

Multiple-view feature modelling for integral product development

W.F. Bronsvoort *, A. Noort

Computer Graphics and CAD/CAM Group, Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology, Mekelweg 4, NL-2628 CD Delft, The Netherlands

Abstract

To allow a designer to focus on the information that is relevant for a particular product development phase, is an important aspect of integral product development. Unlike current modelling systems, multiple-view feature modelling can adequately support this, by providing an own view on a product for each phase. Each view contains a feature model of the product specific for the corresponding phase. An approach to multiple-view feature modelling is presented that supports conceptual design, assembly design, part detail design and part manufacturing planning. It does not only provide views with form features to model single parts, as previous approaches to multiple-view feature modelling did, but also a view with conceptual features, to model the product configuration with functional components and interfaces between these components, and a view with assembly features, to model the connections between components. The general concept of this multiple-view feature modelling approach, the functionality of the four views, and the way the views are kept consistent, are described.

Keywords: Product development, Feature modelling, Feature views, Consistency maintenance, Conceptual design, Assembly design, Part detail design, Part manufacturing planning

1 Introduction

A major goal of integral product development, which is an important aspect of product lifecycle management, is to allow the designer of any development phase to focus on the information that is relevant for that phase, without being diverted by information that is relevant for other phases only. On the other hand, the information for all phases should be integrated, so that no inconsistency can arise.

Current modelling systems do not adequately support this. Although many systems do integrate information for assembly design and part detail design, these systems do not have specific models for these development phases, but instead have geometric or feature models for both phases [1–3]. For parts, the same information is presented to the designer in both the part modelling context and the assembly modelling context, thus bothering the designer in the assembly context with information that is specific for the part context, and vice versa. In the rare case that separate models are provided for different phases, these models are not very well integrated. For example, Parametric Technology Corporation provides Pro/CONCEPT to support conceptual design, in addition to Pro/ENGINEER, but does not maintain the consistency between the model for the conceptual design phase and the model for the other design phases [4].

Multiple-view feature modelling, on the other hand, can adequately support the ideal situation, by providing an own view on a product for each phase, and integrating all views. Each view contains a feature model of the product specific for the corresponding phase. Obviously, the feature models of all views should represent the same product, and therefore have to be kept consistent [5].

Although a lot of research has been done on multiple-view feature modelling during the last decade, there are still at least two major shortcomings of the current approaches. First, they are restricted to views for the later product development phases, in which the geometry of the product has to be fully specified by form features, and, second, they deal with views for single parts only, whereas real products rarely consist of a single part.

In this paper, an approach to multiple-view feature modelling for integral product development is presented that overcomes these shortcomings. It supports initial product design and design of products with multiple parts, by providing feature views for conceptual design and assembly design, with several new types of features, in addition to form feature views for part detail design and part manufacturing planning.
manufacturing planning. The process that makes sure that all these feature models remain consistent is called consistency maintenance; several new techniques have been developed for this.

Section 2 provides an overview of existing approaches to multiple-view feature modelling. Section 3 introduces the new multiple-view feature modelling approach. Sections 4 to 6 describe the functionality of, respectively, the conceptual design view, the assembly design view, and the part detail design and part manufacturing planning views. Section 7 describes the consistency maintenance of these views. Section 8 gives information on the most important underlying models and methods that have been used in the prototype implementation of the approach. Finally, Section 9 gives some conclusions.

2 Other approaches to multiple feature views

Several other approaches to multiple feature views exist. The approaches that only provide design by features or feature recognition can support the concept in a rudimentary way only. The more advanced approaches make use of feature conversion.

In design by features, a designer constructs a product model using features, such as slots and blends, which is considerably easier and faster than working with low-level geometric elements as in geometric modelling [6]. If multiple views are required, design by features can be used to separately create a feature model for each view. If a change is made in the feature model of one view, then this change needs to be manually propagated to the feature models of the other views, because the views are independent of each other.

Feature recognition is the process of deriving a feature model from a given geometric model. Many different feature recognition methods are available now [7]. If multiple views are required, the feature model of each view can be recognised from the given geometric model. Any change required in the product model has to be made in the geometric model, after which feature recognition has to be re-executed for each view.

Feature conversion is the process of deriving a new feature model for some view from an existing feature model for another view. An advantage of feature conversion over feature recognition is the possibility to use non-geometric information stored in the existing feature model to derive the new feature model.

In one-way feature conversion, typically, the feature models for the analysis and planning views are derived from the feature model of the design view. If the analysis or the planning results in the need to change the product model, then the changes have to be made in the feature model of the design view, and the feature models for the other views can be automatically updated from that model.

Cunningham and Dixon [8] propose a multiple-view system architecture with one-way feature conversion. In their proposal, design by features is used to build the feature model of the design view, and feature conversion is used to derive the feature models of, for example, the finite-element analysis view, the manufacturing planning view and the assembly planning view.

Anderson and Chang [9] propose another approach to one-way feature conversion that mimics process planning procedures using geometric reasoning, to convert design features into manufacturing features.

De Martino et al. [10,11] describe a system for multiple-view feature modelling including one-way feature conversion. The feature model of a view is built from instances of feature classes from a view-specific library. The feature models of the different views are integrated by an intermediate model that contains, among others, a boundary representation of the product and a representation of the product with so-called shape features, i.e. generic protrusions and depressions. The shape features are extracted from the boundary representation of the product by the shape-feature recogniser, which uses a geometric-reasoning approach. A new feature model for a view is created from the model with shape features by a so-called application-feature converter, which uses information from the feature library of the view to find features.

Suh and Wozny [12] propose a multiple-view feature modelling system with one-way feature conversion in which features are defined as a subset of the boundary elements of a part model. A feature conversion algorithm is used to propagate the changes of a designer from the design view to the feature models of the other views. The feature conversion approach uses an intermediate model that is built from so-called fundamental features, i.e. faces, edges and vertices, and fundamental spatial relationships, such as angle and coplanar, that relate them. The feature model for another view is extracted from the intermediate model by a hybrid feature recognition method that combines graph-based and hint-based techniques.

The approaches discussed above only support feature conversion in a single direction, which is far from ideal. If, for example, a designer in an analysis view discovers that a dimension of the product has to be adjusted, then this adjustment cannot be made in that view, but has to be made in the design view. In the design view, however, the feature model is different from the one in the analysis view, and it may be quite difficult to find the right adjustments of the feature parameters in that view. To allow the model of a product to be changed in the feature model of the view in which the need for it arises, multiple-way feature conversion has been developed.

Bronsvoort and Jansen [5] were among the first to propose multiple-way feature conversion as the ideal approach to support multiple feature views. They describe an exten-
sion to the GeoNode feature modelling system [13] that would provide multiple feature views and allow changes to be made in an arbitrary view. Later on, de Kraker et al. [14,15] and Bronsvoort et al. [16] describe a new approach to multiple-way feature conversion and its implementation in a prototype multiple-view feature modelling system called SPIFF. An important aspect of this approach is the use of a cellular model for the representation of the geometry of the feature models. This will be described in Section 8.

Jha and Gurumoorthy [17,18] have taken a somewhat different approach. They describe algorithms to automatically extract a feature model of a part from the geometric model of the part, and to propagate an adjustment in the feature model of one view to the feature models for other views. Feature classes are specified in terms of the characteristic arrangement of the faces of its instances that will overlap with existing faces of the model, and the faces of its instances that will form new faces in the feature model. The algorithm to extract a feature model from a geometric model uses the characteristic topology of the faces of a particular feature class to find instances of that class. The algorithm to propagate adjustments is based on a data structure that incorporates the feature models for all views. Disadvantages of this approach are that it only supports features that are based on a (linear) sweep solid, and that it is quite inefficient in propagating changes.

Hoffmann and Joan-Arinyo [19,20] propose an architecture in which the models for different applications are spread over different CAx systems, and a master model is used to associate the models. Each system is a client of the master model server. A client can deposit some of its product information in the master model, but also associate private information with the information in the master model. For example, a CAD client may deposit a boundary representation of the product to the master model, and a manufacturing planning application may associate private information with the elements of this boundary representation. The master model server informs the clients after a change of information in the master model with which they have associated private information, but the way the client reacts to this information is determined by the client itself. A major advantage of this approach is that it supports a product model to be distributed over different CAx systems, and therefore the integration of existing commercial applications. A disadvantage is that it only allows minor changes to be propagated back to the CAD client. Further, no implementation of the architecture has yet been reported.

The approaches to multiple features views described above, which all use form features, share a number of shortcomings.

First, all approaches are restricted to the later product development phases, in which the geometry of the product has to be fully specified. They therefore cannot be applied in the early product development phases, such as conceptual design, in which the geometry does not have to be fully specified yet. The multiple-view feature modelling approach described in this paper does provide a view to support the conceptual design phase.

Second, all approaches deal with single parts only. Real products, however, rarely consist of a single part. Dealing with products that consist of multiple parts does not only involve dealing with the separate parts, but also dealing with the relations between the parts. The multiple-view feature modelling approach described in this paper does support the design of products that consist of multiple parts, by having the conceptual design view distinguish multiple parts and relations between them, by providing an assembly design view that takes into account multiple parts and the connections between them, and by allowing the designer to simultaneously work on multiple parts using a manufacturing planning view for each part.

3 The new multiple-view feature modelling approach

The multiple-view feature modelling approach described here extends the modelling capabilities to views with incompletely specified geometry and multiple parts. It supports four product development phases: the conceptual design phase, the assembly design phase, the part detail design phase, and the part manufacturing planning phase.

The first phase that is supported is conceptual design. Two important trends in conceptual design exists; functional design and shape design. Functional design focuses on the function of a product, and includes dynamic, mechanical and electrical simulation approaches to create a product with all required functions. Shape design focuses on the shape of a product, and includes sketching approaches [21] and configuration design approaches [22].

Here, the focus is on conceptual shape design, in particular configuration design, since it best fits to the other supported development phases. It allows the designer to model the configuration of a product by specifying functional components, which are to be implemented by one or more parts, and interfaces between them, which are to be implemented by a connection. The complete geometry of the components does not have to be specified.

Components are specified by means of a base shape with shape concepts and references on it. The base shape is used to attach and position shape concepts and references, and gives an impression of the shape of the component. Shape concepts are used to specify functional requirements on the geometry of a component, e.g. that there should be a depression, possibly with a cylindrical shape, on the component. References are used to position interfaces, shape concepts and other references. Interfaces between components are specified by means of degrees of freedom between references on the components.

An example of a model of a modern version of the
The historical high-wheel bicycle that results from the conceptual design phase is shown in Fig. 1(a). The model is built from the components of the product, with their important aspects only, and the interfaces between the components (see Fig. 1(b)). Conceptual design is elaborated in Section 4.

The second phase that is supported is assembly design. Many approaches to assembly design exist. Some support all phases of product design that involve multiple parts, and provide several tools to analyse, for example, whether the components can be assembled [23]. Others focus on a specific aspect, such as the mate and contact relations between components [24], ways to represent complex assemblies [25], or the use of features to represent the way components are connected [26].

The approach to assembly design that is supported here focuses on connection design. It allows the designer to specify the type of connection between components, and the geometry for the connection on the components. In this phase, in fact the interfaces that have been specified in conceptual design are refined.

A component in assembly design can be a single component, which represents a single part, or a compound component, which represents two or more connected subcomponents. The geometry of components only needs to be specified as far as it is involved in connections. Connections are specified on regions of the connected components, such as a rib or a slot, and represent a reduction of the degrees of freedom between the components.

An example of an assembly model for the high-wheel bicycle that results from the assembly design phase is given in Fig. 2. The example shows the connection of the rear wheel to the rear fork, with an additional axis component that is connected to the rear fork with a fixed pen-hole connection, and to the rear wheel with a rotating pen-hole connection. Assembly design is elaborated in Section 5.

The third phase that is supported is part detail design. Here, the shape of a component as it has been specified by shape concepts in conceptual design is refined. The shape of a part is specified by means of high-level model
elements, such as corner protrusion and passage, which are related by constraints, such as distance and parallel constraints.

The model in the detail design view on a part of the high-wheel bicycle is given in Fig. 3(a). The example shows the detail geometry of the part that is used to fix the saddle to the frame.

![Image](a)

Fig. 3. The model of the part that is used to fix the saddle as it results from the part detail design phase (a) and the manufacturing planning phase (b).

The fourth phase that is supported is part manufacturing planning. Here, the manufacturing operations that are needed to create a part are determined, taking into account the capabilities of the available manufacturing equipment.

The model in the manufacturing planning view on the part of the high-wheel bicycle from Fig. 3(a) is given in Fig. 3(b). It represents the regions of a stock of material that have to be removed by manufacturing operations, in order to obtain the part geometry as it has been specified in the part detail design phase.

Part detail design and manufacturing planning are elaborated in Section 6.

To support the product development phases described above, the concept of a view is used. Each phase has its own view on a product, or part of a product. A view contains a feature model that represents only those aspects of the product, or part, that are relevant for the corresponding phase.

For each type of view, there are one or more libraries of feature classes. These store pre-defined, parametrised, functional aspects from the phase that is related to the view. Classes need to be specified only once, and instances of them can subsequently be used in the feature model of this type of view on any product. Feature class specification involves specification of feature elements, validity conditions, and the interface to the designer.

A feature element of a feature class can be a simple element such as a face or an algebraic variable, but also a complex element such as a complete form feature, i.e. a region of the geometry of a product consisting of several faces that has some functional meaning.

The validity conditions of a feature class relate the feature elements of the class in such a way that they form a valid functional unit. They are specified by constraints on the elements, and a feature is said to be valid if all constraints are satisfied.

As a consequence of this declarative way of specifying feature classes, each feature class has an explicit interface, which specifies the data that has to be provided to create an instance of the feature. For example, a certain type of connection feature may require two components, one form feature on each component, and a dimension to create the connection feature, and a protrusion form feature may need a face to attach it, two other faces and two values to position it on the attach face, and values for its height, length and width.

The feature model of a view can be specified from instances of the feature classes in the feature libraries of the view, and, depending on the view, some additional entities, such as model validity conditions.

A feature model should always remain valid, i.e. all validity conditions in the feature model should always remain satisfied. Feature validity maintenance is the process that ensures this. It checks the validity of the model after new model elements have been added, after the values of parameters of existing model elements have been changed, and after a model element has been removed, and it ensures that the validity is recovered when the model has become invalid [27].

A feature model can be visualised using several types of cameras. The cameras use geometry, icons and text to show the shape of the product that is represented by the feature model, the structure of the feature model, and the functional information in the feature model. Most types of cameras have parameters that specify the way the feature model is shown.

A complete product model consists of a feature model of the conceptual design view, a feature model of the assembly design view, and a feature model of the detail design view and the manufacturing planning view for each part in the product (see Fig. 4).

Each feature model represents the aspects of the product that are relevant for its associated product development phase. However, feature models of different views may represent a same aspect, and inconsistencies between the feature models can occur if such aspect is specified dif-
Fig. 4. The views in the multiple-view feature modelling approach.

ferently in two views. Whether or not the feature models of two views are consistent is determined on the basis of the consistency conditions that have been defined between their elements; the models are consistent if all consistency conditions are satisfied. Examples of consistency conditions are that the feature models of two views should represent the same geometry, and that connected components in one view that are associated to connected components in another view should have the same degrees of freedom in both views.

Consistency maintenance is used to make sure that the feature models of the views remain consistent after a model operation in one of the views. It consists of consistency checking, which is used to check whether the feature models of two views are still consistent, and consistency recovery, which is used to make the feature models of two views consistent again if they have been found to be inconsistent. Consistency maintenance is elaborated in Section 7.

The multiple-view modelling approach has been implemented in the prototype SPIFF system, already mentioned in Section 2. Information on this implementation is given in Section 8.

4 Conceptual design view

The conceptual design view supports configuration design, i.e. design of the way a product is built from functional components, and of how these components are related to each other. Neither the components, nor their relations, have to be completely specified.

There are several other modelling systems that support configuration design; three of these will be mentioned here.

Van Emmerik [13] describes the GeoNode system, which supports a hierarchical design scheme in which the product is represented by assemblies, sub-assemblies and components. Assemblies are specified by means of connections between sub-assemblies and components, and the dimensions of such sub-assemblies and components.

Guan et al. [28] describe the GEMCON system, which allows a designer to incrementally model geometric configurations, by allowing vague, along with precise, geometric information, and the gradual evolving of configurations into concrete and precise models. The system adheres to the minimal commitment principle, i.e. it does not require the designer to specify aspects of the model that are not relevant yet.

Csabai et al. [29] describe the 3D Layout Module, which also adheres to the minimal commitment principle. A product is represented by a layout, which specifies the arrangement of the functional components of the product. The abstract elements for representing the functional components are called design spaces. Relations between design spaces are defined by constraints, and determine the kinematic behaviour of the layout. These constraints are defined on interface features, which are references attached to the related design spaces.

Several entities found in these systems, are offered in the conceptual design view of the multiple-view feature modelling approach.

The feature model of the conceptual design view is built from conceptual components, shape concepts, references and interfaces. It can contain several configurations, i.e. sets of conceptual components that are related to each other by interfaces, but also several unrelated components.

A conceptual component consists of a base shape with shape concepts and references specified on it. Furthermore, a conceptual component has a number of physical attributes and a function description.

The base shape, which the designer can build from simple shapes, represents the global shape of the conceptual component, and is used as a kind of outline to position shape concepts and references on. The geometry of a base shape should preferably resemble its detailed counterpart, although this is not mandatory. An example of the base shape of a conceptual component for the base grip component of the bench vice of Fig. 5 is given in Fig. 6.
Fig. 6. Three attached blocks form the base shape of the fixed grip conceptual component of the bench vice.

Physical attributes represent physical properties of the conceptual component, such as volume, weight and material type. In many situations, the value of such attributes is restricted by requirements on the component, such as the restriction that the weight should be less than 10 kg to avoid the product to become too heavy. Four types of restrictions are available to restrict the value of a physical attribute: equal to, less than, greater than, and within a specified range.

The function description allows the user to describe the component; what it is, where it is used for, and any other related information useful during the design process that cannot be specified in the model in another way.

Shape concepts are model elements that can be used to specify functional requirements on the shape of the component.

A shape concept is defined on a component, and specifies a depression or a protrusion that is required for the function of the component. In addition, for each shape concept the way it is attached to the component and the type of shape, i.e. block, trapezoidal block, cylinder or cone, may be included or may be specified to be unknown. If the type of shape has been included, restrictions can be specified for the dimensions of the shape, just like for the physical attributes of a conceptual component.

Classes of shape concepts have been defined, based on their nature, i.e. depression or protrusion, and the way they are attached to the component. A special class “unknown” is available for shape concepts whose nature and/or the way they are attached is not known.

References are model elements that are not part of the geometry of the model, but are used to position other model elements, such as shape concepts, and to relate interfaces to a component.

A reference consists of a geometry, and a positioning and orientation scheme. The geometry of the reference can be a single point, line, arrow or surface, or a combination of such elements with their relative position and orientation specified by constraints. The positioning and orientation scheme places the reference within the model, and consists of geometric constraints.

An interface represents a reduction of freedom between the conceptual components on which it has been defined. It uses constraints to represent the reduction of freedom, but does not represent the way this reduction is enforced. Several classes of interfaces exists, such as the prismatic slide and spherical interface. Additional classes may be added by the user.

Two things are important in an interface class. First, the number of components to be related and the classes of the references on these components are specified. For example, for a prismatic slide interface, a reference with a line and an arrow could be used on each component; the line to indicate the sliding direction and to be able to partially fix the orientation of the components with respect to each other, and the arrow to be able to further fix the orientation of the components. Second, the constraints between the references that represent the freedom that is reduced between the conceptual components are specified. For example, the prismatic slide interface has a co-linear constraint to specify that the components may slide with respect to each other along the lines of the references, and a parallel constraint to fix the orientation of the components with respect to each other.

The feature model of the conceptual design view is represented by configuration graphs.

A configuration graph contains information on the way components are related by interfaces, and information on the shape concepts on the components. All these elements are represented by nodes, and relations between them are represented by edges between the nodes. Edges between a node that represents an interface and nodes that represent a component indicate an interface between the components; an edge between a node that represents a shape concept and a node that represents a component indicates that the shape concept has been specified on the component. As an example, the configuration graph for the bench vice of Fig. 5 is given in Fig. 7.

Fig. 7. The configuration graph for the bench vice of Fig. 5

The feature model of the conceptual design view can be visualised using geometry, graph and table cameras. Geometry cameras visualise a component with its
base shape, shape concepts and references, either separately or together with other components in a configuration. They visualise a base shape by means of its geometry, a shape concept by means of an icon, or its geometry if the nature, attachment, shape type and size have been specified, and a reference by means of an icon.

An example of a geometry camera that visualises the conceptual fixed grip component of the bench vice of Fig. 5 is given in Fig. 8. The camera shows the base shape of the component, an icon for the protrusion shape concept with block shape (square connected to arrow) that functions as the yaw of the fixed grip, an icon for the passage shape concept with block shape (two connected squares) that functions to connect the moving grip to the fixed grip, and an icon for the prismatic slide reference (line connected to arrow) that is used for the prismatic slide interface between the fixed grip and the moving grip.

![Fig. 8. A geometry camera.](image)

Graph cameras visualise the structure of a configuration, i.e. the components and the interfaces between them. They use nodes containing different icons to represent the components and the interfaces, and show how these are related by edges between the nodes. An example of the graph camera for the bench vice of Fig. 5 is given in Fig. 9.

![Fig. 9. A graph camera that shows the fixed grip, moving grip, and wringe component of the bench vice, and the prismatic slide, cylindrical, and hinge interfaces between them.](image)

Table cameras show a table with all the shape concept instances present on a component, their class and their parameters, allowing the user to have an overview of all shape concepts, including the ones that do not allow a graphical representation.

The conceptual design view as described above, adheres to the least commitment principle: only an outline of the geometry of a component has to be specified, restrictions may, but need not, be specified on the attributes of components, the shape type of a shape concept may be specified unknown, and so on. This allows the designer to specify only the most important functional aspects of a product, and leave out irrelevant details.

5 Assembly design view

The assembly design view supports design of connections between components. For each connection, the geometry used for the connection and the type of connection are specified.

There are several other approaches to feature-based assembly design; four of these will be mentioned here.

Deneux [30] presents an approach that supports creation and management of complex assemblies. Assembly features are defined as a generic solution to a design problem of relating components. They are created in a number of steps, each of which further refines the assembly feature and its components.

Cugini [31] presents a feature-based design approach to support top-down design of assemblies used for aeronautics, leaving the design of sub-systems and details to a following phase. An assembly feature expresses a relationship that exists between two or more parts within an assembly. Assembly features are created in a way similar to that described by Ref. [30].

Van Holland and Bronsvoort [26] show that the feature concept is useful in both assembly modelling and planning. An assembly is built from, among others, single parts and connection features between them. A connection feature includes information on the involved form feature types, e.g. a rib and a slot form feature for a rib-slot connection, the final position of the connected components with respect to each other, and the internal freedom of motion, i.e. the relative freedom of the components after they have been connected.

In Section 4, the 3D Layout Module of Csabai et al. [29], which supports top-down design of a product, has already been mentioned. During the materialisation phase of a layout, connection features can be used to realise relations that have been specified between design spaces. A connection feature is here a form feature that is attached to an interface feature of a design space.

The assembly or connection feature entity, which is exploited by all four approaches, is also the most important entity in the assembly design view of the multiple-view feature modelling approach.
Fig. 10. The assembly graph of the bench-vice assembly of Fig. 5.

The feature model of the assembly design view is built from components and connection features, which can be combined into assemblies. It can represent multiple assemblies and components.

Components are the elements of the assembly feature model that are combined, and on which assembly information is defined. They have form features required for the connection features between them attached, but no other form features. The form features are similar to the form features used in the part detail design and part manufacturing planning view (see Section 6).

A component is either a single component or a compound component. A single component represents a part in the assembly model. A compound component encapsulates an assembly for further assembly modelling purposes, by hiding the structure of components and connection features, and dealing only with the boundary of the assembly.

In general, the geometry of the form features of several connection features on a component is disconnected. To give a better insight into the geometry of the feature model, and to ease the specification of form features, the reference geometry, i.e. the geometry of the part or assembly related to that component, is added to the component. Form features can be specified with respect to faces of the reference geometry.

Connection features contain assembly information, which is mainly represented by constraints. Many different classes of connection features are possible, such as the dove-tail and the pin-hole class [26]. New classes may be added by the user.

New connection feature classes can be specified by the user in three steps. First, the form feature classes needed for the connection are specified (see Section 6). Second, the constraints between the form features that reduce the freedom of the connected components are specified. Examples of these constraints for a dove-tail connection are the geometric co-planar constraints between the corresponding side faces of the dove-tail rib and slot. Third, optional attributes for the connection feature may be specified. An example of such an attribute for a dove-tail connection is an offset to specify the relative position of the components after the connection has been established.

The feature model of the assembly design view is represented by assembly graphs, which represent the structure of the assemblies, and geometric models, which represent the geometry of the assemblies.

An assembly graph contains information on the way components are connected by connection features, and subcomponents are combined into compound components. Both components and connection features are represented by nodes, and the relations between them are represented by edges between the nodes. As an example, the assembly graph of the bench-vice assembly of Fig. 5 is given in Fig. 10.

The geometric model of an assembly represents the reference geometries of the components in the assembly and the form features that are used for the connections, and consists of a cellular model (see Section 8).

The feature model of the assembly design view can be visualised using geometry and graph cameras. Geometry cameras are used to visualise the geometry of components and assemblies. A geometry camera on a component visualises the reference geometry without shading, and the geometry of the form features of the connection features on the component with shading, see Fig. 11(a). A geometry camera on an assembly visualises the reference geometries of the components in the assembly without shading, and the geometry of the form features of the connection features in the assembly with shading, see Fig. 11(b).

Two types of graph cameras exist to visualise the structure of assemblies: hierarchical and relational graph cameras [32].

Hierarchical graph cameras visualise the hierarchy of an assembly with its components and subcomponents. They use nodes containing an icon to represent compo-
components, and represent the hierarchy between compound components and their subcomponents by edges between their nodes. Nodes representing compound components can be expanded to give a better insight into the model.

Relational graph cameras visualise the connections between the components. They use nodes containing different icons to represent the components and the connection features between the components, and show how these are related by edges between the nodes. Nodes representing compound components can again be expanded.

The assembly design view described above supports both top-down and bottom-up design of assemblies, by allowing either newly created geometry or existing geometry to be used as a form feature for a connection feature. By also supporting both approaches to be used alternately, the designer is provided maximum flexibility to design an assembly.

6 Part detail design and manufacturing planning views

The part detail design view and the part manufacturing planning view are two part-oriented views that have been developed to support the part detail design phase and the part manufacturing planning phase, respectively.

The feature models of both views are built from form features, model constraints, and references.

Form feature classes can be specified in three steps. First, the shape of the feature class, including shape faces and geometric validity conditions within the shape, and the nature of the class are specified [33]. The shape can be either a predefined base shape or a user-defined compound shape. The nature indicates whether the instances of the feature class represent material added to or removed from the model.

Second, validity conditions, i.e. constraints, can be specified. There can be attach constraints that specify the way the shape of a feature is attached to the model, dimension constraints that specify a dimension of the feature to be within a given range, boundary constraints that specify a feature face to be partly or completely on the boundary of the product or not, and interaction constraints that specify, for example, that no additive feature may overlap with a given subtractive feature.

Third, the positioning and orientation scheme of the feature class is specified. In general, an instance of a feature class is added to a model by attaching it to existing feature faces in the model, and subsequently specifying its position and orientation by giving the distance and angle with respect to existing feature faces.

The feature classes in the feature library of the part detail design view provide the designer with the functionality to specify the detail geometry of a part. An example of these feature classes is a corner protrusion, i.e. a protrusion with block shape of which a corner is aligned with the corner of another feature.

The feature classes in the feature library of the manufacturing planning view represent the operations of some “classical” manufacturing machines, such as drilling and milling machines, and are therefore subtractive. An example of these feature classes is a blind hole, i.e. a depression with cylindrical shape and only one entrance face, which can be manufactured using a drilling operation.

Model constraints specify additional validity conditions for a single feature instance or a set of feature instances, and can be of any type of constraint mentioned above.

References are similar to those used in the conceptual design view (see Section 4), and can again be used to position other model elements in the view.

The feature model of a part view is represented by a feature dependency graph and a cellular model (see Section 8).

The feature dependency graph represents all feature instances of the feature model of a view, each of them with its set of elements (e.g. shape, parameters and constraints), model constraints, and references. These model elements are interrelated by dependency relations. A dependency relation appears if an entity is attached, positioned, or in
some other way related to another entity. As an example, the dependency graph of a simple feature model of a part detail design view is given in Fig. 12.

![Dependency Graph](image)

Fig. 12. An example of a dependency graph with features, model references and a model constraint.

The feature models of the part detail design view and the part manufacturing planning view can be visualised using geometry cameras. Geometry cameras can visualise the feature models in many different ways. They can visualise a model as a line image, with the hidden lines removed or dashed as option, as a shaded image, or as a hybrid image, with lines superimposed on the shaded image. In addition, they can visualise a selected subset of features in a different way, e.g. the model with a line image and the selected features shaded in the colours that have been specified for their feature class, and they can even visualise the faces of the selected features that are not on the boundary of the model [32].

Fig. 13(a) shows a geometry camera on the feature model of the part detail design view of which the dependency graph was given in Fig. 12. The blind hole and pocket design features have been shaded, and the rest of the model has been visualised with dashed hidden lines. Fig. 13(b) shows a geometry camera on the corresponding feature model in the part manufacturing planning view, in which all tool-entrance faces have been shaded, and the rest of the model has again been visualised with dashed hidden lines.

7 Consistency maintenance

An important issue in multiple-view feature modelling is that all feature models that represent (part of) the same product need to be kept consistent. This section first describes how the supported views can be used, e.g. the order in which their feature models can be specified, and the relations between these models. After that, it describes consistency maintenance for the pairs of views for which this is needed.

The development of a product can be started in the conceptual design view.

![Geometry Camera](image)

Fig. 13. A geometry camera can show different aspects of the geometry of a feature model.

After the conceptual design has become more or less stable, the development can be continued in the part detail design views. The set of parts that is created for a conceptual component should satisfy the requirements on the component.

After the geometry of some parts has been developed such that the form features for a connection feature can be created on it, assembly design can be started. For each part, a single component has to be created by giving it a name and relating it to the part. For some assemblies, a compound component can be created. Further, a connection feature should be specified for each interface in the conceptual design view.

After some geometry in the detail design view of a part has been created, the development can be continued in the part’s manufacturing planning view to create a manufacturing plan for the part.

Instead of the order suggested above, also other orders of specifying the feature models of the different views are allowed. In addition, the feature model of a “previous” view can always be adjusted.

Each conceptual component in the conceptual design view has to be related to one or more parts that together implement the functionality of the conceptual component. Parts that do not implement functionality of a conceptual component, may implement functionality of an interface, e.g. an axis that allows two components to rotate with respect to each other, or some other miscellaneous function. All parts have a part detail design view and a part manufacturing planning view.

Each shape concept in the conceptual design view
has to be related to a form feature in either the part detail design view or the assembly design view that implements the shape concept, i.e. the shape concept itself has to be related to the form feature, and the parameters of the shape concept have to be related to the parameters of the form feature. However, form features in the part detail design view or assembly design view may exist that are not related to a shape concept in the conceptual design view. Such form features have a miscellaneous function, e.g. to reduce weight.

Each interface in the conceptual design view has to be related to a connection feature in the assembly design view that implements the interface. Connection features in the assembly design view that are not related to an interface may be used to connect components that together implement the functionality of a conceptual component.

Each single component in the assembly design view has to be related to a part that implements the single component, and each part has to be related to a single component in the assembly design view.

Single components in the assembly design view are related to the conceptual component whose functionality is implemented by the parts that are related to the single components. Obviously, single components that are related to parts that implement miscellaneous functions are not related to a conceptual component.

In many situations, a change in the feature model of one view may conflict with the feature models of other views. In such situations, the feature model of some view needs to be changed in order to make all feature models consistent. The feature model of a view may only be changed to overcome a conflict with a feature model of another view if the constraints with highest priority remain unchanged.

Constraints that are specified in the conceptual design view have higher priority than constraints in all other views. However, the value of parameters in the feature model of the conceptual design view may be updated based on constraints in another view, as long as they keep satisfying the constraints in the conceptual design view. Constraints in the assembly design view generally have higher priority than constraints in the part-oriented views. However, the dimensions of form features and the reference geometry of a single component in the assembly design view may always be automatically updated after the feature model of the detail design view has been changed. Constraints in the part detail design view generally have higher priority than constraints in the part manufacturing planning view. However, some constraints in the part manufacturing planning view may get higher priority than constraints in the part detail design view, in order to ensure that the product can be manufactured.

Consistency maintenance is used to update the feature model of a view to make it consistent with a feature model of another view. Because of the different characters of the views, the feature model of a view cannot always be derived from the feature model of an arbitrary other view, and different techniques are needed for different pairs of views. Consistency maintenance is performed only for the views that are connected in Fig. 4. For each pair, the consistency definition, the way the consistency can be checked, and the way the consistency can be recovered if needed, are now described.

### 7.1 Part detail design view and part manufacturing planning view

The part detail design view and the part manufacturing planning view are consistent if they represent the same geometry [14].

In case the feature model of one of the two views (the source view) is changed, a linking scheme is used to try to propagate the change to the feature model of the other view (the target view), in order to avoid the feature model of the other view to become inconsistent. This scheme uses links between the two feature models, to dimension and position the features of the target view based on the features of the source view. Links are constraints that specify two feature faces, one from either view, to be co-planar [34]. Their implementation is described in Section 8.

After the source view has been changed, consistency checking is performed in order to check whether the target view is still consistent. This is the case if the evaluated geometry of the feature models of the views overlap completely. If the target view has been found to be inconsistent with the source view, its consistency has to be recovered.

Consistency recovery removes existing features from, and adds new features to the feature model of the target view, in order to make it consistent with the feature model of the source view. A feature is removed from the feature model of the target view if it has become invalid or if it overlaps with a region that represents material in one view and empty space in the other view. After all necessary features have been removed, feature recognition is used to search new features to be added to the model of the target view in order to make it consistent [15]. See again Section 8 for the implementation of the feature recognition approach.

### 7.2 Part detail design view and assembly design view

The feature model of the assembly design view is consistent with the feature models of the part detail design views, if each single assembly component and its related part are consistent. A single component and a part are consistent if they have the same geometry, i.e. if no region exists that represents material in one view and empty space in the other view.
Consistency maintenance for the feature models of the part detail design views and the feature model of the assembly design view is similar to consistency maintenance for the feature model of a part detail design view and the feature model of a part manufacturing planning view. Propagation of changes via links is again used to try to avoid the feature model of the assembly design view to become inconsistent with the feature models of the part detail design views.

To determine the consistency of a single component in the assembly design view and the feature model of the part detail design view on the associated part, it is again checked whether their evaluated geometry completely overlaps.

If a change to the feature model of a single component caused the feature model of its associated part to become inconsistent, then the feature model of the part can be made consistent again automatically with feature recognition. If a change to the feature model of a part caused the feature model of the associated single component to become inconsistent, then the feature model of the single component has to be made consistent by the designer.

An example of automatic recovery from inconsistencies between the feature model of the assembly design view and the feature model of a part detail design view, will be given here. The base grip component of the assembly design view on the bench vice of Fig. 5 is adjusted to facilitate a nut-bolt connection feature between the base grip component and the wringe component. The model of the part detail design view on the part that is related to the component and the component itself are shown in Fig. 14(a+b). When the nut form feature (a kind of through hole) for the nut-bolt connection is added to the component, the component becomes inconsistent with the part detail design view on the related part; compare the model of Fig. 14(a) with the model of Fig. 14(c) that contains the new form feature. To make the model of the part consistent with the component again, the system uses feature recognition. The updated model for the part includes a new, automatically created through hole form feature (see Fig. 14(d)).

See Ref. [35] for more details on the integration of part and assembly modelling.

7.3 Conceptual design view and other views

The feature model of the conceptual design view is consistent with the feature models of the assembly design view and the part detail design views if the conceptual components are consistent with the related assembly components and parts, and the interfaces between the conceptual components are consistent with the related connection features between the assembly components.

A conceptual component is consistent with its related assembly components and parts, if the requirements on the volume, material and weight attributes of the conceptual component are consistent with the properties of the parts, and the shape concepts on the conceptual component are consistent with the related form features on the assembly components and parts. A shape concept is consistent with its related form feature, if its class and its shape-type attribute are consistent with the form feature, and the dimensions specified for the shape concept are consistent with the related dimensions of the form feature. The class of a shape concept is consistent with a form feature, if the form feature is attached as intended in the shape concept, and the nature of the form feature is equal to the nature intended in the shape concept. The shape-type attribute is consistent with a form feature if the shape of the form feature has the specified shape type. A dimension requirement of a shape concept is consistent with a related dimension of a form feature, if the value of the dimension of the form feature satisfies the requirement.

An interface is consistent with its related connection feature if their degrees of freedom and, if applicable, the direction of their freedom, are consistent.

A designer can specify different types of associations to relate model elements from the conceptual design view to model elements from the assembly design view and part detail design views.

Component associations can be used to relate a conceptual component to one or more assembly components or parts. If an assembly component or part has already been associated to a part or assembly component, respectively, then an additional association is automatically created be-
between the conceptual component and that other element. If the assembly component happened to be a compound component, then additional associations are automatically created between the conceptual component and all parts that have been associated with a single component within the compound component.

A shape concept - form feature association can be used to relate a shape concept of a conceptual component to a form feature of a related assembly component or part.

A parameter association can be used to relate a parameter of a shape concept, e.g. a dimension, to a parameter of a related form feature. When a shape concept is associated to a form feature, associations are created automatically between the parameters of the shape concept and the form feature that have identical names.

An interface - connection feature association can be specified to relate an interface to a connection feature.

The consistency of the feature models of the conceptual design view and the assembly design and part detail design views is checked by the system after each modelling operation. For each type of association, a straightforward scheme has been developed to check the consistency conditions involved [36].

If consistency checking finds an inconsistency between the feature models of two views, one of the models has to be changed to make them consistent again.

The system provides tips on the aspects of the feature models that have become inconsistent to inform the designer who is supposed to make the feature models consistent again.

The contents of a tip is based on the association that caused an inconsistency to be found and the view of the designer. It specifies the elements of the association and, if applicable, the involved attributes of these elements. The tip is expressed in terms of the view of the designer. For example, in case an inconsistency occurs between a requirement on the volume of a conceptual component and the actual volume of its related part, the tip for the designer in the conceptual design view would be that a requirement on the volume of the conceptual component is not satisfied by the part, whereas the tip for the designer in the part detail design view would be that the volume of the part does not satisfy a requirement on the volume of the conceptual component.

8 Underlying models and methods

The most important underlying models and methods that have been used in the prototype implementation of the multiple-view feature modelling approach are briefly discussed here.

8.1 Constraint model and solvers

The constraint model represents the model elements and constraints from the conceptual graphs (see Section 4), the assembly graphs (see Section 5), the feature dependency graphs (see Section 6), and the link constraints and associations between them (see Section 7). The constraint model is stored as a graph data structure with a node for each constraint variable and constraint, and an edge between each related constraint variable and constraint.

The constraints in the constraint model are solved by a number of constraint solvers and checkers that have been integrated in a solving scheme. An algebraic and geometric constraint solver are repeatedly applied, until none of them changes the model any more [37]. An interaction constraint checker is applied after that. The constraint solvers deal with low-level constraints and constraint variables that have been mapped from the constraints and constraint variables in the constraint model. After a constraint solver has solved such a low-level constraint model, the relevant constraint variables of the high-level constraint model are updated from the variables in low-level constraint model.

The geometric constraint solver is based on the constraint solving approach described by Kramer [38,37], which uses degrees of freedom analysis. The constraint model here contains a constraint variable for each face of a feature and reference element in the high-level constraint model, and one or more low-level geometric constraints for each geometric constraint in the high-level constraint model. The algebraic constraint solver is based on the Sky-Blue solver [39], which uses local propagation. The constraint model here contains a constraint variable for each dimension of a feature and user-defined variable in the high-level constraint model that is used in algebraic constraints, and one or more low-level algebraic constraints for each algebraic constraint in the high-level constraint model.

The interaction constraint checker checks whether no disallowed interactions between feature shapes occurs. It uses information from the cellular model (see next subsection) to check whether the shapes of two features overlap and whether feature faces are on the boundary, and thus to check interaction and boundary constraints. The algorithm of the constraint checker is described in detail in Ref. [27].

8.2 Cellular model

A cellular model represents the evaluated geometry of a feature model. It is a non-manifold geometric model that integrates the contributions from all features in a feature model. It represents a geometry as a set of volumetric cells of arbitrary shape that do not geometrically overlap, in such a way that each cell either lies completely inside
the shape of a feature, or completely outside it. The decomposition of a cellular model into cells is determined by the overlaps between the shapes of the features in the feature model that it represents, i.e. intersections between feature shapes introduce additional cells. Each cell contains information on the features whose volume overlaps with the volume of the cell, and each cell face contains information on the feature faces that overlap with it. In addition, a cell also contains information on the fact whether its volume represents material, i.e. the cell has additive nature, or not, i.e. the cell has subtractive nature; a cell face also contains information on the fact whether it represents part of the boundary of the feature model, i.e. has material on one side and no material on the other side, or not. This information is stored by means of so-called owner lists. Each cell and cell face has such an owner list, which indicates to which features and feature faces it belongs in each view [40]. The nature of a cell in a view is determined by the features in that view that overlap with the cell and the dependencies between them [27]. The cellular model, including its attribute mechanism to maintain the owner lists, was implemented using the Cellular Topology Husk of the 3D ACIS Modeler [41]. Fig. 15 shows the cellular model for a part, in particular the cells that originated from the overlapping chamfers; for simplicity, only one view is assumed here.

The evaluated geometry of each assembly and part is represented by a cellular model. A single component shares its cellular model with the associated part, a compound component with the encapsulated assembly.

The cellular model of an assembly, and its associated compound component, represents the evaluated geometry of the form features of the connection features and the reference geometries of the components. The integration of the evaluated geometry of the assembly and its associated compound component in a single cellular model, supports maintenance of the consistency between the part and the component.

The cellular model is used to check the consistency of the feature models of the views on a part, or the feature model of a view on a part and the feature model of its associated single component. The feature models are consistent if the cells in their cellular model have the same nature for both models. In addition, the cellular model supports the creation of links and form feature recognition.

Links are created between faces of features from different views that share a cell face that is on the boundary of the product, in order to avoid an inconsistency between the feature models of two views in case one of them is changed. As an example, a subset of the links between the feature models of the part detail design view and the part manufacturing planning view on a part is given in Fig. 16. When the feature model of one of these views (the source view) is changed, i.e. when one or more of the feature faces that are on the boundary of the model are moved, then the links propagate this change to the feature model of the other view (the target view), i.e. move the linked faces of the target view, to avoid an inconsistency [34].

8.3 Form feature recognition

Form feature recognition is used to derive the form feature model of a target view from the form feature model of the source view, such that it becomes consistent with the form feature model of the source view. The feature recognition algorithm uses geometric reasoning, and applies a strategy that specifies the order of the form feature classes to search instances for [15]. The algorithm consists of a shape recognition phase, a parameter determination phase, an extraction phase, and an organisation phase.

In the shape recognition phase, the candidates for the shape of a new instance of a feature class are recognised in the cellular model using the shape type and the attach constraints specified in a feature class. A candidate shape consists of a set of cell faces, one for each face of the shape of the feature, which satisfy the geometric constraints of the shape, and the boundary and attach constraints of the feature. For each candidate shape that is found, the param-
Fig. 16. Feature faces of the part detail design view (a) are linked with overlapping feature faces (b) of the part manufacturing planning view (c).

In the parameter determination phase, first, the dimensions of the candidate shape are determined from the location of the cell faces of the candidate shape. After that, an instance of the candidate shape is created. Finally, the remaining constraints of the feature class are checked. The largest candidate shape that satisfies all constraints is selected as the feature shape. If no candidate shape satisfies the constraints, recognition of the feature fails and, again, the strategy prescribes the next feature class.

In the extraction phase, the feature shape is added to the feature model of the target view, thus reducing the inconsistency between the feature models of the source and target views.

In the organisation phase, faces relative to which the feature is positioned, are selected among the existing feature faces in the model. The faces are selected based on information from the feature class of the new feature, such as its shape type, the faces used for positioning, and the attach faces.

9 Conclusions

Up to now, multiple-view feature modelling approaches were merely multiple-view form feature modelling approaches, and therefore only applicable to support the later phases of the product development process. The multiple-view feature modelling approach for integral product development presented in this paper can also support the earlier phases of the product development process.

The successful development of the conceptual design view and the assembly design view has shown that the view concept is also useful in more abstract models and/or with multiple parts and relations between them. The implementation of the views confirmed the advantages of having different views for different product development phases. Designers in a particular development phase are no longer bothered by aspects of the model that result from other development phases, but can fully concentrate on the aspects that are relevant in their phase.

Because of the diverse characters of the views, an accurate consistency definition between the views could not always be automatically created, and therefore the designer sometimes has to be involved in this. He can specify associations between related entities in different views, in addition to the associations that are automatically generated by the system. A valuable extension would be an approach to automatically associate features in the conceptual design view with features in the assembly design and part detail design views.

Automatic consistency checking algorithms have been developed, which check whether the feature models of the views remain consistent after a modelling operation. Again, because of the diverse characters of the views, the consistency cannot always be automatically recovered. In such situations, the system does, however, provide tips to the designer to make the models consistent again.

Altogether, the approach minimises the user interaction that is needed to maintain the consistency of the views, by consulting the user only if the model does not contain enough information to automatically maintain the consistency.

The multiple-view feature modelling approach for integral product development is very flexible with respect to the order in which the development phases have to be performed. The approach supports a designer to start specifying a model of a product in an arbitrary view, and continue with the model for any other view, as preferred. So it supports top-down as well as bottom-up development of products in an integrated way, which is an important requirement for future product development systems.

Acknowledgements

We thank Rafael Bidarra, Maurice Dohmen, Winfried van Holland and Klaas Jan de Kraker for their contributions.
to the work described in this paper.

References


