Developments in Feature Modelling

Willem F. Bronsvoort, Rafael Bidarra and Paulos J. Nyirenda

Delft University of Technology, w.f.bronsvoort/r.bidarra/p.j.nyirenda@tudelft.nl

ABSTRACT

Feature modelling is nowadays the predominant way of product modelling, in which geometric and functional information is stored in a single product model. The basic concepts of feature modelling in general, and the state of the art in commercial feature modelling systems, will be presented first. This will be followed by an overview of four major developments that solve shortcomings in such systems. First, in semantic feature modelling, it is possible to more adequately specify and maintain the meaning of features. Second, in multiple-view feature modelling, there is a specific feature model for each product development phase. Third, in collaborative feature modelling, teams of users can collaborate on the development of a product with full feature modelling functionality. Fourth, in freeform feature modelling, features with freeform shapes are made available. Some other developments are mentioned as well.

Keywords: feature modelling, feature semantics, multiple views, collaborative modelling, freeform features.

1. INTRODUCTION

During the last decade, feature modelling has become the most popular way of product modelling. In addition to geometric information, functional information can be stored in a feature model, and this can considerably enhance the capabilities of a modelling system.

Considering the functionality of current commercial feature modelling systems, one can conclude that, on the one hand, these systems provide very advanced modelling facilities, but that, on the other hand, there are still several major shortcomings. This paper presents developments that can solve some of the most important shortcomings. All these developments are explained in the context of research projects performed at Delft University of Technology.

The first shortcoming in current feature modelling systems considered here is that the meaning, or semantics, of features is not adequately maintained during the whole modelling process. The semantic feature modelling approach does handle this. How the semantics of all features in a feature model is specified in this approach, and how the validity of the model is maintained during the whole modelling process, is described.

The second shortcoming is that product models with multiple, integrated feature views for different product development phases are not yet possible. A view is an application-specific feature model, with features relevant for that application. An approach that does support integrated views for conceptual design, assembly design, part detail design and part manufacturing planning is presented.

The third shortcoming is that collaboration of several users in developing a product with a feature modelling system is not yet adequately supported. A web-based collaborative feature modelling approach that offers interactive modelling facilities, but also functionality such as validity maintenance and multiple views, is described.

The fourth shortcoming is that mostly only regular-shaped and simple freeform features can be used, whereas in practice products often contain more complex freeform shapes. Because of the many benefits of feature modelling, it is worth developing techniques for freeform feature modelling as well. Some initial developments in this direction are discussed.

The structure of this paper is as follows. In Section 2, some basic concepts of feature modelling will be introduced. In Section 3, the state of the art in current commercial feature modelling systems will be discussed, and the shortcomings mentioned above will be revisited. In Sections 4 to 7, the developments to solve these shortcomings will be elaborated. In Section 8, some conclusions about the developments discussed in this paper, and about other developments in feature modelling, are given.

2. FEATURE MODELLING

Feature modelling can be considered as an extension of geometric modelling. Whereas in a geometric model only geometric information is stored in a product model, in a feature model functional information is associated with the geometric information. In this section, some basic terminology of feature modelling is introduced, and also some ways of working with features are mentioned.

A generally accepted definition of a *feature* is that it is a representation of shape aspects of a product that are mappable to a generic shape and functionally significant for some product life-cycle phase. Functional information, e.g. on the use of the shape for the end-user or on the way the shape can be manufactured, can be associated with the shape information [1].

Several types of features can be distinguished; protrusions, holes, slots and pockets being typical examples of frequently used ones. Although several attempts have been made to develop a complete classification of features, this has turned out to be very difficult. It is therefore important that new types of features can be easily introduced in a feature modelling system.

All properties of a feature type are specified in the corresponding *feature class*, which defines a template for all its instances. This always includes the generic shape of the feature, and a number of *parameters* and *constraints* that characterise this shape. Examples of parameters are the length, width and depth for a rectangular pocket; examples of constraints are that the side faces of a rectangular pocket are pairwise parallel and pairwise perpendicular, and that the radius of a blind hole should be between 1 and 3 cm. Constraints are also used to define the parameterisation of a feature class.

By specifying values for the parameters, an *instance* of a feature class can be created and then be added to a *feature model*. The instance is normally *attached* to other features in the model, i.e. some of its faces are coupled to faces of other features. Additional constraints can be used to relate the instance to other feature instances in the model. Examples of such constraints are that (the side faces of) two slots should be parallel, at a certain distance, and that the diameter of a hole should be half of the width of the protrusion it is attached to. So a feature model essentially consists of a set of feature instances and a set of constraints.

A feature model is usually represented by a graph and a geometric model of the resulting shape. The graph contains all feature instances, with their shape, constraints and attach relations, and all additional constraints. The geometric model can be a boundary representation, but also a more extended representation, such as a cellular model [2], that is more suitable for advanced facilities such as validity maintenance (see Section 4).

There are many applications of features. In product design, generic shapes with some function for the end-user of the product can be considered as features. In process planning for manufacturing, volumes in a product that can be manufactured with a single or a sequence of machining operations are considered as features. Other examples can be given, but it is important to realize here that each application can have its own way of looking at a product, i.e. its own feature model of the product, with features relevant for that application. Such an application-specific feature model is called a *view* on the product [3].

Independent of the application, there are basically three ways to determine features in a product model: design by features, feature recognition and feature conversion.

In *design by features*, a designer specifies a feature model in an interactive modelling system, via a graphical user interface. He can create instances of feature classes in a library, by specifying values for the parameters, and add them to the model. In some systems, also new feature classes, so-called *user-defined features* [4] can be created by a user, after which instances of these classes can be added to the model. Feature instances can also be modified, by changing their parameters, or be removed from the model. The features that are specified may have a functional meaning for the end-user of the product, but can also be manufacturing features. In the latter case, the designer specifies a model that, more or less, corresponds to the way the product will be manufactured.

In *feature recognition*, features are recognised from a geometric model of a product. Historically, this was the first method to identify features in a model, introduced in the context of manufacturing planning. Many methods for feature recognition exist, each one with its own advantages and disadvantages [1]. Among the most important categories of feature recognition methods are rule-based methods, graph-based methods, and geometric reasoning methods. Some successful feature recognition systems use combinations of these methods.

Once a feature model of a product has been created, either by design by features or by feature recognition, other feature models of that product, which correspond to other views, can be derived by *feature conversion*. For example, a manufacturing planning view can be derived from a design view. Feature conversion is a relatively new technique, and forms the basis for multiple-view feature modelling systems (see Section 5).

The rest of the paper will mainly be about design by features, i.e. feature modelling approaches in which feature models are created from scratch.

3. STATE OF THE ART IN COMMERCIAL SYSTEMS

The past two decades have shown a significant development in product modelling in general. More and more powerful CAD systems began to emerge, providing, for example, better user-interaction facilities. In particular, 3D geometric modelling systems became more popular to industry, such as the car design and manufacturing sector. Further down the years, in the past decade, the geometric modelling systems evolved into even more powerful modelling systems embracing the feature concept.

Several commercial feature modelling systems are now available [5], including, for example, Autodesk Inventor®, CATIA Designer®, Pro/Engineer®, SolidWorks®, and Unigraphics®. These systems offer all sorts of advanced, easy-to-use modelling facilities. The functionality of the systems is more or less the same, generally transparent in their interfaces, particularly the user interface. On the basis of functionality, every system can be seen as a suite of programs used for many purposes, namely, design, analysis and manufacturing, of a virtually unlimited range of products. Some systems are solid modellers, whereas others include surface modelling facilities.

Part models can be created starting from sketches or from imported geometric models. The systems also provide libraries of parts, including platform-specific standard parts, with powerful search facilities that often include graphical feedback. Features can be added to a part through menus. In addition, there are tools to parameterize parts using geometric constraints. Models often evolve into functional designs, which engineers generally call feature models. Assembly models can be created from parts models, normally including relations between them.

The graphical user interfaces, with high-quality images of models, enable users to effectively interact with the models. With these, the user is able to navigate or explore models, e.g. with real-time panning, zooming, view-orientation, selection of specific parts or geometric elements, and many more.

However, to date these systems suffer, in one way or another, from a number of major shortcomings, including the following: features are only a kind of macros, there exists only one feature model of a product, there are limited facilities for collaborative modelling, and there is a limited shape domain. These shortcomings are now explained in some detail.

Firstly, in many current feature modelling systems a ‘feature’ is only a kind of shape macro. The result is that, whereas these macros manifest themselves as features at the user interface, only the geometry resulting at the time of creation of the feature is stored in the product model. Such systems are, in fact, only advanced geometric modelling systems. Other systems store some information about features in the product model, but this information is not consistently checked to ensure that the meaning of all features is adequately maintained during the modelling process. For example, in Fig. 1, a through hole in a model is turned into a blind hole by blocking one of the openings of the hole with a stiffener, without the system even notifying this change. Although the new model is geometrically correct, it is incorrect in the sense that the meaning, or semantics, of the feature has been changed from a through hole into a blind hole.

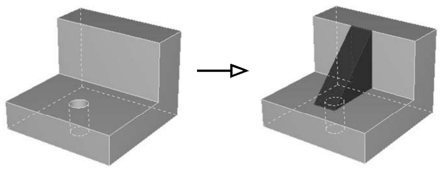

Fig. 1. Changing a through hole into a blind hole.

Secondly, many existing systems work with a single feature model of a product, which might be called a design view. Multiple, integrated views on a product are not supported. If another view on a product is needed, e.g. for a manufacturing planning application, this view has to be derived by a special feature recognition or conversion module, and the new view is not integrated with the design view.

Thirdly, many systems lack high-level feature-based collaborative modelling facilities. Available collaborative facilities comprise a set of technologies that enable companies, together with their customers, to collaboratively design, produce and manage products. The problem of these facilities is that the level of communication is not feature-based, so that functional information is not fully accounted for. This often causes loss in design intent.

Finally, current systems only provide a limited set of shapes for features, namely regular shapes and simple freeform shapes. This includes, for example, (rounded) prismatic shapes, cylindrical shapes, and shapes resulting from the extrusion of a freeform curve, but not general freeform shapes.

In the next four sections, semantic, multiple-view, collaborative and freeform feature modelling are presented in that order. These sections address the aforementioned shortcomings faced in most of the current feature modelling systems.

4. SEMANTIC FEATURE MODELLING

The first shortcoming of current feature modelling systems considered in this paper, is that the semantics of features is not adequately maintained during the whole modelling process. An example of this, in which a through hole was turned into a blind hole, was given in the previous section. This is undesirable, because one of the basic ideas of feature modelling is that functional information can be associated to geometric information. This association, however, becomes useless when the shape imprint of a feature, once added to the model with a specific intent, is significantly modified later by a modelling operation.

Ideally, it should be possible to precisely specify the semantics of features by a set of validity conditions. In addition, all validity conditions of the features in a model should be checked by the system after each modelling operation. If some validity condition is no longer satisfied, e.g. one of the openings of a through hole is blocked, this should be notified by the system, and preferably the user should be assisted in overcoming this situation.

An approach that supports these ideas is *semantic feature modelling* [6]. This approach guarantees that all design intent once captured in a model is maintained, bringing feature modelling to a level really higher than advanced geometric modelling. It involves, in particular, specification of the semantics of features in their respective classes, and maintenance of this semantics during modelling.

Validity conditions in a feature class can be classified into two categories: geometric and topologic. For both, constraints are used. These *feature constraints* are members of the feature class, and are therefore instantiated automatically with each new feature instance.

The geometry of a feature may be constrained by *geometric constraints*, e.g. requiring that some faces should be parallel or perpendicular. Another way of constraining the geometry of a feature, is by restricting the set of values allowed for a shape parameter with *dimension constraints*. For example, the depth of a slot feature could be limited to values between 1 and 10 cm. Finally, the geometry of a feature may be constrained by means of explicit relations among its parameters. These relations can be simple equalities between two parameters, e.g. between width and length of a square passage feature, or, in general, algebraic expressions involving two or more parameters and constants. For this, *algebraic constraints* are used.

The shape of a feature is represented by a set of faces that provides full coverage of the feature boundary. However, for most features, not all faces are meant to effectively contribute to the boundary of the modelled product. Some faces, instead, have a closure role, delimiting the feature volume without contributing to the boundary. The specification of such properties is called *topologic validity* specification. For this, two sorts of constraints are used: boundary constraints and interaction constraints.

A *boundary constraint* states the extent to which a feature face should be on the model boundary. An example of this is a pocket class for which the top face has a *notOnBoundary* constraint, which specifies that a pocket should remain open at the top, whereas the bottom face has an *onBoundary* constraint, which specifies that a pocket should remain closed at the bottom.

Boundary constraints are insufficient to fully describe several other functional requirements that can be inherent to a feature class as well. These are better described in terms of the feature volume or feature boundary as a whole, and can be violated by feature interactions caused during incremental editing of the model. *Feature interactions* are modifications of shape aspects of a feature that affect its functional meaning. An example of this is that a slot is being split into two separate, smaller slots by intersecting it with another slot; this is called *splitting interaction*. *Interaction constraints* are used in a feature class to indicate that a particular interaction type is not allowed for its instances.

In addition to feature validity conditions, there can also be *model validity conditions*, which specify conditions on or between specific instances in a feature model. For specifying such conditions, the same set of constraints is available as for specifying validity conditions in feature classes.

Feature model validity maintenance is the process of monitoring each modelling operation in order to ensure that all feature instances still conform to the validity criteria specified for them, and that all model validity conditions are still satisfied. Validity maintenance can be split into two types of tasks: (i) validity checking, performed at key stages of each modelling operation; and (ii) validity recovery, performed when a validity checking task detected a violation of some validity criterion.

During validity checking, it can be detected that a constraint is no longer satisfied because of a modelling operation, e.g. that a notOnBoundary constraint on the top face of a pocket is violated by blocking the face with another feature. In that case, the model enters an invalid state, and a valid model should be achieved again. This is straightforward if the operation is cancelled: all that is needed is to backtrack to the valid model state just before executing it, by 'reversing' the invalid operation.

However, to always have to recover from an invalid operation by undoing it is too rigid. It is often much more effective to constructively assist the user in overcoming the constraint violations, in order to restore model validity. In most cases, if the user receives appropriate feedback on the causes of an invalid situation, it is likely that other corrective actions might preferably be chosen. The process of validity recovery therefore includes reporting to the user constraint violations, documenting their scope and causes, and, whenever possible, providing context-sensitive corrective hints. These automatically generated hints suggest solutions for violated constraints, and vary with the type of constraint involved. In the example of the blocked top face of a pocket, the feature that blocks the face is reported to the user, who can then reposition it to solve the problem.

In order to overcome the invalid model situation, the user can specify several modelling operations in a batch. Execution of these *reaction operations* follows the same scheme as normal operations, which means that their outcome is checked for validity. At any stage when the model is invalid, the user may give up attempting to fix it, and backtrack to the last valid stage.

The semantic feature modelling approach has been implemented in the SPIFF prototype feature modelling system developed at Delft University of Technology. The system uses a feature dependency graph and a cellular model [2] as main model representations. The cellular model can be used to check, for example, that the top face of a pocket is still open, and that a slot has not been split. Although a cellular model has numerous advantages, e.g. its evaluation is very efficient [7], other systems based on ideas from the semantic feature modelling approach use other representations [8, 9].

5. MULTIPLE-VIEW FEATURE MODELLING

One of the latest trends in product development is *Product Lifecycle Management* (PLM), which involves a set of technologies and processes that enable companies, together with their suppliers and customers, to collaboratively design, make and manage products throughout their entire lifecycle [10]. An important requirement for PLM is that a designer working in any product development phase can work with information that is relevant for that phase, without being diverted by information that is relevant for other phases only. However, the information for all development phases should be integrated, so that no inconsistencies arise.

Current feature modelling systems do not adequately support this. Some systems have specific models for different development phases, but these models are not very well integrated. Other systems do integrate information for, for example, assembly design and part detail design, but these systems typically do not have specific models for these development phases.

Multiple-view feature modelling can do better here, by providing an own view on a product for each development phase, and integrating all views. Each view contains a feature model of the product specific to the corresponding phase. Since the feature models of all views represent the same product, they have to be kept *consistent*.

Quite a lot of research has been done on multiple-view feature modelling during the last years. Most of the approaches that have been developed work on a single part of a product. A new view on a part can be derived from an existing view by feature conversion, the process of converting one feature model into another feature model. Several views on a part can be simultaneously maintained by the system. Modifications made in one view, are automatically propagated to the other views on the part, also by feature conversion, although not all approaches propagate modifications in both directions. The approaches can be used to, for example, support design for manufacturing: while a part is being built with design features, these feature can be immediately converted into features in a manufacturing planning view, which can be used to concurrently check the manufacturability of the product.

Such initial approaches to multiple-view feature modelling have been presented by, among others, de Kraker et al. [11], De Martino et al. [12] and Hoffmann and Joan-Arinyo [13]. All these approaches share two major shortcomings. First, the approaches focus on the later product development phases in which the geometry of the part has been fully specified, and, therefore, cannot be applied in the early product development phases such as conceptual design. Second, the approaches deal with a single part only. Real products, however, rarely consist of a single part. Dealing with products that consist of multiple parts, i.e. assemblies, does not only involve the separate parts, but also the relations between these parts.

To counteract the disadvantages mentioned, and thus to better support the concept of PLM, an approach to multiple-view feature modelling has been developed that supports more product development phases, in particular conceptual design, assembly design, part detail design, and part manufacturing planning [14].

In the *conceptual design view*, the product configuration can be modelled with conceptual components, which are to be implemented by one or more parts, and interfaces between these conceptual components, which are to be implemented by a connection. Components are built from a base shape, concepts, such as depressions and protrusions, and reference elements. Interfaces between components are characterised by means of degrees of freedom between the components. The complete geometry of the components does not have to be specified. For example, for some concept only certain properties, such as its maximum volume, might be specified.

In the *assembly design view*, the connections between the components in the product can be modelled with *assembly features*, in particular *connection features* [15]. A connection feature needs to be created for each interface in the conceptual design view, and linked to that interface. The interface and the connection feature should reduce the same freedom. In order to accommodate the connection feature, features may have to be created on the components in the assembly design view, e.g. a pin feature and a hole feature for a pin-hole connection feature.

In the *part detail design view*, the detail shape of the parts can be modelled, typically with features such as through holes and rectangular protrusions. It allows the designer to refine the parts that are represented by the components in the conceptual design view, and which may have been refined in the assembly design view to accommodate connection features. Features may be created for concepts in the conceptual design view, and linked to those concepts. A feature should satisfy the requirements specified for the corresponding concept, e.g. that the volume should be less than 80 cm^3.

In the *part manufacturing planning view*, the way each part is to be manufactured is determined. Manufacturing planning features are similar to detail design features, but note that the set of features for the manufacturing planning view on a particular part can be quite different from the set of features that has been used to design the part, simply because a product can look different from the points of view of design and manufacturing planning. This view allows the designer to analyse the parts for manufacturability and to create a manufacturing plan for them. The feature model in a manufacturing planning view is automatically linked to the feature model in the corresponding part detail design view. See Fig. 2 for the part detail design view and the part manufacturing planning view on a part.

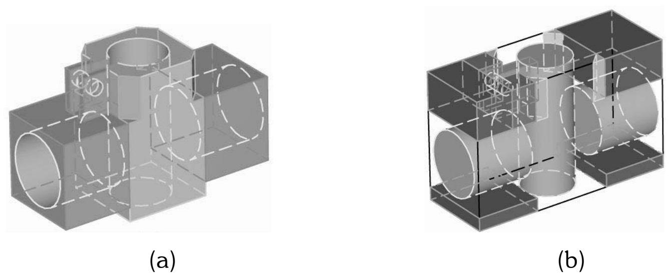
(a) (b)

Fig. 2. Part detail design view (a) and part manufacturing planning view (b) on a part.

Consistency maintenance integrates all views on a product, by ensuring that their feature models remain consistent. It checks the consistency of pairs of feature models, based on consistency definitions specified for these pairs. If an inconsistency is found, it recovers the consistency of the models. This involves, among other things, constraint checking and feature conversion.

The multiple-view feature modelling approach has been implemented in the SPIFF prototype feature modelling system.

6. COLLABORATIVE FEATURE MODELLING

Another important requirement for PLM, in addition to the availability of multiple views on a product, is that several designers can collaboratively model a product. *Collaborative modelling systems* are distributed multiple-user systems aimed at supporting engineering teams in co-ordinating their modelling activities. So far, mainly tools have been developed that somehow support collaborative design activities. For example, tools for collaborative *viewing and markup* of models via Internet are available, providing concepts such as shared cameras and telepointers. However, such tools are primarily focused on inspection, e.g. using simple polygon mesh models, and do not support real *modelling* activities. In other words, they are valuable assistants for teamwork, but no real modelling systems.

At Delft University of Technology, the prototype collaborative feature modelling system webSPIFF has been developed that offers advanced feature modelling facilities to its users, while at the same time providing them with the necessary co-ordination mechanisms to support an effective collaboration. webSPIFF provides a scheme for hierarchically structuring a product into components and parts, and meanwhile assigning each component and part to one or more team members, responsible for its actual development [16]. Typically, in a collaborative session to develop a component or part, participants would be provided with their own, application-specific views on the product model. The co-ordination mechanisms should solve the critical problems of concurrency and synchronisation that characterise collaborative design environments. *Concurrency* involves management of different processes trying to simultaneously access and manipulate the same data. *Synchronisation* involves propagating modified data among users of a distributed application, in order to keep their data consistent.

webSPIFF has a *client-server architecture* [17]. The clients perform operations locally as much as possible, in particular regarding visualisation of, and interaction with, the feature model. Only high-level information messages, e.g. specifying modelling operations, as well as a limited amount of model data necessary for updating the client data, are sent over the network. The server co-ordinates the collaborative session, maintains a central product model, and provides all functionality that cannot, or should not, be implemented on the client. In particular, as soon as real feature modelling computations are required, these are executed at the server, on the central product model, and their results are eventually exported back to the clients. An important advantage of the client-server architecture is that there is only one central product model in the system, thus avoiding inconsistency between multiple versions of the same model. This type of architecture is therefore also used by other collaborative modelling systems [13].

As a basis for the server, the SPIFF feature modelling system was taken, which offers several advanced modelling facilities, including validity maintenance functionality and multiple views, as discussed in the previous sections. A *session manager* stores information about an ongoing session and its participants. It manages all communication between webSPIFF clients and the SPIFF modelling system. Since several session participants can send modelling operations and queries to the server at the same time, concurrency must be handled at the session manager. It is also the task of the session manager to synchronise session participants, by sending them updated model data when needed.

The clients of webSPIFF are using standard web browsers. A user can join an ongoing collaborative session, or start a new one, by specifying the product model he wants to work on and the desired view. The current version of webSPIFF provides views for assembly design, part detail design and part manufacturing planning. Information on the feature model of the view is retrieved from the server.

Obviously, clients should be able to specify modelling operations in terms of features and their entities; for example, a feature, to be added to a model, should be attachable to entities of features already in the model. After a feature modelling operation, with all its operands, has been fully specified, it is sent to the server, where it is checked for validity and scheduled for execution. This can result in an update of the product model on the server, and thus also of the feature model in the view of each session participant.

In addition to the above functionality, several visualisation and interactive facilities of the SPIFF system have also been ported to the clients. webSPIFF clients provide two ways of visualising the product model. First, a high-quality *feature model image*, generated at the server, can be displayed. Second, a *visualisation model* can be rendered that supports interactive modification of viewing parameters, e.g. rotation and zoom operations. After the desired viewing parameters have been interactively set, they are sent to the server, where a feature model image corresponding to the new parameter values is rendered and sent back to the client, where it is displayed. webSPIFF also uses a *selection model*, providing facilities for interactive specification of modelling operations, e.g. selecting features or feature entities on a feature model image.

To support all these facilities, the webSPIFF clients need to locally dispose of some model data. This data is derived by the server from its central model, but it does not make up a real feature model. webSPIFF clients have just enough model information to be able to autonomously interact with the feature model, i.e. without requesting feedback from the server. Client model data is never modified directly by the clients themselves. Instead, updated model data is sent back to the client after a modelling operation has been executed at the server. In addition, appropriate updated model data is sent to all other session participants as well. This consists of, possibly several, new feature model images, a new visualisation model, and an incremental update of the selection model.

A good distribution of the functionality between the server and the clients has resulted in a well-balanced system. On the one hand, the server provides participants in a collaborative modelling session with the advanced functionality of the SPIFF feature modelling system. On the other hand, all desirable interactive modelling functionality is offered by the clients, ranging from display of feature model images, to interactive model specification facilities.

7. FREEFORM FEATURE MODELLING

In almost all current feature modelling systems, only regular-shaped and simple freeform features, called simple features in this section, can be used. However, products in practice often contain more complex freeform shapes. Feature modelling with a shape domain extended to such freeform shapes is called *freeform feature modelling*.

Freeform features can be defined in a similar way as simple features. They still correspond to generic shapes, but there is more modelling freedom for the shapes. A distinction can be made between freeform surface features and freeform volumetric features. *Freeform surface features* consist of one or more freeform surface patches, and can be used for modelling thin artifacts like panels and car bodies; see Fig. 3 for three examples. *Freeform volumetric features* consist of a volume bounded by a set of freeform faces, and can be used for modelling volumetric parts with freeform faces;

features like protrusions and depressions can still be used in this context, but now with freeform faces. Typically, the freeform surface patches or freeform faces are modelled with NURBS [18].

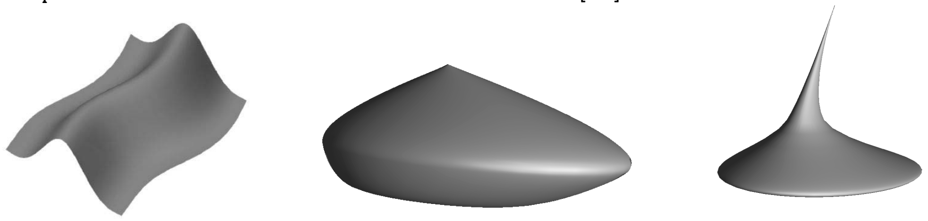
Fig. 3. A ridge, a bump and a peak freeform surface feature.

Many concepts mentioned in Section 2 for simple features can be mapped to freeform features in a straightforward way. A detailed survey of the basic ideas of freeform feature modelling is given in [19]. That survey shows that research on freeform feature modelling is still in its preliminary stage. Here, some recent developments are mentioned, and a number of open issues are indicated. The following will refer to both freeform surface feature and freeform volumetric features, unless stated otherwise.

The definition of a *freeform feature class* is more complicated than for simple features. The generic shape now has to be modelled with, for example, NURBS. In addition, a meaningful set of parameters has to be chosen that makes intuitive instantiation and modification of the feature possible, and a mapping between these parameters and the low-level definition entities of the NURBS has to be established.

Because of the large shape domain of freeform features, it is impossible to predefine all feature classes in the context of freeform feature modelling. It is, therefore, very important that new user-defined feature classes [4] can be created. A way to do this for freeform volumetric features, which makes use of special definition points and constraints between these points, is presented in [20], and a way to do it for freeform surface features, which makes use of a general classification of freeform surface features [21], is presented in [22].

Freeform feature instances can again be created in a model by determining values for the parameters, but their 'attachment' to other features is more complicated than for simple features, as the feature faces that are coupled can have complex shapes, and therefore usually do not match. A way to realise attachment for some types of freeform volumetric features is presented in [23].

A freeform feature model can still be represented by a graph, with all feature instances, attach relations and model validity constraints, and a geometric model of the resulting shape, e.g. a boundary representation or a cellular model with NURBS patches for freeform volumetric features.

Many difficult problems remain to be solved to develop freeform feature modelling into a mature concept. Some of these problems are indicated here.

The idea that properties that correspond to functional information can be included in freeform feature classes and in freeform feature models, by *validity conditions*, and that these properties are maintained during the whole modelling process, has hardly been explored, but seems nevertheless very promising. One can easily imagine conditions that all instances of some freeform feature class have to satisfy, e.g. that the curvature of a surface patch or a face is limited, or conditions on interaction between feature instances, e.g. that splitting by another feature is not allowed. Such validity conditions can again be specified with constraints, which may become quite complex. *Validity specification* and, in particular, *validity maintenance* for freeform feature modelling are major issues of our current research.

Several methods have been proposed for *freeform feature recognition* [19], but better and more general methods are required. One interesting method that is being developed further is working with template matching [24]. Considering the large variety of available methods for recognition of simple features, much more research is to be expected in this area.

In a joint project with colleagues in the Faculty of Design Engineering of Delft University of Technology, we are currently integrating freeform feature recognition and design by freeform features in a single system. Given a geometric model, a freeform shape in the model may be recognised as a freeform feature, after which new instances of the corresponding feature class, satisfying the constraints included in the class, may be added to the model.

Feature conversion and *multiple-view feature modelling* are subjects that have not yet been tackled in the context of freeform features. Several topics mentioned above come together here, such as validity specification and maintenance, and feature recognition.

8. CONCLUSIONS

Feature modelling is an attractive way of product modelling, because of the possibility of integrating functional and geometric information, but current feature modelling systems still have major shortcomings. Developments to solve four of these shortcomings were discussed in this paper.

In this section, some conclusions about these developments, some future work related to these developments, and some other interesting research areas in feature modelling are discussed.

The semantic feature modelling approach has turned out to be a very useful and powerful way of fulfilling a major goal of feature modelling: capturing and maintaining design intent in a product model. However, more research should be done on which semantics is relevant for product modelling, and how to specify and maintain it.

Multiple-view feature modelling is an attractive approach to support multiple product development phases, e.g. in a PLM context. Many extensions can be thought of. In particular, new types of views can be introduced, e.g. idealised views of models for finite-element analysis, and conversion between views can be further automated.

Collaborative feature modelling is a very interesting option for collaborative modelling, because it can offer all advantages of multiple-view feature modelling to the participants in a collaborative session. Several aspects require further attention, e.g. authorisation rights and security issues.

Freeform feature modelling is a very promising and useful extension of feature modelling to cover a larger shape domain. However, many intricate problems remain to be solved, in particular validity maintenance and conversion of freeform feature models.

Beside the developments discussed in this paper, there is much more work going on to further enhance the functionality of feature modelling systems. This includes research on:

- more powerful constraint solving techniques [25]
- families of objects with well-defined semantics [26]
- deformation techniques for freeform features [27]
- exchange of feature models between different systems, without losing semantic information [28]
- exploitation of the feature concept in other applications, e.g. advanced manufacturing [29].

Altogether, the developments mentioned will contribute to making feature modelling more mature, and thus even more attractive as the way of product modelling in the future.

9. ACKNOWLEDGEMENTS

We thank Eelco van den Berg, Maurice Dohmen, Winfried van Holland, Klaas Jan de Kraker, Rick van der Meiden and Alex Noort for their contributions to the work described in this paper.

10. REFERENCES

[1] Shah, J. J. and Mäntylä, M., Parametric and Feature-based CAD/CAM - Concepts, Techniques and Applications, John Wiley & Sons, New York, 1995.

[2] Bidarra, R., de Kraker, K. J. and Bronsvoort W. F., Representation and management of feature information in a cellular model, *Computer-Aided Design*, Vol. 30, No. 4, 1998, pp 301-313.

[3] Bronsvoort, W. F. and Jansen, F. W., Feature modeling and conversion - Key concepts to concurrent engineering, *Computers in Industry*, Vol. 21, No. 1, 1993, pp 61-86.

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

[5] http://www.geometricsoftware.com/

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

[7] Bidarra, R., Madeira, J., Neels, W.J. and Bronsvoort, W. F., Efficiency of boundary evaluation for a cellular model, *Computer-Aided Design*, Vol. 37, No. 12, 2005, pp 1266-1284.

[8] Gao, S., Chen, Z. and Peng, Q., Feature validity maintenance based on local feature recognition, In: *CD-ROM Proceedings of the 2000 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 10-13 September, Baltimore, USA, ASME, New York, 2000.

[9] Chen, G., Ma, Y. S., Thimm, G. and Tang, S. H., Knowledge-based reasoning in a unified feature modeling scheme, *Computer-Aided Design and Applications*, Vol. 2, Nos. 1-4, 2005, pp 173-182.

[10] Stark, J., Product Lifecycle Management - 21st Century Paradigm for Product Realisation, Springer-Verlag, London, 2005.

[11] de Kraker, K. J., Dohmen, M. and Bronsvoort, W. F., Maintaining multiple views in feature modeling, In: *Proceedings of Solid Modeling '97, Fourth Symposium on Solid Modeling and Applications*, 14-16 May, Atlanta, USA, Hoffmann, C.M. and Bronsvoort, W.F. (Eds.), ACM Press, New York, 1997, pp 123-130.

[12] De Martino, T., Falcidieno, B. and Hassinger, S., Design and engineering process integration through a multiple view intermediate modeller in a distributed object-oriented system environment, *Computer-Aided Design*, Vol. 30, No. 6, 1998, pp 437-452.

[13] Hoffmann, C. M. and Joan-Arinyo, R., CAD and the product master model, *Computer-Aided Design*, Vol. 30, No. 11, 1998, pp 905-918.

[14] Bronsvoort, W. F. and Noort, A., Multiple-view feature modelling for integral product development, *Computer-Aided Design*, Vol. 36, No. 10, 2004, pp 929-946.

[15] van Holland, W. and Bronsvoort, W. F., Assembly features in modeling and planning, *Robotics and Computer Integrated Manufacturing*, Vol. 16, No. 4, 2000, pp 277-294.

[16] Bidarra, R., Kranendonk, N., Noort, A. and Bronsvoort, W. F., A collaborative framework for integrated part and assembly modeling, *Journal of Computing and Information Science in Engineering*, Vol. 2, No. 4, 2002, pp 256-264.

[17] Bidarra, R., van den Berg, E. and Bronsvoort, W. F., A collaborative feature modeling system, *Journal of Computing and Information Science in Engineering,* Vol. 2, No. 3, 2002, pp 192-198.

[18] Piegl, L. and Tiller, W., The NURBS Book, Springer, New York, 1995.

[19] 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.

[20] van den Berg, E., van der Meiden, H. A. and Bronsvoort, W. F., Specification of freeform features, In: *Proceedings of the 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.

[21] 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.

[22] Nyirenda, P. J., Mulbagal, M. and Bronsvoort, W. F., Definition of freeform surface feature classes, *Computer-Aided Design and Applications*, Vol. 3, Nos. 1-4, 2006, pp 665-674.

[23] van den Berg, E., Bidarra, R. and Bronsvoort, W. F., Construction of freeform feature models with attachments, In: *CD-ROM Proceedings of the 2004 ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 28 September - 2 October, Salt Lake City, USA, ASME, New York, 2004.

[24] Song, Y., Vergeest, J. S. M. and Bronsvoort, W. F., Fitting and manipulating freeform shapes using templates, *Journal of Computing and Information Science in Engineering*, Vol. 5, No. 2, 2005, pp 43-52.

[25] van der Meiden, H. A. and Bronsvoort, W. F., An efficient method to determine the intended solution for a system of geometric constraints, *International Journal of Computational Geometry and Applications*, Vol. 15, No. 3, 2005, pp 279-298.

[26] Raghothama, S. and Shapiro, V., Topological framework for part families, *Journal of Computing and Information Science in Engineering*, Vol. 2, No. 4, 2002, pp 246-255.

[27] 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.

[28] Pratt, M. J., Anderson, B. D. and Ranger, T., Towards the standardized exchange of parameterized feature-based CAD models, *Computer-Aided Design*, Vol. 37, No. 12, 2005, pp 1251-1265.

[29] Dhaliwal, S., Gupta, S. K., Huang, J. and Kumar, M., A feature-based approach to automated design of multi-piece sacrificial molds, *Journal of Computing and Information Science in Engineering*, Vol. 1, No. 3, 2001, pp 225-234.