Definition of Freeform Surface Feature Classes

Paulos J. Nyirenda, Madhuwanti Mulbagal, and Willem F. Bronsvoort

Delft University of Technology, [p.nyirenda/w.f.bronsvoort]@ewi.tudelft.nl

ABSTRACT

This paper presents a flexible, generic mechanism to define freeform surface feature classes. The method is applicable to pre-defined and user-defined features. A feature class definition includes, firstly, declaring a feature class, and then specifying the class using attributes specific to the feature. A generic taxonomy of freeform surface features is used to facilitate class declaration. Then, the dynamic mechanism facilitates general specification of attributes. It is demonstrated that such a class definition can be consistently archived, retrieved and re-used in an application-independent manner.

Keywords: feature modeling, freeform surface features, user-defined features, taxonomies, feature libraries

1. INTRODUCTION

Feature modeling has become the fundamental design paradigm for Computer-Aided Design (CAD) systems. One main advantage over conventional geometric modeling is the ability to associate functional and engineering information to shape information in a product model [1].

Most current feature modeling systems support only regular and simple freeform shape features, and not general freeform shape features. A distinction can be made between freeform volume and freeform surface features. Although some research has been done in the area of freeform feature modeling, many issues remain to be resolved [2]. The work presented in this paper is on the definition of freeform surface feature classes.

In each feature modeling system, several types of features can be used. All properties of a specific type are defined in the corresponding generic feature class, which prescribes a template for all its instances. An instance of the class can be created by assigning values to its parameters; the instance can then be added to a feature model.

Looking at the definition of feature classes, a distinction can be made between pre- and user-defined classes. The former, commonplace in current commercial systems, represent parametric solutions for standard design problems and primarily constitute a static generation of definitions. Often, they are represented by hard-coded procedures in the system. The latter are customized solutions, not restricted to the pre-defined solutions. They can be completely new definitions, or can be extensions of previously defined classes. The availability of a user-friendly mechanism to define new feature classes is highly desirable in any feature modeling system.

In this paper, a mechanism is presented to define freeform surface feature classes. No distinction is made between pre- and user-defined classes: the mechanism can be used for both pre- and user-defined freeform surface feature classes. Such a mechanism is not yet available, and can significantly contribute to the proliferation of freeform feature modeling.

In order to define a freeform surface feature class, the intended class should be identified as either abstract or specific. An example of an abstract class is a deform class, whereas a concrete class is a bump class. Identification can be done with the help of a taxonomy for features archived in a feature library. With this taxonomy, a level of abstraction can be determined, and a new feature class can be declared.

Definition of a feature class involves, firstly, this declaration of the new class, by deriving from a class in the taxonomy. The new class inherits properties from its parent class. Secondly, the declared class is specified using parameters and constraints that characterize it. Specification generally involves defining the generic shape of the feature, parametrizing it, and adding functional information. A special type of feature class is a compound class, which is a combination of two or more already defined classes. The mechanism presented in this paper provides facilities for all these options.

The paper is structured as follows. In Section 2, related work is discussed. Terminology used in the sequel of the paper is illustratively introduced in Section 3. Declaration of a freeform surface feature class, selected from the feature

taxonomy, is presented in Section 4, and specification of the declared class is presented in Section 5. The definition of a compound class is discussed in Section 6. Finally, Section 7 enumerates some conclusions and future work.

2. RELATED WORK

Freeform feature modeling is a relatively new area of interest in CAD research. Many concepts in freeform feature modeling are similar to concepts found in regular feature modeling. For an overview, please refer to [2].
For this paper, it was, in particular, important to get insight into the shape domain of freeform features. Several freeform feature taxonomies have been used for this, and a taxonomy will also re-occur in our mechanism. In addition, the concept of defining new feature classes in existing systems, often called user-defined features, is relevant. Both are surveyed in this section.

2.1 Freeform Feature Taxonomies

Poldermann et al. [3] introduce four main classes of freeform surface features: primary, modifying or secondary, auxiliary and transition surface features. A primary surface feature describes global shape, whereas a secondary surface feature modifies the primary surface feature. An auxiliary feature includes such mechanical features as holes and slots. A transition surface feature primarily enforces continuity at surface boundary connections, e.g. blends.
Fontana et al. [4] extend the secondary features using a taxonomy encompassing two representations: the first and second types respectively representing deformations and eliminations of areas of the primary surface. Deformations are further classified into border, internal and n-channel, depending on the placement of a secondary on a primary surface feature. This classification, though more formal than that of Poldermann et al., is only a subset of it.
Nyirenda et al. [5] present a generic taxonomy for freeform features using object- and characteristic-oriented hierarchies to derive deform, cut and transition single features, upon further classifying them based on their nature. The taxonomy also includes compound features: composites and patterns. They further classify freeform features on the basis of *relative topology*. By this, every feature can be placed relative to another, possibly in a manner that eliminates explicitly relating intrinsic coordinate systems to each other, say in compound features. In Fig. 1, given an influence area, also called a Region Of Interest (ROI), a freeform surface feature can be disconnected from or connected to the influence area. The former is considered *isolated*, whereas the latter can be either *border*, *channel* or *internal*.

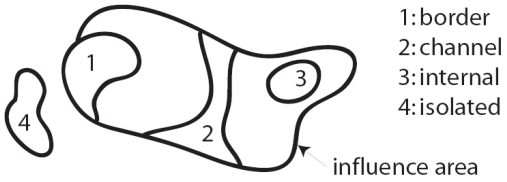


Fig. 1. Relative topology in freeform features.

A freeform surface feature is a border if it intersects its influence area boundary once, a channel if twice, and internal if it does not intersect the border. Note that the influence area can refer to the whole shape outline. Thus, every feature definition includes a defined number of curves (limiting lines), as one of the attributes, used to specify the influence area. From the aforementioned, Poldermann et al. and Fontana et al. present freeform features as a means to modify a given primary surface feature. Poldermann et al. distinguish between adding, removing and deforming regions of the primary surface, and the features are associated to these operations. Both their works are restricted to particular application domains: Fontana et al. confine to aesthetic design, and Poldermann et al. oppose generic taxonomies by stating that design requires application-specific taxonomies. We argue against application-specific taxonomies and support generic taxonomies so that the user can create customized features in his own right per intent, and not by domain limitations. Semantics of the feature are the most important distinguishing property that becomes indispensable when defining a feature. This paper, therefore, is based on the generic taxonomy previously developed by the authors of this paper.

2.2 User-defined Features

Existing commercial systems provide pre-defined features only. The lack of generic means to define new freeform feature classes represents a major shortcoming in these systems. The need for generic, yet (re-)usable and extendable, solutions to support custom feature libraries that satisfy specific application needs cannot be over-emphasized.

Bidarra et al. [6] develop a declarative scheme to define new feature classes for regular shape features. This scheme provides a unified description of the shape, including validity aspects. It also includes a flexible configuration of a new feature class interface. Hoffmann and Joan-Arinyo [7] also present a mechanism to realize user-defined regular shape features, particularly focusing on compound features. van den Berg et al. [8] present a generic approach to specify volumetric freeform features. In this approach, a new freeform feature class is interactively specified using a prototype instance and constraints. The constraints are used to define intuitive parameters and validity conditions for the class. A prototype-driven constraint solving mechanism is also developed to ensure unambiguous definition of freeform features. That work focused on freeform volumetric features. In this paper, freeform surface feature class definition is presented.

3. TERMINOLOGY

In feature-based design, functional units can be described as features containing geometry. Therefore, an advanced design system can apportion its functionality based on two types of expositions: *geometry-* and *semantic-oriented* expositions.

Geometry-oriented exposition of features focuses on mathematical descriptions defining *spatial relationships* characterizing space using geometric entities, e.g. points, curves, surfaces and solids. A feature is associated with a generic shape. The generic shape is a structured grouping of the geometric entities without any semantic representation of the geometry.

Semantic-oriented exposition of features primarily focuses on features that are defined in an application-specific context. The semantic information may be represented by symbolic descriptions associated with the generic shape definition. This association is known as *parametrization* of the shape of the feature. The semantics express properties of the shape of a feature, represented as constraints in the feature class specification. In this paper, an object-oriented approach is taken that utilizes semantic features because their exposition facilitates a class-oriented management of freeform surface features. The semantics is generally known as *functional information*.

Whereas a feature is treated as a *specific shape* in the design interface (at instance level), it is defined (at feature class level) as a *generic shape* associated with the semantics of its engineering meaning. This includes functional attributes, expressing *properties* of the feature. Other useful information may include, for example, a self-explaining sketch of the feature presented as an icon during feature selection from a taxonomy.

In this paper, freeform surface feature class definition includes shape, parameters and constraints definition.

3.1 Shape

The generic shape associated with freeform surface features is described using geometric entities. Freeform shapes defined with NURBS [9] make modeling more complicated, due to many degrees of modeling freedom, than modeling with regular shapes. It therefore seems reasonable to have a structured description of the functional information associated with the freeform surface features to enhance their management.

Nyirenda et al. [5] decompose a freeform feature shape into *basic* and *fine* shapes. A basic shape specifies the structure of a freeform feature as a composition of feature entities, and their relationships, constituting its shape as a whole. A fine shape specifies details of the shape of a freeform feature constituting the internal areas or regions of the basic shape elements. Pernot et al. [10] describe freeform surfaces using freeform deformation techniques, where surfaces can be manipulated using different levels of multi-minimization techniques. The notion of describing such surfaces using constraints to define global and local shapes is similar to ours. However, our approach specifies feature shapes using higher level parameters, e.g. height for certain freeform features.

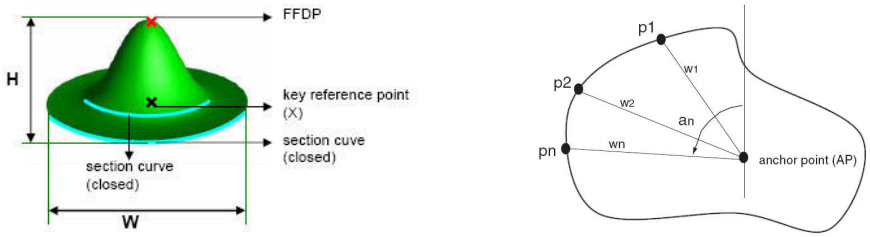
3.2 Parameters

Parameters of a freeform surface feature can generally be categorized on the basis of their influence on the shape: *positioning* and *shape* parameters. The former do not change the shape directly but are used to position the freeform feature relative to other features in the feature model or in a compound feature. The latter directly control the shape.

A positioning parameter can be either a *reference* or an *auxiliary parameter*. The former is a global datum used to position the whole feature, whereas the latter is a local datum used to position only one or more entities making up the feature shape. A freeform surface feature has one reference parameter, e.g. the *key reference point* in Fig. 2a, but can have more auxiliary parameters, e.g. *anchor point*s used to position sections in Fig. 2b. Auxiliary parameters are always specified relative to a reference parameter.

Three categories of shape parameters can be identified [5]: *input*, *computed* and *control* parameters, as shown in Fig. 3.

Input parameters are those values that must always be provided externally to instantiate the freeform feature class, e.g. height (H) and width (W) in Fig. 2a, and angle (a_n) and radius (w_n) for positioning a Freeform Feature Definition Point (FFDP) in Fig. 2b. An FFDP is a point in 3D space meant to facilitate intuitive definition of geometry. For example, a set of FFDPs can be interpolated by a NURBS curve or surface. An FFDP is defined by constraints on angle (a_n) and radius (w_n) imposed on its position. For details on FFDPs, please refer to [8].
Computed parameters are those determined from constraints imposed on input parameters. The user does not have direct control over this type of parameter, e.g. the coordinates of an FFDP such as p1 in Fig. 2b.



(a) bump freeform surface feature (b) 1D section parameter (curve)

Fig. 2. Parameters of a freeform bump surface feature.

Control parameters are the interface to the shape of the freeform surface feature. Unlike a computed parameter, a control parameter can be directly accessed by the user. Examples include a section and an influence area. Their actual values are either provided externally from input parameters or else computed from equations and constraints. Two types of control parameters can be identified: *basic shape* and *fine shape parameters*.

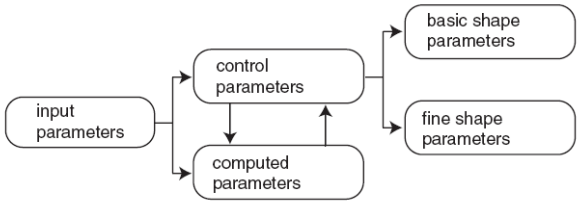


Fig. 3. Types of freeform surface shape parameters.

On the one hand, a basic shape parameter specifies the elements describing the global structure of a freeform surface feature, e.g. a complete section in the feature shape. On the other hand, a fine shape parameter alters internal areas or regions of the basic shape elements (e.g. section, surface). Such a local shape, which may be viewed as a superposition on the global shape, is the fine shape, e.g. a dimple on the bump feature in Fig. 2a. Fine shape parameters are used to specify the fine shape, e.g. a single FFDP or a subset of FFDPs used to specify the section parameter (Fig. 2b) can be used to locally modify the bump basic shape. For another example, consider a regular block feature. Its global shape is characterized by H, L and W. If one of the faces in the block is deformed, e.g. by tweaking, the deform is a fine shape on the global shape. This fine shape of the block feature has its own fine shape parameters, e.g. some height h, always specified with respect to the basic shape elements.

3.3 Constraints

Constraints are rules and conditions used to express the shape properties, primarily derived from the semantics, of the feature. Freeform surface feature constraints can be grouped according to their influence on the feature shape: *shape* and *positioning* constraints. Shape constraints provide a mechanism to determine the shape from the parameters. The nature of this mechanism may be different for different types of shapes. Positioning constraints fix the position of the feature or its entities. The feature or feature entity position can be constrained using explicit coordinate values, whereas the feature position can be constrained, in addition, through relative topology.
In a freeform surface feature, two types of shape constraints are identified: *basic* and *fine shape constraints*. Basic shape constraints are analogous to those in regular shape features, as in [1]. However, the fine shape constraints are crucial in freeform shape definition.
Basic shape constraints include the following: *dimension*, *algebraic*, *on-model*, and *interaction constraints*. A dimension constraint specifies a range of numeric values allowed of a particular parameter. An algebraic constraint

associates a parameter to another through an algebraic relation, e.g. width = 0.25*height. On-model constraints specify to what extent the features should be on the model boundary. On-model constraints are of two types: *on-model*, meaning that the feature should be present on the model, and *not-on-model* which means that the feature should not be present on the model. Further, both types of on-model constraints are parametrized, stating whether the presence or absence on the model is *completely* or *partially* required. For example, a hole feature on a planar face may be declared completely or partially not-on-model; in the first case the hole may not be covered by any other feature at all, in the second case it may be partially covered. This idea is adapted from [1], on volumetric features. Interaction constraints specify the permitted interactions for a feature class in order to preserve the functional meaning of the feature, e.g. in a splitting interaction constraint it is specified whether the feature shape may be split into several parts.

Fine shape constraints include the following: *placement*, *scalar*, *passing-point*, *carrier*, *pressure*, *transformation*, *geometric contact order k*, and *micro-geometry constraints*. A placement constraint specifies relative position and orientation among geometric entities in a global or local coordinate system. Standard types are available such as tangency, (co)incidence, (co)radial, equal, symmetric, midpoint, collinear, offset, parallel, concentric, and so on. A scalar constraint represents a mathematical law on parameters that can be linear, curvilinear or any other, e.g. cosine law. A passing-point constraint restricts the fine shape to interpolate a given FFDP on which a transition, e.g. tangency, can be specified. A carrier constraint specifies whether an entity should be a child of (dependent on) another, e.g. that a set of FFDPs used to specify a section also belongs to the face of the bump freeform feature. A pressure constraint can be a concentrated spot or point, distributed (e.g. a patch) or directional (e.g. curve). A transformation constraint restricts certain transformations to restrict degrees of freedom on freeform shape entities, e.g. on sub-features of a compound feature. Geometric contact order k constraints are those used to specify continuity. Micro-geometry constraints specify such information as roughness of surfaces.

Positioning constraints are specified either on the positioning parameters quantitatively (Fig. 2b), or qualitatively using relative topology (Fig. 1). Quantitative positioning fixes the position, and possibly orientation, using explicit coordinates, whether in absolute or relative terms. An FFDP is constrained through distance between two and an angle between three FFDPs [8]. Qualitative positioning fixes a feature's relative topology, i.e. its relative position to another feature (see Section 2.1).

3.4 Defining a Feature Class

A freeform feature class can be defined as a structured description of all properties pertaining to a given feature type, defining a template for all its instances. Properties of the feature are described using attributes, e.g. parameters and constraints; and one or more methods that operate on the attributes. For example, a computed parameter can be determined in a method.

Definition involves, firstly, declaring the feature class itself without recourse to attributes and, then, specifying the declared class with attributes for the intended feature.

4. FEATURE CLASS DECLARATION

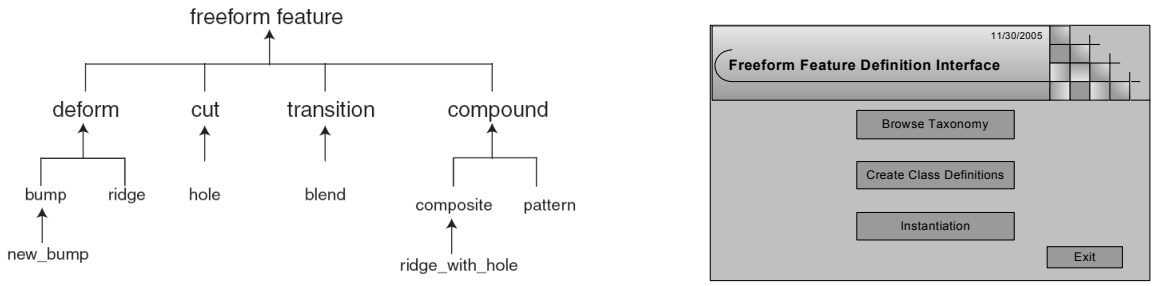
In this section, declaration of a feature class is presented as the first step in the process of defining a feature class. The generic approach we have taken makes use of the taxonomy presented in [5].

4.1 Taxonomy

The taxonomy for freeform features is shown in Fig. 4a. The highest level, the 'freeform feature' class, is the common class, with general attributes universal to all freeform features, e.g. information about type, relationships, and any generic laws. The next lower level derived from this class contains generic abstract types: deform, cut, transition and compound features. From this level, specific classes can be derived, e.g. bump and ridge, by adding specific attributes. These can include parameters and constraints to define a specific shape. Parameters can also include feature entities of the shape, subject to certain constraints. Constraints are the conditions the parameters should satisfy.

This taxonomy, in Fig. 4a, is by no means complete. Only a few specific classes are shown, e.g. bump, ridge, and blend. Further, specific classes can also be further specialized by a user, e.g. new_bump and ridge_with_hole.

The taxonomy is availed whenever "Browse Taxonomy" is selected in a typical class definition interface shown in Fig. 4b. The taxonomy browser includes icons for identifying types of features, including their position in hierarchies of existing definitions. More specific information about the selected feature is displayed if the class definition already existed. Once a parent feature has been identified, the new class can be declared, and the user can name it.



(a) taxonomy (incomplete) (b) class definition interface

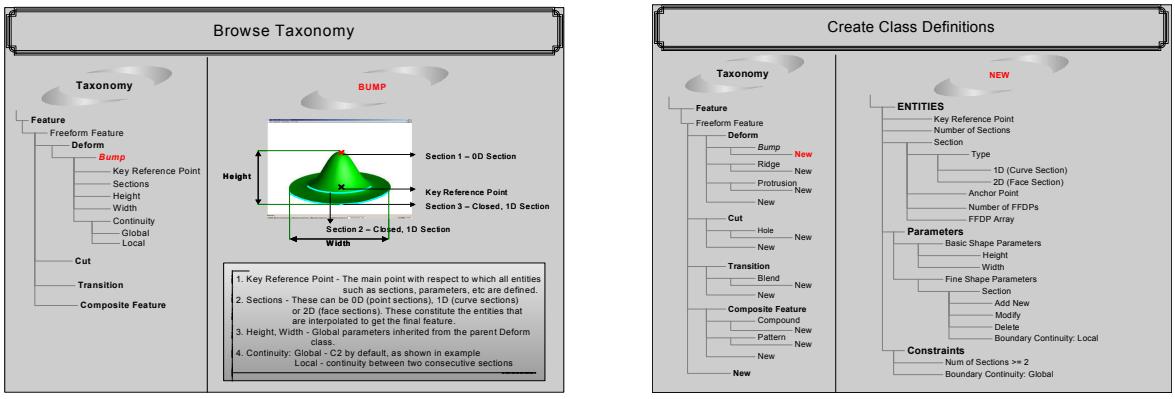
Fig. 4. Freeform surface feature definition.

4.2 Deriving a Feature Class

In this section, derivation of a feature class is presented. As mentioned earlier, a user-defined freeform surface feature can be derived from a pre-defined standard freeform surface feature class or from another user-defined freeform surface feature class. Derivation can be made from an abstract level, e.g. deform level, or from a specific level, i.e. a complete class definition.

As a first case, suppose a new freeform surface feature class called bump is created by deriving from abstract deform feature class, as shown in Fig. 5a. Only general attributes common to deform type are available, as shown in the left hand side pane, e.g. key reference point, sections, height and width. Depending on the selected item in this pane, the right side pane displays associated information. The bump is selected in the figure, and general information is presented; in this case, notes have been added to the bump feature description, as shown in Fig. 5a.

In the second case, suppose another new class is created, but this time derived from a specific feature class. Assume it is derived from the class we defined, earlier, in the first case – bump feature class and call it new_bump (see Fig. 4a). Before the class is named, a label 'New' is displayed, and since in this example the new class derives from the bump class, all parameters of the bump class are displayed, in the right side pane, as shown in Fig. 5b. The user can then specify the class: name it new_bump, add new parameters and constraints, and so on.



(a) browsing taxonomy (b) class specification attributes

Fig. 5. Browsing the feature taxonomy.

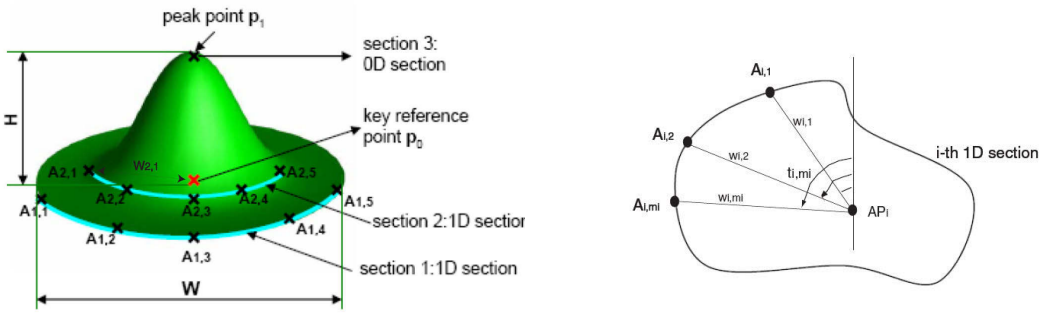
As components of the newly derived class, parent features carry with them their own attributes, thus providing natural "*a kind of*", and "*a part of*" inheritance. A kind-of hierarchy is an object-oriented inheritance which describes subtyping of a parent to derived freeform feature classes, e.g. deform and cut classes share the parent freeform feature class, whereas bump and ridge classes share the deform parent class. This describes different levels of abstraction in the taxonomy. A part-of hierarchy describes distinguishing properties among freeform features, at a given level in the taxonomy.

5. FEATURE CLASS SPECIFICATION

In this section, it is shown how a declared feature class is characterized using attributes. This is called specification of the feature class. Attributes such as geometrical entities and constraints are necessary to define a particular shape. A derived class inherits all the attributes from the parent class, all of which have to be considered together.

5.1 Flexible Specification Mechanism

In this section, the characterization of the declared feature class, using specific attributes, is presented. There are several ways to specifying a feature class, differing particularly with respect to the representation of geometrical data. Firstly, our mechanism is flexible. This means that the class allows for definition of entities with any defined number of attributes, at the same time providing for the capture of geometrically similar objects (same number and type of attributes) within the entity. Secondly, all entities in our class have names so that the user has at least some semantic information at class definition phase. In addition, a feature class definition can also include the parent classes, facilitating data encapsulation. These aspects facilitate custom-tailored specification of a feature class.
Specification of the class involves: *determining basic shape geometry*, *parametrizing* and *constraining* the shape, *determining fine shape parameters and constraints*, and *introduction of additional functional information*. It often involves *validation* through prototyping [2]. The bump feature class declared in Section 4, and annotated in Fig. 6, will be used here to walk through the specification.



(a) Bump feature parameters (b) 1D section details

Fig. 6. Bump shape geometry.

Note that methods in a feature class definition are used to specify the behavior of a class instance. For example, the computed parameters mentioned earlier are determined in a method. Also the basic shape geometry is determined in a method of the freeform feature class, with one or more parameters as arguments.

5.2 Basic Shape Geometry

The basic geometry definition of the bump, in Fig. 6a, will use the following entities:

- position of key reference point, $\mathbf{p_0}$
- $\mathbf{N}$: total number of sections that make up the bump freeform surface geometry, with $\mathbf{N} = \mathbf{N_{0D}} + \mathbf{N_{1D}}$, where
  - $\mathbf{N_{0D}}$: the total number of 0D sections
  - $\mathbf{N_{1D}}$: the total number of 1D sections
- $\mathbf{AP_i}$: the anchor point of the $\mathbf{i}^{th}$ section, representing its local coordinate position. It is defined using the FFDP. Thus there are a total of $\mathbf{N_{1D}}$ anchor points for the 1D sections, and $\mathbf{N_{0D}}$ for the 0D sections. Note that every anchor point is defined with respect to the key reference point, $\mathbf{p_0}$
- $\mathbf{m_i}$: the total number of FFDPs that make up the $\mathbf{i}^{th}$ 1D section, except the anchor point ($\mathbf{AP_i}$)
- let $\{\mathbf{A_{i,j}}\}$ be the set of FFDPs that make up a 1D section
  - i: section index count; [1, $\mathbf{N_{1D}}$]
  - j: FFDP index count on a 1D section; [1, $\mathbf{m_i}$]
- each $\mathbf{A_{i,j}}$ is specified at distance $\mathbf{w_{i,j}}$ and angle $\mathbf{t_{i,j}}$ from its anchor point, $\mathbf{AP_i}$ as shown in Fig. 6b.

From this information, each 1D section (curve) geometry can be easily determined using interpolation of an FFDP set $\{\mathbf{A_{i,j}}\}$ with i fixed. Similarly, the shape of the whole bump freeform feature can be determined by standard interpolation algorithms through the $\mathbf{N_{1D}}$ 1D sections, e.g. lofting or skinning.

The aforementioned parameters are, by default, private to the feature class definition. However, a subset of them can be declared as input parameters. The input parameters occupy public fields so that they can be accessed from external interfaces.

5.3 Parametrizing and Constraining the Shape

Parametrization involves adding constraints on parameters used to control the shape of the feature. In the example Fig. 6a, we assume circular sections.
The height (H) can be defined as follows:

- let $\mathbf{h_i}$ be the distance of an auxiliary parameter for the i^{th} section (here, $\mathbf{AP_i}$), along the vertical axis of the bump, from the reference parameter ($\mathbf{p_0}$)
- H = max {$\mathbf{h_i}$}, i : [1, **N**], in which H is algebraically related to all $\mathbf{h_i}$.

Changing the height (H) will also change all the h-values proportionally, by means of the algebraic constraint.
The width (W) can be done as follows:

- let $\mathbf{w_{i,j}}$ = distance [$\mathbf{AP_i}$: $\mathbf{A_{i,j}}$], the local width of the $\mathbf{i}^{th}$ section at the $\mathbf{j}^{th}$ FFDP, $\mathbf{A_{i,j}}$, from the auxiliary parameter $\mathbf{AP_i}$
- W = max {$\mathbf{w_{i,j}}$}, i: [1, **N**]; j: [1, $\mathbf{m_i}$] in which W is algebraically related to all $\mathbf{w_{i,j}}$.

Therefore, changing the width, W, affects all sections by the algebraic constraints.

5.4 Fine Shape Geometry

In this section, we describe the fine shape definition. It represents a modification of the basic shape. For example, changing the parameters of a section curve modifies the fine shape of the bump freeform feature. This can be done through changing, for example, one or more parameters of an FFDP, i.e. its angle and/or distance.
In addition, with the flexible definition mechanism, new parameters and constraints can be added to facilitate specific fine shape control. For example, a pressure constraint can be a spot (for a point), directed (for a line) or distributed (for an area or patch). These constraints may be used to define specific parameters for the bump feature class.

5.5 Another Example

The above example provides an insight into how our approach could be used define a new feature class. More complex types of features can also be defined. For example, in Fig. 7, a peak feature is defined from a class derived from the new_bump feature class (see Fig. 4a and Fig. 6a). The section parameters need not be planar; neither should they be (co-)axial. Validity condition may also be specified in the class, e.g. constraints can be declared on parameters as required.

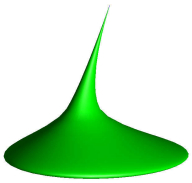


Fig. 7. A variant of a bump freeform feature - a peak.

6. COMPOUND FEATURES

In this section, definition of a compound feature is presented. Most aspects described in earlier sections can be applied in defining a compound feature. Therefore, we focus on the distinguishing concepts. A compound feature treats a group of features as a single unit, e.g. if a feature is so complex that a single feature definition is insufficient. User-defined freeform surface features can be compounded to become larger, functionally meaningful design units.
The compound feature definition is a composition of other freeform feature class definitions. These features become sub-features of the compound class. This structure has the advantage that sub-features can be made to have greater functional independence from each other, while associating them through global attributes in the compound feature class itself.
A compound feature can be composite or pattern [5]. A composite freeform surface feature is a collection of individual features joined to form a more complex feature, e.g. a ridge with a hole considered as a single feature as shown in Fig. 8. A pattern freeform surface feature is a repetitive sequence of a particular feature or groups of features in a prescribed manner, e.g. if the hole feature in Fig. 8 is repeated 3 times where the next instance is linearly displaced and positioned a distance L from the previous.

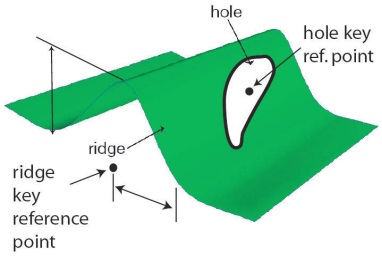


Fig. 8. Composite freeform feature – 'ridge_with_hole'.

6.1 Declaring a Compound Feature

A compound feature is declared just like a single feature using the taxonomy presented in Fig. 4a. Derivation can be made from the compound feature class level or from the pattern/composite class levels. Consider that a new compound freeform feature class is created, derived from the composite class (see Fig. 4a), say bump_with_hole in Fig. 8. Of course such a configuration shown in the figure can be realized in existing modeling methods at instance level from modeling operations involving individual features. In this paper, we argue that such configuration can be made during feature class definition, as single compound features. In addition, it can be argued that a compound feature class declaration can derive from multiple parents, which can be its components. For a compound freeform feature class definition, it is cardinal that every component is represented as a discrete entity in order to avoid possibly conflicting properties (attributes). In that case, if feature classes are added as attributes, their properties become local properties of the compound feature, whereas those properties of the compound feature class itself are global properties that describe the relationships among its constituent entities. With the newly declared compound feature class, its specification can be carried out.

6.2 Specifying a Compound Feature

The main difference between a single feature class definition and a compound class definition are the other class definitions that are, in compound feature class, attributes. In addition, relationships between these features are a requirement for its class definition. All these are presented to the user for specification.

A compound feature class specification can be divided into *local* and *global* scopes. Local scope refers to specifying individual feature classes in the normal manner as described in previous sections. Global scope encompasses adding general parameters and constraints common to all its components, often in relative terms. For example, the key reference points of the respective features in Fig. 8 are defined with respect to a key reference point (global) specified for the compound feature. In that case, all the key reference points of the individual features are considered as auxiliary parameters in the compound feature.

In addition, carrier constraints (see Section 3) are mandatory between parent and child features because they determine dependency between the two features. In this dependency relationship, the hole feature is a child and the ridge is the parent. This dependency implies that the key reference point for a child feature is defined relative to the key reference of the ridge. In general a carrier constraint can specify the coupling between related entities. It can be specified whether deforming the ridge (parent) may correspondingly deform the hole (child) or not. This way, the influence of the parent on the child can be controlled.

Another important property is the nature associated with a freeform feature class definition: additive, subtractive or substitutive. In the example of Fig. 8, the hole is subtractive and therefore, as a child feature, it eliminates material from its parent feature.

6.3 Methods in a Compound Feature Class

In a compound freeform surface feature there are also local and global methods. The local methods are those in the individual class definitions. Global definitions belong to the compound feature class. Thus, the global methods can also invoke one or more of methods from individual feature class definitions depending on purpose, e.g. determining the combined shape geometry of the hole and bump freeform surface features in Fig. 8.

7. CONCLUSIONS

A generic mechanism for defining a freeform surface feature class, pre- or user-defined alike, has been presented. To facilitate generic, custom-tailored features, a taxonomy has been central to declaring new feature classes. This facilitates browsing the taxonomy for feature definitions previously stored in a library, retrieving the definitions, and declaring a new feature class.

The declared feature class can be specified with attributes and/or methods. An important aspect in modeling is ensuring that definitions are valid. Validation can range from trivial syntax checks to more involved prototype instancing.

Validated definitions are then stored in a library. The main objective when creating a feature class definition is the need to archive all information about the feature in the library. Some of the information includes the name of a feature, attributes (parameters, constraints, etc) with semantics represented by a text string, generic shape information about representation of geometry (e.g. groups of entities like faces and relationships), and methods that allow regeneration of shape if design parameters are changed.

Future work will involve methods to completely validate new class definitions as well as the development of a detailed structure of the library. This will be implemented with the help of a Freeform Feature Library Management System (FFLMS) that will serve as an interface between the design interface and the library of features.

8. ACKNOWLEDGEMENTS

This work is supported by the Netherlands Organization for Scientific Research (STW); project code DIT. 6240.

9. REFERENCES

[1] Bidarra, R. and Bronsvoort, W. F., Semantic feature modeling, *Computer-Aided Design*, Vol. 32, No. 3, 2000, 201-225.

[2] van den Berg, E., Bronsvoort W. F. and Vergeest J. S. M., Freefrom feature modelling: concepts and prospects, *Computers in Industry*, Vol. 49, No. 2, 2002, pp 217-233.

[3] Poldermann, B. and Horváth, I., Surface-based design based on parametrized surface features, In: 'Horváth I. and Varadi K. (Eds.), Proceedings of the International Symposium on Tools and Methods for Concurrent Engineering, 29-31 May, Budapest, Hungary, Institute of Machine Design, Budapest', 1996, pp 432-446.

[4] Fontana, M., Giannini, F., and Meirana M., A freeform feature taxonomy, In: Brunet, P., Scopigno, R. (Eds.), Proceedings of Eurographics, Computer Graphics Forum, Vol. 18, No. 3, 1999, pp 107-118.

[5] Nyirenda P. J., Bronsvoort W. F., Langerak T. R., Song Y., and Vergeest J. S. M., A generic taxonomy for defining freeform feature classes, *Computer-Aided Design and Applications*, Vol. 2, Nos. 1-4, 2005, pp 497-506.

[6] Bidarra R., Idri A., Noort A., and Bronsvoort W. F., Declarative user-defined feature classes, In: CD-ROM Proceedings of the 1998 ASME Design Engineering Technical Conferences, 13-16 September, Atlanta, USA, ASME, New York.

[7] Hoffmann, C. M. and Joan-Arinyo, R., On user-defined features, *Computer-Aided Design*, Vol. 30, No. 5, 1998, pp 321-332.

[8] van den Berg E., van der Meiden, H. A. and Bronsvoort, W.F., Specification of freeform features, In: Proceedings Solid Modeling '03, Eighth ACM Symposium on Solid Modeling and Applications, 16-20 June, Seattle, USA, Elber, G. and Shapiro, V. (Eds), ACM Press, New York, 2003, pp 56-64.

[9] Piegl, L. and Tiller, W., The NURBS Book, Springer-Verlag, 1997.

[10] Pernot, J. P., Gillet, S., Léon, J. C., Falcidieno, B. and Giannini, F., Shape tuning in fully free-form deformation features, *Journal of Computing and Information Science in Engineering*, Vol. 5, No. 2, 2005, pp 95-103.