A Generic Taxonomy for Defining Freeform Feature Classes

Paulos J. Nyirenda[1], Willem F. Bronsvoort[1], Thomas R. Langerak[2], Yu Song[2] and Joris S.M. Vergeest[2]

Delft University of Technology
[1]Faculty of Electrical Engineering, Mathematics and Computer Science, p.nyirenda@ewi.tudelft.nl
[2]Faculty of Industrial Design Engineering

ABSTRACT

Feature modeling has simplified, improved and accelerated the Computer-Aided Design (CAD) process. However, this modeling paradigm still has some serious shortcomings. Firstly, its domain is currently restricted to regular and simple freeform shapes. Secondly, most current feature modelers do not allow new class definitions, but instead provide design with predefined classes. In this paper, we propose a method to solve these shortcomings as follows. The shape domain is extended to include more general freeform features. To achieve this, a generic freeform feature taxonomy is developed. New feature classes can then be defined by deriving from this taxonomy, at any level of abstraction per design intent, through an object-oriented approach. In particular specification of class parameters and constraints, in the new classes, is demonstrated.

Keywords: feature modeling, freeform features, generic taxonomy, user-defined classes, validity maintenance.

1. INTRODUCTION

Feature modeling involves creating product models through high-level entities known as features. A feature is defined as a representation of shape aspects of a product that are mappable to a generic shape and functionality significant for some product life-cycle phase [1]. The advantage of a feature model over a geometric model is its property of associating functional information to shape information in the product model. In essence, a feature has a well-defined meaning, or semantics, for a particular life-cycle activity. It can comprise, not only shape, but also other properties such as the function of some shape aspect for the end-user, or the way some shape aspect can be manufactured. As a result, feature modeling simplifies, improves and speeds up the CAD process. While conventional CAD systems, which are pure geometric modelers, only refer to geometric information to create shape models, using features offers the possibility to add more information, in particular semantics, to the model. A feature is fully defined by its *form* and its *semantics*. The semantics provides additional information about usage, technology, associations between several geometric entities, constraints, etc, all in the *feature model*. However, current feature modeling systems still suffer some serious shortcomings described in the following paragraphs.

Firstly, the domain of current systems is restricted to regular and simple freeform shapes [2]. Related to this, attempts to support features still lack a high-level representation of the features. Instead, a geometric representation is used for the product model, and features only occur at the user-interface level, and their meaning is not adequately maintained in the modeling process [1].

Secondly, current systems do not adequately allow a user to define new feature classes. In a modeling process, a product model results from a combination of several features by means of parametrising, transforming and connecting these features. Design systems must support not only such use of existing features, but also the definition of new features. Definitions should be supported, in addition, at various levels of abstraction. Instead, current modeling systems usually offer design with predefined features, often targeting specific design problems, e.g. aesthetic design problems only.

This paper presents a generic approach that will support specification of a variety of freeform feature classes. A generic freeform feature taxonomy is developed from which new generic, high-level parametric feature classes can be derived. Defining these classes involves specifying parameters and constraints for its shape. By adding constraints, typically dictated by technical conditions, the feature can be seen as an intelligent design object, which facilitates the expression of design intent in the product. Utilizing the defined semantics of features, together with other relevant aspects, feature model validity maintenance then becomes feasible throughout the design cycle.

Section 2 outlines research related to user-defined features as well as freeform feature taxonomies. Then, the need for a generic taxonomy is discussed in Section 3, followed by the generic freeform feature taxonomy in Section 4. Section 5 presents the specification of new freeform feature classes, and finally conclusions are drawn in Section 6, with prospects for future work.

2. RELATED RESEARCH

Feature technology has generally shown to be an intuitive means of solving recurrent design problems, by taking advantage of existing solutions. This implies that basic elements are reused in new products. The process of detailing a design fully specifies the elements so that the recurrent solutions have to be defined as parametric objects. These recurrent solutions are thus usable for different product developments. In a feature-modeling environment, it is therefore appropriate and useful to store features in libraries. These features do not have fixed parameter values, or, in other words, they are generic. Whenever design uses such objects, their parameters and constraints have to be specified by values to instantiate specific feature models.

Features may be divided into two types on the basis of morphology: regular-shaped features having regular geometry, and freeform features having freeform shape geometry. Freeform features are further subdivided into freeform surface features and freeform volumetric features [2]. There is a significant amount of literature revealing research on regular-shaped features. However, with the ever-increasing demands for freeform shapes, such as in aesthetic design, complicated freeform features are getting more and more attention.

2.1 User-Defined Features

Hoffmann et al. [3] addressed the issue of user-defined features for regular-shaped features. They mention the need to provide design with mechanisms to define new features that fit the design, thus giving design the option of building custom feature libraries. They use a procedural mechanism for defining and deploying the feature classes, including compound features. Bidarra et al. [4] presented a declarative scheme to define new feature classes. The developed scheme provides a unified description of the shape, including validity issues. They also demonstrated a flexible configuration of the new feature class interface.

2.2 Freeform Feature Classifications

Literature reveals a couple of attempts to classify freeform features ([5]; [6]). Poldermann et al. [5] provide a general classification of freeform surface features that resembles the traditional feature taxonomies, but extended with more complex surfaces. They introduce four main classes of freeform surface features: primary, modifying, auxiliary and transition surface features. The primary surface is defined as that which describes the global shape of the product, whereas the modifying surface features represent secondary features that modify the primary surface features. The auxiliary surface feature class includes mechanical features such as holes and ribs. Transition surface features, corresponding to blends, are included to be able to enforce continuity at the surface boundary connections. Fontana et al. [6] present a freeform feature taxonomy for secondary features. It contains two types of features: the first type representing deformations of a freeform surface, and the second type eliminations of areas of a freeform surface. Deformations are split into three types: border, internal and n-channel deformations, depending on the placement of the feature on the surface. Two types of eliminations are defined: sharp cut and finished cut. Although the freeform features of Fontana et al. are more formally defined, they are a subset of those described by Poldermann et al.

van den Berg et al. [2] present a more detailed summary of these classifications than this outline. It is clear that Poldermann et al. and Fontana et al. present freeform features as a means to modify a given freeform surface, called a primary surface. Poldermann et al. distinguish between adding, removing and deforming regions of the primary surface, and their features are related to these operations. This can be applied to both volumetric and surface freeform features. Both taxonomies are restricted to particular domains: Fontana et al. focus on deformations targeting aesthetic designs, whereas Poldermann et al. argue against generic taxonomies by stating that one requires application-specific taxonomisation.

In this paper, we present an approach to define freeform feature classes by means of a mechanism supported by a generic freeform feature taxonomy. With the mechanism, the user can derive a new class from any level of abstraction, and specify new parameters and constraints for the class. The taxonomy can be updated with newly defined classes.

3. THE NEED FOR A GENERIC TAXONOMY

A feature modeling system should offer service to a wide range of domains. In this respect, such as system should have the capability to process a variety of freeform features. Such a situation can, on the one hand, be resolved by offering a

designer a large collection of static feature definitions, e.g. via templates. On the other hand, the designer can be offered the possibility to define new feature classes.
The first option would result in large object libraries, difficult to maintain, because of the large variety of freeform features that can be identified, as evidenced by the wide shape domain. Also, the limiting nature of using templates on variety makes definitions for reuse is far from sufficient. The second option is expedient, because only design-intended features have to be defined in the system. Poldermann et al. [5] and Au et al. [7] also mention this as a requirement. A freeform feature library management system (FFFLMS) should then be included in the modeling system for defining new feature classes, and for modifying or deleting feature classes. It should, in particular, offer a mechanism to define feature classes at various levels of abstraction. Any defined feature class can be stored in the library, and retrieved whenever required for instantiation in a particular product.
To ease its use, the library requires a meaningful taxonomy of the various types of freeform features. We develop a generic freeform feature taxonomy to support domain-independent definition mechanisms in feature modellers. The taxonomy is derived from a *classification* of several generic freeform features. The main advantage of such a taxonomy is that it provides a set of typical features that is easy to explore, and hence a means for quick access of the features in the library under certain search criteria. Figure 1 shows the relation among user-defined freeform features, freeform feature library and freeform feature taxonomy.

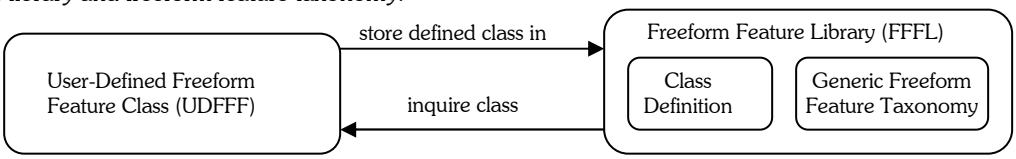


Fig. 1. Relation among UDFFF, FFFL and taxonomy.

In our approach, the classification is based on principles of group and manufacturing technologies. We use some of the criteria utilized there to derive our taxonomy. Group technology here is restricted to refer to an *information-reduction technique* that exploits similarities in the (design) attributes of a product. Manufacturing technology will refer to the nature of a freeform feature and the topology of the feature in a product model. Section 4 describes the criteria.
Most current feature modeling systems classify features by creation methods, mostly procedural, e.g. sweep and fill. Classification by creation does not provide for generic parametric definition of a (freeform) feature, because there are usually different methods to create the same feature. In some systems, the creation method reflects the representation of the feature in its library. For example, a cylindrical surface will be geometrically stored as an elementary cylindrical surface in a declarative definition, and as a NURBS swept surface representation in a procedural definition. These differences may be attributed to the lack of generic definitions of features. It may also explain why some of the current systems generally handle simple freeform features only.
Alternatively, using generic parametric definitions makes the whole design paradigm more intuitive, even for very complex shapes. A generic taxonomy can help a designer in creating new User-Defined Freeform Features (UDFFF) in a natural way. The next section presents such a taxonomy.

4. THE GENERIC FREEFORM FEATURE TAXONOMY

In deriving a generic taxonomy, we firstly characterize product shapes and their properties in generic taxonomic schemes defining *"a part of"* hierarchy. When class B is a part of class A, class A's definition includes one or more components whose types are class B, e.g. a tree will have a root, trunk, branch and leaf. Secondly, we group this hierarchy into "*a kind of*" hierarchy, where, in the example, derived classes would have all the tree's attributes and behaviors, plus possibly some specialized attributes or behaviors, e.g. palm, treefern, cycad, broadleaf, etc. This is done at different levels of generality of information structures and/or functionality. The implication is that when a new shape appears that does not have a known set of properties, such a shape becomes a candidate prototype for a new taxonomy. In such a case, a comprehensive characterization of shape and other properties is called for. We refer to "*a part of*" hierarchy as a *characteristic-oriented classification,* and to "*a kind of*" hierarchy as an *object-oriented classification.*

4.1 Characteristic-Oriented Classification

When the designer conceptualizes about a product, the design thought is naturally characteristic-driven, followed by specification (including global dimensions and final details). Similarly, in deriving the taxonomy, we begin with conceptualizing about a freeform feature by its characteristics in order to discriminate features into families. Using information-reduction per family, detailed characteristics further discriminate sets of a family. Similarities are identified

within carefully selected characteristics. Essentially, each family is associated with a unique data set. The following information, associated with a feature, can be used for characterization: availability, geometrical, functional, technological, and identifying characteristics. Availability refers to the presence or absence of something, depending on context. For example, the presence of a feature, or a sequence of features, say describing a pattern. This can also explain the nature of a feature: material displacement, removal or addition. The *shape* and *size* (metrics) of a product belong to geometric characteristics. This also includes tolerances to quantify the allowed deviation of actual from expected values, roughness, texture, etc. Functional aspects relate to the tasks the feature is qualified for. If a feature is not suitable for clamping tasks, it is not useful to specify a clamping force. Technological characteristics refer to aspects such as material properties, limiting stress, etc., which are generally dependent on manufacturing characteristics. Identifying characteristics refer to the name of the freeform feature, such as bump. It is generally straightforward to identify a form constituting a feature from its name.
Characteristics are related to the semantics of a feature. Thus, the above characteristics can be mapped to feature semantics, without loss in generality. Feature semantics can then be classified into four types:

- availability semantics, e.g. relating to presence, patterns, etc.
- geometric-oriented semantics, e.g. geometric entities and relationships between adjacent entities
- function-oriented semantics, e.g. determining functional interconnection between various features in a freeform feature model
- technology-oriented semantics, e.g. parameters used in calculation of stability, deformation, etc.

4.2 Object-Oriented Classification

With different families derived from the characteristic-oriented approach, we then apply the object-oriented approach to derive members (hierarchy) within every family. We represent families as classes. The object-oriented point of view pertains to inheritance of data while allowing definition of extra data, to further specify the derived features. Each class is associated with unique data derived from characteristics the feature has. The generic features are classified within an overall structure according to different criteria.
A distinction is made between different definitions of a freeform feature class (see Figure 2). A generic definition is a reusable, parametric description, which is essentially a collection of information about a freeform feature class. A generic feature class can be parametrised to describe a family of freeform features; in either a *declarative* or a *procedural* way. The earlier refers to the definition of parameters, e.g. scalar values, related by underlying mathematical descriptions, whereas the latter refers to parameter definitions subject to a predefined sequence of mathematical functions, e.g. via an interactive deformation function.

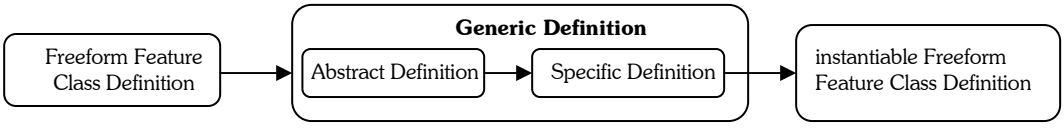


Fig. 2. Freeform Feature Class Definition.

A generic definition can either be *abstract* or *specific*. An *abstract generic definition* refers to a family of similar features, describing a class with the purpose of being inherited by other definitions in the object-oriented paradigm. This offers a common base to specific types. It can never be instantiated since not all parameters are defined, and may even have no parameter. A *specific generic definition* describes a set of similar features that is a subset of an abstract generic definition, or a specialised form of the abstract definition. In other words, an abstract definition is specialised into specific definitions, in which parameters are completely declared. For example, a class quadric may be viewed as an abstract class definition, whereas a class cylinder, derived from the class quadric, is a specific definition which may be instantiable, or further specialized to, say, a hole feature class. A specific generic definition may also have an explicit representation of its *properties* defined by its characteristics. If all parameters have been defined for the shape of a generic definition, then its shape is said to be parametrised to a specific generic feature, otherwise it remains an abstract definition. A user has more freedom in specifying a new class if he derives from an abstract than from a specific generic definition level, because at that level more parameters have to be specified to define the feature. Predefined feature class definitions, as they occur in current feature modeling systems, are derivatives of specific generic definitions, or of even more specialized (derived) classes. It should be noted that if the user derives from a level lower in the hierarchy than the abstract definition, and if its consequent properties are significantly different from all others already in the taxonomy, that then the resulting new class is a potential candidate in a new taxonomy.

Specific generic definitions are fully characterized and, therefore, instantiable. However, it is possible to further specialize them, through inheritance. Declaring parameter names implies declaring instance variables, e.g. height and radius of a cylindrical-like surface, for each instance. An instance describes a single feature that results once all parameter values of a generic definition have been specified. These parameter values, in combination with the declarative or procedural description of the corresponding generic definition, can be used in any feature model.

4.3 Deriving the Taxonomy

This subsection consolidates the method used to classify freeform features. Firstly, specific shape aspects are utilized. This is followed by the characteristic-oriented classification (*a part of hierarchy*) as discussed in Subsection 4.1, and the object-oriented classification (*a kind of hierarchy*) as discussed in Subsection 4.2. Finally, the topology of a freeform feature is considered. The aspects are presented in that order in the next subsections.

Shape Aspects

Aspects generally provide a means for abstracting information (or data) from a concept. In this paper, a freeform feature can be volume-based (3D shape aspects) or surface-based (2D shape aspects).

A Part of (Characteristic-Oriented) Hierarchy

The characteristics identified in Section 4.1 are used to classify freeform features in "a part of" hierarchy. Firstly, *availability characteristics* reveal the presence of a feature. By this means, it is possible to determine, for example, whether a sequence (e.g. pattern) of freeform features exist. Secondly, we observe the *geometric characteristics* of the freeform features by capturing those that are geometrically similar. Those with common character are grouped. Geometric information includes shape and size. Also details of the presence of discontinuities and undulations (curvature behavior), such as creases and ridges, can be identified here. It is clear that every product is a result of a manufacturing process one way or another. Thus, using manufacturing technology, together with this geometric information, a feature can be associated with the following operations: material displacement (e.g. metal forming, pressing, tweaking, die-punching), material removal (e.g. milling, perforation, turning, electro-discharge machining) or material addition (e.g. welding, riveting, layering from rapid-prototyping). We respectively name these using generally accepted terms: *deform*, *cut* and *transition features*. However, there are some operations that result from combined material removal and addition, e.g. blending that can result from a weld and precision grinding. They are generally called filling methods, so that they are largely addition operations or *transitions*. This implies such a type has to be further discriminated using the next criteria in the list of Subsection 4.1. In particular, *functional* and/or *technological characteristics* are utilized: transition features enforce model continuity. Their definition is a function of one or more other features in the model. At this stage, the available families have *identifying characteristics* clearly distinguished. Names are therefore given to the types derived this far as Deform, Cut and Transition Freeform Features. Whenever the availability characteristics reveal multiple features resulting from combinations of these types, a compound freeform feature is defined. If the compound feature has a repeating sequence of a type or types, then a pattern compound freeform feature is specified, otherwise it is composite.

A Kind of (Object-Oriented) Hierarchy

Within the characteristic-oriented approach, whenever common information (one-to-many relationship) appears between families, the object-oriented approach is used. This implies that a common class is inherited by one or more other classes whose information is sharable. For example, the freeform feature class is common to the deform and the cut feature classes, and the deform feature class is common to the ridge and bump feature classes. See Figure 3 for the resulting object-oriented classification.

Topology

Now that the object-oriented classification has been made, it is possible to exploit information about the topology of features in a model. We identify three types of topology: *border, channel and internal* (see Figure 4). A freeform feature is *internal* if it does not intersect the model or its influence area boundary. If it *intersects* the boundary once, it is a *border* feature, and if twice, it is a *channel*.

It should be noted here that this topology may be applied to an influence area; a portion of the whole model. Some shapes, as a whole, may have a non-distinct boundary in 3D space, e.g. a sphere. Since this topologic classification may not apply to such cases, such shapes will only have those applicable defined, e.g., in a complete sphere border

topology is non-applicable. Such issues are not discussed further in this paper. Figure 5 is an example freeform shape, showing a ridge, bump and hole features.

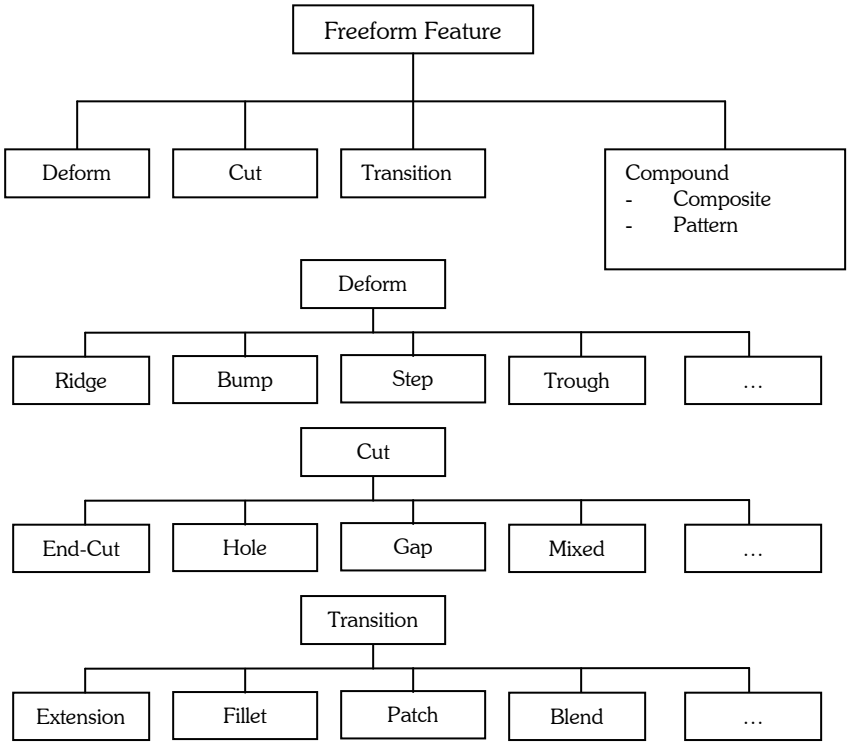
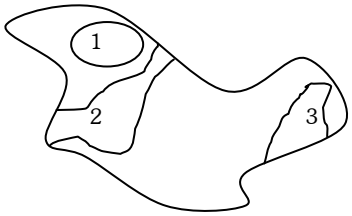


Fig. 3. The object-oriented classification.



1. internal
2. channel
3. border

Fig. 4. Topology of a freeform feature with respect to model or influence area.

In Figure 5, the hole (cut feature) has internal topology with respect to the whole model. However, its topology is border with respect to the feature ridge (assuming the selected influence area includes only the hole and ridge).

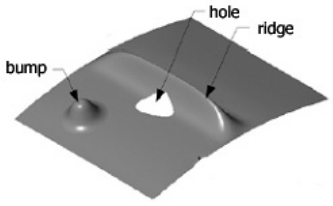


Fig. 5. Freeform feature examples.

4.4 The Taxonomy

From the aforementioned results the freeform feature taxonomy shown in Figure 6.

The *Freeform Feature Class* represents a virgin freeform feature without any specific characteristics, except for general information indicating whether the shape aspects are 2D or 3D. It comprises geometric as well as topologic characteristics common to all freeform features.

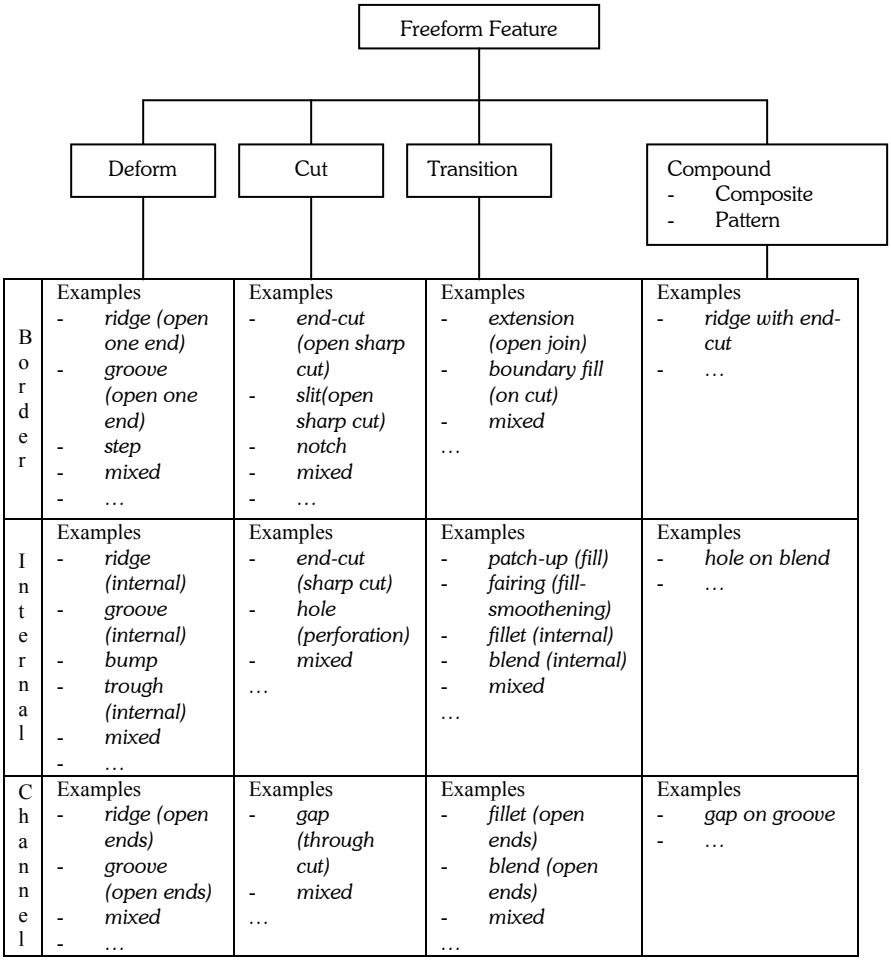


Fig. 6. The Freeform Feature Taxonomy.

Generic Abstract Freeform Feature Classes define deform-, cut- and transition-type freeform features. The process of classification has been accounted for in Subsection 4.3. The compound type is of completely different character as its characteristics are a compatible combination of different generic specific features.

The *Deform Class* may be viewed as displaced regions normally characterized by a height (for extrusions) or depth (for intrusions) function relative to a reference position on the feature. Their instantiation does not alter topology. A good example, a bump, is a rise in the form of a rounded spot or budge. In a modeling operation, it may result from an inflate-and-tug operation on a restricted influence area of a feature model. A ridge is a narrow elevation of a section or sections of a model. A notch is a kind of ridge intersecting the model boundary and is considered as a boundary deformation.

The *Cut Class* represents a vacant position or void in the model. It is associated with removal of material. Complex trimming operations sometimes create this feature, e.g. a sharp hole (with non-smooth edges) or a hollow (with smooth edges).

The *Transition Class* connects two or more other features, usually to maintain model continuity. It can be simply or non-simply connected to the other features. Constant radius fillets are an example of a simple transition feature, whereas a blend between more than two features can be quite complex.

The *Compound Class* is a result of a logical combination of other freeform features. It describes a conglomerate of features, and is always associated with relational parameters and constraints. Two types of compound features are: composite feature and pattern. Mortenson defines a composite feature as a collection of individual surface patches joined to form a continuous, more complex surface [8]. An example is a bump (deform type) with a perforation (cut type). A pattern is a repetitive sequence of another feature or groups of features.

The classes derived from any of these four above are more specialized, e.g. an internal ridge. They are examples of pre-defined classes current systems may provide the user with. The examples are by no means exhaustive. The user can derive a new freeform feature class from any abstraction level in the taxonomy.

5. SPECIFICATION OF FREEFORM FEATURE CLASSES

To define a new feature class, a designer can select a basic feature class from our taxonomy. Note that if he is not satisfied with the types available at any level of abstraction, our system will allow other user-defined types derived from even the most general class: the freeform feature class (see Figure 6). In fact, this level is not restricted to surface-based freeform features, but also applies to volumetric, and even regular features. Thus, the methodology in specifying user-defined freeform features starts with the object-oriented approach (deriving from the taxonomy) for any desired level of abstraction. When a new class has been derived from the taxonomy, parameters and constraints further defining the shape of the freeform feature have to be added. The parameters and constraints are derived from characteristics of the desired freeform feature. The level from which the new class is derived determines the number of parameters the user will have to add to define the feature class. For example, defining from the top-most (freeform feature class) requires more input from the user, whereas deriving from lower (more characterized types) classes (e.g. deform) correspondingly reduces the number of parameters and constraints required to define the feature class.

Parameters are geometrical and topological entities necessary to define a particular shape, and constraints are conditions these parameters have to satisfy. We generically consider parameters to be 0D (e.g. a scalar value), 1D (e.g. a curve), 2D (e.g. a surface patch) or even 3D (e.g. a volumetric deformation tool). Details of generic parameters are postponed to future work.

Specifying parameters of a feature class means defining the characteristics of, or customizing the shape of, the feature. Parameters may be grouped into three types: input, computed, and control parameters. Input parameters refer to handles exposed to the designer, which are useful in feature class instantiation. These will be the public members of a class. Computed parameters are those over which the user has no direct control, but are instead, for example, determined by algebraic relations within a class method, or from reference datums. Control parameters have direct effect on the feature shape. They are divided into two types, corresponding to basic and fine shape definitions: *basic* and *fine shape parameters*. The earlier pertain to global (structural) specification, whereas the latter pertain to local (detail) specifications. Figure 7 shows their relationship.

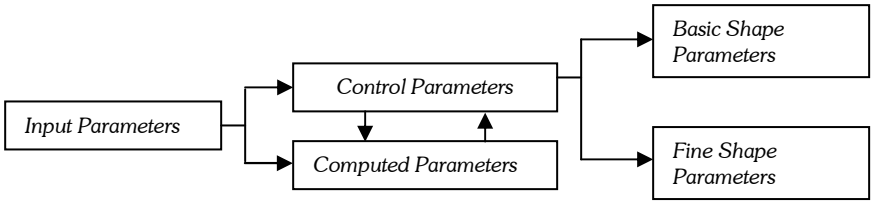


Fig. 7. Freeform feature class definition parameters.

Basic shape parameters are the basis to a class definition. Depending on the level of abstraction in the taxonomy and type of feature, these parameters are sometimes enough to instantiate a freeform feature, e.g. in most cases inheriting a specific generic definition as opposed to deriving from an abstract definition. They can be considered as specifying explicit parameters of a freeform feature, e.g. height and width in a specific generic class definition. Fine shape

parameters specify shape details of the freeform feature, e.g. local shape information related to surface creases. These parameters actually make freeform features different from regular features, because in the latter one normally would specify shape parameters without making a distinction between basic and fine ones.

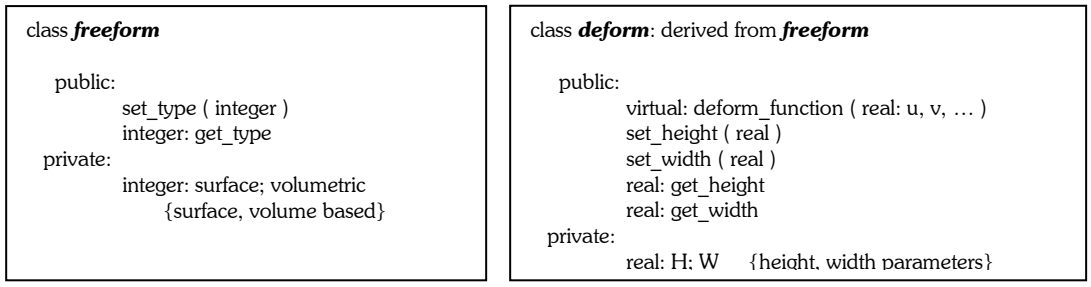
Different conditions apply to different types of features. Constraints are introduced on the parameters to impose such conditions.

5.1 Adding Parameters and Constraints to a Class

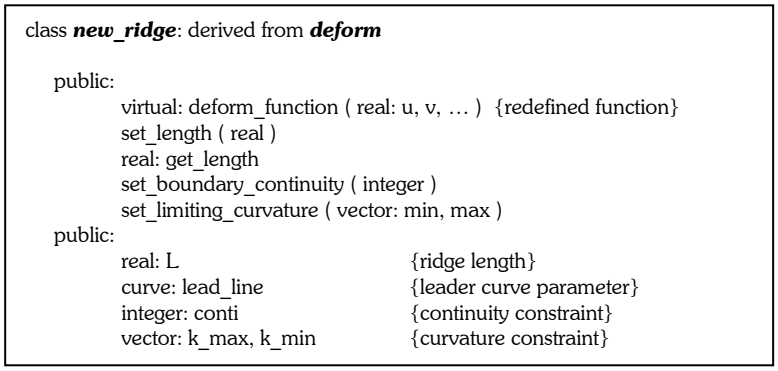
The taxonomy is structured in a hierarchical order (*a part of and a kind of hierarchy*), which helps the user to traverse the whole taxonomy. The user selects a class from the taxonomy as a basis by means of a graphical user interface. With an appropriate editor, parameters are added to the freeform feature class, which are then visible to the user in the interface. In addition to specifying simple parameters with their names and type, e.g. real, we even allow specifying mathematical functions for a parameter, e.g. a deform function for deform type features.
Basic shape parameters are firstly specified. These are normally derived from geometric characteristics of the freeform feature, and relating to shape and size. The user may add basic geometric constraints on the dimensions, e.g. geometric tolerances, and relations between parameters, say, radius = 0.5*height.
Fine shape parameters may include additional parameters, such as those relating to micro-geometry: roughness, texture, etc. from type of material quality. Other parameters are computed parameters that include those relating connectivity to other features in the model, e.g. continuity.
Detail constraints are also specified, such as the maximum allowable curvature in some principle direction. Note that a curve can also be used as a (1D) parameter on which constraints such as limited acceleration and tension can be applied. An example is a leading line for controlled deformation in aesthetic design.

5.2 Example

Suppose the user wants to define a freeform feature class for a new type of ridge, say *new_ridge*. If the ridge can be considered a variant of the existing ridge, he would use the predefined *ridge* class that already exists in the taxonomy. If, however, the new ridge has properties that clearly distinguish it from the existing ridge, and the user is experienced, he will derive his own *new_ridge* class. In this example, we assume the latter.



(a) **(b)**



(c)

Fig. 8. Example freeform feature class definition.

The definition of the most generic *freeform* feature class, or the base class, is shown in Figure 8a. Then, the *deform* feature class (see Figure 6), which has been derived from the class *freeform* is shown in Figure 8b. The user wants to derive his *new_ridge* class from this *deform* class level of abstraction. This class is not the predefined ridge class in the taxonomy, but a new one. He thinks about parameters and constraints. The parameters could be the length: L, height: H, base width: W, and leading line: lead_line, and the constraints could be geometric continuity at boundaries: conti, and limited curvature on surface: k_min, k_max. The class details are shown in Figure 8c.

The common parameters height and width are available in the *deform* class. Note that the length parameter is not included here because some deform freeform features do not use it, e.g. a bump. However, we do need it for the *new_ridge*, just like the leading line, as shown in the class *new_ridge*. Also note that one can optionally use a 1D parameter (curve) to represent the boundary of the feature instead of directly using the width (W) parameter.

Note that the function deform_function is here defined to override that in the super-class. Pre-defined abstract classes can be used to implement this function. The fields "set_" are the ones exposed to the user in the interface edit fields as input parameters, or handles, to instantiate the class.

6. CONCLUSIONS

This paper has argued for a generic freeform feature taxonomy using a characteristics-oriented approach and an object-oriented approach. The fact that every product results from manufacturing in one way or another, justifies the use of manufacturing technology in association with geometric characteristics. Group technology, with information-reduction, was used to obtain the 'a part of' hierarchy and 'a kind of' hierarchy to derive the taxonomy.
We have shown that with the generic taxonomy the user is enabled to create his own feature classes, through adding shape parameters and constraints through a user interface. What is also worth noting is that, since the user can derive a class from different levels of abstraction in the taxonomy, both novice and more experienced users are able to work with the taxonomy. The latter can even specify their own functions defining the shape of the feature by overriding existing ones in the classes of the taxonomy. Novice users can simply specify parameters for the shape without thinking about complex mathematical functions.
Altogether, the generic freeform feature taxonomy, and the approach to define new feature classes from it, can effectively support the incorporation of user-defined features in freeform feature modeling.

ACKNOWLEDGEMENT

This research (coded DIT. 6240) is supported by the Technology Foundation STW, Applied Science Division of NWO and the technology programme of the Ministry of Economic Affairs, The Netherlands.

7. REFERENCES

[1] Bidarra, R. and Bronsvoort, W. F., Semantic feature modelling, *Computer-Aided Design*, Vol. 32, No. 3, 2000, pp 201-225.
[2] van den Berg, E., Bronsvoort W. F. and Vergeest J. S. M., Freeform feature modeling: concepts and prospects, *Computers in Industry*, Vol. 49, No. 2, 2002, pp 217-233.
[3] Hoffmann, C. M. and Joan-Arinyo, R., On user-defined features, *Computer-Aided Design*, Vol. 30, No. 5, 1998, pp 201-225.
[4] Bidarra, R., Abdelfettah, I., Noort, A. and Bronsvoort, W.F., Declarative user-defined feature classes, in: 'CD-ROM Proceedings of the 1998 ASME Design Engineering Technical Conference', 13-16 September, 1998, Atlanta, USA, ASME, New York.
[5] Poldermann, B. and Horváth, I., Surface-based design based on parametrized surface features, in: 'I. Horváth, K. Varadi (Eds.), *Proceedings of the International Symposium on Tools and Methods for Concurrent Engineering*, 29-31 May, 1996, Budapest, Hungary, Institute of Machine Design, 'Budapest', pp 432-446.
[6] Fontana, M., Giannini, F. and Meirana M., A freeform feature taxonomy, in: Brunet, P., Scopigno, R. (Eds.), *Proceedings of Eurographics 1999', Computer Graphics Forum*, Vol. 18, No. 3, 1999, pp 107-118.
[7] Au. C. K. and Yuen, M. M. F., A semantic feature language for sculptured object modeling, *Computer-Aided Design*, Vol. 32, No. 1, 2000, pp 63-74.
[8] Cavendish, J. C., Integrating feature-based surface design with freeform deformation, *Computer-Aided Design*, Vol. 27, No. 9, 1995, pp 703-711.