NISTIR 7173

INFORMATION MODELS FOR PRODUCT REPRESENTATION: CORE AND ASSEMBLY MODELS

Sudarsan Rachuri
Mehmet Baysal
Utpal Roy
Sebti Foufou
Conrad Bock
Steven J. Fenves
Ram Sriram
Eswaran Subrahmanian
Kevin Lyons

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7173

INFORMATION MODELS FOR PRODUCT REPRESENTATION: CORE AND ASSEMBLY MODELS

Sudarsan Rachuri
Mehmet Baysal
Utpal Roy
Sebti Foufou
Conrad Bock
Steven J. Fenves
Eswaran Subrahmanian
Kevin Lyons

December 2004

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary
TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arden L. Bement, Jr., Director

ABSTRACT

This report presents a revised and unified view of the NIST Core Product Model (CPM) [1; 2] and the NIST Open Assembly Model (OAM) [3]. The CPM provides a base-level product model that is not tied to any vendor software; open; non-proprietary; simple; generic; expandable; independent of any one product development process; and capable of capturing the engineering context that is shared throughout the product lifecycle. The OAM represents the function, form, and behavior of assemblies and defines both a system level conceptual model and the associated hierarchical relationships. The model provides a way for supporting tolerance representation and propagation, representation of kinematics, and engineering analysis at the system level.

The objectives of the report are:

(1) to report on the revisions and extensions of the two models since their initial documentation;
(2) to link the two models more explicitly than was done before;
(3) to present the revised CPM/OAM as the basis, or organizing principle, of a product information-modeling framework that can support the full range of Product Lifecycle Management (PLM) information needs about a product;

A case study illustrates the use of the CPM and OAM models to capture information during various design phases.

Keywords
Product modeling, information modeling, data modeling, artifact, form, function, behavior, entity-relationship data model, next-generation product development tools, Core Product Model.

TABLE OF CONTENTS

1 Introduction **8**

2 Overview of the Core Product Model **9**

3 Overview of the Open Assembly Model **13**

3.1 OAM Review **13**

3.2 Usage Patterns for Reusable Subassemblies and Parts **17**

4 Applications **23**

4.1 Design of a Planetary Gear **23**

4.1.1 Conceptual design 24

4.1.2 Preliminary design 26

4.1.3 Solid Modeling of parts and assemblies 33

4.1.4 Detailed Design: Tolerance assignment 35

4.2 Usage Example **37**

5 Future Work **39**

6 Conclusions **40**

LIST OF FIGURES

Figure 1: Class diagram of the Core Product Model 10

Figure 2: CPM object classes 11

Figure 3: Class diagram of the Open Assembly Model 15

Figure 4: Tolerance model 17

Figure 5: Repeated subassemblies, parts, and artifact associations in OAM 18

Figure 6: OAM part hierarchy and ArtifactAssociation 19

Figure 7: Usage pattern 19

Figure 8: Example of ReusableElements, Usages, and Contexts in OAM 20

Figure 9: Part hierarchy in OAM 21

Figure 10: Usage metamodel in OAM 22

Figure 11: ArtifactAssociation and FeatureAssociation classes in OAM 23

Figure 12: Conceptual representation of planetary gear in CPM/OAM 25

Figure 13: Basic planetary gear system 26

Figure 14: Preliminary representation of the planetary gear's behavior in CPM/OAM 27

Figure 15: Coordinate systems assigned to KinematicPairs 28

Figure 16: Size constraints of gearbox 34

Figure 17: Geometrical and dimensional tolerancing on sungear (Art7) 35

Figure 18: Tolerancing for the whole gearbox assembly 36

Figure 19: Planet-gear-carrier assembly 37

Figure 20: Instance diagram of the Planet-gear-carrier assembly 38

Figure 21: Instance diagram of the planet-gear-carrier assembly with the usage pattern 39

LIST OF TABLES

Table 1: Usage pattern in OAM 20

Table 2. List of sub-artifacts of planetary_reduction_gear 26

Table 3. KinematicPairs[2] 28

Table 4. RevolutePairs[2] 29

Table 5. GearPairs[2] 29

Table 6. Artifacts[1] 30

Table 7. Artifact associations[2] 30

Table 8. Connections[2] 31

Table 9. List of assemblies[2] 31

Table 10. Assembly associations[2] 32

Table 11. Assembly feature associations[2] 32

Table 12. Assembly constraints[3] 33

Table 13. Assembly features[2] *defined using the user interface [11]* 35

Table 14. Tolerance[3] *information for Features and Artifcats* 36

Table 15. OAMFeatures[3] *(only toleranced features are listed)* 37

LIST OF ABBREVIATIONS

Designation	IDs	Example
Association	A	A1, A2, …
Behavior	B	B1, B2, …
Feature	F	F1, F2, …
Form	Form	Form1, Form2, …
Function	Fun	Fun1, Fun2, …
Constraint	C	C1, C2, …
Representation	R	R1, R2, …
Artifact	Art	Art1, Art2, Art3, …
ArtifactAssociation	ArtA	ArtA1, ArtA2, ArtA3, …
Assembly	Ass	Ass1, Ass2, Ass3, …
AssemblyAssociation	AA	AA1, AA2, AA3, …
AssemblyFeature	AF	AF1, AF2, AF3, …
AssemblyConstraint	AC	AC1, AC2, AC3, …
AssemblyFeatureAssociation	AFA	AFA1, AFA2, AFA3, …
AssemblyFeatureAssociationRepresentation	AFAR	AFAR1, AFAR2, AFAR3, …
PositionOrientation	PO	PO1, PO2, …
FixedConnection	FC	FC1, FC2, …
MovableConnection	MC	MC1, MC2, …

1 Introduction

Globalization of markets and products has led to product development that is increasingly performed by geographically and temporally distributed teams with a high level of outsourcing of many phases of the product development process. This new context demands that new tools be developed to address a broader spectrum of product development activities than those covered by traditional Computer Aided Design and Engineering (CAD/CAE) systems. Next-generation tools will require information models and representations that allow all information used or generated in the various product development activities to be transmitted to other activities by way of direct electronic interchange. Furthermore, product development across companies, and even within a single company, takes place and will be increasingly so, within a heterogeneous software environment.

The NIST Core Product Model (CPM) and its extension provide the required base-level product model that is open, non-proprietary, generic, extensible, independent of any one product development process and is capable of capturing the full engineering context commonly shared in product development [1; 2]. The Open Assembly Model (OAM) Model extends CPM to provide a standard representation and exchange protocol for assembly [3]. 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) [4].

The development of OAM is geared towards overcoming the interoperability issues between different CAD tools during different phases of an assembly design. The main difference of the OAM from any other available standard 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.

The basis of geometry information of the OAM is the STEP application protocol and necessary information may be extracted from the STEP data structure. Other design information related to the function, behavior, design rationale, priorities, tolerance, etc, is built up within the model.

In this report, we give a unified view of the CPM and the OAM models and show how we can use the combination of these models in the exchange and capture of flow of information. Section 2 gives an overview of CPM2: the revised version of CPM [2]. Section 3 gives an overview of the OAM and presents its new extensions. In Section 4, we discuss two applications of these models. We explain (see Section 4.1) the use of these models to realize seamless integration of product information, with an emphasis on assembly, throughout all phases of a product design. We use a gearbox design example to illustrate the process. This example also highlights the value added information that we

are providing in the OAM to realize seamless integration of product information throughout all phases of a product design. We show how the assembly of reusable elements and subparts is improved using the usage pattern presented in Section 3.2. Future extensions and the conclusion of this work are in Sections 5 and 6.

2 Overview of the Core Product Model

The NIST core product model (CPM) is a Unified Modeling Language (UML) based model intended to capture the full range of engineering information commonly shared in product development [1; 2]. It consists of a set of classes, associations and class associations. In order to make the representation as robust as possible without having to predefine attributes that might be relevant only in a given domain, the CPM is limited to attributes required to capture generic product information and to create relationships among the classes. The representation intentionally excludes attributes that are domain-specific (e.g., attributes of mechanical or electronic devices) or object-specific (e.g., attributes specific to function, form or behavior).

A UML class diagram of the core product model is shown in Figure 1. In the text that follows, names of classes are capitalized (e. g., **Information**) and names of attributes are not (e. g., `information`). The classes comprising the CPM are grouped below into four categories: abstract classes, object classes, relationship classes and utility classes.

Five abstract classes are used as base classes for other CPM classes: **CoreProductModel** represents the highest level of generalization; all CPM classes are specialized from it according to the class hierarchy presented in Figure 1. The common attributes `type`, `name` and `information` for all CPM classes are defined in this class. **CommonCoreObject** is the base class for all the object classes. **CommonCoreRelationship** and its specializations, the **EntityAssociation**, **Constraint**, **Usage** and **Trace** relationships, may be applied to instances of classes derived from this class. **CommonCoreRelationship** is the base class from which all association classes are specialized. It also serves as an association to the **CommonCoreObject** class. **CoreEntity** is an abstract class from which the classes **Artifact** and **Feature** are specialized. **EntityAssociation** relationships may be applied to entities in this class. **CoreProperty** is an abstract class from which the classes **Function**, **Flow**, **Form**, **Geometry** and **Material** are specialized. **Constraint** relationships may be applied to instances of this class.

The second set of CPM classes is given in Figure 2. The containment relationship **subArtifacts/subArtifactOf** is illustrated in the figure as an example.

The key object class in the CPM is the **Artifact**. **Artifact** represents a distinct entity in a product, whether that entity is a component, part, subassembly or assembly. All the latter entities can be represented and interrelated through the **subArtifacts/subArtifactOf** containment hierarchy. The **Artifact**'s attributes, other than the common ones described in, refer to the **Specification** responsible for the **Artifact**, the **Form**, **Function** and **Behavior** objects comprising the **Artifact**, i.e., in UML terminology, forming an aggregation with the **Artifact**, and the **Features** comprising the **Artifact**.

A **Feature** is a portion of the artifact's form that has some specific function assigned to it. Thus, an artifact may have design features, analysis features, manufacturing features, etc., as determined by their respective functions. **Feature** has its own containment hierarchy, so that compound features can be created out of other features (but not artifacts).

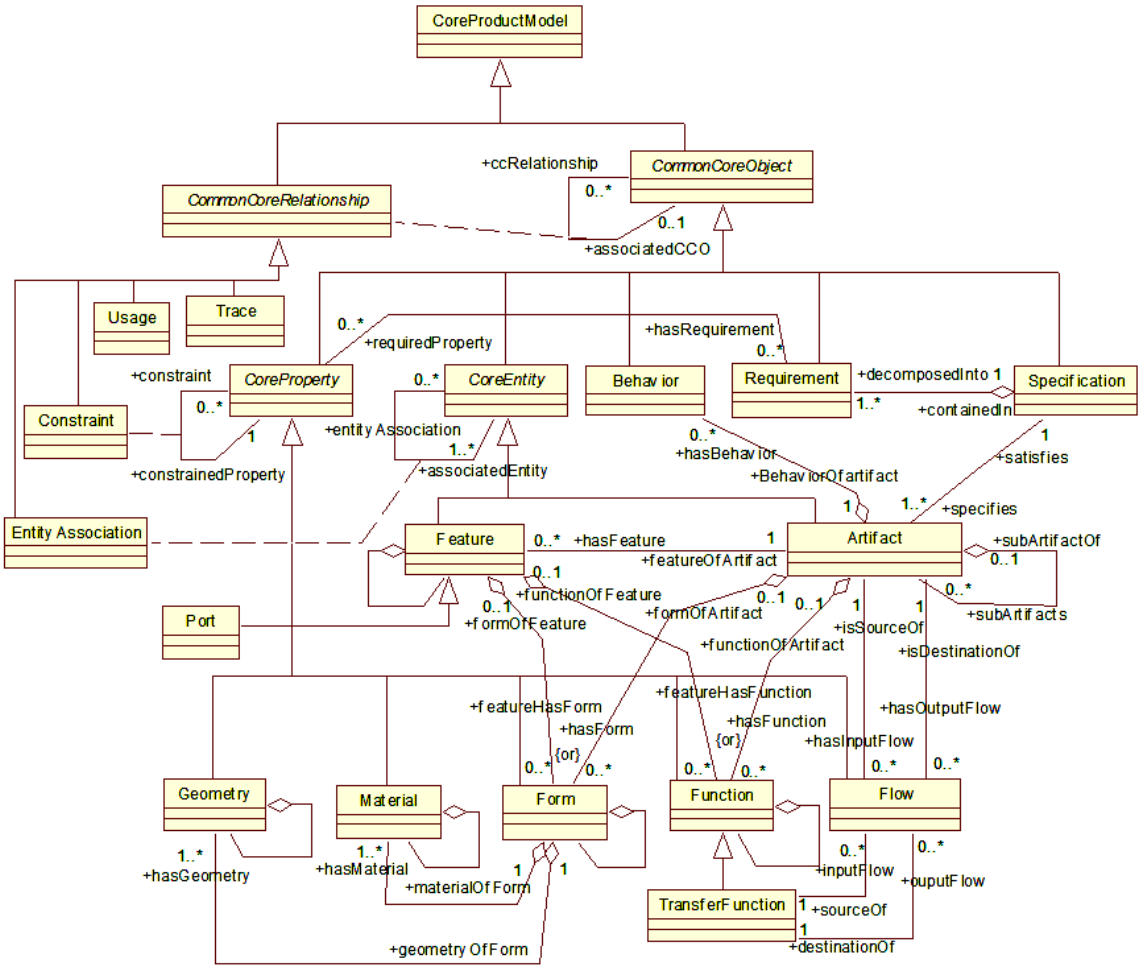

Figure 1: Class diagram of the Core Product Model

A **Port**, a specialization of **Feature**, is a specific kind of feature (sometimes referred to as an interface feature) through which the artifact is connected to (or interfaces with) other artifacts. The semantics of the term port are deliberately left vague; in some contexts, ports only denote signal, control or display connection points, while in other contexts ports are equivalents of assembly features through which components mate.

A **Specification** represents the collection of information relevant to an **Artifact** deriving from customer needs and/or engineering requirements. The **Specification** is a container for the specific requirements that the function, form, geometry and material of the artifact must satisfy.

A **Requirement** is a specific element of the specification of an artifact that governs some aspect of its function, form, geometry or material. Conceptually, requirements should only affect the function, i.e., the intended behavior; in practice, some requirements tend

to affect the design solution directly, i.e., the form, geometry or material of the artifact. Requirements cannot apply to behavior, which is strictly determined by the behavioral model.

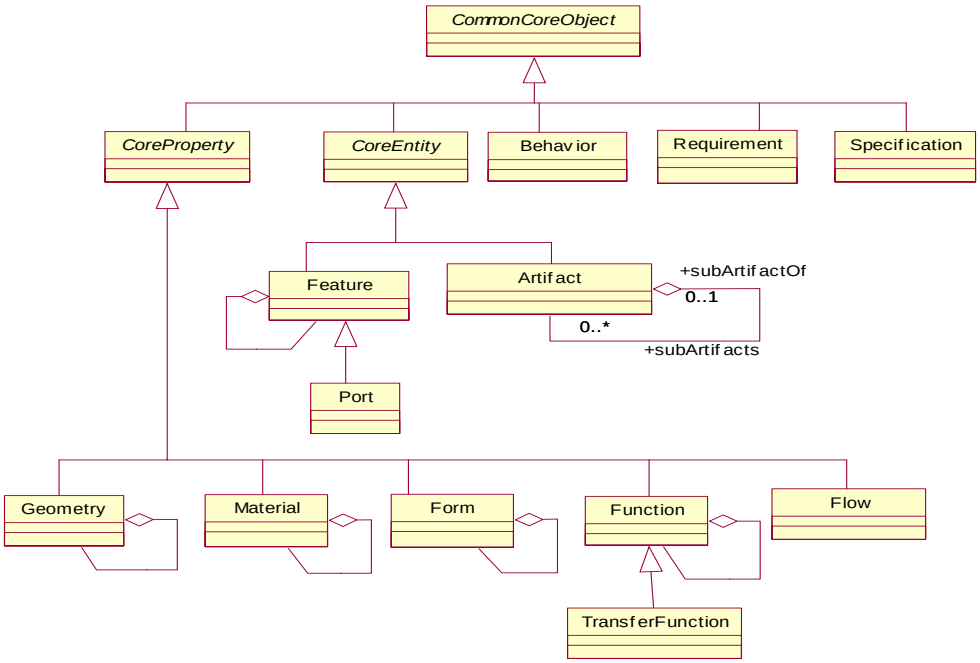

Figure 2: CPM object classes

A **Function** represents one aspect of what the artifact is supposed to do. The artifact satisfies the engineering requirements largely through its function. The term function is often used synonymously with the term *intended behavior*.

A **TransferFunction** is a specialized form of **Function** involving the transfer of an input flow into an output flow. Examples of transfer functions are "transmit" a flow of fluid or current, a message, etc., or "convert" from one energy flow to another or from a message to an action.

A **Flow** is the medium (fluid, energy, message stream, etc.) that serves as the output of one or more transfer function(s) and the input of one or more other transfer function(s). A flow is also identified by its source and destination artifacts.

The **Behavior** describes how the artifact implements its function. **Behavior** is governed by engineering principles that are incorporated into a behavioral or causal model. Application of the *behavioral model* to the artifact describes or simulates the artifact's *observed behavior* based on its form. The observed behavior can then be examined with respect to the requirements to yield the *evaluated behavior*. Consequently, behavior has three specialized attributes or slots to hold the **`behavioralModel`**, the **`observedBehavior`** and the **`evaluatedBehavior`** (typically, URLs to the executable analysis program that embodies the behavioral model, the output of the behavioral model and the output of an external evaluation, respectively).

The **Form** of the artifact can be viewed as the proposed design solution for the design problem specified by the function. In the CPM, the artifact's physical characteristics are represented in terms of its geometry and material properties. This subdivision was introduced because some of the intended applications tend to treat these two aspects quite differently (e. g., the product development process may have a separate task of material selection for a given function and geometry).

Geometry is the spatial description of the artifact. **Material** is the description of the internal composition of the artifact.

The third set of CMP classes represents relationship classes, which are derived from the **CommonCoreRelationship** class. A **Constraint** is a specific shared property of a set of entities that must hold in all cases. At the level of the CPM, only the entity instances that constitute the constrained set are identified. If it is intended to represent a mathematical equality or inequality constraint, the properties slot of the **Information** element associated with the constraint can store the names of the attributes that enter in the constraint as well as the relational operator linking them. The **EntityAssociation** is a simple set membership relationship among artifacts, features and ports. In applications of the CPM, this relationship can be specialized; for example, in the OAM, **EntityAssociation** is specialized to **ArtifactAssociation**. **Usage** is a mapping from **CommonCoreObject** to **CommonCoreObject**. The relationship is particularly useful when constraints apply to the specific "target" entity but not to the generic "source" entity, or when the source entity resides in an external catalog or design repository. **Trace** is structurally identical to **Usage**. The relationship is particularly useful when the "target" entity in the current product description depends in some way on a "source" entity in another product description.

Associations and aggregations are other important and fundamental components of the CPM.

First, all *object* classes, i.e., specializations of the abstract class **CommonCoreObject**, except **Flow**, have their own separate, independent decomposition hierarchies, also known as "partOf" relationships or containment hierarchies[1]. Decomposition hierarchies are represented by attributes such as **subArtifacts/subArtifactOf** for the **Artifact** class.

Second, there are associations between:

(a) a **Specification** and the **Artifact** that results from it;
(b) a **Flow** and its source and destination **Artifact**s and its input and output **Function**s; and
(c) an **Artifact** and its **Feature**s.

Third, and most importantly, four aggregations are fundamental to the CPM:

(a) **Function**, **Form** and **Behavior** aggregate into **Artifact**;
(b) **Function** and **Form** aggregate into **Feature**;
(c) **Geometry** and **Material** aggregate into **Form**; and
(d) **Requirement** aggregates into **Specification**.

[1] For clarity, only the subArtifacts/subArtifactof containment hierarchy of Artifact is labeled in Figures 1 and 3.

The last group of CPM classes is the Utility package. It contains three classes.

The class **Information** is a container consisting of: (i) a brief textual description slot; (ii) a textual documentation string (e. g., a file path or URL referencing more substantial documentation); and (iii) a properties slot that contains a set of attribute-value pairs stored as strings representing all domain- or object-specific attributes. **CoreProductModel** and all its specializations have the attribute **`information`**.

The class **ProcessInformation** represents attributes related to the product development process, such as state and level, as used in [5], alternative and/or version designation or other product development process parameters that may be used in a PLM environment. **`processInformation`** is an attribute of **Artifact**.

The class **Rationale** represents attributes that record explanatory information on the reasons for or justifications of a particular decision in the product development process. **`rationale`** is an attribute of **CoreProperty** and all its specializations.

Extensions and implementations of CPM may explicitly assign attributes to specializations of CPM objects and relationships. This has been done, for example, in the OAM discussed in the next section, to provide interoperability with legacy data models, such as STEP, or existing CAD programs.

A detailed description of CPM, including the semantics of all entities and relationships, a listing of all object attributes and Java and XML implementations, is given in [2].

3 Overview of 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 [6]. 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 STEP.

3.1 OAM Review

Figure 3 shows the main schema of the Open Assembly Model. The schema incorporates information about assembly relationships and component composition; the representation of the former is by the class **AssemblyAssociation**, and the model of the latter uses 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 one fixes an artifact in space for anchoring the entire assembly with respect to the ground. It can also occur when we capture kinematic information between an artifact at an input point and the ground. We can regard such cases 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 a composition of its 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.

ArtifactAssociation 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 where the participating artifacts physically connect only intermittently. **Connector** realizes **Connection**, which is a specialization of the **Artifact**.

OAMFeature has tolerance information, represented by the class **Tolerance**, and subclasses **AssemblyFeature** and **CompositeFeature**. **CompositeFeature** represents a composite feature that is decomposable into multiple simple features. **AssemblyFeature,** a sub-class of **OAMFeature,** by definition represents assembly features. Assembly features are a collection of geometric entities of artifacts. They may be partial shape elements of any artifact. For example, consider a shaft-bearing connection. The 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 tori as assembly features.

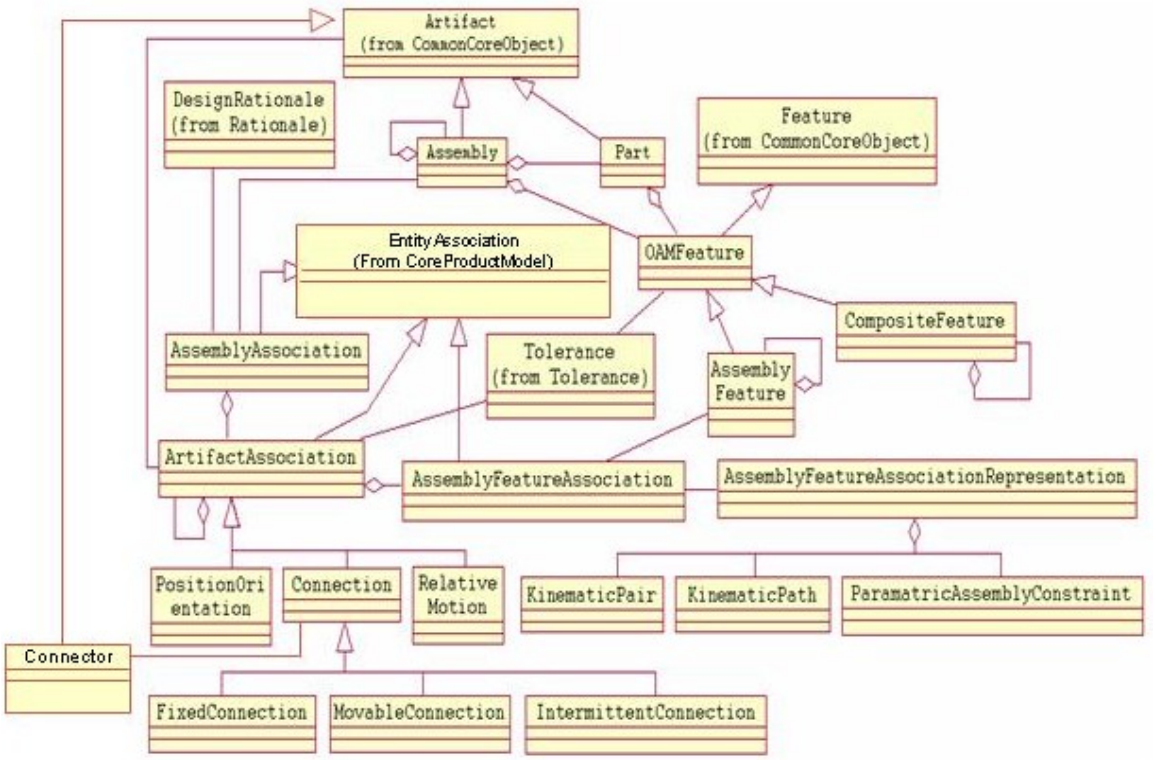

Figure 3: Class diagram of the Open Assembly Model

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.

ParametricAssemblyConstraint 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 [7]. 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 [8]. 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.

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; we can achieve significant gains 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 early 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 4.

DimensionalTolerance typically controls the variability of linear dimensions that describe location, size, and angle; it is also known as tolerancing of perfect form. This concept is included to accommodate the requirements of ISO 1101 standard [9]. **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**.

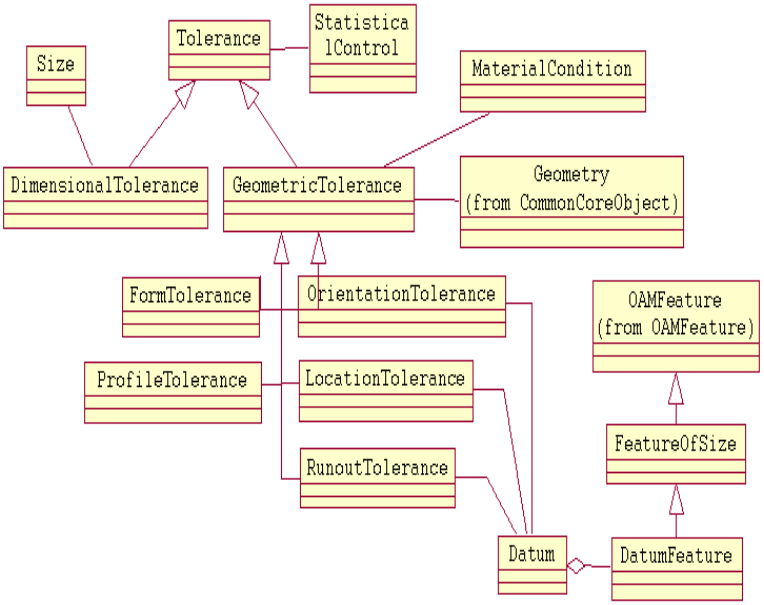

Figure 4: Tolerance model

Datum is a theoretically exact or a simulated piece of geometry, such as a point, line, or plane, which serves as a reference to a tolerance. **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.

A detailed description of OAM, including a case study example is given in [3].

3.2 Usage Patterns for Reusable Subassemblies and Parts

This section describes an update to the OAM to support reusable subassemblies [3]. The current model requires redundant specification of subassemblies, parts, and artifact associations used more than once. For example, if a wheel subassembly in a car is comprised of a tire and a hub, this subassembly must be repeated at least two times in the current model, so that each wheel subassembly could be attached to the correct axle, as shown on the left in Figure 5. Otherwise, there is no way to distinguish the back wheels from the front.[2]

The current model provides no central element to record data about wheels, axles, and their connections. This leads to a number of problems:

- Consistency maintenance is difficult when a reused subassembly is changed, and requires propagation to all its usages. For example, there is no central class to make a change to the parts of wheels, and then propagate to all assemblies using wheels.

[2] Two additional copies are needed to distinguish left wheels from right.

- Finding all the usages of a part or subassembly is unreliable. For example, the only information available about location of wheels usage is in the names "FrontWheel" and "BackWheel". The naming conventions may change over time, and may be different over countries in the model's implementation.

The same problem applies to the artifact associations. For example, the association between wheels and axles may be the same for front and back axles, but this is not apparent from the model, because there is no record of the generic association centrally.

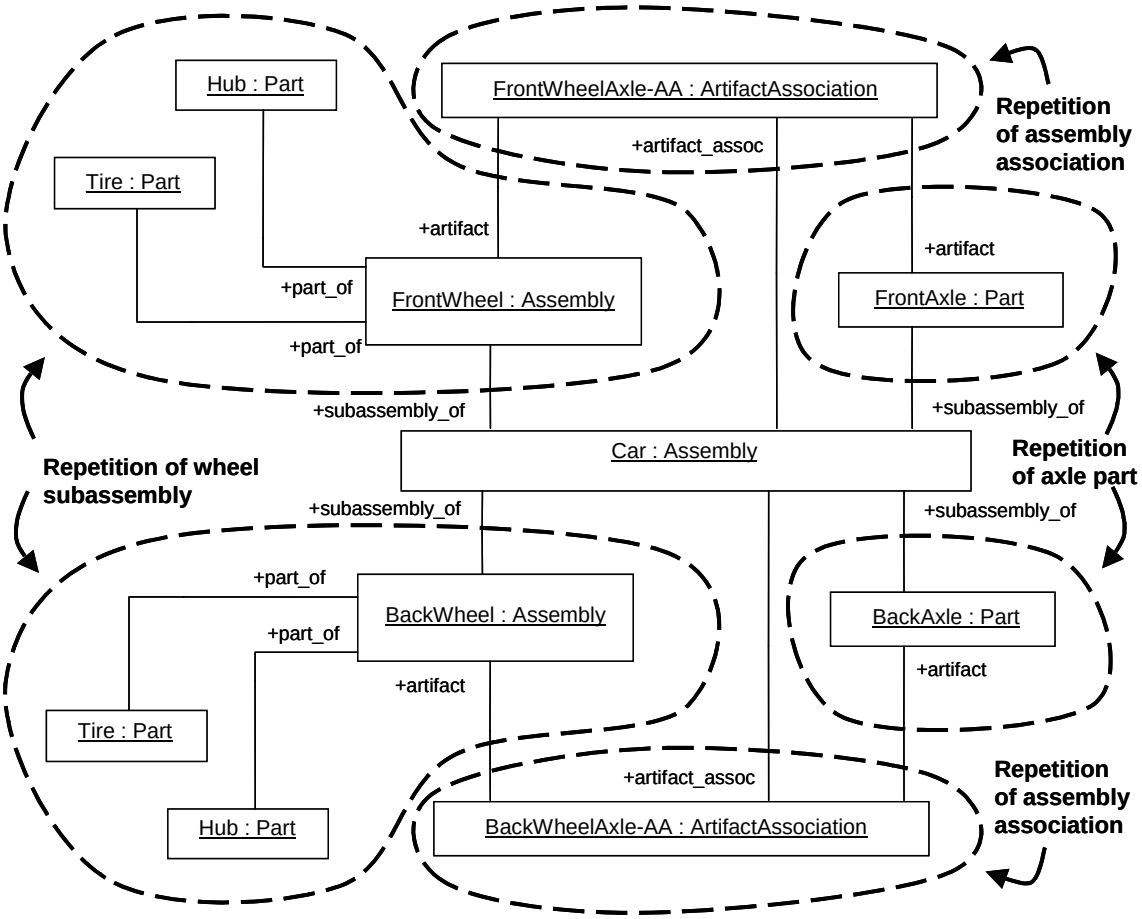

Figure 5: Repeated subassemblies, parts, and artifact associations in OAM

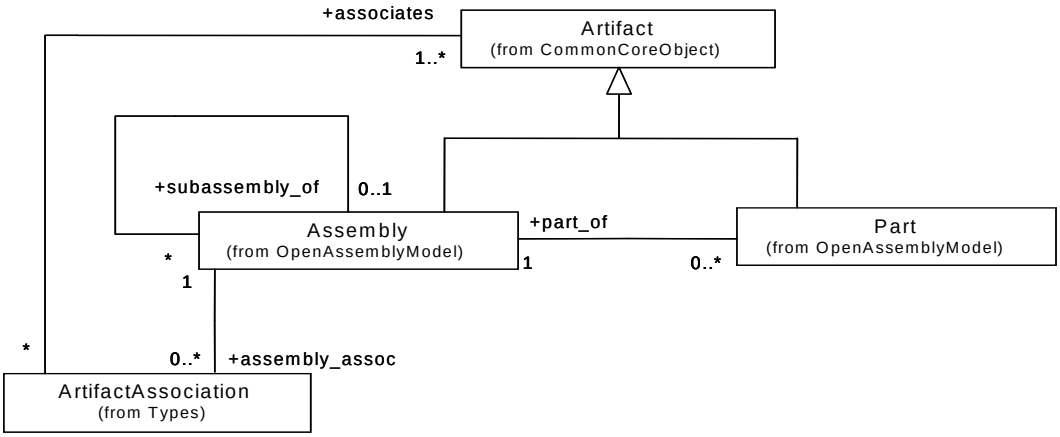

Figure 6: OAM part hierarchy and ArtifactAssociation

The redundancy and resulting maintenance and support problems are due to the direct association between subassemblies, parts, and artifact associations in a larger assembly as shown in
Figure **6**, rather than between usages or roles of these.

The **ArtifactAssociation** connects subassemblies and parts together by referring directly to **Artifact**, which may be an assembly. There is no support for a central element to describe an artifact independently of its connections to other artifacts in an assembly. To support this we introduce the concept of usage in OAM. The general pattern is that for any element of an assembly to define a corresponding usage element that refers to the original element and the context of its use, as shown in Figure 7. Usage applies to a number of OAM elements, as shown in Table 1. Figure 8 is a model of the car example from Figure 5 using some of these concepts.[3] The wheel subassembly once represented allows for reuse at the ends of the axles, as is the axle and wheel-axle association.[4]

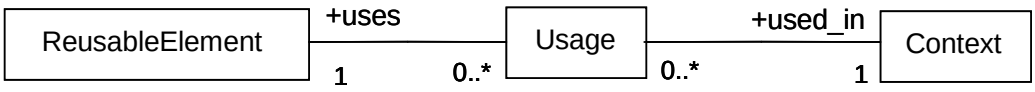

Figure 7: Usage pattern

ReusableElement	**Usage**	**Context**
Artifact	ArtifactUsage	Assembly
AssemblyFeature	AssemblyFeatureUsage	ArtifactUsage
ArtifactAssociation	ArtifactAssociationUsage	Assembly
AssemblyFeatureAssociation	AssemblyFeatureAssociationUsage	ArtifactUsage

[3] A more complete example can show feature usages, which have artifact usages as context.
[4] The classes in Figure 8 could subtype from those in Figure 7 as indicated by the annotations in Figure 8, but it requires association specialization, which is beyond the scope of this paper.

Table 1: Usage pattern in OAM

The model is shown in Figure 9, Figure 10 and Figure 11. Figure 9 models the part-whole hierarchy of artifacts and assemblies, slightly modified from the current OAM. It uses a modified Composite pattern [10].

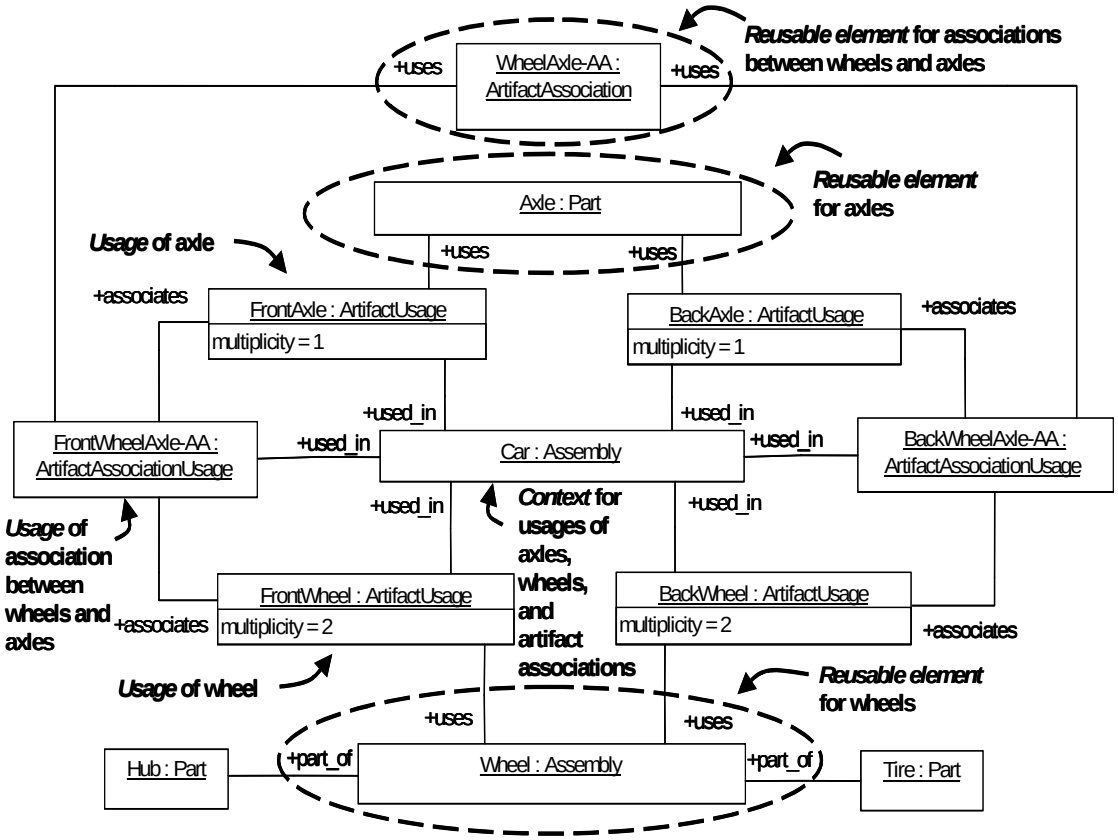

Figure 8: Example of ReusableElements, Usages, and Contexts in OAM

The **Contains** relation is derived as the union of the subassemblies and parts of an assembly, which are subsets of the contents.[5] The multiplicity from **Part** to **Assembly** is unlimited because the assumption that these elements represent the generic reusable forms, rather than usages, allows the reuse of the same kind of part in many assemblies.

Figure 10 describes the Usage model. **Assembly** contains two kinds of usage, one for using artifacts, **ArtifactUsage,** and one using associations that connect artifacts in the assembly, **ArtifactAssociationUsage.** The associations between artifacts reference usages, not artifacts directly, so two uses of the same kind of artifact are distinguished in the assembly, enabling each to have different associations, as wheels does in Figure 8. The multiplicity of a usage indicates the number of represented instances of the used

[5] In the UML's Object Constraint Language (OCL), this derivation is expressed as:
`context` Assembly `inv`
`contains = subassembly->union(part)`
Note: All OCL expressions will appear in special font, for example `context.`

artifact. In the wheel example, there are two wheels for each axle. Figure 10 has a second layer of usages for features and the connections between features[6]. This is for a more detailed design stage where the identification of associations between individual features takes place. For example, the connection between the axles and wheel will use a feature of the hub, like a hole, and some feature of the axles, like a shaft and flange.[7]

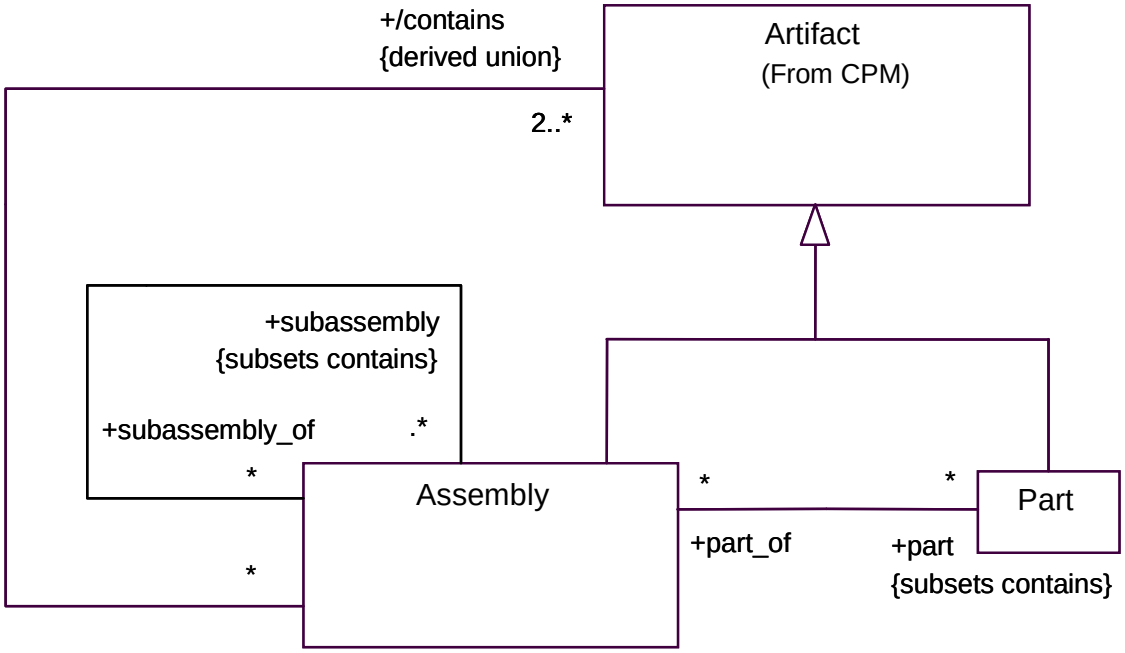

Figure 9: Part hierarchy in OAM

[6] The black diamond notation indicates that deleting an element at the diamond end will delete the element at the other end, which is called *strong composition*. For example, in Figure 10 deletion of an assembly will lead to deletion of its usages. This is an effect on the repository or database storing the design, as instances of the metamodel, not on real-world subassemblies or parts. Strong composition is not used in

Figure 9 between and assembly and its subassemblies and parts, because these are reusable elements in the design repository. If an assembly is deleted, and no longer uses a subassembly or part, the subassembly or part remains in the repository to be reused.

[7] The relations for

Figure 9 are derivable from those in Figure 10. For example, parts and subassemblies of an assembly can be derived from usages of artifacts that are assemblies and parts. In OCL this is expressed as:

```
context Assembly inv
composed_of = artifactUsage->collect(usage_of)
```

Normally in UML, derived associations are marked specially to indicate that they should not be stored, only calculated as needed. The OAM does not treat relations as derived if they are useful at different points in the design and production cycle. For example, the PART and SUBASSEMBLY relations are useful early in the design cycle to sketch the hierarchy of subassembly, as in Figure 25 of [3]. It is also useful later in the cycle as a bill of materials. For the same reasons, artifact associations are not marked as derived, even they could be derived from feature associations. In OCL this is:

```
context ArtifactAssociationUsage inv
associates = featureAssociationUsage->collect(associates)->collect(used_in)
```

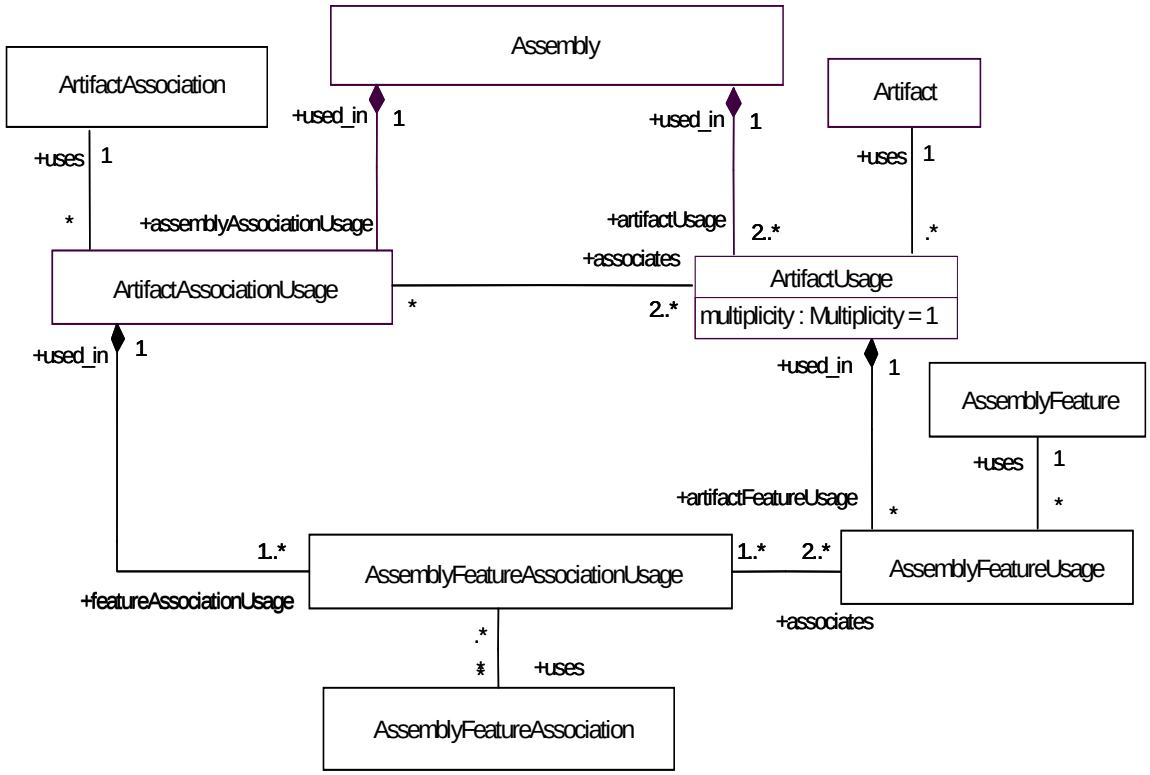

Figure 10: Usage metamodel in OAM

Artifact associations and feature associations come in various kinds, as shown in Figure 11. These metamodels simplify the OAM by collapsing OAM's **AssemblyFeatureAssociationRepresentation** into **AssemblyFeatureAssociation**. These two are one-to-one in the current OAM. OAM constrains the relation between **AssemblyFeatureAssociationUsage** and **AssemblyFeatureAssociation** to be at most one of **KinematicPair** and at most one of **KinematicPath**.[8] The OAM specifies additional specializations of these concepts, see [3].

[8] In OCL this is:

```
context AssemblyFeatureAssociationUsage inv
uses->select(isKindOf(KinematicPair))->size() < 2
and uses->select(isKindOf(KinematicPath))->size() < 2
```

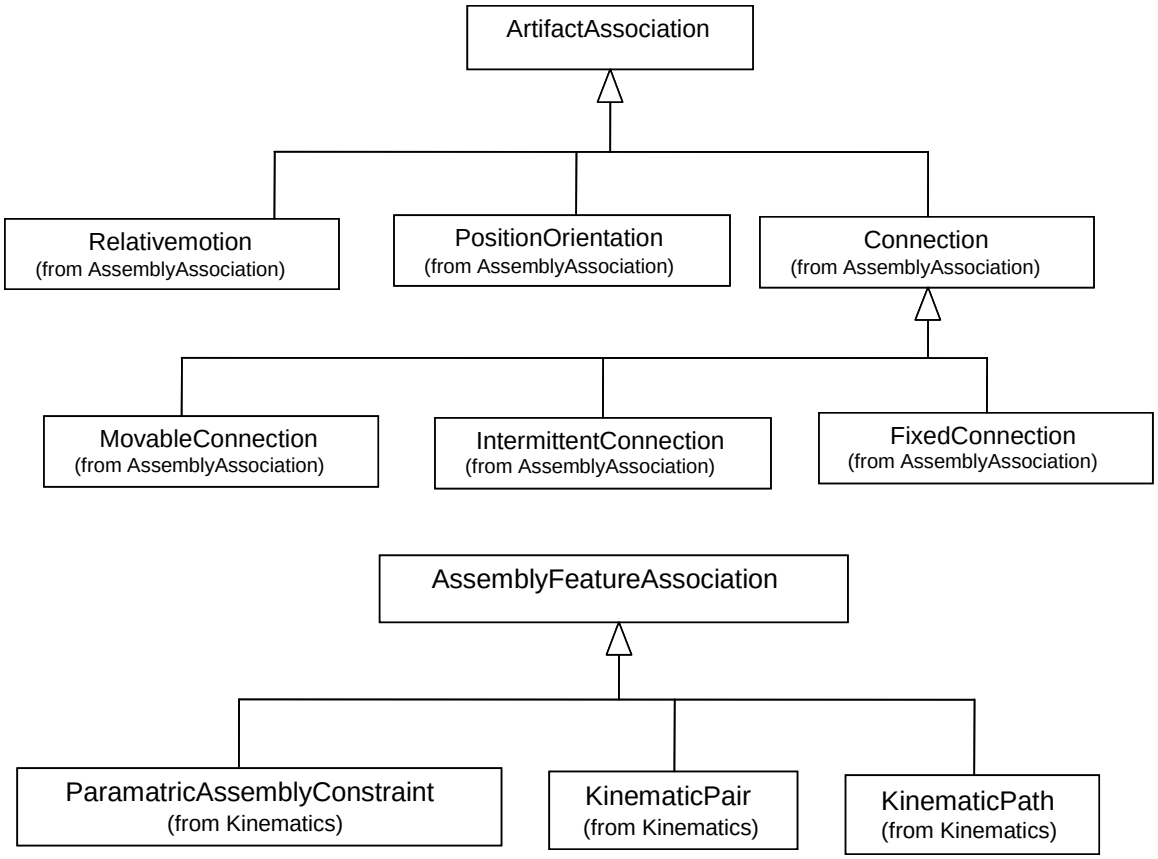

Figure 11: ArtifactAssociation and FeatureAssociation classes in OAM

4 Applications

In this section, we present two examples of the applications of the CPM and OAM models.

The first example illustrates the use of CPM and OAM during different design phases and shows how the relevant design information is captured and updated throughout the design process. The example uses the Planetary Gear System (PGS) initially described in [5] and presented in detail in [3]; it is adapted from [5] with slight modifications.

The second example illustrates the application of the usage patterns discussed in Section 4.2.

4.1 Design of a Planetary Gear

The design information generated and used by the various designers and tools throughout the entire PLM process needs to be stored and retrieved in a reliable way. In particular, there is the need to define and capture the evolutionary nature of the product information as that information progresses from abstract, conceptual representations to the most detailed ones. Obviously, the representation of the evolving artifact throughout the design process requires large volumes of domain-specific data that are not included in the CPM

or OAM. Our working hypothesis is that CPM and OAM can provide the organizing principles for all the phase-, domain- and application-specific data. In the presentation that follows, we keep information "external" to CPM and OAM to a minimum and concentrate on the role of CPM and OAM in the organization of the design information. A full description of the information generated, as well as the description of the supporting tools and interfaces, may be found in [11].

As an example we follow the design information flow in the design of the planetary gear described in [3] and show how the OAM-organized information is developed and populated throughout the design process.

Information captured throughout this example design session is represented at multiple levels, as discussed in [5]. Superscripts (1, 2, …) that follow some entries in the information shown denote information captured and updated in different design phases. The assignment of two superscripts to the same attribute implies that the attribute values were created or modified in those two design phases. The superscripts designate the design phases as follows:

1. Conceptual design,
2. Preliminary design,
3. Solid Modeling of parts and assemblies,
4. Detailed Design: Tolerance assignment

4.1.1 Conceptual design

The initial phase of *customer need elicitation* provides user requirements for an **Artifact** called "portable drill" such as:

- Provide sufficient torque to drill holes and drive screws
- Provide forward and reverse rotation
- Adequate and variable speed for all different operations
- Comfortable operation
- Small size and weight
- Affordable cost.

These **Requirements** are grouped in the **Artifact**'s **Specification**.

An early stage of *conceptual design* decomposes the **Artifact** "portable drill" into its major **subArtifacts**, among which is the **Artifact** "speed reducer" which has the **Function** of converting the input energy from the motor into the output energy for the chuck. The next stage of conceptual design chooses a planetary gear for instantiating the "speed reducer," primarily in response to the **Requirement** of small size.

From this point on, we follow only the design of the planetary gear. The *functional design* of the planetary gear generates the following further **Requirement**s:

- Function Requirements:
  - Input kinetic (rotational) energy.

        - inputSpeed = 900 rpm (low) and 1800 rpm (high).
        - inputTorque = 2.26 N•m (maximum value).
    - Output kinetic (rotational) energy.
        - outputSpeed = 300 rpm (low) and 600 rpm (high).
        - outputTorque = 6.78 N•m (maximum value).
- Form Requirements:
    - Concentric shafts
    - Length to allow mating with output shaft of motor and input shaft of chuck.
    - Size restrictions
    - length < 100 mm
    - width < 60 mm
    - height < 60 mm.

Using the terminology of [5], the requirements specify some functional parameters as bound parameters, i. e., inputSpeed = 1800 rpm, inputTorque = 2.26N-m, etc., while others remain unbound parameters at this stage, i. e., outputSpeed, outputTorque, speedRatio, formSize and formWeight. At this stage, three instance entities describe the planetary gear, as shown in Figure 12[9].

art:Artifact	**fun:Function**	**form:Form**
name: planetary_reduction_gear type: gearbox	name: gear_function type: energy_transfer subFunctions: heat_generation properties: {Bound_parameters: {Speed_in=1800rpm, Torque_in=2.26N-m} Unbound_parameters: {OutputSpeed, OutputTorque, SpeedRatio, Form}}	name: gear_form type: symbolic_description properties: {Bound_parameters: {LT<100mm DT<60mm} Unbound_parameters: {D1, D3, D4, L1, L2, L3, L4, L5}}

Figure 12: Conceptual representation of planetary gear in CPM/OAM

[9] The properties of CPM/OAM entities are shown as attributes of the entities rather than as slots in the **Information** attribute

A subfunction, heat_generation of the artifact's function is defined so that subsequently the thermal performance of the gear can be evaluated and, if necessary, constrained. At this stage, there is no **Form** instance.

4.1.2 Preliminary design

4.1.2.1 Decomposition and preliminary Analysis

Based on the above conceptual product description, a constrained optimization-based design synthesis [12], [13], [14] (in consultation with an existing design repository/library) is carried out to select an appropriate planetary gear system to attain the desired functional objective. The planetary gearbox solution establishes an abstract form as shown in Figure 13. This abstract form includes the various components: sun, planet and ring gears and their connection arms. An instantiation of a planetary gear train is made, and the resulting subartifacts are shown in Table 4.

Id	Name	Multiplicity
Art1	PlanetGearPin1	3
Art4	PlanetGear1	3
Art7	OutputShaft	1
Art8	SunGear	1
Art9	RingGear	1

Table 2: List of sub-artifacts of planetary_reduction_gear

The artifact now needs to go through an analysis to describe its observed behavior. The *behavior model* is provided by the general equation for a planetary gear train:

$$\frac{\omega_{in}}{\omega_{out}} = 1 + \frac{N_{ring}}{N_{sun}} = 1 + \frac{D_{ring}}{D_{sun}}$$

where ω represents the rotational speed, N the number of teeth and D the pitch diameter of the gears.

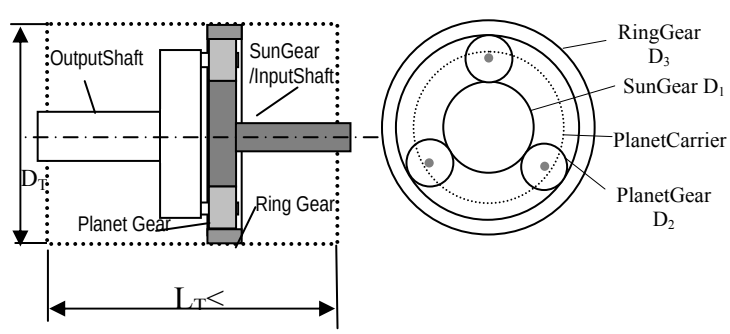

Figure 13: Basic planetary gear system

The preliminary representation of the gear's behavior is described by the entity shown in Figure 14.

beh:Behavior

name:
gear_behavior
type:
calculated_by_formula
subFunctions:
Heat_generation
properties:
{Bound_parameters:
{SpeedRatio = 3.0:1,
OutputTorque = 6.78 N•m,
Shaftsdirection=Collinear }

Figure 14: Preliminary representation of the planetary gear's behavior in CPM/OAM

4.1.2.2 Kinematics and Assembly Details

In order to describe the kinematic behavior of the artifact, coordinate systems are assigned to the **KinematicPair**s in the OAM (see Figure 15). The **KinematicPair** association defines the kinematic constraints between two associated artifacts at a joint as shown in Table 7. Specific associations define the **KinematicPair** information according to the **KinematicPair** type, namely, **RevolutePair** (Table 4) and **GearPair (**Table 5). The **GearPair association also** shows attributes of the gears (radiusfirstlink, radiussecondlink, gearratio, bevel and helicalangle).

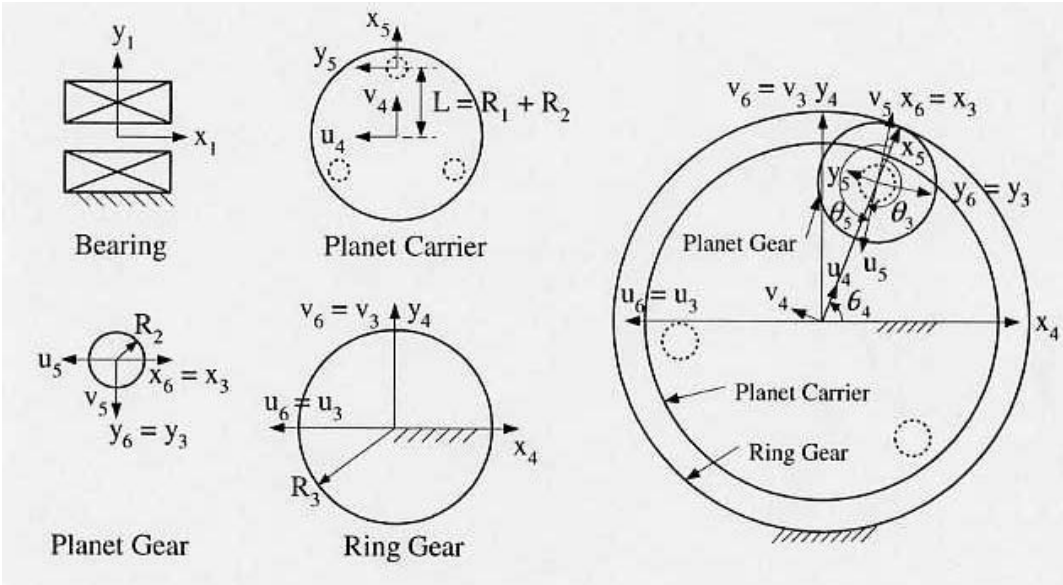

Figure 15: Coordinate systems assigned to KinematicPairs

Id	Name/type	Description	TransformItem1	TransformItem2	Frames	Pair Variables
RP1	RevoluteP[2]	RevolutePair[2]	UnknownSupport	SunGear [2]	{x1 y1 z1 }[2] - {u1 v1 w1 }[2,3]	$\theta 1^{2,3}$
RP2	RevoluteP[2]	RevolutePair[2]	PlanetGear1 [2]	PlanetGearPin1 [2]	{x5 y5 z5 }[2] - {u5 v5 w5 }[2,3]	$\theta 5^{2,3}$
RP3	RevoluteP[2]	RevolutePair[2]	PlanetGear2 [2]	PlanetGearPin2 [2]	-	-
RP4	RevoluteP[2]	RevolutePair[2]	PlanetGear3 [2]	PlanetGearPin3 [2]	-	-
RP5	RevoluteP[2]	RevolutePair[2]	PlanetCarrierSub assembly	Bearing	{x4 y4 z4 }[2] - {u4 v4 w4 }[2,3]	$\theta 4^{2,3}$
GP1	GearPair[2]	GearPair[2]	SunGear [2]	PlanetGear1 [2]	{x2 y2 z2 }[2] - {u2 v2 w2 }[2,3]	$\theta 2^{2,3}$
GP2	GearPair[2]	GearPair[2]	SunGear [2]	PlanetGear2 [2]	-	-
GP3	GearPair[2]	GearPair[2]	SunGear [2]	PlanetGear3 [2]	-	-
GP4	GearPair[2]	GearPair[2]	PlanetGear1 [2]	RingGear [2]	{x3 y3 z3 }[2] - {u3 v3 w3 }[2,3]	$\theta 3^{2,3}$
GP5	GearPair[2]	GearPair[2]	PlanetGear2 [2]	RingGear [2]	-	-
GP6	GearPair[2]	GearPair[2]	PlanetGear3 [2]	RingGear [2]	-	-

Table 3: KinematicPairs[2]

Id	Name	TransformItem1	TransformItem2	PairValue	Frame1	Frame2
RP1	RP1[2]	UnknownSupport [2]	SunGear [2]	Rotation angle = $\theta 1^{2,3}$	{x1 y1 z1 }[2]	{u1 v1 w1 }[2,3]
RP2	RP2[2]	PlanetGear1 [2]	PlanetGearPin1 [2]	Rotation angle = $\theta 2^{2,3}$	{x1 y1 z1 } [2]	{u1 v1 w1 }[2,3]
RP3	RP3[2]	PlanetGear2 [2]	PlanetGearPin2	Rotation angle =	{x1 y1 z1 } [2]	{u1 v1 w1

			[2]	θ3 [2,3]		}[2,3]
RP4	RP4[2]	PlanetGear3 [2]	PlanetGearPin3 [2]	Rotation angle = θ4 [2,3]	{x1 y1 z1 } [2]	{u1 v1 w1 }[2,3]
RP5	RP5[2]	PlanetCarrierSubassembly [2]	Bearing [2]	Rotation angle = θ5 [2,3]	{x1 y1 z1 } [2]	{u1 v1 w1 }[2,3]

Table 4: RevolutePairs[2]

Id	TransformItem1	TransformItem2	Radius First Link	Radius Second Link	Gear Ratio	Bevel (plane angle measure)	HelicalAngle (plane angle measure)
GP1 [2]	Art8 [2]	Art4 [2]	11 [2]	4.5 [2]	2.44 [2]	0 [D]	0 [D]
GP2 [2]	Art8 [2]	Art5 [2]	11 [2]	4.5 [2]	2.44 [2]	0 [D]	0 [D]
GP3 [2]	Art8 [2]	Art6 [2]	11 [2]	4.5 [2]	2.44 [2]	0 [D]	0 [D]
GP4 [2]	Art4 [2]	Art9 [2]	4.5 [2]	20 [2]	4.44 [2]	0 [D]	0 [D]
GP5 [2]	Art5 [2]	Art9 [2]	4.5 [2]	20 [2]	4.44 [2]	0 [D]	0 [D]
GP6 [2]	Art6 [2]	Art9 [2]	4.5 [2]	20 [2]	4.44 [2]	0 [D]	0 [D]

Table 5: GearPairs[2]

At this point, an application would also design the planetary gears' teeth attributes and face widths. After these calculations and material selection, the values of the function-related attributes (pitchdia, facewidth, shaftdia, nteeth, sbmax, scmax, stmax, E, G, rho) would be mapped into the artifact's Function table (Figure 12).

The complete OAM representation of the planetary gear assembly consists of the following:

- List of artifacts (parts/subassemblies) (Table 6)
- List of **OAMFeature**s used in the assembly.
- List of artifact associations (Table 7)
- List of **Connections** with assembly features (Table 8),
- List of assemblies (Table 9)
- List of Assembly Associations (Table 10)
- List of Assembly Feature Associations (AFA) (Table 11)
- List of Assembly constraints (Table 12)
- For each AFA define AFAR (Relationship) subclasses [15] with (explicit geometric constraints between artifacts) **ParametricAssemblyConstraint** **KinematicPair** (according to mate groups in AFA) **KinematicPath**

Id	Name	Defin.	Group/ Type	Form	Requires
Art1	PlanetGearPin1 [2]	Part [2]	Locator[2]	Form1	Art7
Art2	PlanetGearPin2 [2]	Part[2]	Locator[2]	Form1	Art7
Art3	PlanetGearPin3 [2]	Part[2]	Locator[2]	Form1	Art7
Art4	PlanetGear1 [2]	Part[2]	PT[2]	Form2	Art1, Art7
Art5	PlanetGear2 [2]	Part[2]	PT[2]	Form2	Art2, Art7
Art6	PlanetGear3 [2]	Part[2]	PT[2]	Form2	Art3, Art7
Art7	OutputShaft [2]	Part[2]	PT[2]	Form3	UnDef
Art8	SunGear [2]	Part[2]	PT[2]	Form4	UnDef
Art9	RingGear [2]	Part[2]	PT[2]	Form5	UnDef
Art10	PlanetCarrierSubassembly [2]	SA[2]	PT, Locator[2]	Form7	Art1, Art2 Art3, Art7
Art11	PlanetGearCarrierSubassembly [2]	SA[2]	PT[2]	Form8	Art4, Art5, Art6, Art7, Art10
Art12	PlanetGearSubassembly [2]	SA[2]	PT[2]	Form9	Art8, Art9, Art11
Art13	PlanetaryGearBoxSubassembly. [2]	SA[2]	PT[2]	Form10 [1]	Art12, UnDef

Table 6: Artifacts[1]

Id	Artifacts	Assembly Constraints	Connection/ Position Orientation/ RelativeMotion -- Type	Type
AA1	Art7, Art1 [2]	AC1, AC2 [3]	FC7 [2]	Conn
AA2	Art7, Art2 [2]	AC3, AC4 [3]	FC8 [2]	Conn
AA3	Art7, Art3 [2]	AC5, AC6 [3]	FC9 [2]	Conn
AA4	Art1, Art4 [2]	AC7, AC8 [3]	MC2 [2]	Conn
AA5	Art2, Art5 [2]	AC9, AC10 [3]	MC3 [2]	Conn
AA6	Art3, Art6 [2]	AC11, AC12 [3]	MC4 [2]	Conn
AA7	Art7, Art8 [2]	AC13 [3]	PO1 [2]	PO
AA8	Art7, Art9 [2]	AC14 [3]	PO2 [2]	PO

Table 7: Artifact associations[2]

Id	Type	**Assembly Constraint**	**Artifacts**	AssemblyFeatures	KinematicPair
FC7 [2]	Fixed [2]	AC1, AC2 [3]	Art7, Art1 [2]	PinHole3, PinCylinder3 [2,3]	Null
FC8 [2]	Fixed [2]	AC3, AC4 [3]	Art7, Art2 [2]	PinHole4, PinCylinder4 [2,3]	Null
FC9 [2]	Fixed [2]	AC5, AC6 [3]	Art7, Art3 [2]	PinHole5, PinCylinder5 [2,3]	Null
MC2 [2]	Movable	AC7, AC8 [3]	Art1, Art4 [2]	PinCylinder6,	RP2 [2]

	[2]			GearJournalSurface1 [2,3]	
MC3 [2]	Movable [2]	AC9, AC10 [3]	Art2, Art5 [2]	PinCylinder7, GearJournalSurface2 [2,3]	RP3 [2]
MC4 [2]	Movable [2]	AC11, AC12 [3]	Art3, Art6 [2]	PinCylinder8, GearJournalSurface3 [2,3]	RP4 [2]
MC5 [2]	Movable [2]	-	Art4, Art8 [2]	Teeth7, Teeth1 [2,4]	GP1 [2]
MC6 [2]	Movable [2]	-	Art5, Art8 [2]	Teeth9, Teeth2 [2,4]	GP2 [2]
MC7 [2]	Movable [2]	-	Art6, Art8 [2]	Teeth11, Teeth3 [2,4]	GP3 [2]
MC9 [2]	Movable [2]	-	Art4, Art9 [2]	Teeth8, Teeth4 [2,4]	GP4 [2]
MC10 [2]	Movable [2]	-	Art5, Art9 [2]	Teeth10, Teeth5 [2,4]	GP5 [2]
MC11 [2]	Movable [2]	-	Art6, Art9 [2]	Teeth12, Teeth6 [2,4]	GP6 [2]

Table 8: Connections[2]

Id	Art Id	SubAssemblyof	Subassemblies	Parts	OAMF	AA	Kinematic Relation
Ass1 [2]	Art10 [2]	Ass2 [2]	-	Art1, Art2, Art3, Art7 [2]	OAMF11, OAMF12, OAMF13 [2,3]	AA1, AA2, AA3 [2]	-
Ass2 [2]	Art11 [2]	Ass3 [2]	Ass1 [2]	Art1, Art2, Art3, Art4, Art5, Art6, Art7 [2]	OAMF5, OAMF6[3]	AA4, AA5, AA6 [2]	RelativeMotion
Ass3 [2]	Art12 [2]	Ass4 [2]	Ass1, Ass2 [2]	Art1, Art2, Art3, Art4, Art5, Art6, Art7, Art8, Art9 [2]	OAMF1, OAMF2, OAMF3, OAMF4, OAMF7, OAMF8, OAMF9, OAMF10 [2,3]	AA7, AA8, AA9, AA10, AA11 [2]	RelativeMotion
Ass4 [2]	Art13 [2]	-	Ass1, Ass2, Ass3 [2]	Art1, Art2, Art3, Art4, Art5, Art6, Art7, Art8, Art9, UnDef [2]	UnDef [3]	UnDef [2]	

Table 9: List of assemblies[2]

Id	Artifacts	**Artifacts**	**Type of Conn**
AA1 [2]	Art7, Art1 [2]	OutputShaft, PlanetGearPin1 [2]	Fixed [2]
AA2 [2]	Art7, Art2 [2]	OutputShaft, PlanetGearPin2 [2]	Fixed [2]
AA3 [2]	Art7, Art3 [2]	OutputShaft, PlanetGearPin3 [2]	Fixed [2]
AA4 [2]	Art1, Art4 [2]	PlanetGearPin1, PlanetGear1 [2]	Movable [2]
AA5 [2]	Art2, Art5 [2]	PlanetGearPin2, PlanetGear2 [2]	Movable [2]
AA6 [2]	Art3, Art6 [2]	PlanetGearPin3, PlanetGear3 [2]	Movable [2]
AA7 [2]	Art7, Art8 [2]	OutputShaft, SunGear [2]	-
AA8 [2]	Art7, Art9 [2]	OutputShaft, RingGear [2]	-
AA9 [2]	Art4, Art8 [2]	PlanetGear1, SunGear [2]	Movable [2]
AA10 [2]	Art5, Art8 [2]	PlanetGear2, SunGear [2]	Movable [2]
AA11 [2]	Art6, Art8 [2]	PlanetGear3, SunGear [2]	Movable [2]

Table 10: Assembly associations[2]

Id	Art1	Art2	AF1	AF2
AFA1 [2]	Art7 [2]	Art1 [2]	PinHole3 [2,3]	PinCylinder3 [2,3]
AFA2 [2]	Art7 [2]	Art2 [2]	PinHole4 [2,3]	PinCylinder4 [2,3]
AFA3 [2]	Art7 [2]	Art3 [2]	PinHole5 [2,3]	PinCylinder5 [2,3]
AFA4 [2]	Art1 [2]	Art4 [2]	PinCylinder6 [2,3]	GearJournalSurface1 [2,3]
AFA5 [2]	Art2 [2]	Art5 [2]	PinCylinder7 [2,3]	GearJournalSurface2 [2,3]
AFA6 [2]	Art3 [2]	Art6 [2]	PinCylinder8 [2,3]	GearJournalSurface3 [2,3]
AFA7 [2]	Art4 [2]	Art8 [2]	Teeth7 [2,4]	Teeth1 [2,4]
AFA8 [2]	Art5 [2]	Art8 [2]	Teeth9 [2,4]	Teeth2 [2,4]
AFA9 [2]	Art6 [2]	Art8 [2]	Teeth11 [2,4]	Teeth3 [2,4]
AFA10 [2]	Art4 [2]	Art9 [2]	Teeth8 [2,4]	Teeth4 [2,4]
AFA11 [2]	Art5 [2]	Art9 [2]	Teeth10 [2,4]	Teeth5 [2,4]
AFA12 [2]	Art6 [2]	Art9 [2]	Teeth12 [2,4]	Teeth6 [2,4]

Table 11: Assembly feature associations[2]

Id	Artifacts	MF (MatingFeatures)	Type	Assembly Name
AC1 [3]	OutputShaft, PlanetGearPin1 [3]	CylindricalPinHoleSurface1, CylindricalPinSurface1 [3]	Coaxial [3]	PlanetCarrier Subassembly [3]
AC2 [3]	OutputShaft, PlanetGearPin1 [3]	FlatPinHoleSurface1, FlatPinSurface1 [3]	CoincId. /Parallel [3]	PlanetCarrier Subassembly [3]
AC3 [3]	OutputShaft, PlanetGearPin2 [3]	CylindricalPinHoleSurface2, CylindricalPinSurface2 [3]	Coaxial [3]	PlanetCarrier Subassembly [3]
AC4 [3]	OutputShaft, PlanetGearPin2 [3]	FlatPinHoleSurface2, FlatPinSurface2 [3]	CoincId. /Parallel [3]	PlanetCarrier Subassembly [3]
AC5 [3]	OutputShaft, PlanetGearPin3 [3]	CylindricalPinHoleSurface3, CylindricalPinSurface3 [3]	Coaxial [3]	PlanetCarrier Subassembly [3]
AC6 [3]	OutputShaft, PlanetGearPin3 [3]	FlatPinHoleSurface3, FlatPinSurface3 [3]	CoincId. /Parallel [3]	PlanetCarrier Subassembly [3]
AC7 [3]	PlanetGearPin1, PlanetGear1 [3]	CylindricalPinHoleSurface1, JournalSurfaceGear1 [3]	Coaxial [3]	PlanetGearCarrier Subassembly [3]
AC8 [3]	PlanetGearPin1, PlanetGear1 [3]	FlatSurfaceOutputShaft, FlatSurfacePlanetGear1 [3]	CoincId. /Parallel [3]	PlanetGearCarrier Subassembly [3]
AC9 [3]	PlanetGearPin2, PlanetGear2 [3]	CylindricalPinHoleSurface2, JournalSurfaceGear2 [3]	Coaxial [3]	PlanetGearCarrier SAssembly [3]
AC10 [3]	PlanetGearPin2, PlanetGear2 [3]	FlatSurfaceOutputShaft, FlatSurfacePlanetGear2 [3]	CoincId. /Parallel [3]	PlanetGearCarrier SAssembly [3]
AC11 [3]	PlanetGearPin3, PlanetGear3 [3]	CylindricalPinHoleSurface3, JournalSurfaceGear3 [3]	Coaxial [3]	PlanetGearCarrierSAssembly [3]
AC12 [3]	PlanetGearPin3, PlanetGear3 [3]	FlatSurfaceOutputShaft, FlatSurfacePlanetGear3 [3]	CoincId. /Parallel [3]	PlanetGearCarrierSAssembly [3]
AC13 [3]	OutputShaft, SunGear [3]	CylindricalSurfaceOutputShaft, JournalSurfaceSunGear [3]	Coaxial [3]	PlanetGear Subassembly [3]
AC14 [3]	OutputShaft, RingGear [3]	CylindricalSurfaceOutputShaft, JournalSurfaceSunGear [3]	Coaxial [3]	PlanetGear Subassembly [3]

Table 12: Assembly constraints[3]

4.1.3 Solid Modeling of parts and assemblies

In creating the example, we used a typical modeling system, Solid Works™ to model all the parts and subassemblies. We assigned the assembly relationships (assembly constraints/mates) and created the entire assembly. By using STEP files (or using appropriate APIs for the 3D CAD modeler), all assembly-related information for each artifact is extracted and mapped into the database tables used to implement the OAM. Since not all features are available in the STEP files, entry of data to the OAM database required a separate user interface.

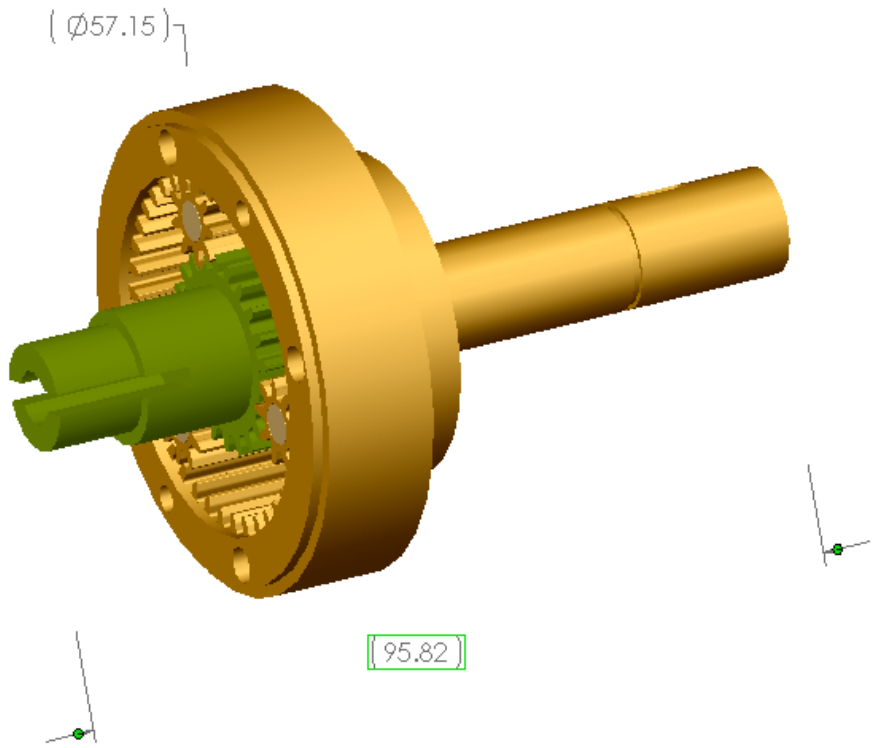

Figure 16: Size constraints of gearbox

Id	Name	Artifact	AFA	Definition / Parameters
AF1 [2]	PinHole3 [2,3]	Art7 [2]	AFA1 [3]	D= D_{ph3} , L= L_{ph3} [2,3]
AF2 [2]	PinHole4 [2,3]	Art7 [2]	AFA2 [3]	D= D_{ph4} , L= L_{ph4} [2,3]
AF3 [2]	PinHole5 [2,3]	Art7 [2]	AFA3 [3]	D= D_{ph5} , L= L_{ph5} [2,3]
AF8 [2]	PinCylinder3 [2,3]	Art1 [2]	AFA1 [3]	L= depth of Pinhole1 [2,3]
AF9 [2]	PinCylinder4 [2,3]	Art2 [2]	AFA2 [3]	L= depth of Pinhole2 [2,3]
AF10 [2]	PinCylinder5 [2,3]	Art3 [2]	AFA3 [3]	L= depth of Pinhole3 [2,3]
AF4 [2]	PinCylinder6 [2,3]	Art1 [2]	AFA4 [3]	L= length of GearJournalSurface1 [2,3]
AF5 [2]	PinCylinder7 [2,3]	Art2 [2]	AFA5 [3]	L= length of GearJournalSurface2 [2,3]
AF6 [2]	PinCylinder8 [2,3]	Art3 [2]	AFA6 [3]	L= length of GearJournalSurface3 [2,3]
AF7 [2]	Cylinder [2,3]	0010 [2]	AFA [3]	D= D_{011} , L= L_{0011} [2,3]
AF11 [2]	GearJournalSurface1 [2,3]	Art4 [2]	AFA4 [3]	D= D_{gc1} , L= L_{gc1} [2,3]
AF12 [2]	GearJournalSurface2 [2,3]	Art5 [2]	AFA5 [3]	D= D_{gc2} , L= L_{gc2} [2,3]
AF13 [2]	GearJournalSurface3 [2,3]	Art6 [2]	AFA6 [3]	D= D_{gc3} , L= L_{gc3} [2,3]
AF14 [2]	Teeth1 [2,4]	Art8 [2]	AFA7 [3]	Teethform1 [2,4]
AF15 [2]	Teeth2 [2,4]	Art8 [2]	AFA8 [3]	Teethform1 [2,4]
AF16 [2]	Teeth3 [2,4]	Art8 [2]	AFA9 [3]	Teethform1 [2,4]
AF17 [2]	Teeth4 [2,4]	Art9 [2]	AFA10 [3]	Teethform1 [2,4]
AF18 [2]	Teeth5 [2,4]	Art9 [2]	AFA11 [3]	Teethform1 [2,4]

AF19 [2]	Teeth6 [2,4]	Art9 [2]	AFA12 [3]	Teethform1 [2,4]
AF20 [2]	Teeth7 [2,4]	Art4 [2]	AFA7 [3]	Teethform1 [2,4]
AF21 [2]	Teeth8 [2,4]	Art4 [2]	AFA10 [3]	Teethform1 [2,4]
AF22 [2]	Teeth9 [2,4]	Art5 [2]	AFA8 [3]	Teethform1 [2,4]
AF23 [2]	Teeth11 [2,4]	Art6 [2]	AFA9 [3]	Teethform1 [2,4]
AF24 [2]	Teeth10 [2,4]	Art5 [2]	AFA11 [3]	Teethform1 [2,4]
AF25 [2]	Teeth12 [2,4]	Art6 [2]	AFA12 [3]	Teethform1 [2,4]
AF26 [2]	Cylinder [2,3]	Art10 [2]	AFA13 [3]	L= width of BearingJournal [2,3]

Table 13: Assembly features[2] defined using the user interface [11]

4.1.4 Detailed Design: Tolerance assignment

We illustrate the one aspect of detailed design directly supported by the CPM/OAM: tolerance assignment. For tolerance synthesis and analysis, we need a detailed kinematic description of the assembly. The kinematic definitions include the location, size, and form of the associated mating features. The engineering requirements on the desired product's function, specified as customer needs in the early stages of design, do not directly provide these kinematic specifications. These specifications evolve slowly with the assembly as the design takes concrete shape and size in later phases of early design. Tolerance synthesis and analysis needs a complete functional analysis to make sure that the identified functional requirements between the mating components of the assembly are met. Tolerance needs to be suitable described in the form of critical toleranced dimensions/sizes/forms or in the form of toleranced gaps. Roy et al [16; 17] have been developing a tolerance and synthesis representation that uses the small deviation torsor scheme to represent the variations associated with each feature of a part in the assembly. Using this technique, we define the tolerance information generated for the sun gear and the gearbox as a whole in Figure 17 and Figure 18 respectively, and tabulate the design data in Table 14 and Table 15.

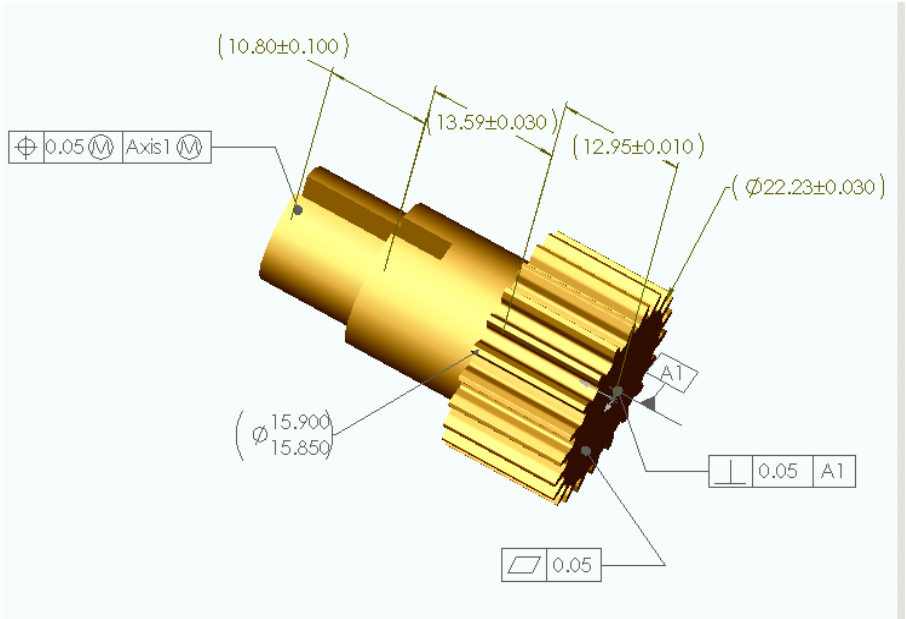

Figure 17: Geometrical and dimensional tolerancing on sungear (Art7)

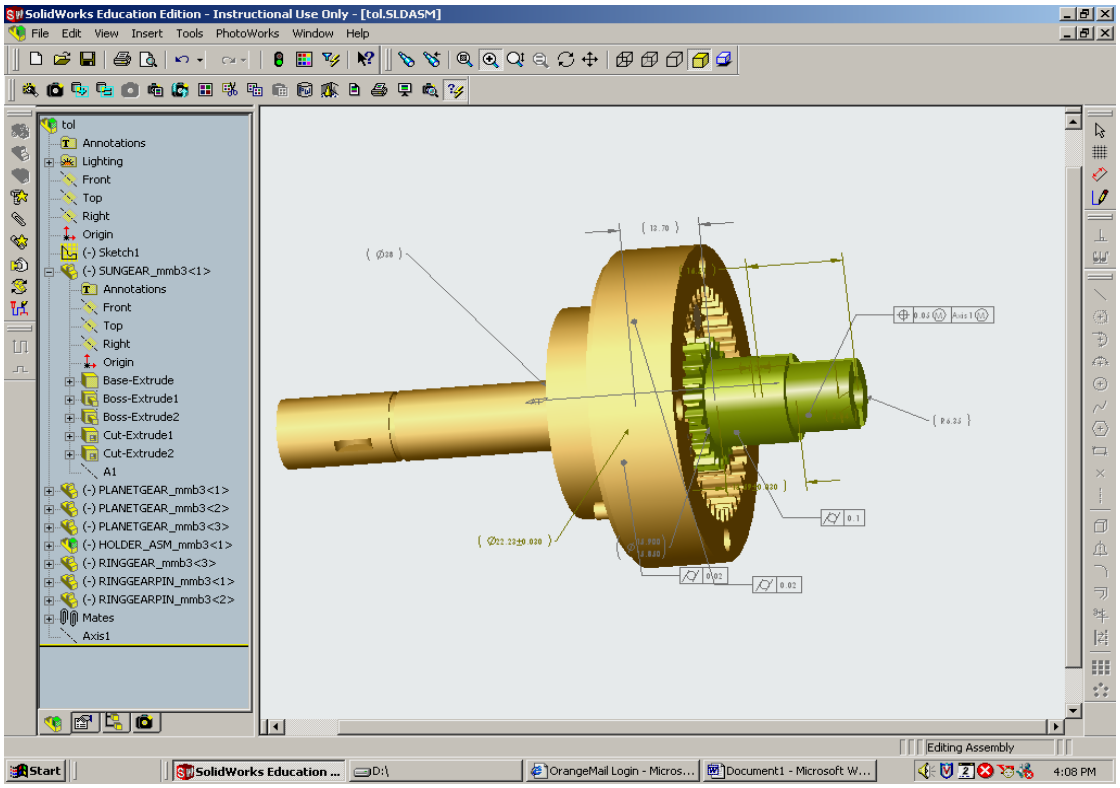

Figure 18: Tolerancing for the whole gearbox assembly

Id	Name	Artifact	OAMF	Magnitude	Datum	MMC
GT1 [3]	PerpTol1 [3]	Art8 [3]	OAMF EndSurface1 [3]	0.05 [3]	DatumAxis1 (A1) [3]	
GT2 [3]	Flattol1 [3]	Art8 [3]	OAMF Endsurface1 [3]	0.05 [3]	-	
DT1 [4]	DimTol1 [4]	Art8 [4]	OAMF Endsurface1 [4]	(ϕ22.23)±0.03 [4]		
DT2 [4]	DimTol2 [4]	Art8 [4]	OAMFSunGearteeth	(12.95)±0.01 [4]		
GT3 [3]	CylTol1 [3]	Art8 [3]	OAMF Shank [3]	0.1 [3]		
DT3 [4]	DimTol3 [4]	Art8 [4]	OAMF Shank [4]	(13.59)±0.03 [4]		
DT4 [4]	DimTol4 [4]	Art8 [4]	OAMF Shank [4]	ϕ15.85~ ϕ15.85 [4]		
GT4 [3]	PosTol1[3]	Art8 [3]	OAMFinputshaft [3]	0.05 [3]	DatumAxis1 (A1) [3]	MMC
DT5 [4]	DimTol5 [4]	Art8 [4]	OAMFinputshaft [4]	(10.85)±0.10 [4]		
GT5 [3]	CylTol2 [3]	Art4 [3]	PinHole6:AF [3]	0.02 [3]		
DT6 [4]	DimTol6 [4]	Art4 [4]	PinHole6:AF [4]	(ϕ4.90)±0.01 [4]		
DT7 [4]	DimTol7 [4]	Art4 [4]	GearCylinder1 [4]	(12.70)±0.10 [4]		
DT8 [4]	DimTol8 [4]	Art4 [4]	GearCylinder1 [4]	(ϕ10.16)±0.01 [4]		
GT6 [3]	ParTol1 [3]	Art9 [3]	rimsurface [3]	0.05 [3]	DatumPlane3 (A3) [3]	
DT9 [4]	DimTol9 [4]	Art9 [4]	GearTeethHole [4]	(ϕ42.62)±0.01 [4]		
DT10 [4]	DimTol10 [4]	Art9 [4]	PinHole1,2 [4]	(ϕ3.30)±0.05 [4]		
GT7 [3]	PerpTol2 [3]	Art7 [3]	EndSurface2 [3]	0.03 [3]	DatumAxis2 (A2) [3]	
GT8 [3]	Flattol2 [3]	Art7 [3]	EndSurface2 [3]	0.06 [3]	-	
GT9 [3]	TotRun1 [3]	Art7 [3]	outputShaftShank [3]	0.1 [3]	DatumAxis2 (A2) [3]	
GT10 [3]	ProfSurfTol1 [3]	Art7 [3]	Keyway [3]	0.1 [3]	-	

Table 14: Tolerance[3] information for Features and Artifacts

Id	Feature Name	Artifact	TolId
OAMF1	EndSurface1	Art8	GT1, GT2, DT1
OAMF2	SunGearteeth	Art8	DT2
OAMF3	Shank	Art8	GT3, DT3, DT4
OAMF4	inputshaft	Art8	GT4, DT5
OAMF5	PinHole6:AF	Art4	GT5, DT6
OAMF6	GearCylinder1	Art4	DT7, DT8
OAMF7	rimsurface	Art9	GT6
OAMF8	GearTeethHole	Art9	DT9
OAMF9	PinHole1	Art9	DT10
OAMF10	PinHole2	Art9	DT10
OAMF11	EndSurface2	Art7	GT7, GT8
OAMF12	outputShaftShank	Art7	GT9
OAMF13	Keyway	Art7	

Table 15: OAMFeatures[3] (only toleranced features are listed)

4.2 Usage Example

In order to show how to apply the usage pattern presented previously in Section 3.2, in the context of the planetary gear example [3], we start by analyzing the planet-gear-carrier assembly. As shown in Figure 19, this assembly is composed of the planet-carrier subassembly and three planet gears. First, three planet gear pins are assembled with the output shaft, through a tight fit, to form the planet-carrier subassembly. Second, the other end of each planet gear pin is assembled, by a loose fit, to a planet gear to form the complete planet-gear-carrier assembly.

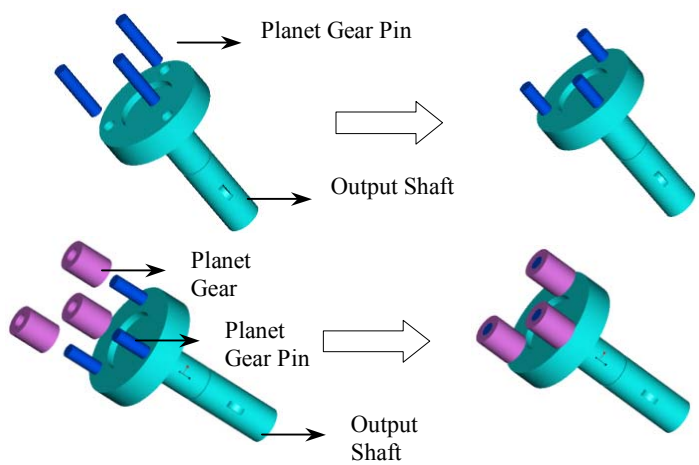

Figure 19: Planet-gear-carrier assembly

Figure 20 shows the UML instance diagram of the complete assembly at the **ArtifactAssociation** level (the decomposition of these **ArtifactAssociation**Usage into **AssemblyFeatureAssociation** and **AssemblyFeatureAssociationRepresentation** is not shown in this diagram; we refer the reader to [3; 15] for the complete diagram).

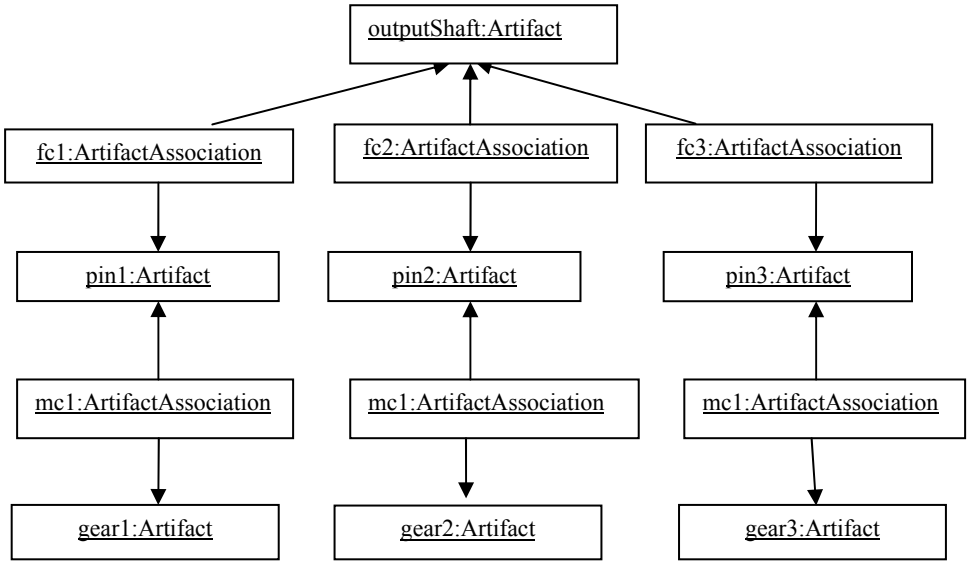

Figure 20: Instance diagram of the Planet-gear-carrier assembly

In this assembly, the three planet gear pins are supposed to be identical, but when we see the instance diagram of Figure 20, we cannot affirm that we are using three identical pins with the same attribute values. There is no way to ensure that these three pins have equal lengths, equal base radii or the same material. A similar comment applies to the three planet-gears, which are supposed to be identical, but the instance diagram does not show this.

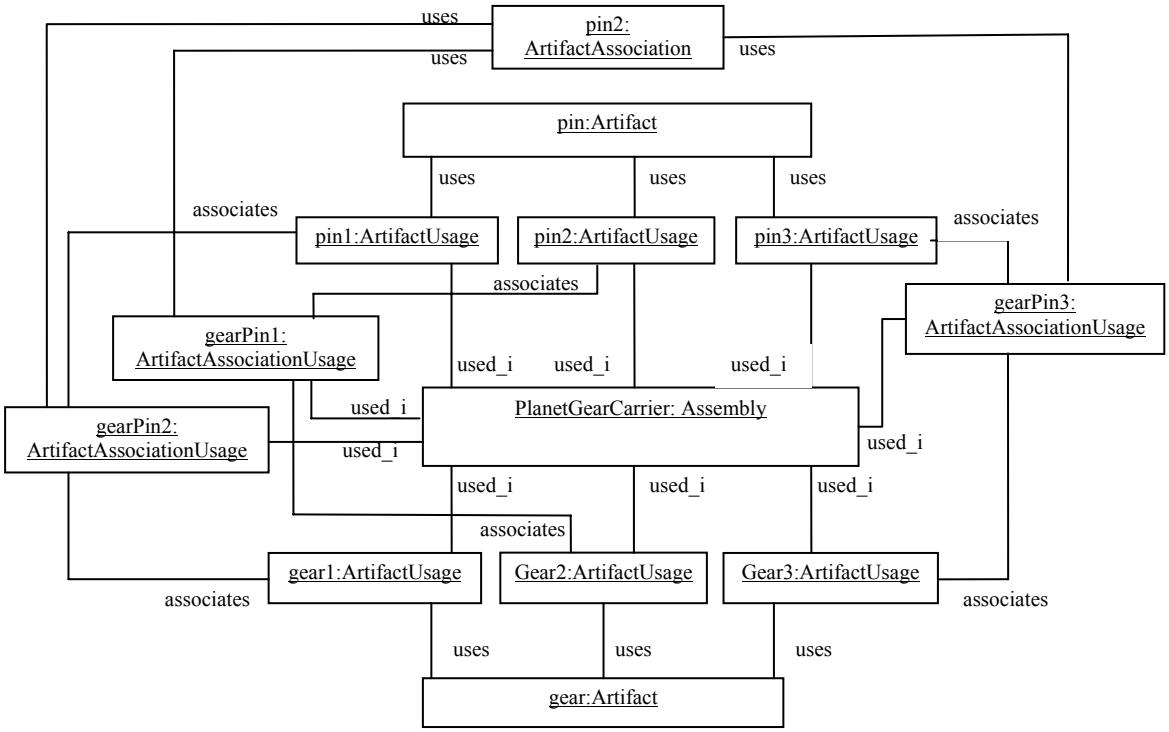

Figure 21: Instance diagram of the Planet-gear-carrier assembly with the usage pattern

The use of the ***usage*** pattern (see Figure 21) solves this problem. We use a "pin", a generic usable **Artifact** for pins, in the context of planet-gear-carrier assembly playing three different roles: pin1; pin2; and pin3, represented as instances of **ArtifactUsage**. This means that the three pins are copies of the same reusable **Artifact**. Similarly, one reusable "gear" **Artifact** is playing three different roles: gear1; gear2; and gear3, represented as instances of **ArtifactUsage**. We then use the three instances of the **ArtifactAssociationUsage** to associate gear-pin couples.

5 Future Work

We are looking to future extensions of this work from two different points of view: (1) CPM/OAM model revisions to achieve completeness and comprehensiveness; and (2) model implementation.

First, from the model revision point of view, potential future development of this work could include:

– Introduce tolerances on usages. We associate tolerance in OAM to artifact associations. When we use an artifact association in any particular assembly, there may be additional constraints placed on tolerance inherited from the generic artifact association. In many cases, tolerance determination is by the context of usage. The

extension to introduce tolerances on usage will provide the flexibility to choose the least expensive tolerance that will meet requirements for that usage of the artifact.

- Demonstrate the information added into STEP through OAM using an industrial case study.

- Convert to UML's composite structure model and its extension for systems engineering, in particular SysML ([18], [19]). These modeling languages have a composition model based on the usage pattern, that is integrated with other aspects of structure and behavior modeling that would be of benefit to CPM/OAM. Then the various kinds of artifact association, feature association, and tolerances would be available from a widely used structure model.

- Investigate an alternative usage model, based on a flexible relation of types and instances. OAM and CPM elements in Table 1 are assumed reusable, in particular, it is assumed that they represent the types rather than instances. Many subject matter experts in engineering do not make a strong distinction between type and instance. For example, when we think of a design for a car it could imply the design of an actual car, or a prototypical car, and the notion of usage is often implicit. It is left for future work to investigate alternative approaches to the class/instance/usage distinctions presented here.

Second, from the implementation point of view, we are pursuing eXtensible Markup Language(XML) and Ontology Web Language (OWL) implementations of CPM and OAM. The corresponding XML schemas will be used to facilitate interactions between these two models and other existing systems. The exchange of geometric and design data will be tested. Our interest in an OWL implementation arises from the reasoning and consistency checking facilities available in OWL tool implementations. The other issues for building an OAM are (i) constraints satisfaction and (ii) the maintenance of product functionality. We intend to use key characteristics (KCs) [20] to define constraints and parameters, which are important to maintain product functionality and manufacturability.

6 Conclusions

Product data management is an important issue in any collaborative product design and concurrent engineering. The foci of existing product data models are towards the end of a design process and are difficult to be used during early phases of the design. Though the current assembly models describe the structure of an assembly, they cannot provide for the integration of functional needs and technical solutions, as in the case of STEP. Hence, the existing product models are not very useful in all phases of collaborative design. The OAM is a significant step towards addressing this limitation. In this report, we discuss the use of an OAM in a real life gearbox design to show the utility of this model. The OAM as an extension to CPM promises to provide support for the product's entire lifecycle, from the first conceptualization to disposal, by providing a uniform framework for product information for accessing, storing and reusing all of the product information throughout the entire lifecycle.

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

1. Fenves, S. J., *A Core Product Model For Representing Design Information*, National Institute of Standards and Technology, NISTIR6736, Gaithersburg, MD 20899, USA, 2001.

2. 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.

3. Sudarsan, R., Young-Hyun H, Feng, S. C., Roy, U., Fujun W., Sriram, R. D., and Lyons, K. W., *Object-oriented Representation of Electro-Mechanical Assemblies Using UML*, National Institute of Standards and Technology, NISTIR 7057, Gaithersburg, MD 20899, USA, 2003.

4. ISO TC 194/SC4, Official *TC184/SC4 Web Site*, http://www.tc184-sc4.org/.

5. Shooter, S. B., Keirouz, W. T., Szykman, S., and Fenves, S. J., *A Model for the Flow of Design Information in Product Development*, Engineering with Computers, Vol. 16, 2000, pp. 178-194.

6. *ISO 10303-1: 1994, Industrial automation systems and integration -- Product data representation and exchange -- Part 1: Overview and fundamental principles*. Geneva, Switzerland, ISO.

7. ISO/DIS 10303-108:2003, *Product data representation and exchange: Integrated application resource: Parameterization and constraints for explicit geometric product models*. International Organization for Standardization (ISO), Geneva, Switzerland.

8. ISO10303-105:1996, *Industrial automation systems and integration -- Product data representation and exchange -- Part 105: Integrated application resource: Kinematics*. International Organization for Standardization (ISO), Geneva, Switzerland.

9. ISO 1101:1983, *Technical drawings -- Geometrical tolerancing -- Tolerancing of form, orientation, location and run-out -- Generalities, definitions, symbols, indications on drawings*. International Organization for Standardization, Geneva, Switzerland.

10. Gamma, E., Johnson, R., and Vlissides, J., *Design Patterns*, Addison-Wesley 1995.

11. 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*, Salt Lake City, Utah, 2004.

12. Shooter, S. B., Keirouz, W. T., Szykman, S., and Fenves, S., *A Model for the Flow of Design Information in the Open Assembly Design Environment (OpenADE)*, National Institute of Standards and Technology, NISTIR6746, Gaithersburg, MD, USA, 2001.

13. Pramanik, N., Roy, U., Sudarsan, R., Sriram, R. D., and Lyons, K. W., *Synthesis Of Geometric Tolerances Of A Gearbox Using A Deviation-Based Tolerance Synthesis Scheme*, International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Chicago, Illinois, September 2-6, 2003, 2003.

14. Chang, E., Li, X., and Schmidt, L. C., *The need for Form, Function, and Behavior-based Representations in Design*, Department of Mechanical Engineering, University of Maryland, College Park, MD, USA, 2000.

15. Sudarsan, R., Roy, U., Young-Hyun H, Feng, S. C., Fujun W, Sriram, R. D., and Lyons, K. W., *Object-Oriented Representation of Electro-Mechanical Assemblies Using UML*, IEEE International Symposium on Assembly and Task Planning, Besancon, France, July 9-11, 2003.

16. Roy, U., Sudarsan, R., Pramanik, N., Sriram, R. D., and Lyons, K. W., *Function-to-Form Mapping: Model, Representation and Applications in Design Synthesis*, Computer-Aided Design, Vol. 33, 2001, pp. 699-719.

17. Roy, U., Pramanik, N., Wang, H., Sudarsan, R., Sriram, R. D., and Lyons, K. W., *Tolerance Synthesis Scheme*, National Institute of Standards and Technology, NIST IR 6836, Gaithersburg, MD 20899, USA, 2003.

18. Bock, C, Systems *Engineering in the Product Lifecycle*, International Journal of Product Development, Special Issue on Product Lifecycle Management, 2004.

19. *OMG: UML 2.0 Superstructure Specification*. 2003.

20. Rezayat, M., *Knowledge-based product development using XML and KCs*, Computer-Aided Design, Vol. 32, 2000, pp. 299-309.