A product information modeling framework for product lifecycle management

R. Sudarsan*, S.J. Fenves, R.D. Sriram, F. Wang

Manufacturing Systems Integration Division, Manufacturing Engineering Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA

Accepted 2 February 2005

Abstract

The Product Lifecycle Management (PLM) concept holds the promise of seamlessly integrating all the information produced throughout all phases of a product’s life cycle to everyone in an organization at every managerial and technical level, along with key suppliers and customers. PLM systems are tools that implement the PLM concept. As such, they need the capability to serve up the information referred to above, and they need to ensure the cohesion and traceability of product data.

We describe a product information-modeling framework that we believe can support the full range of PLM information needs. The framework is based on the NIST Core Product Model (CPM) and its extensions, the Open Assembly Model (OAM), the Design-Analysis Integration model (DAIM) and the Product Family Evolution Model (PFEM). These are abstract models with general semantics, with the specific semantics about a particular domain to be embedded within the usage of the models for that domain. CPM represents the product’s function, form and behavior, its physical and functional decompositions, and the relationships among these concepts. An extension of CPM provides a way to associate design rationale with the product. OAM defines a system level conceptual model and the associated hierarchical assembly relationships. DAIM defines a Master Model of the product and a series of abstractions called Functional Models—one for each domain-specific aspect of the product—and two transformations, called idealization and mapping, between the master model and each functional model. PFEM extends the representation to families of products and their components; it also extends design rationale to the capture of the rationale for the evolution of the families.

The framework is intended to: (1) capture product, design rationale, assembly, and tolerance information from the earliest conceptual design stage—where designers deal with the function and performance of products—to the full lifecycle; (2) facilitate the semantic interoperability of next-generation CAD/CAE/CAM systems; and (3) capture the evolution of products and product families. The relevance of our framework to PLM systems is that any data component in the framework can be accessed directly by a PLM system, providing fine-grained access to the product’s description and design rationale.

© 2005 Elsevier Ltd. All rights reserved.

Keywords: Product Lifecycle Management (PLM); Core Product Model; Open Assembly Model; Interoperability; Ontology; Standards

1. Introduction

PLM is generally defined as ‘a strategic business approach for the effective management and use of corporate intellectual capital’ [1]. PLM systems are gaining acceptance for managing all information about a corporation’s products throughout the products’ full lifecycle. Global competition is one of the key drivers for many organizations to adopt the PLM concept and implement PLM systems. The PLM concept aims to streamline product development and boost innovation in manufacturing. Hence the PLM concept is a strategic business approach for the effective creation, management and use of corporate intellectual capital, from a product’s initial conception to its retirement [1].

Even in the current (2003) economic downturn, many manufacturing companies are investing in PLM systems—to the tune of $2.3 billion this year [2]. We believe the reason why these companies are willing to take the risk is that these companies see PLM’s potential to vastly improve...
their ability to innovate, get products to market faster, and reduce errors. According to industry analyst CIMdata, “For an enterprise to be successful in today’s and tomorrow’s global markets, PLM is not an option—it is a competitive necessity” [1].

A critical aspect of PLM systems is their product information modeling architecture. Here, the traditional hierarchical approach to building software tools presents a serious potential pitfall: if PLM systems continue to access product information via Product Data Management (PDM) systems which, in turn, obtain geometric descriptions from Computer-Aided Design (CAD) systems, the information that becomes available will only be that which is supported by these latter systems.

In this paper, a different approach to serving up information to PLM systems is proposed: a single PLM system support framework for product information that can access, store, serve, and reuse all the product information throughout the entire product lifecycle. This framework and its components are presented after a brief discussion of the PLM concept and of the major PLM system architecture and interoperability issues.

1.1. The PLM concept

PLM holds the promise of seamlessly integrating and making available all of the information produced throughout all phases of a product’s life cycle to everyone in an organization, along with key suppliers and customers. Manufacturers can shrink the time it takes to introduce new product models in a number of ways. Product engineers can dramatically shorten the cycle of implementing and approving engineering changes across an extended design chain. Purchasing agents can work more effectively with suppliers to reuse parts. Executives can take a high-level view of all important product information, from details of the manufacturing line to parts failure rates culled from warranty data and information collected in the field.

Because PLM systems grew out of product design software, company management tends to delegate the PLM concept to engineering executives, who traditionally have managed their own technology rollouts. While this hands-off approach works for choosing point solutions, such as CAD tools, it does not work well for a company-wide integrated platform. Different business functions generate and deal with product data in disparate ways. Manufacturing and engineering, for instance, work with a different version of a bill of materials—a listing of parts and subassemblies making up a product—than does purchasing, which also relies on approved vendor lists and catalogs.

For the PLM concept to be successful, issues such as establishing data standards and designing corporation-wide integration architectures need to be addressed so that formerly fragmented information can be served up to individuals in a format they can use. That way, people in various divisions are equipped to make key decisions—such as what products to introduce or what features to include in a product’s design phase—when they are most cost-effective, rather than midstream in the parts procurement stage or even during manufacturing.

1.2. PLM system architecture and interoperability issues

PLM systems are tools that assist a corporation in the implementation of PLM concepts. One of the main questions regarding PLM systems is: “What constitutes the PLM systems’ functionality?” The full PLM system functionality can be achieved by the specific components illustrated in Fig. 1. These are: (1) an Information Technology (IT) Infrastructure; (2) a Product Information Modeling Architecture; (3) a Development Toolkit and Environment; and (4) a set of Business Applications. The IT infrastructure is the foundation that includes hardware, software, and Internet technologies, underlying representation and computing languages, and distributed objects and components.

The product information modeling architecture includes product ontology and interoperability standards. The development toolkit and environment provide the means for building Business Applications that provide the initial functionality and enhance and extend the functionality of the PLM concept and could include kernels (e.g. geometry, math), visualization tools, data exchange standards and mechanisms, and databases. The business applications provide the PLM functionality that processes the corporate intellectual capital.

In two recent NIST workshops held in 2003, attempts were made to describe an architecture for the lifecycle-wide management and integration of product data [4,5]. The architecture, as described in the working draft of the workshops’ summary report, is intended to provide a roadmap for the application of the diverse information technologies and computer science concepts that may be used to build and operate PLM systems supporting the full product lifecycle [6].

The domain of application for the resulting PLM system considered in the workshops deals with complex

Fig. 1. A conceptual PLM system architecture.
engineered-to-order systems, such as found in the aerospace and defense industries. The architecture defines two classes of views of product data: semantic views define constraints on the interpretation and usage of the information; while infrastructural views relate to the encoding and composition of data in the processes and tools in which it is used. Potentially applicable technologies are discussed in the working draft with respect to these two classes of views.

Some of the principal concerns expressed in the NIST planning meetings were the cohesion and traceability of product data. The conclusion was that current data management practices do not provide sufficient support of data cohesion and traceability. Cohesion and traceability, however, are complex and abstract goals when viewed as attributes of an information system. Information technology does not address these goals directly; rather, certain other qualities help to support these goals. Among the major constituent properties of cohesion and traceability identified were associativity across viewpoints and logical consistency [6].

PLM systems form the apex of the corporate software hierarchy and frequently implemented so that they depend on subsidiary systems for detailed information capture and dissemination. PLM systems therefore tend to delegate the task of managing the information describing the product itself to Product Data Management (PDM) systems. Furthermore, in many organizations, only the geometric description of products generated by Computer-Aided Design (CAD) systems is managed directly; in these organizations PDM systems rely on the CAD systems for managing product descriptions.

The above segmentation of PLM and subsidiary software systems results in three shortcomings. First, while PLM systems can track changes through the products’ lifecycle from conception to disposal, the information that describes the actual changes can be found only through the subsidiary PDM systems, and the reason for the changes may not be recorded in computer-processable form anywhere. Thus, there is a need to make product descriptions and their design rationale directly accessible from PLM systems, with no intermediary layers of software. Second, CAD representations of form (geometry) arise only at later stages of design, after a geometry has been assigned to the product concept; therefore, PLM systems tied only to CAD representations of products cannot be useful before the form is assigned. In order to realize the PLM concept’s full potential, PLM systems need to interact with product information used in the early stages of conception and ideation, where designers and planners deal with the function and performance of products, and not yet with their form. Third, at the opposite end of the product’s lifecycle, during manufacturing, installation, operation, maintenance and, eventually, disposal, the form of the product changes little, while much information is gathered about the product’s behavior in these lifecycle stages. Here again, PLM systems tied only to CAD representations of products cannot be useful; PLM systems need to interact with product behavior information in the late stages of the lifecycle.

PLM systems are still in the very early stages and are in a flux. This may lead to the development of many proprietary systems and interfaces, which would result in additional interoperability problems. Hence we need national and international efforts to develop standards to alleviate future interoperability problems for PLM systems. We have made it our goal in the Product Engineering Program at the National Institute of Standards and Technology in the US to establish a semantically based, validated product representation scheme as a standard that supports the seamless interoperability among current and next generation computer-aided design (CAD) systems and between CAD systems and other systems that generate and use product. As part of this effort, we are developing a framework and a representation scheme that will address some of the above-mentioned issues.

The focus of this paper is the second component of the PLM system architecture presented in Fig. 1, namely, the product information modeling architecture. The aim of the paper is to argue that the Product Engineering Program’s approach can: (1) support the full range of PLM information needs; and (2) overcome the three shortcomings of the PLM software segmentation discussed above. The paper is organized as follows. In Section 2 we introduce the NIST information-modeling framework. In Section 3, we describe the four components of the NIST information modeling framework. Further research issues to be addressed are discussed in Section 4. Finally the conclusions are given in Section 5.

2. The NIST information modeling framework

The exchange of product, part and assembly information between heterogeneous modeling systems is critical for collaborative design and manufacturing. Interchange standards for product geometry are in wide use. However, little has been done in terms of developing standard representations that specify the full range of design information and product knowledge. The NIST information-modeling framework is intended to address this issue.

The conceptual product information modeling framework under development at NIST has the following key attributes: (1) it is based on formal semantics, and will eventually be supported by an appropriate ontology to permit automated reasoning; (2) it is generic: it deals with conceptual entities such as artifacts and features, and not specific artifacts such as motors, pumps or gears; (3) it is to serve as a repository of a rich variety of information about products, including aspects of product description that are not currently incorporated; (4) it is intended to foster the development of novel applications and processes that were not feasible in less information-rich environments; (5) it incorporates the explicit representation of design...
rationale, considered to be as important as that of the product description itself; and (6) there are provisions for converting and/or interfacing the generic representation schemes with a production-level interoperability framework. An implementation of the information modeling framework will: (1) provide a generic repository of all product information at all stages of the design process; (2) serve all product description information to the PLM system and its subsidiary systems using a single, uniform information exchange protocol; and (3) support direct interoperability among CAD, CAE, CAM and other interrelated systems where high bandwidth, seamless information interchange is needed.

To better understand the high-level view of PLM framework we adapted the epicycle diagram from [7] to describe the process and information flows in any product lifecycle. The Figs. 2 and 3 explains the epicycle nature of PLM. The Fig. 2 characterizes the information flow pattern in the PLM, as it perceived today. In Fig. 3 the mediation of information flow across the activities of PLM are done through a common set of ontological structure, and information models to represent product and process. The concept of product information model framework is derived from the traditional engineering design, functional reasoning, and product modeling [8–11]. The main focus of this paper is to synthesize these representations in the context of engineering information exchange as well as in the context of computational models.

3. Components of the information modeling framework

The NIST information-modeling framework consists of the four major components as sown in Fig. 4. The dependency relationships (represented by dashed arrows) among these packages show that there exist certain association or generalization relationships among classes in the different packages. In this paper, we only give brief descriptions of these packages. The models are explained in more detail elsewhere, using an example [12–14].

3.1. The core product model

The primary objective of the Core Product Model (CPM) is to provide a base-level product model that is open, non-proprietary, generic, extensible, independent of any one product development process and capable of capturing the full engineering context commonly shared in product development [15]. Throughout the paper we use the notation and class diagrams of the Unified Modeling Language (UML) [16], we also use bold face font for UML classes.

---

1 We thank E. Subrahmanian (Carnegie Mellon) in developing this figure.
and packages, where a package in UML is a collection of classes that can be used as a namespace.

Fig. 5 illustrates the entities comprising the CPM. All entities are specializations of the abstract class CommonCoreObject. CoreEntity and CoreProperty are abstract classes. The former specializes into Artifact and Feature, and the latter into Function, Form, Geometry, and Material. A DesignRationale class (discussed in Section 3.4) is associated with CoreProperty.

Artifact is the aggregation of Function, Form, and Behavior. Form in turn is the aggregation of Geometry and Material. In addition, an Artifact has a Specification and is an aggregation of Features. Feature represents any information in the Artifact that is an aggregation of Function and Form. Artifact, Feature, Function, Form, Geometry and Material are each aggregates of their own containment hierarchies (part-of relationships).

Semantically, Artifact represents a distinct entity in a product, whether that entity is the entire product or one of its subsystems, parts or components. Function represents what the artifact is intended to do. The distinct representation of Function renders the core product model and its extensions capable of supporting functional reasoning in the absence of any information on the artifact’s form, thus providing support for the conceptual phases of design.

Form may be viewed as the proposed design solution to the problem specified by the function and consists of the artifact’s Geometry (shape and structure may be synonymous to geometry in some contexts) and the Material it is composed of. Behavior represents how the artifact’s form implements its function; one or more causal models, such as Finite Element Analysis (FEA) or Computational Fluid Mechanics (CFM) models, may be used to evaluate it. Cost, manufacturability, durability, etc. are examples of other behavioral models that may be incorporated. As stated above concerning function, this extended representation of behavior renders the core product model and its extensions capable of supporting behavioral reasoning at all stages of the product’s lifecycle.

Feature represents a subset of the form that has some function assigned to it. CPM does not treat pure form elements as features, nor does it support the independent behavior of features.

Fig. 6 shows the relationships in the CPM. All relationships are subclasses of the abstract class CommonCoreRelationship and are all UML association classes.
Requirement is an association class between the Specification and a CoreProperty of the artifact; each requirement applies to some aspect of the function, form, geometry or material of the artifact (purists in design theory may argue that requirements may only address function, but in practice many aspects of form may be specified without giving a specific functional justification). Constraint links a set of CoreProperty entities that share an attribute that must hold in all cases.

There are two specializations of SetRelationship: UndirectedSetRelationship groups objects into a set, while DirectedSetRelationship groups them into two subsets with different roles (e.g. a controlling subset and a controlled-by subset). AssemblyRelationship is implemented in the CPM as an undirected set of artifacts and features; it is specialized in the Open Assembly Model described below. Finally, a Reference links or cross-references entities.

3.2. 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 [17]. 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) [3,18]. Fig. 7 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 Associations.

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. 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 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 geometric 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 ArtifactAssociation 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 [19]. 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 [20] 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 [20] 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 subassemblies 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 Fig. 8.

**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 [21]. **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 tolerated feature in manufacturing.

![Fig. 8. Tolerance model.](image-url)
3.3. The design-analysis integration model

Computer-Aided Design, for generating a product’s geometry, and Computer-Aided Engineering, for analyses of its behavior, are both in common use today. Typically, a product’s behavior needs to be analyzed in several functional domains (e.g. structural, thermal, kinetics, economics) and the results of the analyses may suggest design changes for improving or optimizing the behavior. However, the integration of the efforts of the professionals in the two disciplines of spatial and functional design is not as complete as it should be, resulting in the limited interoperability of the two sets of tools.

The Design-Analysis Integration Model (DAIM) is a conceptual data architecture that provides the technical basis for tighter design-analysis integration than is possible with today’s tools and information models. It is also intended to make analysis-driven design (often referred to as form-to-function reasoning) more practical. Eventually, it should also support opportunistic analysis, where the system tracks the geometric design process and notifies the designer when sufficient geometric information has been generated to initiate a functional analysis [22].

The class diagram of the DAIM is shown in Fig. 9. MasterModel and FunctionalModel are both specializations of the CPM Artifact class; the latter also serves as the organizing principle for all information in the DAIM. The Master Model serves as the global repository of information on a product; in practice, it may be implemented as a centralized, distributed, federated or virtual database. Each FunctionalModel represents an abstraction of the product of interest to a specific functional domain at a particular stage in the lifecycle of a product. The figure shows three representative specializations: a StrengthView for finite element modeling and analysis; a ShapeView for classical CAD geometry modeling; and a KinematicsView for kinematic modeling and analysis.

Two association classes link the master and functional models. Idealization provides the transformation that creates a functional model specific to a particular domain from the master model; this is typically an abstraction operation removing detail irrelevant to the particular function, but more general transformations may also be used. Mapping provides the reverse transformation of updating the master model based on changes in the domain-specific functional model; it is conceptually the more difficult transformation to define and develop for the various functional domains of interest, as it is responsible for maintaining full logical consistency between the two models.

3.4. The product family evolution model

Many manufacturing concerns develop product families so as to offer a variety of products with reduced development costs [23]. The Product Family Evolution Model (PFEM) represent the evolution of product families and of the rationale of the changes involved [24]. The model consists of three sub-models: family, evolution, and evolution rationale.

A product is made up of components that usually have their own family definitions. Therefore, product and component families are modeled separately, and configuration relationships established between products and their components. The class PFEM_Artifact, a specialization of the CPM Artifact class, represents the design information about an artifact in the family.

Fig. 10 shows the class diagram. Family, Series, and Version are subclasses of FamilyDesignation. Family is the designation for an entire artifact family, a collection of Series that may have sub-series. Series, in turn, are composed of a chronologically sequenced chain structure of Versions.

ProductFamily and ComponentFamily, ProductSeries and ComponentSeries, and ProductVersion and ComponentVersion are subclasses of Family, Series, and Version, respectively. Configuration is the association class between ProductVersion and ComponentVersion that defines the actual configuration of component versions in each of the product versions. Family Evolution.
consists of two aspects: Family Derivation and Design Evolution. Family Derivation contains the precedence relationships between series and versions in the evolution of the product line. Design Evolution contains the design information characterizing the changes between particular series or versions and their predecessor(s).

Fig. 11 shows the class diagram of family evolution. The class Evolution is the aggregation of FamilyDerivation and DesignEvolution. FamilyDerivation is specialized into SeriesDerivation and VersionDerivation. SeriesDerivation is the association class between a series and its predecessor series, and VersionDerivation is the association class between a version and its predecessor version(s). DesignEvolution is the association class between a PFEM_Artifact of a series or version and that of its predecessor series or version.

Evolution Rationale. While Family Evolution captures what has changed, Evolution Rationale captures the reasons for the changes. The evolution rationale includes two aspects: FamilyDerivationRationale and DesignEvolutionRationale. Family Derivation Rationale captures the driving factors for the changes in the product line while Design Evolution Rationale records the reasons for the design changes.

The class EvolutionRationale is defined in the package Rationale, shown in Fig. 12. The classes DesignRationale and EvolutionRationale are subclasses of Rationale. The class DesignJustification defines the justification of the design decision to use the associated artifact, and is the principal contents of the design rationale.

A representative set of specializations of DesignJustification is shown in the figure. DesignEvolutionRationale and FamilyDerivationRationale are subclasses of EvolutionRationale. DevelopmentSpecificationEvolution represents the evolution of DevelopmentSpecification, the driving factors that are the justifications of the changes in the product family. Requirement, Regulation, and Technology are the subclasses of DevelopmentSpecification currently supported.

4. Further research needs

A number of issues have to be investigated before implementation of a PLM system support and
interoperability platform based on the proposed product information-modeling framework can begin. First, the framework presented is but a first step towards a complete product modeling architecture supporting the PLM concept. A search needs to be made to identify other framework components that need to be modeled and integrated.

Second, a focused search of the PLM literature and current PLM system products needs to be made so as to clarify all product information needs throughout the PLM process to develop a conceptual Application Programming Interface (API) that can serve all product information to all PLM process components. As part of such a conceptual interface specification, considerable attention needs to be given to the possible interactions between the product data served by the framework and metadata about the product data maintained by the PLM system.

Third, recognizing that product information modeling frameworks of the scope contemplated here will be heterogeneous, rather than single-language, single-vendor homogeneous systems, research is needed to identify, and if necessary develop, information exchange standards that can provide the degree of interoperability that will be necessary.

5. Conclusions

Until quite recently, computer support for product development tended to cover a narrow slice of a product’s lifecycle, typically the segment from the product’s engineering specification to its physical embodiment. The PLM concept promises to provide support for the product’s entire lifecycle, from the first conceptualization to the disposal of its last instance. The volume, diversity, and complexity of information describing the product will increase correspondingly.

This paper makes a proposal for a single PLM system support framework for product information that can access, store, serve, and reuse all the product information throughout the entire lifecycle. The guiding principles for such a framework are outlined, and four components that constitute the kernel of such a framework are described. Further research is needed to identify and model the other components of the framework, to develop a conceptual API between PLM systems and the framework, and to identify or develop standards for the information interchange. The proposed product information modeling architecture framework is contemplated to have a broader scope than just being a product information server to PLM systems. Design and manufacturing process components interoperate by exchanging large volumes of product information, and the proposed product information modeling framework needs to support such ‘horizontal’ information exchanges as readily as the ‘vertical’ exchanges among process components, PLM systems and any intermediary systems, such as PDM and Enterprise Resource Planning (ERP) systems.

6. 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 the materials or equipment identified are necessarily the best available for the purpose.

References


Sudarsan Rachuri is a Research Professor with the Department of Engineering Management, George Washington University, Washington DC. He is a Guest Researcher in the Design and Process Group, Manufacturing Systems Integration Division, National Institute of Science and Technology (NIST), Gaithersburg, MD. Presently, his work at NIST includes development of information models for product lifecycle management, assembly models and system level tolerancing, and standards development. He coordinates research projects with industry and academia. He closely works with various standard bodies including ISO TC 184/SC4. He is a member of ASME Y14.5.1 and Advisory Group Member for ISO/TC 213/AG12, Mathematical Support for GPS. He is the regional editor (North America) for the International Journal of Product Development, and associate editor for International Journal of Product Lifecycle Management. His areas of interest include scientific computing, mathematical modeling, product lifecycle management, ontology modeling, system level tolerancing, quality, object oriented modeling, and knowledge engineering. Rachuri Sudarsan received the MS and PhD degrees from the Indian Institute of Science, Bangalore.

Steven J. Fenves is University Professor Emeritus of Civil and Environmental Engineering at Carnegie Mellon University and is a Guest Researcher at NIST. He received his degrees in Civil Engineering from the University of Illinois and has taught at the University of Illinois, Carnegie Mellon, MIT, National University of Mexico, Cornell and Stanford. His research deals with computer-aided engineering, design standards, engineering databases, and structural analysis and design environments. He is the author of six books and over 300 articles and is a member of the National Academy of Engineering and an Honorary Member of the American Society of Civil Engineers.
Ram D. Sriram, Senior Member IEEE is currently leading the Design and Process group in the Manufacturing Systems Integration Division at the National Institute of Standards and Technology, where he conducts research on standards for interoperability of computer-aided design systems and on healthcare informatics. Prior to that he was on the engineering faculty (1986–1994) at the Massachusetts Institute of Technology (MIT) and was instrumental in setting up the Intelligent Engineering Systems Laboratory. At MIT, Sriram initiated the MIT-DICE project, which was one of the pioneering projects in collaborative engineering. Sriram has co-authored or authored more than 175 papers, books, and reports in computer-aided engineering, including thirteen books. Sriram was a founding co-editor of the International Journal for AI in Engineering. In 1989, he was awarded a Presidential Young Investigators Award from the National Science Foundation, USA. Sriram has a BS from IIT, Madras, India, and an MS and a PhD from Carnegie Mellon University, Pittsburgh, USA.

Fujun Wang has worked on the research and development of collaborative design for 10 years. He is now working as a system engineer at the Shared Service Group, the Boeing company. Before joining Boeing, he had worked as a guest researcher for 4 years at the Manufacturing Systems Integration Division, National Institute of Standards and Technology. Dr Wang had worked at the Automation and Robotics Research Institute, University of Texas at Arlington, and the Key Center of Design Computing, University of Sydney, Australia during 1998 to 2000. Dr Wang received his PhD from the Beijing University of Aeronautics and Astronautics, China, in 1997.