicates the shortened name of the schema: ap203 in our example.

While STEP considers all instances to be different, OWL does not have the unique name assumption, i.e., in OWL a same object can be identified with two different names. The solution to capture the semantics of EXPRESS is to declare all the created individuals as different.

The treatment of an unknown fact is another major difference between EXPRESS and OWL. In EXPRESS, any unknown fact is supposed to be false. For example, if an instance of product is not known to be instance of product_category, the system assumes it is not. This behavior is called the Close World Assumption (CWA), because it supposes that the world is limited to what is stated. OWL uses the Open World Assumption (OWA): unless a reasoner proves a fact is false, that fact is unknown. Hence the translation sometimes requires additional information to capture the semantics of EXPRESS in OWL. The difference between CWA and OWA causes a translation problem when an instance is constrained to have one attribute. The attribute id of the entity product is not declared optional, so it should be instantiated for all the instances of product. In the EXPRESS logic, the lack of data will raise an error. In OWL, even if we do not define an id for an instance of product, the reasoner does not detect an inconsistency: the instance is still considered to have an unknown id. To allow the reasoner to detect an inconsistency in case of missing id, it would be required to declare explicitly that that instance of product has no id.

To fully translate the STEP APs, the translation of some additional concepts, such as derived data types, is required to be introduced. Our proposed translation from EXPRESS to OWL for these additional concepts is presented in the next section.

3.3 Mapping additional concepts

Let us now consider some additional concepts of EXPRESS and, when possible, propose their translation in OWL. Unfortunately, some constructs of EXPRESS, such as functions, cannot be automatically translated: these constructs usually define entity constraints and attributes computation and may rely on complex algorithms. OWL, as it is based on Description Logic, does not contain any procedural aspects. This section focuses on the EXPRESS language aspects that can be automatically translated to OWL concepts. Some EXPRESS concepts, such as SELECT, ENUMERATION and UNIQUE are all translated in OWL and the details of the translation are provided in [14].

3.3.1 Data types

EXPRESS defines simple data types to cover a wide range of information. These simple types are presented and the definition of constructed types is explained.

EXPRESS includes all the data types required to capture the common product information. OWL inherits the data types defined in the XML Schema Definition (XSD) language. In EXPRESS, some types, like boolean and string, have the exact equivalent in OWL, while other types, like number and real, are represented in a slightly different way in OWL. For example, we translate the real data type in EXPRESS as a double in OWL, even if the precision of those two data types is different. This solution should not lead to major problems since a 32-bit approximation of real numbers is usually sufficient in the product domain. The translation of the logical and binary data types is outside the scope of this paper since these data types are not contained in the AP203.

EXPRESS allows the creation of data types derived from the simple types previously presented. In order to deal with these derived types in OWL, we build a type hierarchy and we apply the concept of data wrapping.

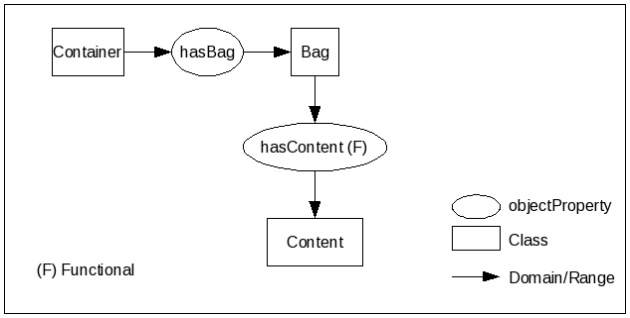
For example, we define a class String that has a DataProperty to the string data type. It is then sufficient to subclass the class String to translate all the user-defined data types related to string (Label in this case). This concept organization allows us to translate all the user-defined data types related to string only by subclassing the class String. Because of the possible use of functions, we cannot guarantee the correctness of an automatic translation of data type restrictions. Using a manual case-by-case translation, most of the types defined in AP203 can be translated.

3.3.2 Aggregations

EXPRESS provides four different types of aggregations: set, bag, list, and array. Each type of aggregation has order policies and duplication policies. For the attribute declarations, the type of content and the number of elements of the aggregation are defined. The detailed mapping of these types are explained in [14]. Here, as an example, we provide the detailed mapping of bag.

Bags are unsorted collection of elements. The only difference between sets and bags is the duplication policy: the same element can be repeated several times in a bag. Because object properties in OWL do not allow duplications, we create the concepts structure shown in Figure 1. A new class, called Bag, is inserted between the Container class and the Content class. The property hasContent is declared functional in order to associate only one element for each instance of Bag.

Figure 1: Bag (Class level)



3.3.3 Abstraction

A supertype in EXPRESS may be declared as abstract. The meaning is the same as in object-oriented programming: an abstract entity cannot be directly instantiated.

OWL does not provide any feature to translate the ABSTRACT keyword and, even if it had, it would not work as expected. Because of the OWA, an ontology is assumed to be incomplete so that the non-instantiation of a concrete entity does not lead to inconsistency. To overcome this problem, we could have declared the subtypes classes as the partition of the supertype. A partition forces the instances of the supertype to belong to at least one subtype. This is achieved by declaring that the set of instances of the supertype is covered by the sets of instances of the subtypes. In that case, if an individual was declared as an instance of the supertype class and not an instance of the subtype class, the reasoner would detect an inconsistency. However this solution only works when both the supertype and all the subtypes are declared within the same schema. Because of these reasons, we choose to ignore the ABSTRACT keyword. It is then impossible for the reasoner to check that abstract entities are not directly instantiated.

3.3.4 Inheritance

In order to specify the allowed combination of subtypes for an entity, EXPRESS provides three keywords: ONEOF, ANDOR, and AND. Along with the ABSTRACT keyword, they restrict the usage of the instantiation mechanism.

ONE OF : the ONEOF keyword takes as parameter a list of entities and it specifies that only one of these entities can be instantiated. An equivalent behavior in OWL is obtained by defining the subclasses as disjoint: an inconsistency is detected when an individual is an instance of two of these subclasses. We mark the set of classes contained in a ONEOF as all disjoint. Another solution could be to use the logical definition of XOR: we could use in OWL the intersection, the union and the complement to translate and, or and not. However this increases the complexity of the ontology, as the length of the formula increases dramatically with the number of elements involved. For this reason we choose the first solution.

ANDOR: when no specific constraints are defined, the default keyword for the instantiation is ANDOR: the instance can belong to more than one subclass. In OWL a set of entities joined by an ANDOR is translated by a union of the corresponding classes in OWL: we first represent the union of the subclasses by using the `ObjectUnionOf` construct and we then declare this union to be equivalent to the parent class.

AND: the AND operator imposes that the object be an instance of all the subclasses. In order to respect this constraint in OWL, we use the `ObjectIntersectionOf` to link the subclasses.

3.4 Benefits

OWL-DL semantics is based on Description Logic (DL), a family of knowledge representation languages. These languages are used to define domain concepts according to a predefined and well understood formalism. Concepts are used to represent the domains objects, while roles are used to represent relationships between these concepts. Concepts and roles are the main components of the knowledge base[1].

OWL-DL provides the "maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time)" [2]. Expressiveness, computational completeness, and decidability enable reasoning mechanisms, e.g., consistency checking. These mechanisms are applied by reasoners to find implicit consequences based on the explicit information provided in a knowledge base. Many reasoners have already been developed. For the purpose of our project we choose Pellet [15].

3.4.1 Consistency checking

The consistency checking procedure can be applied at two different levels: schema level and instances level. At the schema level, the consistency checking procedure checks whether a concept can be instantiated at least once. At the instances level, the consistency checking procedure checks whether an individual declared as an instance of a concept is really instance of that concept.

Currently, libraries are available to check the consistency of EXPRESS schemas and Part21 files but with OntoSTEP, both kinds of consistency checking are performed by a DL reasoner. Checking the logical consistency of the OWL classes and individuals resulting from the translation is the necessary condition to use an inference procedure.

3.4.2 Inference procedure

An inference procedure uses the data evidence in a context and draws conclusions using certain problem solving strategies [16]. The inference procedure is the process to reach these conclusions and it is performed by a reasoner. Reasoners use a knowledge base as a source of data: concepts, roles, and axioms are elaborated by the reasoner to reach the conclusion. The expressivity of the axioms and concepts definitions is dependant on the logic language used.

OWL2 is based on a SROIQ(D) [12] expressivity. All the operators included in the SROIQ(D) expressivity e.g., transitivity, can be used and combined to express axioms. These axioms can be computed only with a reasoner that supports SROIQ(D) expressivity, e.g., Pellet.

In our work, for example, we use inference procedures to represent ordered lists of instances, i.e., to elaborate functional properties, transitive properties, properties hierarchies, and properties chains. All the axioms used in this representation are provided by the SROIQ(D) expressivity.

---

[1] The terms "knowledge base" and "ontology" are used interchangeably for the purpose of this paper.

Once the reasoner has applied all the inferences procedures on our ontology, new knowledge and data become available. Then one can use a querying mechanism to query these new data, which represents an enriched version of the original ontology.

3.4.3 Queries

Queries are performed to retrieve specific data from a large amount of information: in our case we perform queries to retrieve some specific product information from a CAD file. The information contained in a CAD file is first translated into OWL representation, then checked for consistency and inference, and finally queried.

There are two approaches in vogue today to perform queries on OWL ontologies: the first approach uses a language called SPARQL Protocol and RDF Query Language (SPARQL) [17]. while the second approach uses the Semantic Query-Enhanced Web Rule Language (SQWRL) [18]. None of them is used in OntoSTEP for the reasons explained below.

SPARQL was specifically developed for Resource Description Framework (RDF) models, so we would need to translate our OWL ontology to RDF before performing SPARQL queries. We do not adopt this solution because of two reasons. First, the translation from OWL to RDF increases the computational time. Second, the Pellet reasoner does not support some SPARQL built-ins functions, such as DESCRIBE, OPTIONAL, or FILTER [19], or some classical aggregation functions, such as maximum, minimum, sum, or average.

While SPARQL was developed for RDF, SQWRL was specifically being developed for OWL. Unlike SPARQL, SQWRL is based on SWRL [20] and does not need any RDF bridge: the computation of SQWRL queries is then faster. SQWRL provides not only many built-in functions, but also some classical aggregation functions like maximum, minimum, sum, or average [21], which are missing in SPARQL. We do not adopt this solution for two reasons. First, only a proprietary engine, i.e., Jess, is currently available to process SQWRL queries. Second, SQWRL does not allow combining functions together.

To overcome these drawbacks we choose to perform our queries by using the OWL Application Programming Interface (API) for OWL 2 [22] , which is a Java API. This API enables us to manipulate our ontology and to query it. The next section provides more details about this API and its usage.

4 Implementation

The previous section discussed how to generate an ontology from EXPRESS schemas and instance files. This part presents an implementation of the translation rules previously described. The goal is to create tools that perform this generation automatically, and then use these tools to translate both the AP203 and Part 21 CAD files. Three kinds of technologies are used to achieve the above goal:

1) those related to the EXPRESS schemas translation: EXPRESS Parser [23], Another Tool for Language Recognition (ANTLR) [24] parser generator and OWL API;

2) those related to the instances translation: Standard Data Access Interface (SDAI) [25], STEPTools [26] and OWL API;

3) the one related to a web application that facilitates the use of our tools: Google Web Toolkit framework [27].

The web application contains basic login capabilities and uploads a CAD file. Once the file is retrieved, the translation process is launched, and the resulting file containing the A-Box is created. A query engine selects the products and parts corresponding to the criteria input by the user. A detailed use case is provided in [14].

5 Conclusion and future work

Semantic interoperability between the applications that exchange product information is required to achieve systems integration. STEP is the most known and accepted standard for the exchange of product geometry information: its aim is enabling interoperability between engineering applications.

The main benefits of the semantically enriched STEP information presented in this paper are the ability to check the consistency of EXPRESS schemas, the ability to check the validity of the Part 21 files against their schemas and the opportunity of performing queries on those files. In this paper we presented a mapping to OWL-DL from the STEP AP203 and Part 21 CAD files and we showed the principles we followed to create it. These same principles could be used to create OWL mappings of other STEP APs.

We also developed a web application to allow users to upload their CAD files, to translate them, and to manage and query their product ontologies.

In the future, we plan to combine OntoSTEP with the OWL-DL versions of the CPM/OAM, which are information models to support beyond geometry information. We also plan to develop a plug-in to CAD applications to allow the insertion of this information, which would then be checked for consistency along with the geometry information. We also plan to strengthen OntoSTEP by formalizing at the MetaObject Facility (MOF) [28] level the translation between EXPRESS and OWL. For both these languages, a MOF-compliant metamodel has been developed [29] [30]. A translation between these metamodels would allow a bi-directional robust transformation between EXPRESS and OWL.

References

1. International Organization for Standardization. ISO 10303-1: Industrial automation systems and integration -- Product data representation and exchange -- Part 1: Overview and fundamental principles. 1994.

2. OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/ . 2004.

3. International Organization for Standardization. ISO 10303-203: Industrial automation systems and integration -- Product data representation and exchange -- Part 203: Application Protocol: Configuration controlled 3D design of mechanical parts and assemblies. 1994.

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

5. Subrahmanian, E., Rachuri, S., Fenves, S., Foufou, S., and Sriram, R. D.,"Product lifecycle management support: A Challenge in supporting product design and manufacturing in a networked economy" Int.J.Product Lifecycle Management, Vol.1, No.1, 2005, pp. 4-25.

6. International Organization for Standardization. ISO 10303-11: 1994, Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual.

7. Fiorentini, X., Gambino, I., Liang, V., Rachuri, S., Mahesh, M., and Bock, C., "An ontology for assembly representation," National Institute of Standards and Technology, NISTIR 7436, Gaithersburg, MD 20899, USA, 2007.

8. Fenves, S., Foufou, S., Bock, C., Bouillon, N., and Sriram, R. D., "CPM2: A Revised Core Product Model for Representing Design Information ," National Institute of Standards and Technology, NISTIR 7185, Gaithersburg, MD 20899, USA, 2004.

9. Baysal, M. M., Roy, U., Sudarsan, R., Sriram, R. D., and Lyons, K. W., "The Open Assembly Model for the Exchange of assembly and tolerance information: overview and example," Proceedings of the ASME DETC/CIE'04 Conference,2004.

10. Klein, L., Liutkus, G., Nargelas, V., Sileikis, P., Baltramaitis, T., Schowe-von der Brelie, B., Alfter, A., and Wesbuer, C., "Ontologies derived from STEP data models," S-TEN, Deliverable D3.3, 2008.

11.Zhao, W. and Liu, J. K., "OWL/SWRL representation methodology for EXPRESS-driven product information model," Computers in Industry, Vol. 59, 2008, pp. 580-600.

12.W3C. OWL 2 Web Ontology Language: Model-Theoretic Semantics. http://www.w3.org/TR/2008/WD-owl2-semantics-20080411/ . 2008.

13. Fiorentini, X., Rachuri, S., Mahesh, M., Fenves, S., and Sriram, R. D., "Description logic for product information models," Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference,2008.

14. Krima, S., Barbau, R., Fiorentini, X., Rachuri, S., and Sriram, R., "OntoSTEP: OWL-DL Ontology for STEP," National Institue of Standards and Technology, NISTIR 7561, Gaithersburg, MD 20899, USA, 2009.

15. LLC. Pellet. http://clarkparsia.com/pellet/ . 2008.

16. Sriram, R. D., Intelligent Systems for Engineering: A Knowledge-Based Approach, Springer, 1997.

17. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/ . 2008.

18. SQWRL. http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL . 2009.

19. LLC. Pellet Features. http://clarkparsia.com/pellet/features/ . 2008.

20. SWRL, W3C Member Submission. http://www.w3.org/Submission/SWRL/ . 2004.

21.SQWRL - Aggregation functions. http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL#nidA20 . 2009.

22. University of Manchester. OWL API. http://owlapi.sourceforge.net/index.html . 2008.

23.Joshua Lubell, and Stephane Lardet. Open Source EXPRESS Parser. http://sourceforge.net/projects/osexpress/ . 2001.

24. ANTLR Parser generator. http://www.antlr.org/ . 2008.

25. International Organization for Standardization. ISO 10303-22: Industrial automation systems and integration -- Product data representation and exchange -- Part 22: Implementation methods: Standard data access interface. 1998.

26. STEP and STEP-NC Software for e-manufacturing. http://www.steptools.com/ . 2009.

27. Google Web Toolkit. http://code.google.com/webtoolkit/ . 2008.

28. Object Management Group. Meta Object Facility (MOF) Specification. 2002.

29. Object Management Group. Reference Metamodel for the EXPRESS Information Modeling Language RFC. 2008.

30. Object Management Group. Ontology definition metamodel. 2008.