Proceedings of The 2005 ASME International Mechanical Engineering Congress & Exposition
IMECE2005
November 5 -11, 2005 Orlando, FL

IMECE2005-82812

PRODUCT INFORMATION EXCHANGE USING OPEN ASSEMBLY MODEL: ISSUES RELATED TO REPRESENTATION OF GEOMETRIC INFORMATION

Mehmet Murat Baysal
Knowledge Based Engineering Laboratory
Department of Mechanical and Aerospace Engineering
Syracuse University
149 Link Hall, Syracuse, NY 13244, USA
E-mail: mmbaysal@syr.edu

Utpal Roy*
Knowledge Based Engineering Laboratory
Department of Mechanical and Aerospace Engineering
Syracuse University
Syracuse, NY 13244-1240
Email: uroy@ecs.syr.edu

Rachuri Sudarsan
Manufacturing Systems Integration, Division
National Institute of Standards and Technology USA
and George Washington University,
Gaithersburg, 20899 8263, MD, Washington DC.
E-mail: Sudarsan@nist.gov

Ram D. Sriram
Design and Process Group Manufacturing
Systems Integration Division
National Institute of Standards & Technology
Gaithersburg, MD 20899
Email: sriram@nist.gov

Kevin W. Lyons
Design and Process Group Manufacturing
Systems Integration Division
National Institute of Standards & Technology
Gaithersburg, MD 20899
Email: klyons@cme.nist.gov

ABSTRACT

The objective of this paper is to discuss the main issues for product information exchange through the Open Assembly Model (OAM). The OAM model provides a base level product model that is open, simple, generic, expandable, independent of any vendor software and product development process, and capable of engineering context that is shared throughout the product lifecycle.

Two of the main issues in the OAM model are the representation of geometric information of the artifacts (and assembly features) and maintenance of the consistency of the product information among relevant classes based on geometry information. This paper considers the geometry information at three levels: 1) basic geometric information of artifact with position and orientation information, 2) assembly features and their interrelations, and 3) detailed geometric information of all features in the artifact. In addition to geometric information, other relations/associations between the classes in the Unified Modeling Language (UML) based OAM model are maintained by constraints written in Object Constraint Language (OCL). This information structure in the UML and OCL is then mapped into the Extensible Markup Language (XML) for easy information exchange. XML is commonly used and supported by many softwares. Therefore, integration of XML with UML will provide an excellent tool for internet based collaboration.

1. INTRODUCTION

With an incredible advancement in the computer technology (information and bandwidth wise), real time collaboration between tools and people are now available via internet. However, the success of a project still depends on the delivery of the right information to right engineers at the right time. This becomes very important while people are exchanging product data in any particular design phase through other product life cycle processes and tools. For this reason, many research studies have focused on creating a standard interoperability model or finding out a mapping mechanism between existing standard information models. This paper outlines the main issues for information exchange and discusses ways of mapping information through the OAM model.

This paper intends to answer four basic queries: 1) which information model is better for the OAM model? 2) how can we translate/map the existing information models/standards used in Product Lifecycle (PLM) systems from/to the OAM model? 3) how can we maintain consistency of product information among relevant classes? and 4) what kind of modifications should be incorporated in the OAM model to cover all of these issues?

* Corresponding author.

The paper is organized as follows: section 2 discusses standard information models available for representation of product information. In section 3, representation of associations and constraints in the OAM model is given. Section 4 discusses the geometry information which is the main theme for all those associations, constraints etc. in the OAM model. Finally in section 5, a representation of a sub-assembly in gearbox is used as an example to show representations of geometry, associations and constraints.

2. REVIEW OF STANDARD INFORMATION MODELS FOR REPRESENTATION OF PRODUCT INFORMATION

Interoperability and integration of product information in product life cycle have become very important and are costing about 1 billion dollars in only the automotive industry [Brunnermeier, 99]. Academic and government institutions and companies have therefore made agreements about open standards, so that product and systems made by any group can work together to reduce interoperability costs. The open standards are mainly for representations of product information, not for software applications. One of the most important open standards is STEP (Standard for the Exchange of Product Model Data) which was developed by ISO (International Organization for Standardization) with the help of industrial consortiums such as PDES, Inc. (http://pdesinc.aticorp.org) and ProSTEP (http://www.prostep.de). XML and UML are also open standards, even though they are not ISO standards. XML is developed by the World Wide Web Consortium (W3C, http://www.w3.org), and UML is developed by the Object Management Group (OMG, http://www.omg.org). STEP/EXPRESS, XML, UML and their relevant concepts are described below.

The Standard for the Exchange of Product model data (STEP – ISO 10303) is a family of standards defining a robust and time-tested methodology for describing product data throughout the lifecycle of the product. STEP is widely used in Computer Aided Design (CAD) and Product Data Management (PDM) systems. The objects to be represented and exchanged using STEP, as well as the associations between these objects, are defined in schemas written in EXPRESS (ISO 10303-11). EXPRESS is a data modeling language for use in engineering data exchange standards that combines the entity-attribute relationship.

XML is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. The Web Ontology Language (OWL) is an on-going work of W3C aimed at producing an XML and RDF based language for ontologies. OWL is based on the DARPA Agent Markup Language (DAML+OIL).

An UML model is a structure of interrelated objects that conform to syntax defined in metamodel. The UML diagram, such as a class diagram, is typically not refined enough to provide all the relevant aspects of a specification. There is, among other things, a need to describe additional constraints about the objects in the model. Such constraints are often described in natural language. Practice has shown that this will always result in ambiguities. In order to write unambiguous constraints, so-called formal languages have been developed. The OCL, a formal language has been used to describe expressions in UML models. These expressions typically specify invariant conditions that must hold for the system being modeled or queries over objects described in the model. UML modelers can use OCL to specify application-specific constraints in their models. UML modelers also uses OCL to specify queries on the UML model, which are completely programming language independent [UML 2.0 OCL]. The UML based OAM model (that has been used in this paper) also uses OCL to define specific constraints to provide all the relevant aspects about the objects in its PDM database in an unambiguous way.

2.1. STEP/EXPRESS – XML – UML RELATIONS

There have been many studies for developing standards for representing product information. STEP/EXPRESS is widely used in industry. However, most researchers now prefer XML, UML or OWL over EXPRESS. There are several reasons for this choice including the followings:

1) XML, UML and OWL are commonly used, and therefore related resources (software/books) are broadly available but EXPRESS is used by a very limited community.
2) XML provides a standard syntax to represent structural data.
3) XML, UML and OWL are better models for web applications. This makes distributed collaboration through the Internet easier and more convenient.

Currently research works are focused on the possibility of integrating STEP with XML and UML [Lubell and Peak 2004, Lubel et. al. 2004]. ISO also provides implementation methods for mapping EXPRESS schema to XML and UML [ISO 10303-28]. ISO 10303-28 “Part 28: Implementation methods: XML representations of EXPRESS schemas and data”, not only enable developers to use low-cost, ubiquitous XML software tools to implement file-based exchange and visualization of STEP instances, but also facilitate the use of STEP information in emerging areas such as XML-based Web Services. Another part of STEP, ISO 10303-25 (Industrial automation systems and integration -- Product data representation and exchange -- Part 25: EXPRESS to OMG XMI binding), focuses on EXPRESS schemas rather than STEP data. It is possible to combine Part 28 with XML Schema production rules to generate an early-bound XML Schema. It is a potential alternative to Part 25 and annotation/configuration of the EXPRESS can be done using UML tools. On the other hand, Part 28 focuses on STEP data but it gives an XML vocabulary which, in some cases, eliminates the need for the EXPRESS schema. The studies showed that mapping between AP233

(another ISO standard application protocol on Systems Engineering) and System Modeling Language (SysML) by using Part 25 was successful [SysML, 05].

Figure 1 shows a mapping of STEP/EXPRESS to XML, UML (through XMI) and STEP-Part 21. Since mapping between EXPRESS and XML/UML is now available and product information exchange (especially through internet) is becoming very important, the OAM model needs to provide a structure which is compatible with STEP/EXPRESS and XML.

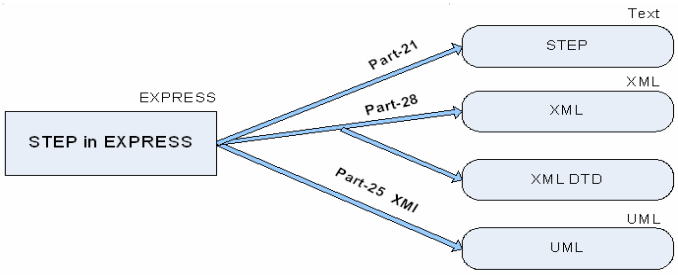


Figure 1. Mapping EXPRESS/STEP to XML and UML (through XMI).

2.2. Overview of the OAM Model

The development of OAM is directed at overcoming the interoperability issues between different CAD tools during different phases of an assembly design. The main difference between OAM and many other available standards is that the assembly model created is not at the end of the product design; instead, the model evolves from an incomplete, preliminary form to a complete model as the design progresses from early design to detailed design phases. Initially, the model starts with customer specified functions and functional requirements. On completion of the design, the OAM databases contain detailed information regarding function, behavior, form/structure, kinematics, assembly and tolerance information for the entire product. For a brief discussion on the OAM model please refer to Appendix I.

In order to facilitate interoperability for product information exchange, the OAM model must be mapped into the XML format as shown in figure 2. The current OAM model in UML has all required relationships among its classes; other constraint information is provided in OCL. This information is then expressed in XML. The OAM model uses XML for information exchange. Currently efforts have been made to map STEP/EXPRESS to UML and XML as well. As we mentioned before, AP233 (a part of STEP) has already been successfully mapped to SysML (in UML) by using Part 25 [SysML]. So, the OAM model in XML will provide an excellent compatibility with other tools written in XML. The next step would be to create an ontology based structure for the OAM model (in OWL) in order to provide a compatible product model for information exchange among different standards/models (Figure 2).

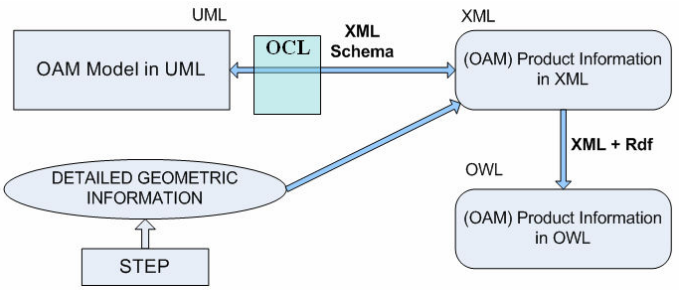


Figure 2. The OAM Model from UML Format to XML and OWL Formats

2.3. Representation of Associations in the OAM Model

In order to represent geometry information, there is a need to understand the representation of associations in the OAM model. The associations in the OAM model are described in three different levels; assembly, artifact (part) and assembly feature. The diagram in figure A1 incorporates information about assembly relationships and component composition. The class AssemblyAssociation represents the component assembly relationship of an assembly. It is the aggregation of one or more Artifact Associations. An ArtifactAssociation class represents the assembly relationship between one or more artifacts. The class AssemblyFeatureAssociation represents the association between mating assembly features through which relevant artifacts are associated. The class ArtifactAssociation is the aggregation of AssemblyFeatureAssociation. These associations and other relevant relations could be better understood if different design phases of a product development process could be followed.

In the conceptual design phase, system level artifacts (main assembly and major parts) are defined with incomplete information. As an example, artifact "Art12" (Planetary Gear Carrier) has been defined (as shown in figure 3) without any information on behavior or form/structure in the beginning of the design. The part level artifacts (e.g. Gear1, Pin1 etc. in figure 3) are introduced in the preliminary design phase and then fully defined in the detailed design phase and other phases.

From the conceptual design stage to the detailed design phases, the associations are specified one by one, beginning with the assembly and artifact associations (in the conceptual and preliminary design phases) to the assembly feature associations and kinematic relations (in the preliminary and detailed design phases). After the artifacts are designed at the detailed design phase, the assembly features and associations between them are then defined as detailed representations of the artifact associations (figure 4). First level association (assembly association) is defined between Assembly1 and Part3 (which are the components of Assembly2) (figure 4 (a)). There is only one connection between Assembly1 and Part3 for assembly level association. In second level association (artifact association) is established between artifacts; ArtifactAccosiation1 between Part1 and Part3,

AssemblyAssociation2 between Part2 and Part3 (figure 4 (a)). In the sublevel of ArtifactAssociation2, there are two assembly associations; AssemblyFeatureAssociation1 between AssemblyFeature2-1 (Part2) and AssemblyFeature3-1 (Part3), and AssemblyFeatureAssociation2 between AssemblyFeature2-2 (Part2) and AssemblyFeature3-2 (Part3) (figure 4 (b)). Namely, assembly feature associations aggregate to artifact associations which also aggregate to assembly associations in the OAM model.

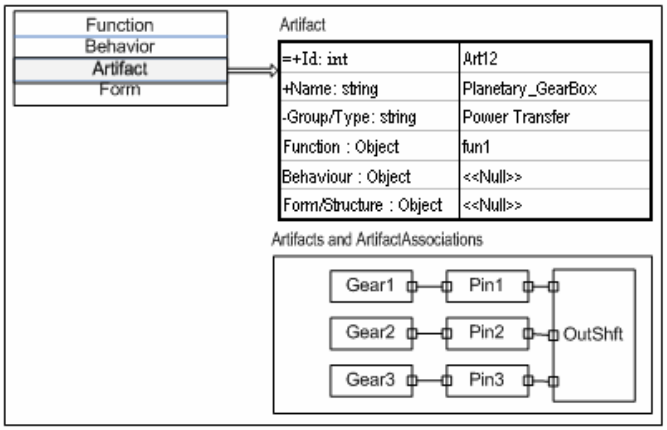
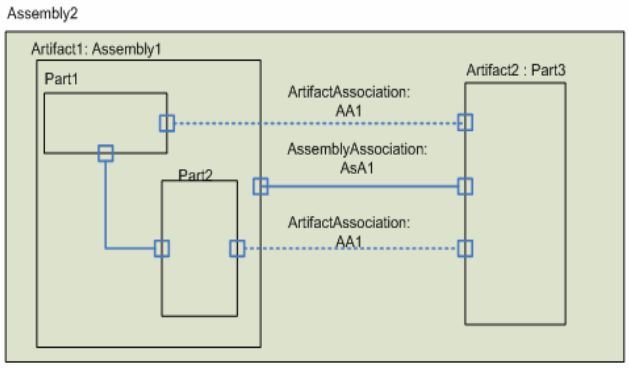
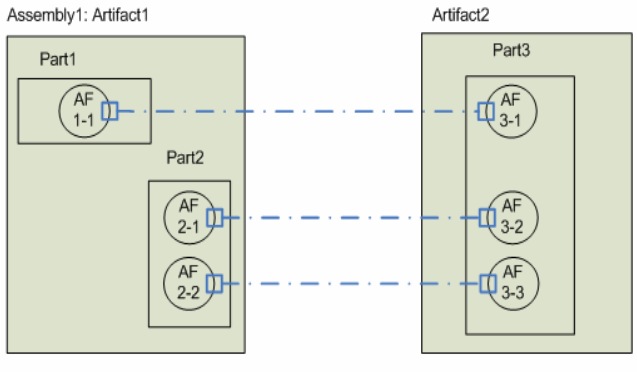


Figure 3. The User Interface for "OAM Objects" Input.



(a) Assembly and Artifact Associations



(b) Assembly Feature Associations

Figure 4. Representation of Associations in the OAM Model

Since, the relations of the three level associations (classes) are defined in the OAM model (figure A1), it is needed to connect the geometry information of those classes by rules/constraints. In following section 3, the information regarding geometry and other constraints among them has been discussed in details.

3. REPRESENTATION OF GEOMETRY INFORMATION AND RELEVANT CONSTRAINTS IN THE OAM MODEL

One of the main issues of the product information representation in the OAM model is the representation of geometry information of the entities in relevant classes in a consistent way. The basis of geometry information of the OAM is the STEP standard. The necessary information for the OAM model may be extracted from the STEP data structure. Other design information related to the function, behavior, design rationale etc, is built up within the model. Geometry information in STEP is very detailed and associated with other standards and parts in ISO-STEP. The OAM model cannot be fully populated and tested without a geometry information structure. Extracting required information from STEP to the OAM model is a complicated and time consuming process. Total mapping of STEP entities (30,000 definitions for transfer from a CAD to another CAD tool [Ray, 2002]) to the OAM model is not possible at present. The work is still on-going. Therefore, we provide this geometry information manually in this work. We decided to classify geometry information in three levels (Table 1); 1) basic geometry information of artifact with position and orientation information of artifact within assembly, 2) assembly features and their interrelationships (type and information) and 3) detailed geometry information of all features in artifact (from STEP), and manually input the information in the OAM model.

Table1. The Geometry Information Levels

1st level: (Artifact)	Artifact:Pin1: geometry_Art (diameter=,length=) Artifact: Gear1: geometry_ Art (hole_diameter = , hole_depth = , pitch_diameter, module, major and minor_diameters, face_width)
2nd level: (Assembly Feature)	AF1:PinCyl1: geometry_ AF.type= Cylindrical : Centre (x, y, z), Radius = , length = , AF2:GearHole1: geometry_AF.type = Cylindrical: Centre(x, y, z), Radius = , length = ,
3rd level: (Detailed)	Artifact geometry_Detailed (since the STEP already includes geometry information and can be translated into XML, we can get detailed relevant geometry information of features in the artifact from STEP files which are translated to XML.)

First two levels of the geometry information is enough to provide the necessary information that is required in the OAM classes/objects for the assembly and tolerance related purposes. The model can then be tested without requiring detailed

geometry information. But these two levels of information are not enough for a complete representation of geometry information of product required to support all other product life cycle tools. This makes the detailed (3rd level) geometry information an important issue which needs to be solved.

Mapping of some basic geometry information (e.g. from EXPRESS to XML and UML) is shown in Appendix II that could be used for the exchanging product information among existing information models. In this section, we show the representation of geometry information and geometry based relations between different classes and constraints to maintain consistency in these geometry related relationships. Let us take an example of the artifact Pin1 (which has been also used in the section 4) to represent its geometry. The following XML definition describes the geometry of the Pin1 in three different levels.

```
<form>
  <geometry>
    <geometry_Art>   "Radius= , Length = "</geometry_Art>
                              // basic geometry information
    <geometry_AF id="AF8.geometry">
        <type> Cylindrical </type>
        <cylindrical>
          <Centre> x, y, z  </centre>
          <radius> R </radius>
          <length> L </length>
        </cylindrical>
    </geometry_AF>                    // assembly feature geometry
    <geometry_Detailed> "pin1.stp" </geometry_Detailed>
                        // detailed info.  STEP file (in Appendix IV)
  </geometry>
  <material> AISI1050 Steel </material>
</form>
```

If required, any necessary information could be extracted from this XML definition using relevant "tags". For example, for the first level geometry (basic size) information of the artifact "Pin1", we can get "Radius" and "Length" information in-between <geometry_Art> tags. Second level geometry information about assembly features is represented as a part of AssemblyFeature in XML as shown in Appendix III.

It should be noted here that the geometry information represented in the three levels of any particular artifact is self-referenced and interconnected. Needs may arise to establish interrelations between geometry information of different artifacts or assembly features. As an example regarding to the artifacts Pin1 and Gear1 (as described in the section 4), there is a design requirement for concentricity of the gear journal and the pin. In order to define these types of relationships whether they are in between different artifacts in assembly or between the geometry information (in three levels) of the same artifact, establishment of "rules/constraints" becomes necessary. As per our design requirements, we now have to establish the equivalence between the concentricity information in the geometry tolerance of the Pin1 and also establish the kinematic pair relationships between Gear1 and Pin1 using constraints.

For concentricity, the constraint becomes:

AssemblyConstraint.Matingfeature.geometry_AF.

Cylindrical.centre ⇐⇒

Tolerance.GeometricTolerance.CrossReferenced.Location.Con

centric.datum.df.geometry.axis

i.e.: AssemblyConstraint class for this association has two cylindrical assembly features (CylindricalPinSurface1 and JournalSurfaceGear1) with center information. This position information needs to be equivalent to the "axis" information in the concentric geometry tolerance for the same assembly features.

For kinematic pair relation and cylindrical assembly feature, the constraint should be:

KinematicPair.RevolutePair.transferitem1.frame

⇐⇒

AssemblyFeature.geometry.geometry_AF.cylindrical.centre

i.e.: RevolutePair class (sub class of KinematicPair class) has two "transfer_item"s (artifacts: Pin1 and Gear1) and those "transfer_item"s have "frame (x, y, z)" attribute for the position information of the artifacts. The same position information needs to be stored in the "centre (x, y, z)" attribute (for cylindrical surfaces) of the assembly features (CylindricalPinSurface1 and JournalSurfaceGear1) of the same artifacts.

Constraints are definitely needed in order to define tolerance in the artifacts and its associations. During the design of assembly, both assembly structure (including associations between artifacts and assembly features) and the associated tolerance information evolve continuously; "tolerance" can be carried to the early design phases of product development. When an association is defined between two artifacts, a potential applicable geometric tolerances object is also created by the model. For instance, to show the *artifact association* between planet gear and pin they must be made concentric with an *assembly constraint*. Thus, a concentricity tolerance is defined for one of them (gear) which is referenced to other artifact (pin) accordingly.

4. EXAMPLE

In this section, we are going to show how the OAM model handles associations, and geometry and tolerance information for a subassembly of the planetary gearbox example. Geometry information for associations needs to be defined in different levels. The necessary geometry information in the artifacts, associations and other classes can then be used by other product life cycle purposes i.e. tolerance analysis, assembly planning etc. In the OAM model, first major functional artifacts are entered to the system (i.e. planetary gear carrier, planetary gear etc.) in the conceptual design stage. In later stages, other

artifacts (pin, output shaft) and associations between all those artifacts (as shown in figure 5) are entered in the assembly and artifact levels. In this example artifacts are;

- Planet Gear Carrier Subassembly (includes shaft, pins and gears)
- Planet Carrier Subassembly (includes shaft and pins)
- Planet Gear 1, 2, and 3
- Pin 1, 2, and 3

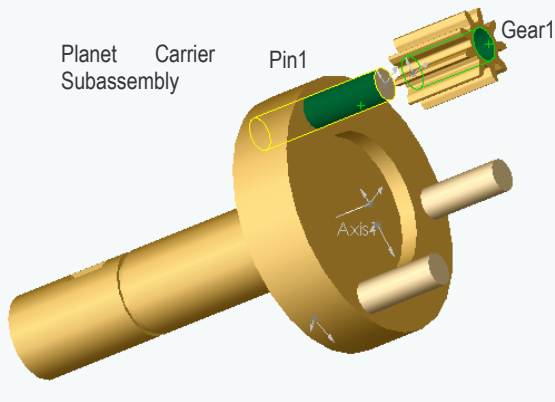


Figure 5. Planet Carrier Subassembly (or Pin1) and Gear1 Association in Artifact Level

Once we have designed artifacts, associations between artifacts are also defined from assembly level to part level. The associations between the artifacts according to assembly associations are defined. Then, the more detailed associations between assembly features are established based on assembly constraints (Table 2). For example, when a "concentricity" assembly constraint between cylindrical surfaces of Pin1 and Gear1 and a "coincident" assembly constraint between flat surfaces of Pin1 (CylindricalPinSurface1) and Gear1 (JournalSurfaceGear1) (figure 6) are defined, it means first there is a association between Planet Carrier Subassembly and Gear1 in assembly level, then there is an association between artifacts Pin1 and Gear1, finally associations (AF1 and AF2) between the assembly features (as shown in figure 7) which are described as mating features in assembly constraints.

Table 2. Association Levels

Association Level	**Associated Elements**
Assembly Level	Planet Carrier – Gear1
Artifact Level	Pin1 – Gear1
Assembly Feature Level	1). CylindricalPinSurface1 (AF 2-2) JournalSurfaceGear1 (AF 5-2)
	2) FlatSurfacePin1 (AF 2-3) FlatSurfacePlanetGear1 (AF 5-1)

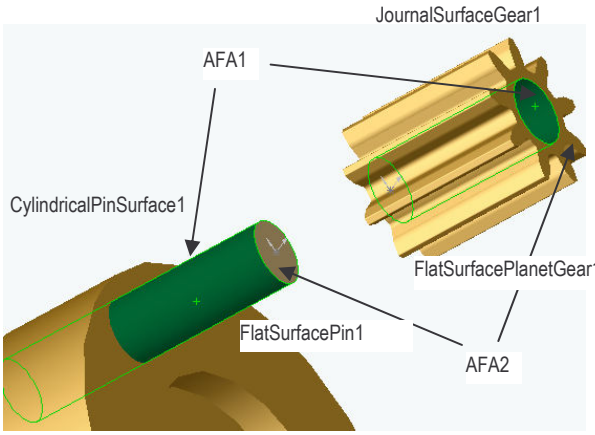


Figure 6. Planet Carrier Subassembly (or Pin1) and Gear1 Association in Assembly Feature Level.

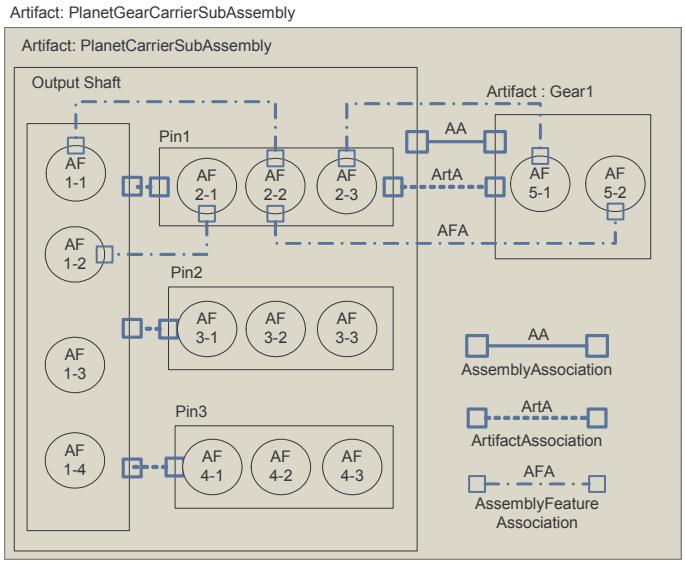


Figure 7. Three Different Associations in Planet Gear Carrier Subassembly.

At this point, the associations between entities in assembly, artifact and assembly feature levels are defined in the OAM model, and the relationships among relevant objects are provided by rules/constraints. For example, relation between *Kinematicpair frame* and artifact (and assembly feature)'s position is defined as

for *kinematicpair.type = revolute*;

CylindricalPinSurface1.geometry.cylindrical.centre = Revolutepair.Pin1.frame
and
JournalSurfaceGear1.geometry.cylindrical.centre = Revolutepair.Gear1.frame

The information (in XML format) about PlanetaryGearPin1 (Pin1), relevant assembly features, kinematic pair and The artifact association also includes kinematic and tolerance information in relevant (KinematicPair and Tolerance) classes. So, any product lifecycle tool (e.g. tolerance analysis tools etc.) can use this information for their particular purposes (i.e. tolerance analysis, assembly planning etc.). In this example, the kinematic pair between Gear1 and Pin1 is a revolute pair which has the "PairValue" and "PairRange" information (based on STEP), and the local coordinate systems of the parts in the "frame" attribute in the RevolutePair class in the OAM model. The local coordinate system for the Pin1 is stored in frame1 $\{x_5, y_5, z_5\}$ and that for the Gear1 is stored in frame2 $\{u_5, v_5, w_5\}$ as shown in figure 8.

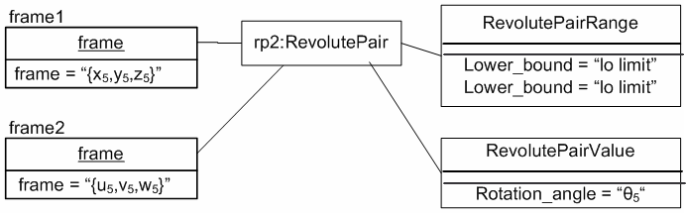


Figure 8. Representation of the KinematicPair information of the Gear1-Pin1 Assembly.

Figure 9 shows the horizontal dimensions of three parts (Gear1, Pin1 and output shaft) and the gap (between Gear1 and Output shaft) as well as related surfaces. In figure, for the number the '21'; first number (2) represents part (Gear1), the second number (1) represents the surface on that part. The dimensional chain and tolerance chain are defined as in equations (1) and (2).

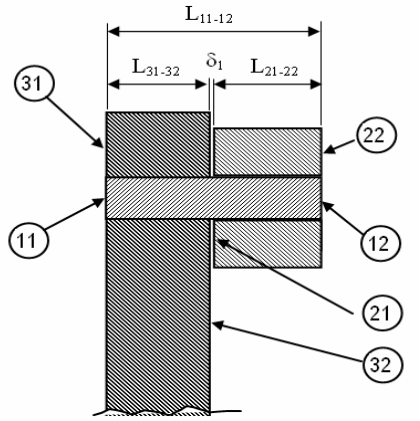


Figure 9. Dimensions and related surfaces in PlanetGearCarrier subassembly.

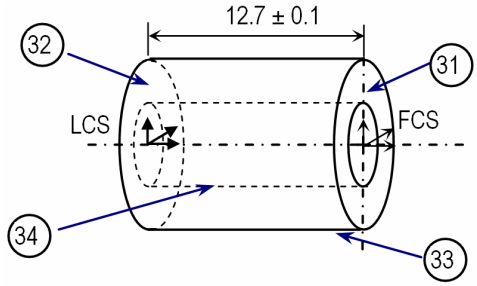
Dimensional chain is;

$$\delta_1 = L_{11-12} - L_{21-22} - L_{31-32} \quad (1)$$

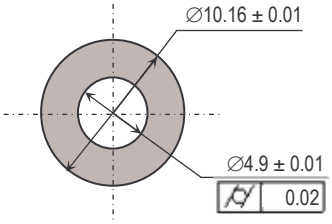
Tolerance chain for δ_1;

$$T_{\delta_1} = t_{11-12} + t_{21-22} + t_{31-32} \quad (2)$$

In the downstream tolerance analysis of the PlanetGearCarrier subassembly, the required information about position and orientation of Gear1 and Pin1, and other assembly features is extracted from the relevant classes (Artifact and AssemblyFeature) in the OAM model. The Local Coordinate System (LCS) gives the positions of the artifacts and the Feature Coordinate System (FCS) gives the center of the assembly feature (figure 10). For the positions of surfaces in horizontal (x) direction, we only need to use the 'x' component of the FCS (for surface 32 is (0, 0, 0); for 31 is (12.7, 0, 0)). After extracting this dimensional information only in x-direction, one can perform the 1-D stack-up analysis.



(a)



(b)

Figure 10. Tolerances and Associated Features of "Gear 1".

Similarly, other geometric tolerances of the artifacts like concentricity, cylindricity etc. (shown in figure 11 and 12) could be extracted from the XML file as follows:

```
<artifact>
   <id> Pin1 </id>
         ……
 <oamfeature>
    <assemblyfeature>
      CylindricalPinSurface1,
       FlatSurfacePin1
    </assemblyfeature>
   ……
    <assemblyfeature>
      <id> CylindricalPinSurface1 </id>
    …….
      <GeometryTolerance>
       <id> GT_CylT1 </id>
       <type> Concentricity_Tolerance </type>
       <magnitude> ± 0.10 </magnitude>
```

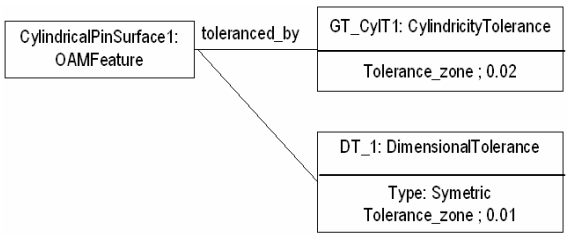


Figure 11. Planet Tolerance information of the "Cylindrical PinSurface1" (Cylindrical Surface of the Pin1) in the OAM Model.

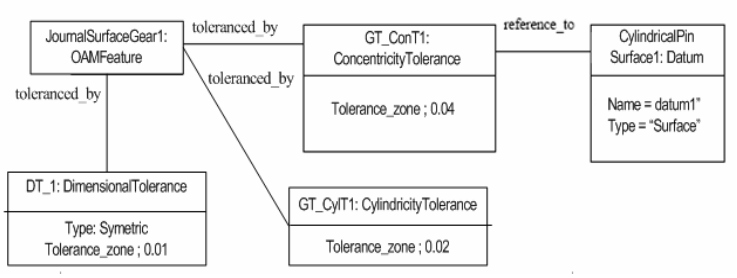


Figure 12. Tolerance information of the "JournalSurfaceGear1" in the OAM Model.

CONCLUSION

The OAM model in XML provides an excellent compatibility with other PLM tools that support XML. In this paper, we showed how to map the OAM model to its XML format and to use it in tolerance analysis. Representation of geometry information in the OAM model is a critical issue, we proposed to represent geometry information in 3 levels and to attach the information to the appropriate OAM classes. Since the detailed geometry information of any artifact is based on the widely used product information exchange model, STEP, it is necessary to map all the geometry related entities from STEP to the OAM model. Since this mapping is not available at present, we manually input the required geometry information of an artifact in the OAM model to test the feasibility of using this OAM model in XML for information exchange. Although, geometry information and relationships between classes/attributes in UML diagram of the OAM model are defined, it is necessary to input constraints between attributes in different classes in OCL format for the consistency of the information. We used an example to illustrate this information representation and extraction process from the OAM model for tolerance analysis of a typical Planet Gear Carrier Subassembly.

Hence, every standard uses their definitions for same information, we plan to create an ontology based structure for the OAM model in order to provide a compatible product model for information exchange among different standards/models.

Disclaimer: No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied. Certain commercial equipments, instruments, or materials are identified in this report in order to facilitate better understanding. Such identification does not imply recommendations or endorsement by the National Institute of Standards and Technology, nor does it imply the materials or equipment identified are necessarily the best available for the purpose.

REFERENCES

Sudarsan R., Mehmet Baysal, Utpal Roy, Sebti Foufou, Conrad Bock, Steven J. Fenves, Eswaran Subrahmanian and Kevin Lyons, 2005, "Information Models For Product Representation: Core And Assembly Models", International Journal of Product Development, Vol. 2, No. 3, 2005.

Baysal, M.M., Roy, U., Sudarsan, R., Sriram, R.D., and Lyons, K.W. (2004) The Open Assembly Model for the Exchange of Assembly and Tolerance Information: Overview and Example, Salt Lake City, Utah.

Ray, S. R., 2002, "Interoperability Standards in the Semantic Web", Journal of Computing and Information Science in Engineering, March 220, Vol. 2, pp.65-69.

Kimber, W. E., 1999, "XML Representation Methods for EXPRESS-Driven Data", NIST GCR 99-781, ISOGEN International Corporation

ISO 10303-28, 2003, "Industrial automation systems and integration -- Product data representation and exchange -- Part 28: Implementation methods: XML representations of EXPRESS schemas and data".

ISO 10303- 25, 2003. "Industrial automation systems and integration -- Product data representation and exchange -- Part 25: EXPRESS to OMG XMI binding",

Lubell, J. and Peak, R., 2004 " STEP, XML, and UML: Complementary Technologies", 24th Annual Computers & Information in Engineering Conference, Salt Lake City, Utah, September 2004, DETC2004-57743.

Lubell, J, Peak, R. S., Srinivasan, V., and Waterbury, S. C., 2004, "STEP, XML, and UML: Complementary Technologies", 24th Annual Computers & Information in Engineering Conference, Salt Lake City, Utah, September 2004, CIE-2004-141.

Brunnermeier, S. B. and Martin, S. A., 1999, "Interoperability Cost Analysis of the U.S. Automotive Supply Chain", Prepared for NIST, RTI Project Number 7007-03, March 1999.

Object Management Group OMG, 2003 "UML 2.0 OCL OMG Final Adopted Specification", http://www.omg.org/issues, 2003.

SysML. 2005, “Systems Modeling Language (SysML) Specification Addendum to SysML v. 0.9 : Profiles and Model Libraries Chapter”, OMG document ad/05-05-01. (http://www.omg.org/cgi-bin/doc?ad/05-05-01)

PDES Inc, http://pdesinc.aticorp.org

ProSTEP http://www.prostep.de

World Wide Web Consortium W3C, http://www.w3.org,

APPENDIX I: THE OPEN ASSEMBLY MODEL

Most electromechanical products are assemblies of components. The aim of the Open Assembly Model (OAM) is to provide a standard representation and exchange protocol for assembly and system-level tolerance information. OAM is extensible; it currently provides for tolerance representation and propagation, representation of kinematics, and engineering analysis at the system level. The assembly information model emphasizes the nature and information requirements for part features and assembly relationships. The model includes both assembly as a concept and assembly as a data structure. For the latter it uses the model data structures of ISO 10303, informally known as the STandard for the Exchange of Product model data (STEP).

Figure A1 shows the main schema of the Open Assembly Model. The schema incorporates information about assembly relationships and component composition; the former is represented by the class **AssemblyAssociation** and the latter is modeled using part-of relationships. The class **AssemblyAssociation** represents the component assembly relationship of an assembly. It is the aggregation of one or more **Artifact Association**s.

An **ArtifactAssociation** class represents the assembly relationship between one or more artifacts. For most cases, the relationship involves two or more artifacts. In some cases, however, it may involve only one artifact to represent a special situation. Such a case may occur when an artifact is to be fixed in space for anchoring the entire assembly with respect to the ground. It can also occur when kinematic information between an artifact at an input point and the ground is to be captured. Such cases can be regarded as relationships between the ground and an artifact. Hence, we allow the artifact association with one artifact associated in these special cases.

An **Assembly** is decomposed into subassemblies and parts. A **Part** is the lowest level component. Each assembly component (whether a sub-assembly or part) is made up of one or more features, represented in the model by **OAMFeature**. The **Assembly** and **Part** classes are subclasses of the CPM **Artifact** class and **OAMFeature** is a subclass of the CPM **Feature** class.

Artifact Association is specialized into the following classes: PositionOrientation, RelativeMotion and Connection. PositionOrientation represents the relative position and orientation between two or more artifacts that are not physically connected and describes the constraints on the relative position and orientation between them. RelativeMotion represents the relative motions between two or more artifacts that are not physically connected and describes the constraints on the relative motions between them. Connection represents the connection between artifacts that are physically connected.

Connection is further specialized as **FixedConnection, MovableConnection,** or **IntermittentConnection**. **FixedConnection** represents a connection in which the participating artifacts are physically connected and describes the type and/or properties of the fixed joints. **MovableConnection** represents the connection in which the participating artifacts are physically connected and movable with respect to one another and describes the type and/or properties of kinematic joints. **IntermittentConnection** represents the connection in which the participating artifacts are physically connected only intermittently.

OAMFeature has tolerance information, represented by the class **Tolerance**, and subclasses **AssemblyFeature** and **CompositeFeature**. **CompositeFeature** represents a composite feature that can be decomposed into multiple simple features. **AssemblyFeature,** a sub-class of **OAMFeature,** is defined to represent assembly features. Assembly features are a collection of geometry entities of artifacts. They may be partial shape elements of any artifact. For example, consider a shaft-bearing connection. A bearing’s hole and a shaft’s cylinder can be viewed as the assembly features that describe the physical connection between the bearing and the shaft. We can also think of geometric elements such as planes, screws and nuts, spheres, cones, and toruses as assembly features.

The class **AssemblyFeatureAssociation** represents the association between mating assembly features through which relevant artifacts are associated. The class **ArtifactAssociation** is the aggregation of **AssemblyFeatureAssociation**. Since associated artifacts can have multiple feature-level associations when assembled, one artifact association may have several assembly features associations at the same time. That is, an artifact association is the aggregation of assembly feature associations. Any assembly feature association relates in general to two or more assembly features. However, as in the special case where an artifact association involves only one artifact, it may involve only one assembly feature when the relevant artifact association has only one artifact.

The class **AssemblyFeatureAssociationRepresentation** represents the assembly relationship between two or more assembly features**.** This class is an aggregation of parametric assembly constraints, a kinematic pair, and/or a relative motion between assembly features. P**arametricAssemblyConstraint** specifies explicit geometric constraints between artifacts of an assembled product, intended to control the position and orientation of artifacts in an assembly. Parametric assembly constraints are defined in ISO 10303-108).

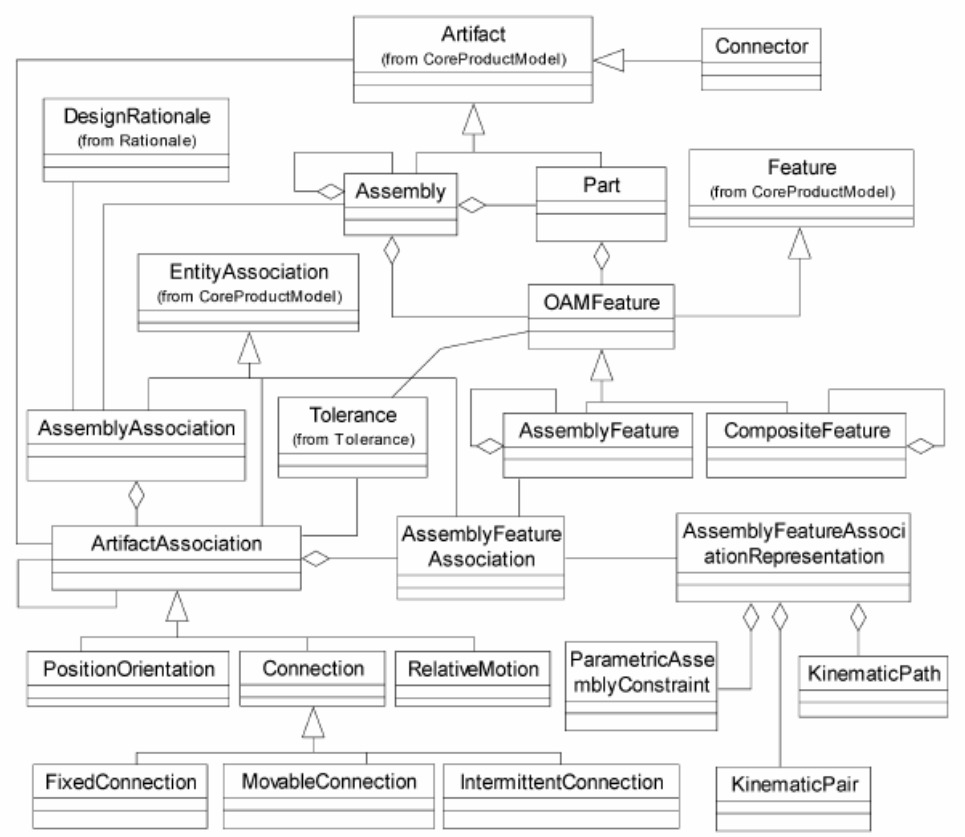


Figure A1: Main Schema of Open Assembly Model

This class is further specialized into specific types: **Parallel**, **ParallelWithDimension**, **SurfaceDistanceWithDimension**, **AngleWithDimension**, **Perpendicular**, **Incidence**, **Coaxial**, **Tangent**, and **FixedComponent**.

KinematicPair defines the kinematic constraints between two adjacent artifacts (links) at a joint. The kinematic structure schema in ISO 10303-105 defines the kinematic structure of a mechanical product in terms of links, pairs, and joints. The kinematic pair represents the geometric aspects of the kinematic constraints of motion between two assembled components. **KinematicPath** represents the relative motion between artifacts. The kinematic motion schema in ISO 10303-105 defines kinematic motion. It is also used to represent the relative motion between artifacts.Tolerancing is a critical issue in the design of electro-mechanical assemblies. Tolerancing includes both tolerance analysis and tolerance synthesis. In the context of electro-mechanical assembly design, tolerance analysis refers to evaluating the effect of variations of individual part or subassembly dimensions on designated dimensions or functions of the resulting assembly. Tolerance synthesis refers to allocation of tolerances to individual parts or sub-assemblies based on tolerance or functional requirements on the assembly. Tolerance design is the process of deriving a description of geometric tolerance specifications for a product from a given set of desired properties of the product. Existing approaches to tolerance analysis and synthesis entail detailed knowledge of the geometry of the assemblies and are mostly applicable only during advanced stages of design, leading to a less than optimal design.

During the design of an assembly, both the assembly structure and the associated tolerance information evolve continuously; significant gains can thus be achieved by effectively using this information to influence the design of that assembly. Any proactive approach to assembly or tolerance analysis in the early design stages will involve making decisions with incomplete information models. In order to carry out early tolerance synthesis and analysis in the conceptual product design stage, we include function, tolerance, and behavior information in the assembly model; this will allow analysis and synthesis of tolerances even with the incomplete data set. In order to achieve this we define a class structure for tolerance specification and we describe this in Figure A2.

DimensionalTolerance typically controls the variability of linear dimensions that describe location, size, and angle; it is also known as tolerancing of perfect form. This is included to accommodate the ISO 1101 standard. **GeometricTolerance** is the general term applied to the category of tolerances used to control shape, position, and runout. It enables tolerances to be placed on attributes of features, where a feature is one or more

pieces of a part surface; feature attributes include size (for certain features), position (certain features), form (flatness, cylindricity, etc.), and relationship (e.g. perpendicular-to). The class **GeometricTolerance** is further specialized into the following: (1) **FormTolerance**; (2) **ProfileTolerance**; (3) **RunoutTolerance**; (4) **OrientationTolerance**; and (5) **LocationTolerance**.

Datum is a theoretically exact or a simulated piece of geometry, such as a point, line, or plane, from which a tolerance is referenced. **DatumFeature** is a physical feature that is applied to establish a datum. **FeatureOfSize** is a feature that is associated with a size dimension, such as the diameter of a spherical or cylindrical surface or the distance between two parallel planes. **StatisticalControl** is a specification that incorporates statistical process controls on the toleranced feature in manufacturing.

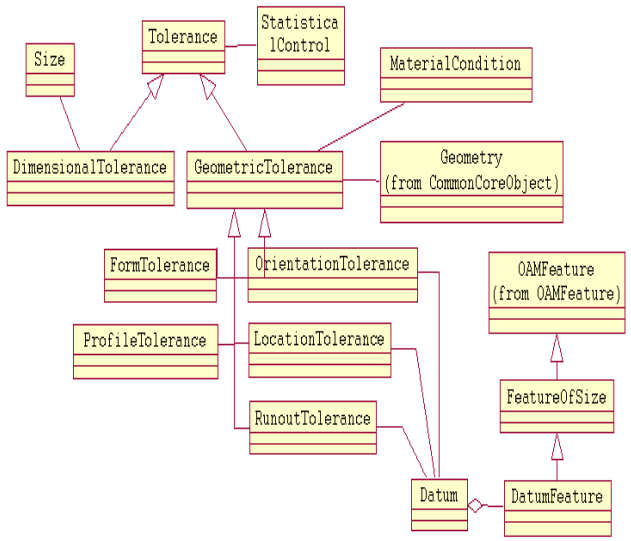


Figure A2: Tolerance Model

APPENDIX II: Some Eamples for Mpping from EXPRESS to XML, UML, and OWL.

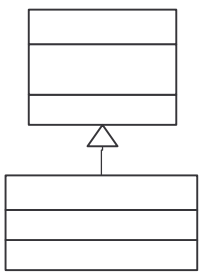
EXPRESS	XML
ENTITY point; x_coord : REAL y_coord : REAL END_ENTITY	<entity name="point"> <attribute name="x_coord"> <REAL/> </attribute> <attribute name="y_coord"> <REAL/> </attribute> </entity>
ENTITY circle: centre : point radius : REAL WHERE radius_non_nega tive : radius >= 0; END_ENTITY	<entity name="circle"> <attribute name="centre"> <type href="#point"/> </attribute> <attribute name="radius"> <REAL/> </attribute> <where-rule name= "radius_non_negative"> <where-rule> </entity>

and also the “Point” and “Circle” entity are shown in part 21 as following;

```
#1=POINT('',(5.E0,5.E0));
#2=DIRECTION('',(1.E0,0.E0));
#3=CENTRE('',#1,#2);
#4=CIRCLE('',#3,2.E0);
```

OWL

```
<owl:Class rdf:ID="Point"/>
<owl:DatatypeProperty rdf:ID="Point.X_ccord">
   <rdfs:domain rdf:resource="&plcs;Point" />
   <rdfs:range  rdf:resource="&xsd;REAL"/>
  </owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Point.Y_coord">
   <rdfs:domain rdf:resource="&plcs;Point" />
   <rdfs:range  rdf:resource="&xsd;REAL"/>
  </owl:DatatypeProperty>
```

EXPRESS	UML
SCHEMA OAMF; ENTITY OAMFeature SUPERTYPE OF (ONEOF (AssemblyFeature)) ID : INTEGER; NAME : STRING; END_ENTITY ENTITY AssemblyFeature SUBTYPE OF (OAMFeature); END_ENTITY	

APPENDIX III: XML Representation of Some of the OAM Classes.

```
<artifact>
    <id> Art1 </id>
    <name> PlanetGearPin1 </name>
    <information> </information>
    <type>  Locator </type>
    <subartifactof> </subartifactof>
    <subartifacts> </subartifacts>
    <function>   </function>
    <form>
        <geometry> </geometry>
        <material> AISI1050 Steel </material>
    </form>
    <oamfeature>
        <assemblyfeature>
            <id> AF8 </id>
            <name> CylindricalPinSurface1 </name>
            <information> “ ”</information>
            <geometry_AF>
                <type> Cylindrical </type>
                <cylindrical>
                    <Centre> x, y, z  </centre>
                    <radius> R </radius>
                    <length> L </length>
                </cylindrical>
            </geometry_AF>
        </assemblyfeature>
    <oamfeature>
</artifact>
```

For he concentricity tolerance between Pin1 and Gear1 is defined as following;

```
<GeometryTolerance>
    <id> GT11 </id>
     <type> Concentricity_Tolerance </type>
   <oamfeature.id>
            CylindricalPinSurface1
    </oamfeature.id>
   <dimension> entity </dimension>
   <magnitude> ± 0.10 </magnitude>
   <datum>
      <id> DatumAxis3 </id>
      <datumfeature>
         <geometry_AF> “axis, x1, y1, z1, x2, y2, z2”
          </geometry>
         <oamfeature.id>
              JournalSurfaceGear1
        <oamfeature.id>
      </datumfeature>
   </datum>
   <mmc> </mmc>
</tolerance>
```

APPENDIX IV: Partial Definition of the Pin1 in STEP Part 203.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''),'2;1');
FILE_NAME('PIN','2004-11-22T',('mmbaysal'),('',
…
DATA;
#1=CARTESIAN_POINT('',(0.E0,0.E0,0.E0));
#2=DIRECTION('',(0.E0,0.E0,1.E0));
#3=DIRECTION('',(1.E0,0.E0,0.E0));
#4=AXIS2_PLACEMENT_3D('',#1,#2,#3);
#6=CARTESIAN_POINT('',(0.E0,0.E0,0.E0));
#7=DIRECTION('',(0.E0,0.E0,1.E0));
#8=DIRECTION('',(-1.E0,0.E0,0.E0));
#9=AXIS2_PLACEMENT_3D('',#6,#7,#8);
#11=DIRECTION('',(0.E0,0.E0,1.E0));
#12=VECTOR('',#11,5.E1);
#13=CARTESIAN_POINT('',(5.E0,0.E0,0.E0));
#14=LINE('',#13,#12);
….
#37=CARTESIAN_POINT('',(0.E0,0.E0,0.E0));
#38=DIRECTION('',(0.E0,0.E0,1.E0));
#39=DIRECTION('',(1.E0,0.E0,0.E0));
#40=AXIS2_PLACEMENT_3D('',#37,#38,#39);
#41=PLANE('',#40);
…
#5=CIRCLE('',#4,5.E0);
#10=CIRCLE('',#9,5.E0);
#23=CIRCLE('',#22,5.E0);
#28=CIRCLE('',#27,5.E0);
….
#42=EDGE_CURVE('',#31,#32,#5,.T.);
#44=EDGE_CURVE('',#32,#31,#10,.T.);
#55=EDGE_CURVE('',#31,#35,#14,.T.);
#57=EDGE_CURVE('',#35,#36,#23,.T.);
#59=EDGE_CURVE('',#32,#36,#18,.T.);
#71=EDGE_CURVE('',#36,#35,#28,.T.);
#97=ADVANCED_BREP_SHAPE_REPRESENTATION
('',(#88),#96);
#104=PRODUCT_DEFINITION('design','',#103,#100);
#105=PRODUCT_DEFINITION_SHAPE('','SHAPE  FOR
PIN.',#104);
#106=SHAPE_DEFINITION_REPRESENTATION(#105,
#97);
ENDSEC;
END-ISO-10303-21;
```