Representation of Heterogeneous Material Properties in the Core Product Model

Arpan Biswas[†] biswas@cae.wisc.edu
Steve J. Fenves[‡] sfenves@cme.nist.gov
Vadim Shapiro[†] vshapiro@engr.wisc.edu
Ram Sriram[‡] sriram@nist.gov

Abstract

The Core Product Model (CPM) was developed at NIST as a high level abstraction for representing product related information, to support data exchange, in a distributive and a collaborative environment. In this paper, we extend the CPM to components with continuously varying material properties. Such components are becoming increasing important and popular due to progress in design, analysis and manufacturing techniques.

The key enabling concept for modeling continuously varying material properties is that of distance fields associated with a set of *material features*, where values and rates of material properties are specified. *Material fields*, representing distribution of material properties within a component, are usually expressed as functions of distances to material features, and are controlled with a variety of differential, integral or algebraic constraints. Our formulation is independent of any particular platform or representation, and applies to most proposed techniques for representing continuously varying material properties. The proposed model is described using system independent Unified Modeling Language (UML) and is illustrated through a number of specific examples.

1 Introduction

1.1 Motivation

Product development is increasingly being performed by geographically and temporally distributed teams with a high level of outsourcing of many phases of the product development process. As the complexity of products increases further, product development becomes even more distributed. Newer tools will be needed to address a broader spectrum of product development activities than do traditional Computer Aided Design and Engineering (CAD/CAE) systems. Next-generation tools will require representation capabilities that allow all information used or generated in the various product development activities to be transmitted to other related activities by way of direct electronic interchange. Furthermore, product development across interrelated companies, and even within a single company, will almost invariably take place within a heterogeneous software environment. As a result, there is a greater need for the support of the formal representation, capture, and exchange of the entire range of information generated and used in the product development process, not just the representation of the product resulting from the completion of the design process.

The ability to effectively and formally capture additional types of information will become a critical issue. NIST has continued work on product representations and interchange standards supporting the entire lifecycle of a product, from conceptualization to disposal, besides capturing all the information relevant to design and manufacturing processes. The key component of the NIST conceptual product information modeling framework is the Core Product Model (CPM). CPM is a base-level product model that is open, non-proprietary, generic, extensible, independent of any one product development process. It is capable of capturing the full engineering context commonly shared in the product's lifecycle. Design related information, such as geometry and material, and functional information, such as behavior specification, are the main components of CPM. Discretely varying material composition can be represented in CPM as a collection of quasi-disjoint regions with constant material attached to each of the regions.

Recently, components with *continuously* varying material properties have become frequent in product modeling – largely due to emerging techniques in design of functionally graded materials and solid free-form fabrication

[†]Spatial Automation Laboratory, University of Wisconsin - Madison
[‡]National Institute of Standards and Technology, Gaithersburg, MD

techniques that allow local material composition control. Powerful analysis and shape optimization methods (e.g., homogenization [2]) are available now for generating parts with variable material properties. Numerous advantages of components with *heterogeneously* and *anisotropically* varying materials are well documented [23, 10, 44], including weight reduction, improved structural and other mechanical properties, promise of embedded sensors, and substantially improved motion and deformation control. Applications of heterogeneous materials range from aircraft structures to medical products. For example, implants (prosthetic hip, dental) are usually designed to have superior wear resistance, superior bonding with bones, maximum strength, and biocompatibility. In this report, we extend CPM to represent such components with continuously varying material properties.

1.2 Concepts in heterogeneous modeling

Figure 1(a) presents a typical example of material modeling, which was discussed in [10]. The solid object is partitioned into a diamond cutter (chip) and a shank that is manufactured from functionally graded composition of SiC and diamond. To control the composition, the shank is subdivided into three parts: a block region with 100% SiC, a prismatic region with 80% SiC and 20% diamond, and the transition region between them where the composition is a gradual blend of the two materials. Ideally, the blend should be smooth and may satisfy additional constraints prescribed by the designer or constrained by the manufacturing process. Figure 1(b) shows an interpolation between the respective fractions of the two materials that vary linearly with the distance between the two regions and add up to 1 at every point of the solid. In general terms the material modeling problem may be defined as the following: given a geometric representation of solid and/or a collection of *material features* with known material properties, construct one or more *material fields*, subject to some given *constraints* (design, manufacturing, etc.) Each material field represents some material property that varies, usually continuously, from point to point throughout the space, including the boundary and the interior of the solid. Three material features are identifiable in the above example: the diamond chip and the two solid subsets of the shank where material properties are known. Two material fields are required to be constructed (one for SiC and another one for diamond), subjected to the following constraints: continuous interpolation of the material fractions specified on each of the features, fractions must add to 1 at all points, and the rate of change of material is linear in the distance from each feature. In a more general case, material features can be pointsets of any dimension, shape, or topology; they may or may not be subsets of a solid object, but provide convenient means for defining material distribution throughout the solid [43]. The material distributions may be given as known continuous functions $F(x, y, z)$ that are subjected to algebraic, differential, integral and/or interpolation constraints.

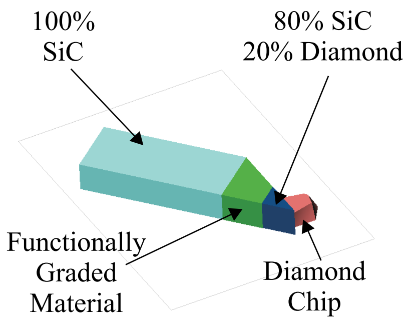
1.3 Outline

The rest of the paper is organized as follows. In Section 2, we summarize the NIST efforts on product modeling along with a description of the CPM, and briefly review material representation techniques. Section 3 contains the main contribution of this paper, an extension of the CPM for representing material heterogeneity [6]. In order to support data exchange between various existing modeling and manufacturing systems, the presented model includes most existing techniques for representing material heterogeneity. Applications of the corresponding UML model are explained with examples in Section 4, followed by summary and conclusions in Section 5.

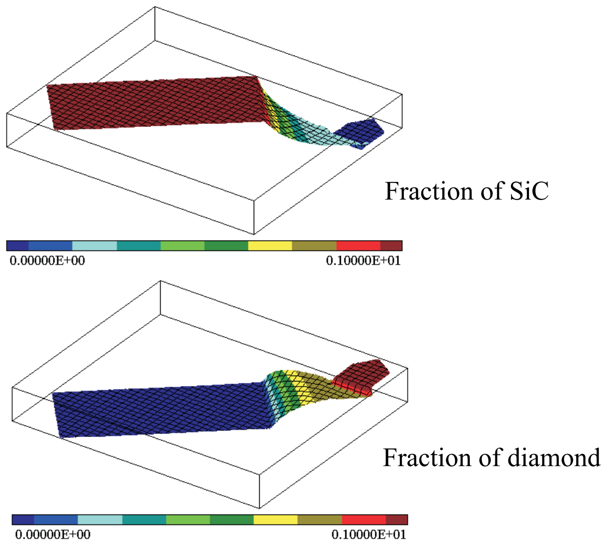
2 Previous work

2.1 ISO background

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. Perhaps the most widely used product interchange standard is ISO 10303, informally known as STEP (Standard for the Exchange of Product model data) [13, 24]. NIST has been active in the development of STEP, and served as the secretariat of the ISO Technical committee that developed it. STEP is a mature and widely used standard for the exchange of product data after that product has been designed. In practice, STEP can only be invoked late in the product development process, after all design decisions have been made and when the product is ready to be purchased, manufactured or assembled. Thus, STEP is used for the exchange of information that is the outcome of design activities, rather than for the information produced and used through the development of the design. STEP provides no support for



(a) Diamond cutter tool consists of several distinct material regions



(b) Fractions of each material vary throughout the solid but always add to 1

Figure 1: A typical material modeling problem requires construction of material property functions that interpolate known material features subject to specified constraints and physical laws. Plots in (b) correspond to the midsection of the cutter shown in (a).

design evolution in the early phases of design, when descriptive information is sparse, or for ready attachment of various forms of knowledge rather than pure data.A recently proposed (but not implemented) addition of ISO 10303-239:2005, ”Product Life Cycle Support,” informally known as PLCS, intends to support a broad range of information about the product, its assembly, configuration change, requirements, predicted and observed states, planning activities, product and activity history. For the actual product representation, this standard depends on AP 239 (Application Protocol for ” Product life cycle support”), for geometry and on a very limited material representation uncoupled from geometry. There is no concept of function or form: requirements are directly ”allocated” or linked to components, as is the representation of predicted and observed states of the product.

NIST has continued work on product representations and interchange standards, supporting the entire lifecycle of a product from conceptualization to disposal, and capturing all the information relevant to the design and manufacturing processes. 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; (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) it supports provisions for converting and/or interfacing the generic representation schemes with a production-level interoperability framework. The conceptual product information model is designed so that implementations of it can: (1) provide a generic repository of all product information at all stages of the design process; (2) serve all product description information to a 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.

2.2 Core Product Model

The Core Product Model (CPM) is a base-level product model that is open, non-proprietary, generic, extensible, independent of any product development process, and capable of capturing the full engineering context commonly shared in the product's lifecycle. In describing the CPM, we use the notation and class diagrams of the Unified Modeling Language (UML)[6] and we also use bold face font for UML classes in this paper. Figure 2 shows the UML model of the CPM, which was introduced in [9]. Refer to appendix for the UML notation used in this paper.

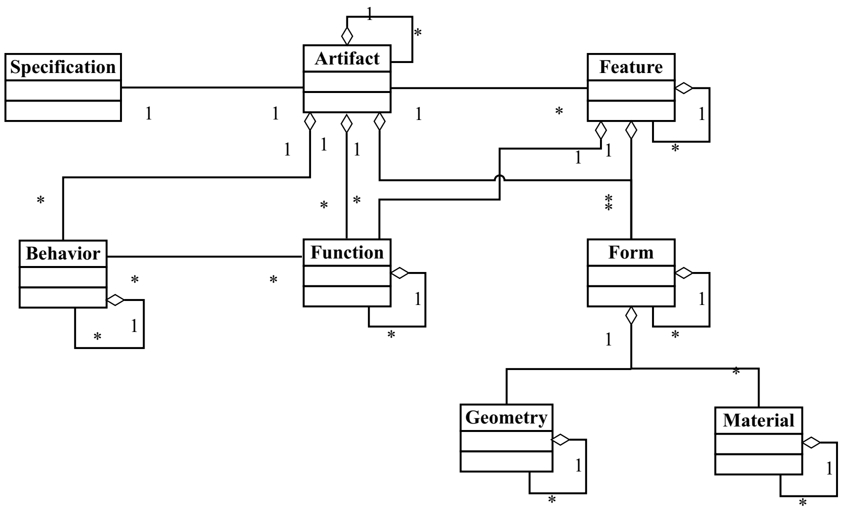


Figure 2: Base-Level Core Product Model (CPM)

All entities in the CPM are specializations of the abstract class **CommonCoreObject**. Classes **CoreEntity**, **CoreProperty**, **Behavior** and **Specification** specialize the class **CommonCoreObject**. The abstract class **CoreEntity** is further specialized into classes **Artifact** and **Feature**, and the other abstract class **CoreProperty** is specialized into **Function**, **Form**, **Geometry**, and **Material**. **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, one of its subsystems, parts or components. **Function** represents what the artifact is intended to do. The distinct representation of **Function** renders the CPM capable of supporting functional reasoning in the absence of any information on the artifact's form, thus providing support for the conceptual phases of product 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 **Material**. **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 CPM 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.

2.3 Heterogeneous Modeling

The need for developing an information model of material properties, for the purpose of data transfer, was explained [27, 28]. In [25], a data model for representing material heterogeneity and composition specifically for layered manufacturing was introduced. A comparative study on various material modeling techniques can be found in [27].

Most existing material modeling techniques rely implicitly or explicitly on distances to construct material fields [8, 43, 22, 15, 4]. The capabilities of manufacturing processes are also commonly described by their ability to modify material as functions of distance (determined either analytically or experimentally) [23, 11]. As a result, most material modeling applications may be reformulated in a terms of *distance fields* [4], which serve as convenient and intuitive parameters for specifying variation in material properties. Informally, a distance field for a given set S (geometry of a material feature) is defined by associating with every point p in the space a value that is equal to the shortest distance from p to S. The specific metric, used to measure the distance, depends on the space in which S is embedded, and the particular material modeling application. The Euclidean metric is the most widely used metric in engineering applications.

The above observations on distance fields lead to a unified and computationally convenient representational framework for continuously varying material properties [4]. Broadly, modeling of such properties involves three steps:

1. specifying values and rates of change in material fields at material features;
2. extending the specified values and rates with distance fields away from the feature; and
3. combining all specified functions with or without constraints.

Major differences between various representation methods may be studied in terms of how distance fields are represented, and how specified values and rates of changes at multiple features are combined, with or without constraints to construct global material fields [7, 43, 26, 29, 22, 15, 20, 19, 4].

The UML model of the CPM extension, proposed below, unifies all such representations into two broad categories: unevaluated and evaluated models of the material fields. This approach is based on recognition that in many ways heterogeneous material modeling is a generalization of classical solid (homogeneous) modeling. Solid modeling representations may be also classified into evaluated (usually combinatorial) representations, and unevaluated (usually implicit) representations [40]. A popular unevaluated representation is Constructive Solid Geometry (CSG), and the widely used evaluated representations include boundary representations and volumetric meshes [31]. It is reasonable to expect that heterogeneous material models must subsume and generalize such representations, and this view is reflected in the proposed UML extension of the CPM.

3 UML representation

3.1 Overview of material function construction

Generally speaking, a material field $F(\mathbf{x})$ can be written in terms of two components [4, 36]:

$$F = P + R, \tag{1}$$

where P is the explicit component satisfying the specified conditions at *all* features S_i, and R is a remainder term that satisfies specified constraints within the solid. The material field F can be constructed with either one of the components P or R separately, or as a combination of the two. Depending on material fields, the P and R components themselves may be scalar, vector or tensor. Vector- and tensor- valued material fields consist of a finite number of scalar components, with each scalar component constructed independently.

Figure 3 shows the classes of the CPM that are specialized to represent the material model. In this diagram, the specialized class **MaterialArtifact**, which represents material models, is constructed with multiple instances of the specialized class **MaterialFeature** (representing material features). The specialized class **DistanceGeometry** represents distance fields of material features and the class **HeterogeneousMaterial** corresponds to value and rates of change specified at material features. The rest of the section contains a detailed description of the UML model with figures. For clarity in explanation, classes of the CPM are shown in black, classes derived from the CPM in green, and classes, which do not belong to the above two categories, are in red. Appendix 5 shows the complete UML model.

By definition, specified values and rates of changes of P are known only at material features, and the field P is constructed by extending these specified conditions using distance fields associated with the material features. In principle, such specifications can be exact, at least until they are enforced computationally for a particular representation scheme and/or machine precision. In contrast, the remainder term R can be *evaluated* only within certain computational accuracy. In order to separate the intrinsic properties of material specifications from (approximately) computed material distributions, we propose to distinguish between two types of representations of material field: unevaluated and evaluated. The proposed material model consists of both of these two types of representation.

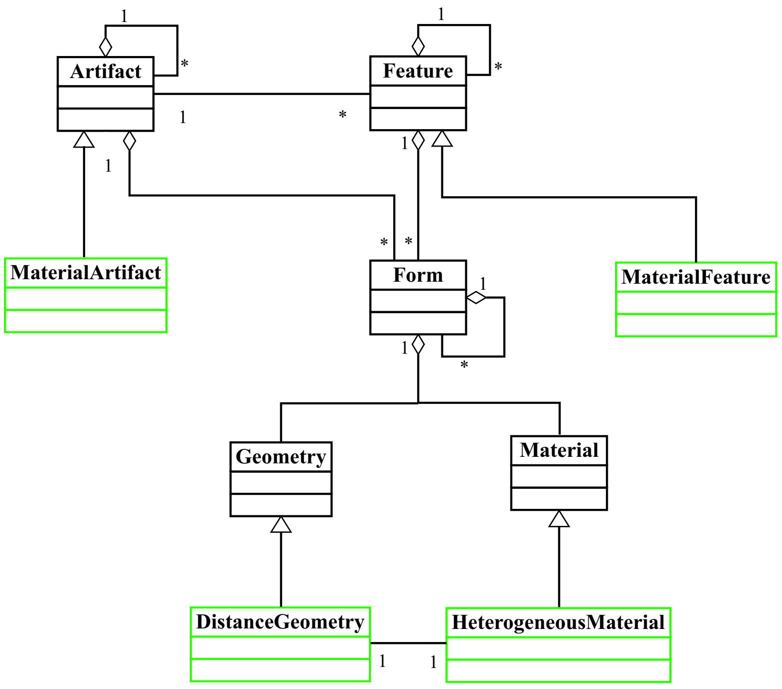


Figure 3: Classes of CPM are specialized to represent heterogeneous materials; and the derived classes are shown with green color.

3.2 Material model

3.2.1 Unevaluated material model

Material fields in the unevaluated form can be represented with two ingredients: specified material fields at material features, and global constraints on the material distribution. Figure 4, proposes the class diagram for the unevaluated representation.

In the diagram, the class **MaterialArtifact**, which captures the material model, specializes the class **Artifact** of CPM. The one to many relation between **Artifact** and **Feature** classes also signify a one to many relation between **MaterialArtifact** and **MaterialFeature**. At each **Form** of class **MaterialFeature**, a material field is explicitly specified as a function of distance u to the geometry of a feature. There can be any number of instances of **MaterialFeature**. The class **InfluenceParameters** associated to a material feature determines the influence with a feature at points away from it, when multiple features are specified. **MaterialArtifact** has an aggregation relation to class **Constraints**; which suggests that any number of constraint of differential, integral, and algebraic type can be specified on the material distribution. In the UML diagram, the classes **Differential**, **Integral**, and **Algebraic** are designated for different types of constraints, and these classes specialize the class **Constraints**.

Unevaluated material fields are theoretically exact, but sooner or later they must be evaluated. Hence, the description of allowable errors is an important part of the unevaluated material specification. The class **ApproximationParameters** represents parameters used to specify the desired quality of approximation in evaluating the material field from its unevaluated representation. The quality of evaluation is governed by two types of approximate computations: (1) satisfying the specified conditions at the material feature (measured by *Error_SpecifyFunction)*, and (2) enforcing various global constraints (up to *Error_Interpolation*) [4, 29]. The specification does not prescribe use of a particular numerical technique, but only the resulting quality of the evaluated representation. Theses errors can be measured with

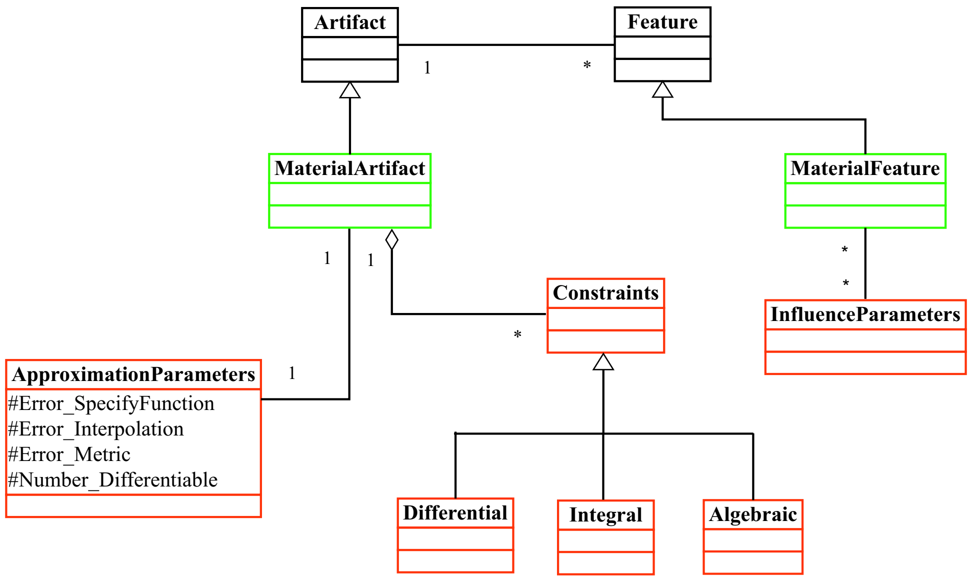


Figure 4: UML representation of unevaluated material model

different metrics, typically an L_p norm, as specified by *Error_Metric*. Finally, since smoothness of material fields is important in many applications, an unevaluated material specification must indicate the continuity class C^k to which the material field belongs using the *Number_Differentiable* parameter. All four parameters, *Error_SpecifyFunction*, *Error_Interpolation*, *Error_Metric*, and *Number_Differentiable* are included as attributes of the class **Approximation-Parameters**.

The unevaluated representation is sufficient for constructing evaluated (instances of) material fields, as we describe below. But unevaluated representations are also useful in their own right, for example in process planning, analysis, and comparison of material fields.

3.2.2 Evaluated material model

The evaluated representation of the material model includes an unevaluated representation material field F, together with specific instances of the material fields computed with the prescribed precision. Figure 5 shows the corresponding UML model for capturing an evaluated material field. The class **MaterialField** corresponds to the evaluated representation of material field, which is associated to the class **MaterialArtifact**. In the evaluated representation, material fields are constructed in the class **MaterialField** with the two fully evaluated components P and R. Accordingly, the class **MaterialField**, which specializes the class **MaterialArtifact**, has aggregate relation to two classes **Explicitly_Def_Field** (for P) and **Remainder_Term** (for R).

For a single material feature, the function P, represented by the **Explicitly_Def_Field** as shown in Figure 5, is determined as a function $P(u)$ of distance field u. We will explain how these functions may be constructed in terms of specified value and rates of changes indicated at the feature in section 3.3. In the case of multiple features, if explicitly constructed functions P^i, $i = 1 \dots n$ are known at all material features S_i, they can be combined to construct a single material field $P(u_1, u_2, \dots, u_n)$ using one of many interpolation techniques. In a typical material modeling problem, the feature S_i can be of any shape and dimension, and the constructed function P must be evaluated with P^i of every material feature S_i. This type of interpolation is known as transfinite interpolation [37, 21, 12]. As with most interpolation techniques, transfinite interpolation can be formulated as a convex combination of P^i as

$$P = \sum_{i=1}^{n} P^i(p) W_i(p), \; i = 1, \dots, n, \tag{2}$$

with each weight function W_i controlling the influence of the feature S_i of the material field. In transfinite interpo-

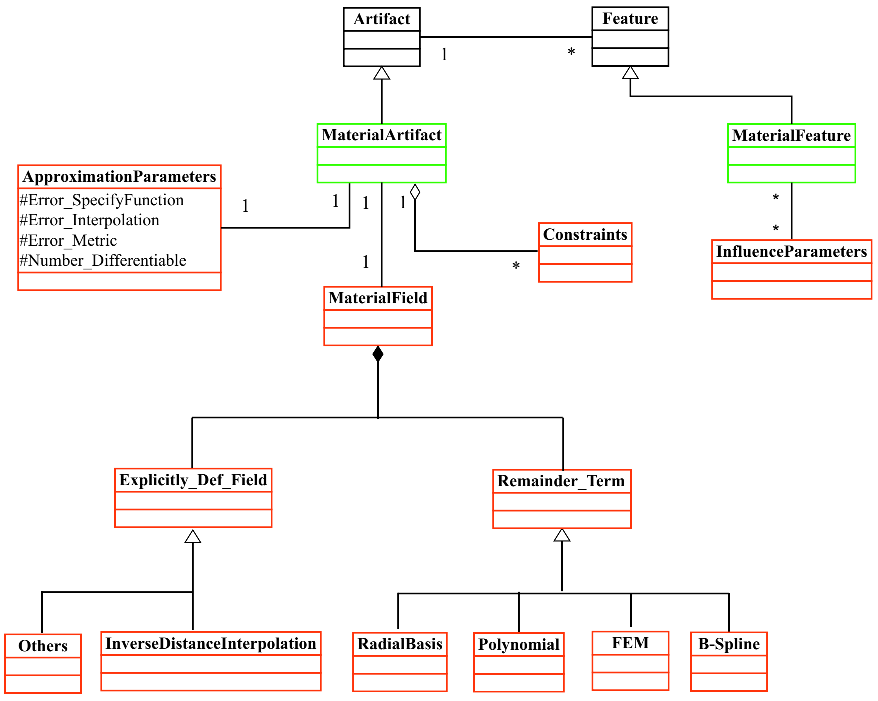


Figure 5: UML representation for evaluated material fields

lation, one of the most popular techniques for constructing W_i is to employ inverse distance interpolation [37]. In this case, weights W_i are inversely proportional to power of distance field u_i^{-k}. For details on the formulation of W_i and properties of the constructed function P refer to [4, 37, 12]. Further control on the interpolation is possible by scaling the inverse power term u_i^{-k} with $\lambda_i(u_i)$, where $\lambda_i > 0$ is the *influence function*[4, 21]. An attractive property of inverse distance interpolation is that, once the distance fields u_i are constructed, no further approximations or discretizations are needed. See [4] for additional details.

In principle, many other techniques can be employed to construct P [12], but virtually all such techniques involve additional approximations and require some form of spatial discretization. In the UML model for evaluated material fields, classes **InverseDistanceInterpolation** and **Others** specialize **Explicit_Def_Function** to represent material fields, constructed respectively with the inverse distance interpolation and other interpolation techniques. The influence function λ is part of the unevaluated material specification, and it is represented with the class **InfluenceParameters** in Figure 3.

In the UML model, the class **Remainder_Term** corresponds to the remainder term R that may be used to enforce, or approximate additional global constrains on the material field F. Such constraints may come in a variety of forms, including integral (e.g., bound on total material), differential (e.g., Laplacian smoothing), or algebraic (e.g. strict positivity or approximating another known function). If no such constraints are present, representation of the remainder term $R = 0$ is not required. When the remainder term is of order k, it has a general form of

$$R = u^k \sum_{i=1}^{n} C_i \chi_i$$

where u is the distance to the specified material features, and $\{\chi_i\}$, $i = 1, \ldots, m$ is a finite number of known basis functions from some theoretically complete space, such as polynomials, B-splines, and others. This representation is complete in the sense that it allows to represent any and all constraints on a material field [47]. Some popular choices of basis functions are reflected in the Figure 5, where the class **Remainder_Term** is specialized to **Polynomial**, **B_spline**, **FEM**, and **RadialBasis**.

3.3 Extending specified values and rates of changes with distances

Above, we tacitly assumed that when material values and rates of changes are prescribed on a single material feature S_i, the explicitly evaluated material field $P(u_i)$ can be constructed as a function of distance to the feature. In this section, we will explain how this may be accomplished and will introduce the UML model for explicitly specifying functions at each **Form** of a **MaterialFeature**. Figure 6 shows the corresponding UML diagram.

We have observed before that the distance field is the most widely accepted parameter for modeling material properties. In an ideal modeling framework any material field can be constructed, controlled and represented with the distance field u_i of a material feature S_i. A systematic method to construct an explicitly defined material field may be by specifying its value and rates of its change at S_i [4], and extending these specified value with distance field u_i of S_i.

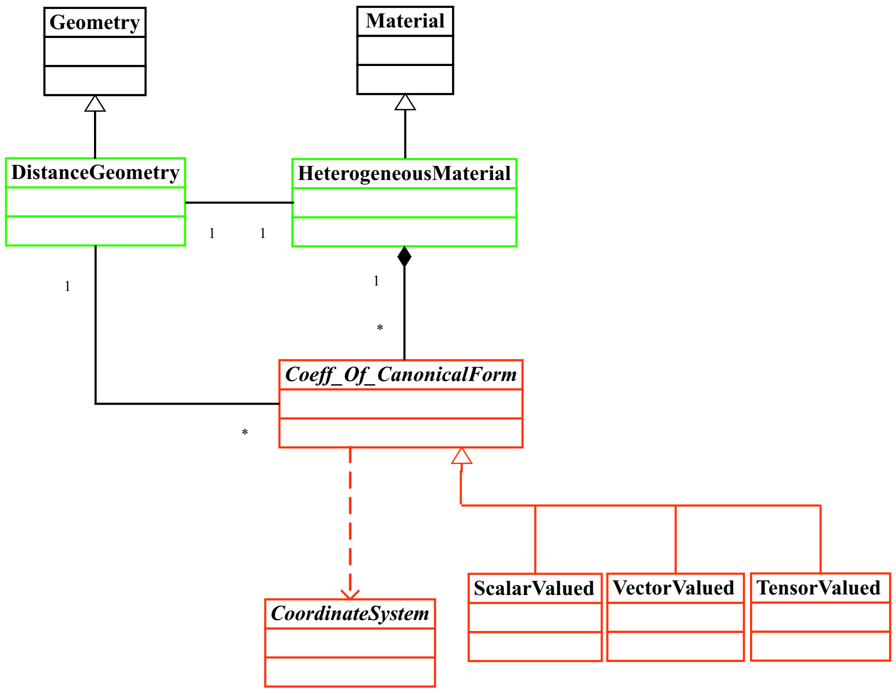


Figure 6: UML representation of explicitly specified functions at features.

Formally, in the usual Euclidean space, any *known* material field may be put in canonical form by expanding it in terms of powers of distances from the feature boundaries [4]. An explicitly specified component $P(u_i)$ of a scalar valued material field can also be represented at a point p as:

$$P(u_i) = F_0 + F_1 u_i + \frac{1}{2!} F_2 u_i^2 + \sum_{k=2}^{n} \frac{1}{k!} F_k u_i^k, \tag{3}$$

where the coefficient F_0 is the value of $P(u_i)$ at S_i, and coefficients F_i are the i-th order derivatives in the direction normal to the boundary ∂S_i. The distance canonical form, which is a generalization of classical Taylor series, is explained in detail in [4, 36].

The distance canonical form also provides a method for systematic constructions of functions $P(u_i)$ from the indicated material values and rates of changes. The method is essentially a syntactic composition of known functions into the canonical form. This technique for constructing material fields with the distance canonical form can be extended to vector valued, and tensor valued functions, by treating individual components as scalar. In the UML model in Figure 6, explicitly specified material fields are represented with the class **HeterogeneousMaterial**. The introduced link between the class **HeterogeneousMaterial** and the class **DistanceGeometry** specifies that the explicitly defined function is constructed with distance fields. The class **Coeff_Of_CanonicalForm**, which represents coefficients of the distance canonical form, is related to **HeterogeneousMaterial** with an aggregate relation. This aggregate relation specifies that an explicitly defined function can be constructed with one or multiple coefficients of the canonical form.

The coefficient F_i may be a constant, a function of position p, or a combination of the two. The constructed function $P(u_i)$ assumes the same value as F_0, and the k-th derivative of $P(u_i)$ in the direction normal to the boundary is equal to F_k. For constant F_ks, straightforward substitution of coefficients in the equation (3) is sufficient. But when coefficients themselves are functions of the coordinate of p, a simple coordinate transformation: $F_k^* \equiv F_k(p - u\nabla u)$ is also required in (3). In the UML model, a dependency relation between the class **Coeff_Of_CanonicalForm** and **DistanceGeometry** states that coefficients of the distance canonical form depends on the distance function. All classes **Scalar**, **Vector**, and **Tensor** specializes the class **Coeff_Of_CanonicalForm** to represent different types of material fields. Specified coefficients may depend on the class **CoordinateSystem**, which in turn represents the associate coordinate system.

The distance canonical form (3) assumes that the underlying metric of the distance field u_i is Euclidean. Generalization of the notion of distance, such as distance measured along a curve or over a surface may be useful in some applications. In addition to the Euclidean metric, the Riemannian metric may be employed to measure distance on surfaces. On a surface, the shortest distance between two points is measured along the geodesic, and can be used in a suitable generalization of distance canonical form. The general strategy for constructing the material field, as a combination of a explicitly specified functions with values and their rates of changes, remains the same and is captured by the diagram in Figure 6.

3.4 Distance fields of material features

We observed that distance fields to material features are the key to material modeling. In principle, the distance field is an intrinsic property of the feature geometry, and as such does not need to be specified or represented. However, there are at least two reasons why distance fields themselves are required to be represented within CPM: (1) there are different ways to measure distances, and (2) smooth approximations of distance fields are often used in place of the exact distance fields. Below, we propose UML classes for specification and representation of such distance fields, to be used as part of the material fields representation. Just like with material fields, the distance field may be specified in an unevaluated form or represented as fully evaluated functions of spatial variables.

3.4.1 Specification of unevaluated distance fields

Figure 7 shows the proposed UML model for distance fields. The class **DistanceGeometry** represents the distance field, which is a specialization of the class **Geometry** in the CPM. The underlying metric of the distance field is represented with the class **Metric**. All classes **Euclidean**, **Riemannian**, and **Others** specialize **Metric** to represent the corresponding Euclidean, Riemannian and user-defined methods for measure of distances. In general, any user-defined real valued function satisfying the well known properties [17] of a metric can be employed to construct distance fields. A metric for the distance field is usually selected based on the material property to be modelled.

The class **Approx_Dist_Field** has an association relation to the class **DistanceGeometry** in order to capture approximate distance fields. Approximate distance fields can be characterized with properties of normalization (*Order_Normalization*) [3], smoothness (*Number_Differentiable*), and deviation from exact distance functions (*Error_Distance*). These three parameters characterize approximation of the distance field locally (near the material feature) and globally.

The distance canonical form of the material field relies on the values and derivatives of the distance field in the direction normal to the boundary at all points of the material features. An approximate distance field may be used in place of the exact distance field, assuming that the two agree on the first m derivatives. Formally, we say that a field is normalized to the $m-$th order to indicate that the field takes on zero values on the set S, its first partial derivative in the direction normal to the boundary ∂S is 1, and all other derivatives up to order m are zero. The concept of normalization may also be extended to situations, where the normal direction may not be unique. (See [3] for more details.)

The smoothness of approximate distance fields can be characterized by the number of times functions are differentiable at points away from the zero sets. Note that exact distance fields are not differentiable at the points that are equidistant from material features. A smooth and normalized approximate distance field may substantially deviate from the exact distance field at points away from the boundary. Bounds on such deviations may be specified in terms of *Error_Metric* to measure *Error_Distance* as shown in the diagram.

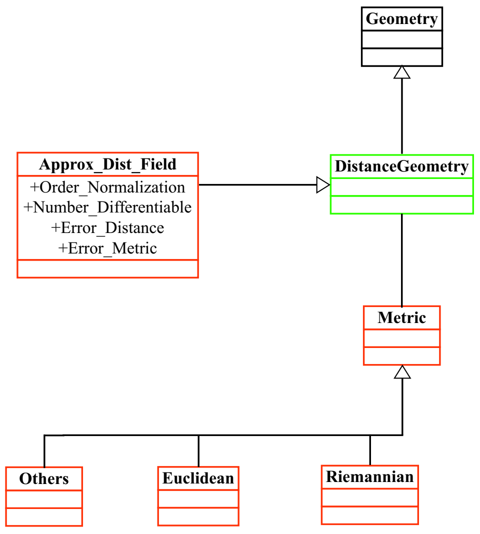


Figure 7: UML representation of the distance field

3.4.2 Evaluated distance fields

In the evaluated representation, the distance field of a material feature S is represented with a scalar valued function u. There may be many reasons for including evaluated representation of the distance fields in a material model. First, though the concept of distance field is well defined, computation of distance fields can be expensive for material features, particularly when they are represented using complex curves and surfaces. Furthermore, different systems may rely on substantially different approximations of distance fields, even if their normalization properties and error measures are similar. Figure 8 depicts a UML diagram for an evaluated distance field. In the UML model, the class **DistanceField** representing an evaluated distance field is associated to the class **DistanceGeometry**. Various techniques are available for representing evaluated distance fields [3]. Classes **Constructive**, **Approximated** and **PotentialFunction** all specialize the class **DistanceField**, and correspond to different methods of constructing and representing distance fields.

In **Constructive**, the distance field of any set S is constructed from the distance functions of its primitive sets. These primitives are obtained from well-formed CSG expressions and the boundary representation [32, 39] of S [41, 3]. Representation techniques in this category rely on R-functions [35, 39], Ricci functions [32] and Blending functions [34, 33]. Distance fields constructed with R-functions are exactly zero at S, can be as smooth as required, and are normalized up to a specific order [3]. The other two techniques, Ricci functions and Blending functions, are suitable for constructing distance fields of blends.

An approximation of the distance field can be constructed by interpolating a sampled distance field of a feature with basis functions such as polynomials, B-splines, radial basis, or other convenient basis functions [30, 14, 38, 46]. In Figure 8, the class **Approximated** represents distance fields which are obtained by interpolation. All classes **RadialBasis**, **FEM**, and **B_Spline** specialize the class **Approximated** to represent distance fields with various types of basis functions. The constructed function can be normalized in two ways: by introducing suitable constraints in the interpolation process or by applying the recursive transformation for constructing normalized functions [3]. Standard approximation methods allow control of the distance field's error, *Error_Distance*. Representation with sampled distance can be generalized to represent geodesic distance of Riemannian metrics on surfaces [18].

Another promising method for approximating distance fields relies on the observation that the reciprocal of a distance field to a set S may be approximated by a potential function (class **PotentialFunction**) as

$$f(\mathbf{x}) = \int_S \frac{1}{r(\mathbf{x})^n} dS, \tag{4}$$

where $r(\mathbf{x})$ measures the distance from point $\mathbf{x}$ to dS [1]. As n increases, the influence of points of S decreases with

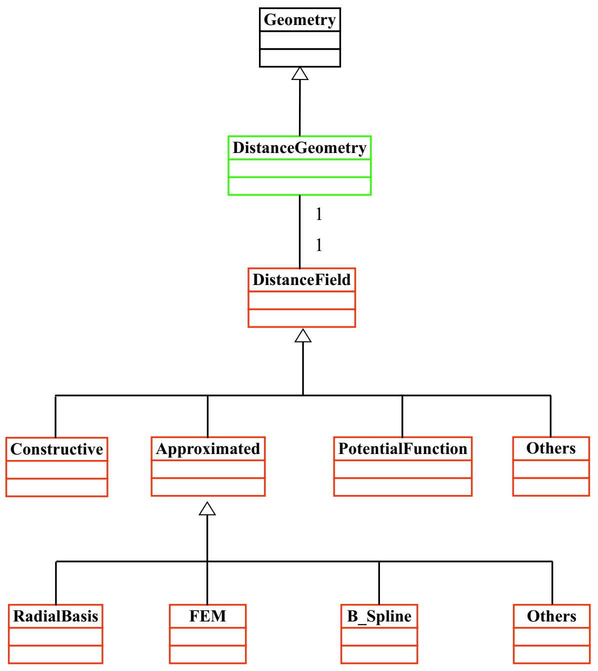


Figure 8: UML representation of the evaluated distance field

the increase in distance from S, giving a measure of distance as $n \to \infty$. In addition to $\frac{1}{r(\mathbf{x})^n}$, a number of other kernels, such as Gaussian, Cauchy, and polynomials can be employed [5, 42]. Potential and convolution fields share many attractive properties. The integral formulation may be convenient for parametric curves and surfaces and may result in closed form representations for some simple sets such as line segments and circular arcs [42]. Differential properties of the constructed functions $f(\mathbf{x})$ depend on the selected kernels. For example, fields constructed with polynomials and exponential kernels are flat on the boundary (that is, the first derivative normal to S vanishes).

4 Uses and examples

We demonstrate the proposed UML model for representing material heterogeneity using three examples from diverse application areas. The first example involves a material model of graded refractive index constructed with a single feature. The second example is from tissue engineering, which relies on multiple material features. Finally, the last example is a heterogeneous turbine blade constructed with multiple material features, and additional constraints. In each case, we describe the material model and show how it may be represented in terms of the proposed UML model.

4.1 Y-shaped solid with GRIN

The three-dimensional example in Figure 9, shows a parabolic distribution of the graded refractive index (GRIN) within the Y-shaped solid. Many applications of GRIN are found in optical fibers for communication and precision lenses for microoptics [23, 16]. In Figure 9, the material feature is the Y-shaped, one-dimensional skeleton constructed as the union of three axes of the cylinders. Expressed in the distance canonical form (3), the refractive index inside the solid is prescribed to vary as $(0.02 + 0.0u - 50u^2)$, where u is the distance field of the feature. That ensures that the value of the refractive index will be equal to .02 at the feature and the second order rate of change in the direction normal to the boundary is -50. Here the material field F is constructed only with the explicit component P. Figure 9(a) shows the resulting distribution of refractive index constructed with a distance field normalized up to the order 12. Figure 9(b) shows a magnified planar section of the distribution, which is almost as uniform as with the exact distance field, but is also smooth everywhere including the points that are equidistant from the axes of the material feature.

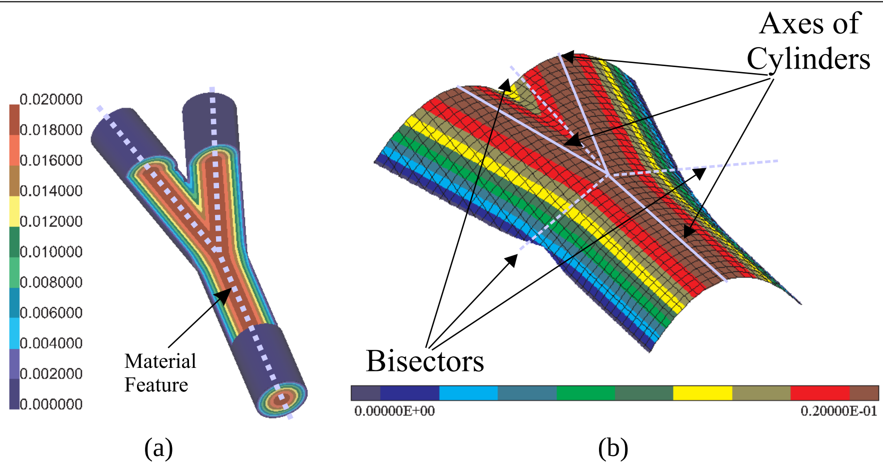


Figure 9: Refractive index distribution of a radial GRIN Y-branching waveguide for optical communications. (a) Distribution $(0.02-50u^2)$ is specified explicitly for the Y-shaped one-dimensional skeleton. (b) Magnified distribution plotted as a surface over a 2D section: analytic everywhere away from the feature, including the points on bisectors where the exact distance field would not be differentiable.

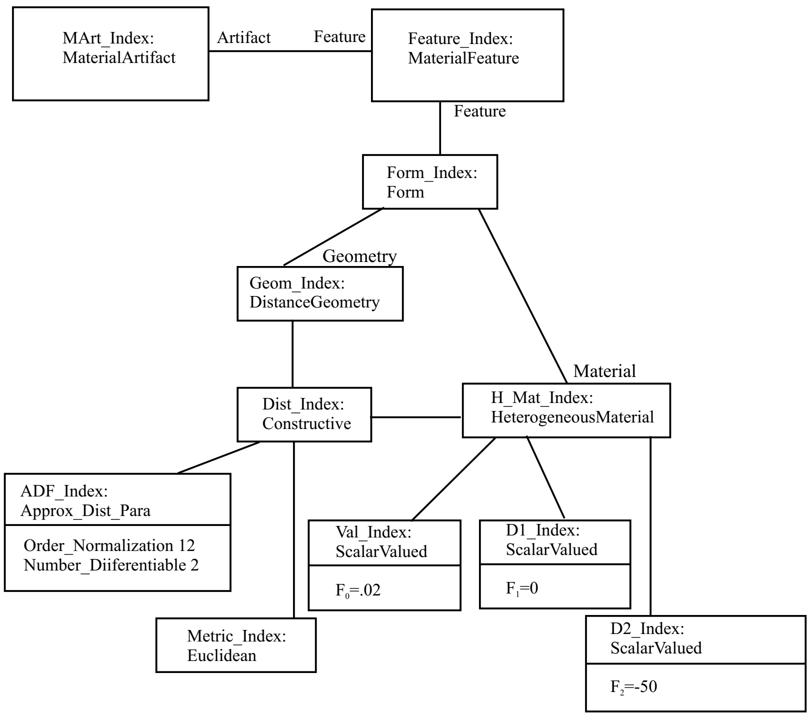


Figure 10: Object diagram representing a material field constructed with values specified at a single feature in the refractive index modeling example.

Figure 10 shows the object diagram of the material model in Figure 9 representing the distribution of refractive index. In the object diagram, object *MArt_Index* is an instance of **MaterialArtifact**. The object *MArt_Index* has a link to the object *Feature_Index* representing a single instance of class **MaterialFeature**. As the material field is constructed with a single feature and no additional constraint was specified, the material model does not involve any evaluated representation. *Feature_Index* has a link to the object *Form_Index* of class **Form**. The *Geom_Index* object, which is an instance of class **DistanceGeomtry**, represents specified geometry and an unevaluated distance field. The geometric representation of the feature, containing parameters such as lengths and split angles, is stored in the object *Geom_Index*. The object *ADF_Index*, which is linked to the object *Geom_Index*, contains specified parameters for characterization of the approximate distance field: order of normalization (*Order_Normalization*=12), and the order of differentiability (*Number_Differentiable*=2). The object *Dist_Index* represents the approximate distance field of the Y-shaped feature constructed by joining the distance fields of three line segments with the constructive technique (R-functions method), that was mentioned in the previous section. With the specified scalar valued coefficients of distance canonical form, the material field is constructed in object *H_Mat_Index*, which is an instance of class **HeterogeneousMaterial**. Object *H_Mat_Index* is linked to objects *Val_Index*, *D1_Index*, and *D2_Index*, which are instances of class **ScalarValued**. Object *Val_Index*, *D1_Index*, and *D2_Index* correspond to coefficients F_0, F_1, and F_2 of distance canonical form and the values specified at the three objects are 0.02, 0, and -50 respectively.

4.2 Tissue modeling

In tissue engineering, a computer model of a tissue for an implant is constructed to create a biologically inspired design that might accelerate wound healing. For example, in Figure 11(a), a segment of human skull is modeled with spatial pattern of two growth factors and the density of fibrin [48]. The dimension independent density factor of fibrin typically decreases smoothly with distance, from 1.0 at the outer surface of the skull to 0.1 at the inner surface. Figure 11 shows the two features involved in the construction and the constructed density distribution within a piece of the skull. The material field F for the skull tissue is constructed only with the explicit component P. The outer and inner surfaces of skull were treated as two separate material features with fibrin density values 1.00 and 0.1 respectively. At the outer feature an influence parameter $e^{(-100u)}$ was specified to diminish the influence of the outer feature away from it. Inverse distance interpolation between the two features with two distance functions results in a smooth density distribution of fibrin between the two features. As no additional constraint was specified on the distribution, no instance of the class **Remainder_Term** was necessary in the object diagram.

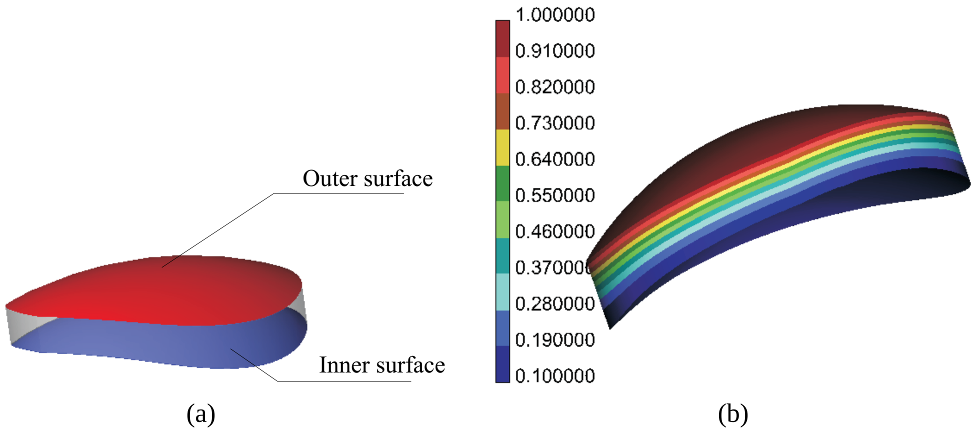


Figure 11: (a) Distribution of the density of fibrin in human skull can be modeled with the outer and the inner surface as two features. (b) shows the distribution of density within a piece of a skull.

Object diagrams of this model are shown in the Figures 12 and 13. Object *MArt_Tissue* is an instance of class **MaterialArtifact**, which is linked to objects *Out_Tissue* and *In_Tissue*, which are in turn instances of the class **MaterialFeature**. Requirements of the material fields are that the allowable error in specifying values *Error_SpecifyFunction* at features is equal to 0 and the field is required to be twice differentiable. The object diagram also introduces an evaluated representation of material field and *Mf_Tissue*, which captures the evaluated representation of the material model. The object *MArt_Tissue* is linked to the object *Mf_Tissue* and the object *Mf_Tissue* in turn is linked to the object *InvDistInter_Tissue*, an instance of the class **InverseDistanceInterpolation**. In the object *InvDistInter_Tissue*, values specified at two features are combined in order to construct the explicit component of material field.

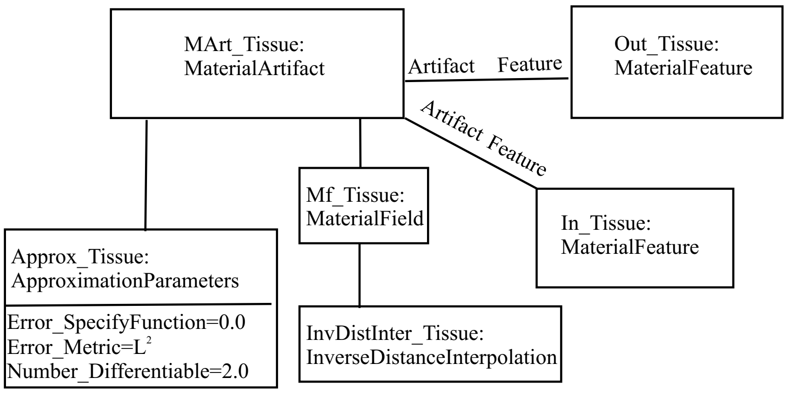


Figure 12: Object diagram representing material field constructed with value and rates specified at two features in the tissue modeling example.

Figures 13 (a) and (b) show objects associated with the two features. In this case, exact distance fields were used to construct the density distribution because they do not introduce any discontinuity within the modeled tissue. *In_Geom_Tissue*, an instance of the class **DistanceGeometry**, captures geometry and unevaluated distance field. The value specified at the feature *In_Tissue* is .1 and it is represented with the object *In_Val_Tissue*. At the feature *Out_Tissue*, influence function e^{-100u} is specified with the object *Out_Influence_Tissue*. The exact distance function of the feature is represented with the object *Out_Geom_Tissue*, which is again an instance of class **DistanceGeometry**. The specified value 1 is captured in object *Out_val_Tissue*, which is linked to the object *Out_H_Mat_Tissue*.

4.3 Turbine blade

Figure 14 shows a 2D sectional view of a heterogeneous turbine blade and the constructed distribution of thermal conductivity within it [45]. The heterogeneous turbine blade has a coating of relatively higher conductive ceramic and a titanium insert of low conductivity but of higher strength. The rest of the material is made of smoothly graded heterogeneous material. Two material features employed in modeling the conductivity are: the outer surface and the insert, as shown in Figure 14(a). The values of material field F specified at the outer feature and the insert are 150 and 7 respectively, and influence parameter $\lambda_o = e^u$ is specified at the outer feature to reduce the influence of the outer surface at points away from it. Apart from the specified values and influence parameter, a differential constraint requiring to minimize ∇F was specified to obtain a smooth distribution of conductivity. In this case, the material field F for the turbine blade is constructed with an explicit component P and the remainder term R. The explicit component P is constructed by interpolating the specified material values at the two features with inverse distance interpolation. The remainder term R, which enforces the prescribed differential constraint on F, is evaluated numerically on a non-conforming grid of B-splines. Figure 14(b) shows the conductivity F obtained by combining the two terms.

The object diagrams in Figure 15 and Figure 16 capture the above construction of the conductivity distribution within the turbine blade. The object *MArt_Turbine* corresponds to the class **MaterialArtifact** in the CPM. As the material field is constructed with two material features, object *MArt_Turbine* has links to the two instances of class **MaterialFeature**, *Out_Feature_Turbine* and *In_Feature_Turbine*. Apart from the conditions on two features a differential constraint minimizing ∇F was also specified on the material distribution. The object *Constr_Turbine*, an instance of class **Differential**, corresponds to the specified differential constraint and it is linked to the object *MArt_Turbine*. The specified differential constraint is required to be evaluated numerically. The object *Approx_Turbine* which is an instance of class **ApproximationParameters** has all the approximation parameters required to evaluate the material field from the unevaluated representation. The material field is evaluated in object *MF_Turbine*; the explicit component

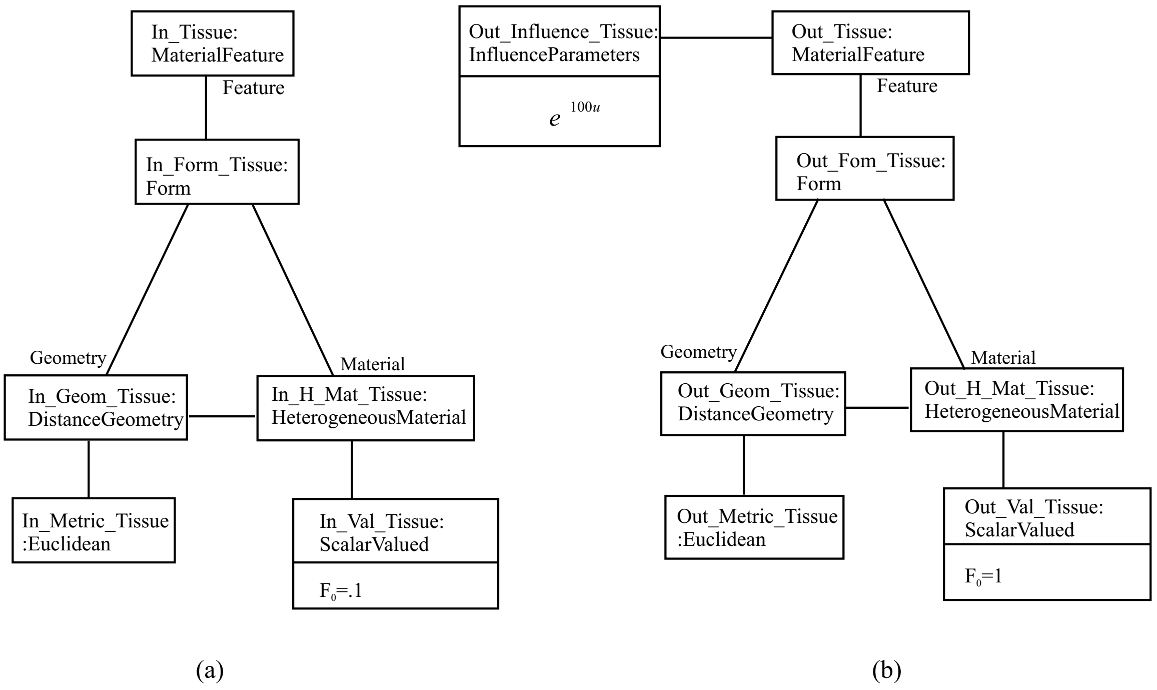


Figure 13: Diagrams (a) and (b) show objects associated to the feature *in_Tissue* and *out_Tissue* respectively for the tissue modeling example.

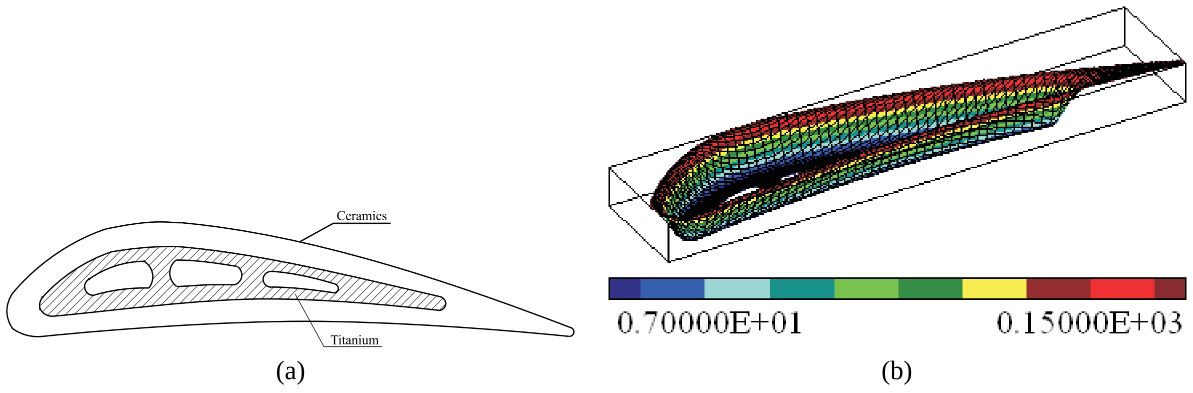


Figure 14: (a) Sectional view of a heterogeneous turbine blade consisting of two features. Plot in (b) corresponds to the distribution of material conductivity obtained by interpolation with differential constraint.

is captured in the object *Interpol_Turbine* and the remainder term is captured with the object *Remainder_Turbine*. In object *Interpol_Turbine*, values specified at features are combined with inverse distance interpolation. Inverse distance interpolation guarantees that values specified at features will be satisfied exactly. The remainder term is represented with bicubic B-splines defined over a 60×60 grid. Figure 16 shows geometry and values specified at the two features *Out_Feature_Turbine* and *In_Feature_Turbine*. The approximate distance field of *In_Feature_Turbine* is captured by the object *In_Dist_Turbine* and the field is represented with the constructive technique, R-functions methods. The distance field of this feature is normalized to the order 2 and is smooth up to the same order. The object *In_Val_Turbine* represents the scalar value specified at the feature *In_Feature_Turbine* and the specified value is 7. Similarly, distance field of feature *Out_Feature_Turbine* is also constructed with the constructive technique, and it is also smooth and normalized to the second order. This distance field of this feature is represented in the object *Out_Dist_Turbine*. The scalar value specified at the feature is 150 and it is captured with the object *Out_Val_Turbine*.

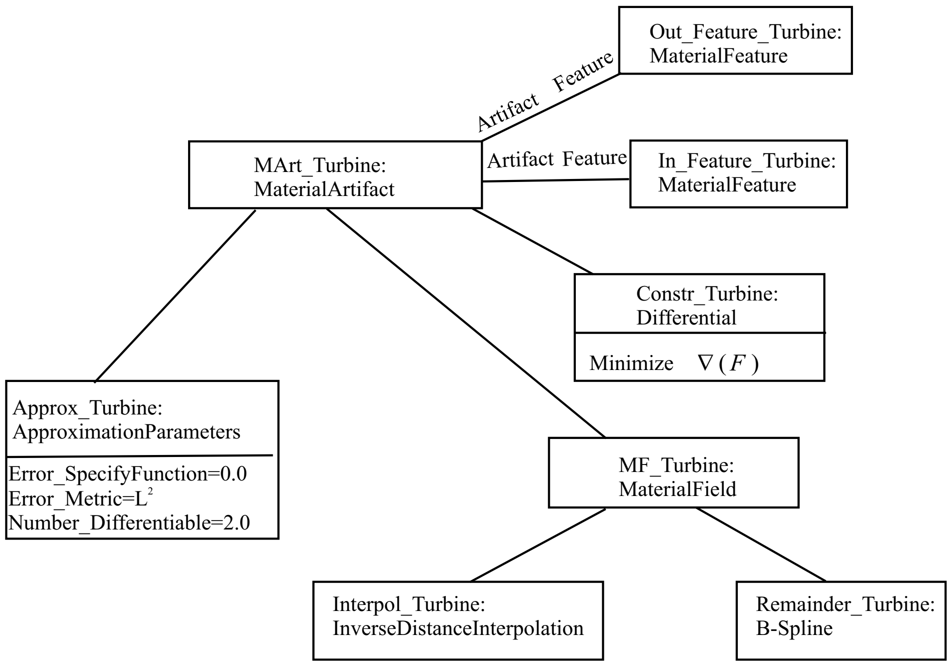


Figure 15: Objects associated to the instance of **MaterialArtifact** class for the turbine blade example, introduced in the Figure 14.

5 Conclusions

The proposed UML extension of CPM to include heterogeneous material modeling is complete in the sense that it applies to any and all types of material fields [4]. In this report, we fleshed out the details of representing typical scalar-valued material properties. Extensions to vector- and tensor-valued material properties, anisotropy, and periodic structures are relatively straightforward. For example, the proposed technique is used in [4] to model multi-material composition, such as used for free-form fabrication, under the constraint that the volume fractions must add to unity. The critical feature of the proposed model is that it subsumes most proposed methods for modeling continuously varying material properties, including those using finite-element methods, triangulations, implicit functions, distances, meshless, and meshfree representations.

The current model was developed to support material model construction, material related queries, data transfer, and model comparison. The construction process is relatively straightforward, involving the definition of material features, and choosing properties for distance fields, influence coefficients, constraints, and basis functions. Material

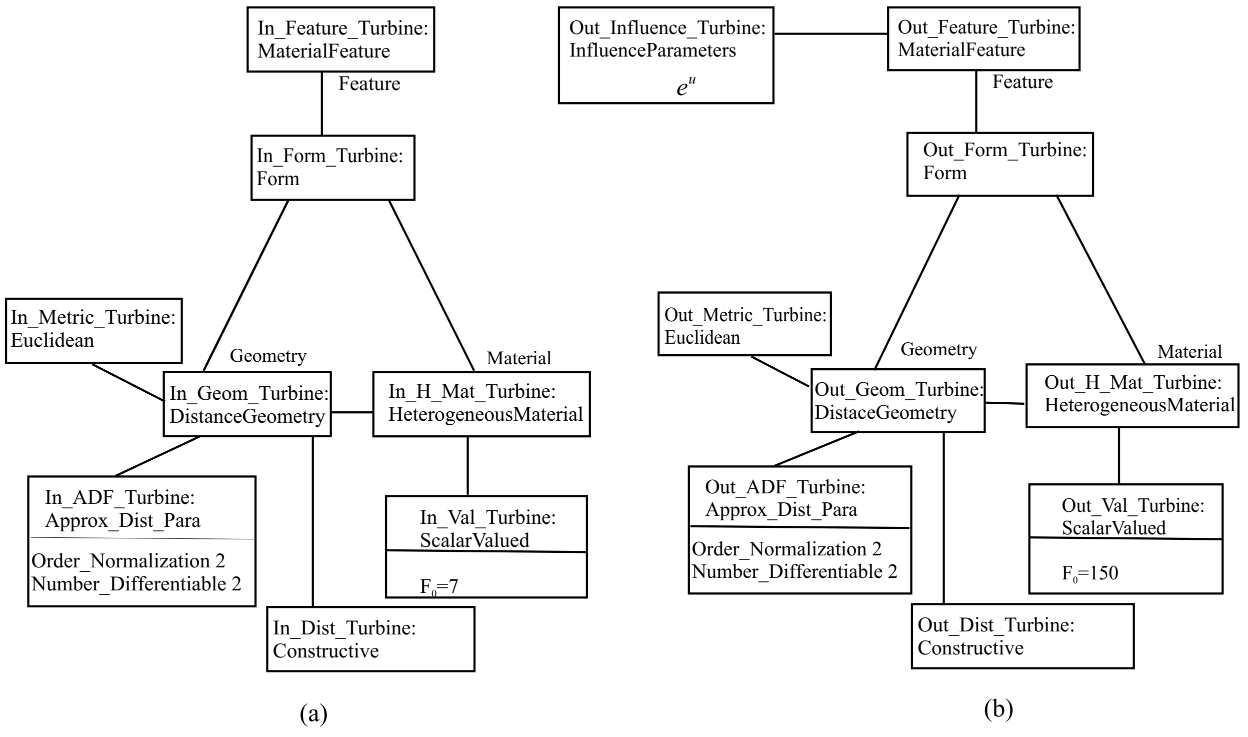
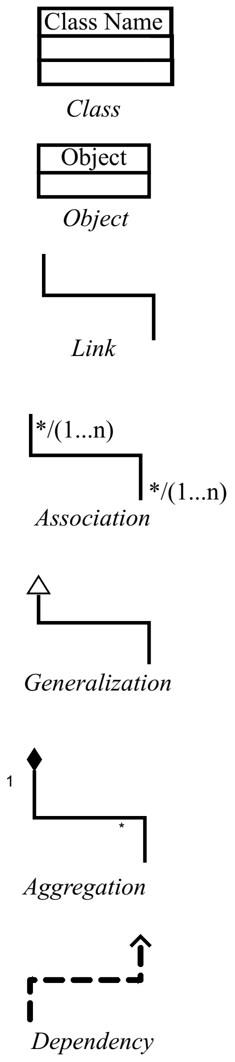

In_Feature_Turbine: MaterialFeature
Feature
In_Form_Turbine: Form
In_Metric_Turbine: Euclidean
Geometry
Material
In_Geom_Turbine: DistanceGeometry
In_H_Mat_Turbine: HeterogeneousMaterial
In_ADF_Turbine: Approx_Dist_Para
Order_Normalization 2
Number_Differentiable 2
In_Val_Turbine: ScalarValued
F0=7
In_Dist_Turbine: Constructive
(a)
Out_Influence_Turbine: InfluenceParameters
e^u
Out_Feature_Turbine: MaterialFeature
Feature
Out_Form_Turbine: Form
Out_Metric_Turbine: Euclidean
Geometry
Material
Out_Geom_Turbine: DistaceGeometry
Out_H_Mat_Turbine: HeterogeneousMaterial
Out_ADF_Turbine: Approx_Dist_Para
Order_Normalization 2
Number_Differentiable 2
Out_Val_Turbine: ScalarValued
F0=150
Out_Dist_Turbine: Constructive
(b)

Figure 16: Objects associated to the two material features (a) *In_Feature* and (b) *Out_Feature* for the turbine blade example.

queries are supported through the usual geometric queries (specifically, point and set membership classification) followed by evaluation of the constructed material properties over the indicated set of points. Data transfer is supported in both unevaluated and fully evaluated form, and model comparison reduces to comparison of the objects in individual material models. Comparison of unevaluated models reduces to comparison of the specified parameters; comparison of fully evaluated fields reduces to classical problems of field comparison. A fully evaluated field may be compared with an unevaluated specification via appropriate field construction and/or evaluation procedures as described in this report. Of course, data transfer and comparison of fields assume the ability to perform these tasks for individual classes in the CPM model, and for all geometric models in particular.

Appendix



A *class* is a description of a set of object that share the same attributes, operations, relationships, and semantics.

An *object* is an instance of a class.

A *link* is a connection among objects.

An *association* is a structural relationship that describes a set of links among classes.

A *generalization* is a specialization/generalization relationship in which objects of the specialized element (the child) are substitutable for objects the generalized element.

An *aggregation* is a special kind of link (association), representing a structural relationship between a whole and its parts.

A *dependency* is a semantic relationship between two things in which a change to one thing (the independent thing) may affect the semantics of the other thing (the dependent thing).

Figure 17: UML notations, which are used in this paper.

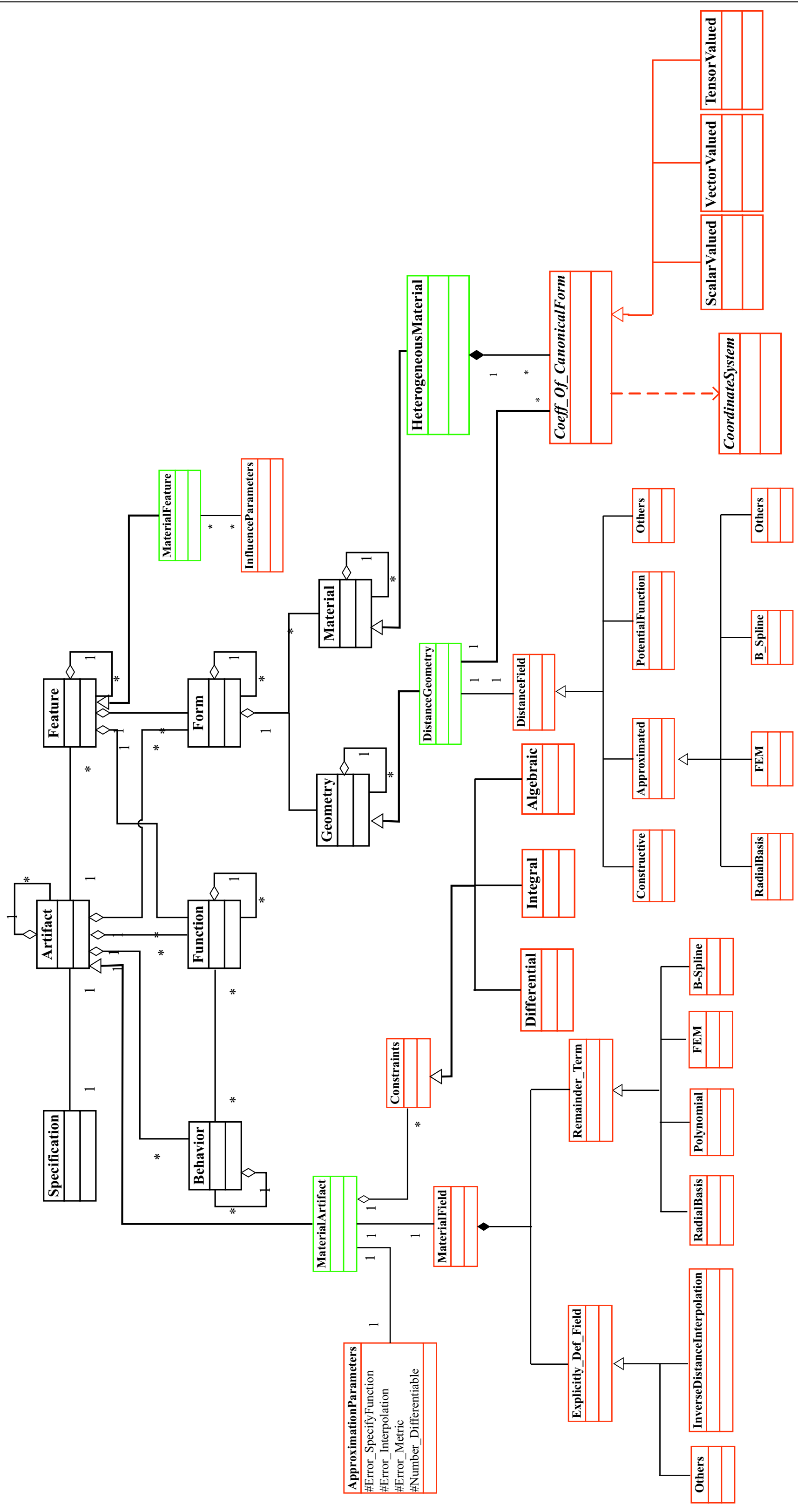


Figure 18: UML representation of the proposed material model.

Acknowledgments

This research is supported in part by the National Science Foundation grants DMI-0323514 and DMI-0115133, and the National Institute of Standards (NIST) grant 60NANB2D0126. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. No approval or endorsement of any commercial product by the National Institute of Standards and Technology or by University of Wisconsin-Madison is intended or implied.

References

[1] Narendra Ahuja and Jen-Hui Chuang. Shape representation using generalized potential field model. *IEEE Transaction Pattern Analysis and Machine Intelligence*, 19(2):169–176, February 1997.

[2] M. Bendsoe and N. Kikuchi. Generating optimal topologies in structural design using homogenization method. *Computer Methods in Applied Mechanics and Engineering*, 71:197–224, 1988.

[3] A. Biswas and V. Shapiro. Approximate distance fields with non-vanishing gradients. *Graphical Models*, 66(3):133–159, 2004.

[4] Arpan Biswas, Vadim Shapiro, and Igor Tsukanov. Heterogeneous material modeling with distance fields. *Computer Aided Geomery Design*, 22(3):215–242, 2004.

[5] J. Bloomenthal. Bulge elimination in implicit surface blends. In *Implicit Surfaces'95*, pages 7–20, Grenoble, France, 1995.

[6] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Massachusetts, 1999.

[7] V. Chandru, S. Manohar, and C. Prakash. Voxel-based modeling for layered manufacturing. *IEEE Computer Graphics and Applications*, 27:42–47, 1995.

[8] W. Cho, E. M. Sachs, N. M. Patrikalakis, M. Cima, H. Liu, J. Serdy, and C. C. Stratton. Local composition control in solid freeform fabrication. In *Proceedings of the 2002 NSF Design, Service and Manufacturing Grantees and Research Conference*, San Juan, Puerto Rico, January 2002.

[9] S. J. Fenves. A core product model for representing design information. Internal Report NISTIR 6736, National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[10] Asish Ghosh, Yoshinari Miyamoto, and Ivan Reimanis. Functionally graded materials: Manufacture, properties and applications. *Ceramic Transactions*, 76, July 1997.

[11] J. B. Holt, M. Koizumi, T. Hirai, and Z. A. Munir, editors. *Ceramic Transactions*, volume 34. The American Ceramic Society, Westerville, Ohio, 1993.

[12] J. Hoscheck and D. Lasser. *Fudamentals of Computer Aided Geometric Design*. A K Peters, 1993.

[13] International Organization for Standardization (ISO), Geneva, Switzerland. *ISO 10303-1:1994, Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 1: Overview and Fundamental Principles*, 1994.

[14] P. Brunet J. Esteve and A. Vinacua. Multiresolution for algebraic curves and surfaces using wavelets. *Computer Graphics Forum*, 20(1):47–58, 2001.

[15] T. R. Jackson. *Analysis of Functionally Graded Material Object Representation Methods*. PhD thesis, MIT, Ocean Engineering Department, June 2000.

[16] Y. Koike. Graded index materials and components. In L. A. Hornak, editor, *In Polymers For Lightwave and Integrated Optics*. Marcel Dekker, Inc., New York, 1992. Special contribution to this book.

[17] A. N. Kolmogorov and S. V. Fomin. *Introductory Real Analysis*. Dover Publication Inc, 1975.

[18] G. V. V. Ravi Kumar, Prabha Srinivasan, V. D. Holla, K. G. Shastry, and B. G. Prakash. Geodesic curve computations on surfaces. *Computer Aided Geometric Design*, 20(2):119–133, 2003.

[19] V. Kumar, D. Burns, D. Dutta, and C. Hoffmann. A framework for object modeling. *Computer-Aided Design*, 31:541–556, 1999.

[20] V. Kumar and D. Dutta. An approach to modeling and representation of heterogeneous objects. *ASME Journal of Mechanical Design*, 120:659–667, 1998.

[21] Peter Lancaster and Kestutis Salkauskas. *Curve And Surface Fitting: An Introduction*. Academic Press, 1986.

[22] Hongye Liu, Wonjoon Cho, Todd R. Jackson, Nicholas M. Patrikalakis, and Emanuel M. Sachs. Algorithms for design and interrogation of functionally gradient material objects. In *Proceedings of ASME 2000 IDETC/CIE 2000 ASME Design Automation Conference*, Baltimore, MD, 2000.

[23] Yoshinari Miyamoto, W. A. Kaysser, and B. H. Rabin, editors. *Functionally Graded Materials: Design, Processing and Applications.* Kluwer Academic Publishers, Boston, 1999.

[24] J. Owen. *STEP: An Introduction*. Information Geometer, Wichster, 1997.

[25] L. Patil, D. Dutta, A. D. Bhatt, K. K. Jurrens, K. W. Lyons, M. J. Pratt, and R. D. Sriram. Representation of heterogeneous objects in iso 10303. In *ASME International Mechanical Engineering Congress and Exposition*, Orlando, FL, November 2000.

[26] J. Pegna and A. Safi. CAD modeling of multi-model structures for freeform fabrication. In *In Proceedings of the 1998 Solid Freeform Fabrication Symposium*, Austin, TX, August 1998.

[27] M. J. Pratt. Modelling of material property variation for layered manufacturing. In *Mathematics of Surfaces IX*, Cambridge, UK, 2000.

[28] M. J. Pratt, A. D. Bhatt, D. Dutta, K. W. Lyons, L. Patil, and R. D. Sriram. Progress towards an international standard for data transfer in rapid prototyping and layered manufacturing. *Computer-Aided Design*, 34(1):1111–1121, December 2002.

[29] Xiaoping Qian and Debasish Dutta. Physics based B-Spline heterogeneous object modeling. In *ASME Design Engineering Technical Conference*, Pittsburgh, September 2001.

[30] Alon Raviv and Gershon Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 246–257. ACM Press, 1999.

[31] A. A. G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, December 1980.

[32] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, May 1973.

[33] A. P. Rockwood. The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics (TOG)*, 8(4):279–297, 1989.

[34] Alyn P. Rockwood and John C. Owen. Blending surfaces in solid modeling. In Gerald E. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 367–383. SIAM, Philadelphia, 1987.

[35] V. L. Rvachev. *Theory of R-functions and Some Applications*. Naukova Dumka, 1982. In Russian.

[36] V. L. Rvachev, T. I. Sheiko, V. Shapiro, and I. Tsukanov. On completeness of RFM solution structures. *Computational Mechanics*, 25:305–316, 2000.

[37] V. L. Rvachev, T. I. Sheiko, V. Shapiro, and I. Tsukanov. Transfinite interpolation over implicitly defined sets. *Computer Aided Geometric Design*, 18:195–220, 2001.

[38] Vladimir V. Savchenko, Alexander A. Pasko, Oleg G. Okunev, and Tosiyasu L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.

[39] V. Shapiro. Real functions for representation of rigid solids. *Computer-Aided Geometric Design*, 11(2):153–175, 1994.

[40] V. Shapiro. Solid modeling. In G. Farin, J.Hoschek, and M.-S. Kim, editors, *Handbook of Computer Aided Geometric Design*. Elsevier Science Publishers, 2002.

[41] V. Shapiro and I. Tsukanov. Implicit functions with guaranteed differential properties. In *Fifth ACM Symposium on Solid Modeling and Applications*, Ann Arbor, MI, 1999.

[42] Andrei Sherstyuk. Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer*, 15(4):171–182, 1999.

[43] Y. K. Siu and S. T. Tan. Source-based heterogeneous solid modeling. *Computer-Aided Design*, 34:41–55, January 2002.

[44] S. Suresh and A. Mortensen. *Fundamentals of Functinally Graded Materials*. The University Press, UK, 1997.

[45] I. Tsukanov and V. Shapiro. Meshfree modeling and analysis of physical fields in heterogeneous media. *Advances in Computational Mathematics*, 2003. accepted for publication.

[46] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 1999.

[47] V.L.Rvachev, T.I.Sheiko, V.Shapiro, and I.Tsukanov. On completeness of rfm solution structure. *Computational Mechanics Journal on Meshfree Methods*, 1999.

[48] L.E. Weiss, C.H. Amon, S. Finger, E.D. Miller, D. Romero, I. Verdinelli, L.M. Walker, and P.G. Campbell. Bayesian computer-aided experimental design and freeform fabrication of heterogeneous matrices for tissue engineering applications. *Computer Aided Design*, 2004. to appear.