assembly features in modelling and planning

ISBN 90-9011056-9
NUGI 855

A PostScript version of this thesis is available via :

ftp://ftp.twi.tudelft.nl/TWI/publications/dissertations/W.vanHolland.ps.gz.

This thesis was typeset using teTEX (version 0.4), a TEX / LATEX distribution that includes TEX, LATEX 2ε, METAFONT and many other programs, such as dvips, xdvi and BIBTEX.
The main part of this thesis was written with the use of the XEmacs editor (version 19.14) using AUC-TEX (version 9.4g) on a LinuX (version 2.0.23) operating system.

Assembly Features
in
Modelling and Planning

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. J. Blaauwendraad
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op maandag 3 november 1997 om 10:30 uur
door

Winfried VAN HOLLAND

informatica ingenieur
geboren te Maarssen

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. F.W. Jansen

Toegevoegd promotor:
Dr. W.F. Bronsvoort

Samenstelling promotiecommissie:
Rector Magnificus, voorzitter
Prof.dr.ir. F.W. Jansen, Technische Universiteit Delft, promotor
Dr. W.F. Bronsvoort, Technische Universiteit Delft, toegevoegd promotor
Prof.dr. I. Horváth, Technische Universiteit Delft
Prof.dr.ir. H.J.J. Kals, Universiteit Twente en Technische Universiteit Delft
Prof.dr. H. Koppelaar, Technische Universiteit Delft
Dr.ir. T. Storm, Technische Universiteit Delft
Dr. D.E. Whitney, Massachusetts Institute of Technology
Prof. J.M. Hennesy, Technische Universiteit Delft, reservelid

Preface

This thesis is concerned with the application of the feature modelling concept, originally developed for single-part manufacturing, to assembly modelling and planning. When you are expecting that the complete assembly process will be automated from now on, then this thesis may disappoint you. However, this thesis shows that some aspects of assembly can be more automated and easier used with the here described assembly features.

The research was carried out at the Computer Graphics and CAD/CAM group of the Department of Computing Science of Delft University of Technology. It was professor Denis McConalogue who headed the CAD/CAM group when I became a PhD student. Because of his superannuation, professor Erik Jansen became my first promoter. Without the ever motivating work of my co-promoter and supervisor, Wim Bronsvoort, who invited me to join the group as a PhD student, this work could not have been finished. He has the perfect gift to drag people through difficult periods. I owe him many red pens for all his corrections made in my documents.

Several people did also provide direct or indirect contributions to my work, I want to thank:

Especially, the other PhD students involved in feature modelling research — Maurice Dohmen, Klaas Jan de Kraker and later Rafael Bidarra — for being sparring partners for my ideas.

The people from the Mechanical Engineering Department — Marcel Tichem, Michiel Willemse, Jan Peter Baartman, Ton Storm, Bart Meijer and professor Nick Reijers — in their ever lasting patience in explaining me, as a computing scientist, the differences between screws and bolts and the more serious parts of mechanical engineering.

The support staff of our group during my stay — Aadjan van der Helm, Peter Kailuhu, Kees Seebregts and πter Jonker — they did spend a lot of time in creating for me a working environment, as comfortable as it could

be, given the means available.

The master students who helped me in solving parts of my problems — Ronald Hupperichs doing research in the functional modelling area, Ronald van Gimst doing research in motion planning and Marco van der Zwet doing research in modelling with assembly features.

The other PhD students of our group — Arjan Kok (my first roommate), Reinier van Kleij (my second roommate), *I.* Ari Sadarjoen, Andrea Hin, Theo van Walsum, Wim de Leeuw, Freek Reinders and Erik Reinhard — for all the nice hours during work and after work, but almost always talking about our work.

I will miss the laughter of Ton Bosman through the long aisles, and the sneezings from Matthijs Sepers, who unconsciously provided the building with some liveliness.

I will miss the people from the First-Aid group, who were always laughing during practising First-Aid on the most disgusting wounds and amputations.

And I will remember my car-pool partners — dr. Cees Witteveen, dr. Leo Boellaard and the one without a grade Martien van Beeck — they have shortened the long traffic jams with all kinds of games and alternative routes through the “Groene Hart” of the Netherlands.

I also want to thank my new colleagues, from Baan Research, who are provided with the habit of continuously reminding me of the fact that there was still something like a thesis that I had to finish.

I want to thank my parents for providing me the opportunity to go to university. And of course, my wife Nelleke, for her love, support and patience and all the work she did for me during the last years, that I was supposed to do, but did not do, because I was busy with what you are reading now.

Soli Deo Gloria – to God alone be glory.

Winfried van Holland
Breukelen, 1997

Contents

Who is wise and understanding among you?
Let him show it by his good life, by deeds done in humility that comes from wisdom.
James 3:13 (NIV)

Chapter 1

Introduction

With the now commonly used tools in computer-aided design (*CAD*) and computer-aided manufacturing (*CAM*), accuracy in part design and precision in production have increased. These improvements made it possible to further reduce the lead time, or time-to-market, of products, for example by further automating the assembly process. The process of linking CAD and CAM is now going on for a couple of decades.

In the 80s, however, the automation of the assembly process suddenly staggered. One of the reasons for this was the disillusion in the possibilities of using *flexible* assembly robots. Although it was possible for such robots to perform a large set of tasks, it was very hard to automatically generate programs to execute these tasks on these robots (Gottschlich et al. 1994). Each task, together with alternatives needed because of uncertainties in the assembly process, had to be completely spelled out. To overcome these problems, this work had to be automated.

Before the production of a new or modified product can take place, it must be preceded by an engineering phase. In this phase, a design or re-design of a product is executed, resulting in a model of the product to produce. Together with this model, a plan is made, describing how the product can actually be produced. The key for automating the assembly planning lies in the use of the product model information for assembly analysis and planning.

This thesis will focus on the automation and integration of modelling and planning, especially for assembly. With the use of new techniques usable in assembly modelling and planning, the automation of the assembly process can make another step forward.

1.1 Assembly modelling

In assembly modelling, a model is created representing a product consisting of several smaller components. Because of these smaller components, the focus in assembly modelling will be not only on these components, but also on the relations between these components. A component that cannot be subdivided into smaller components is called a *single part*. A group of components merged together is called an *assembly*.

Decisions made during the creation of a model can have great impact on the complete life cycle of the product. Wrong decisions made during design can be responsible for time- and money-consuming product re-designs. To avoid such re-designs as much as possible, a designer has to take into account requirements from other disciplines involved in the life cycle of the product. Involved disciplines can be, among others, from the start of the life cycle to its end: marketing, design, manufacturing, assembly, service and finally disassembly, see Figure 1.1.

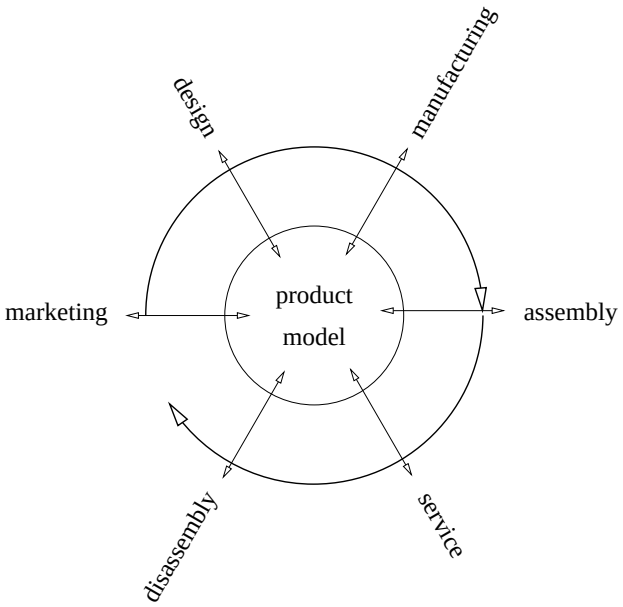


Figure 1.1: Product life cycle

Several analyses can already be performed during modelling to check whether the requirements are met. So, modelling and analysing have to be integrated. Because of the analysing step, modelling cannot be restricted to the design department only. It is possible that a designer is responsible

for the final design of a product, but the modelling process itself still needs input from many other disciplines.

To fulfill all different, and sometimes conflicting, requirements, the designer can use the design for X (*DFX*) concept, where the *X* can be any life cycle phase (Boothroyd 1987, Andreasen et al. 1988, Boothroyd et al. 1994). In DFX, already during design, requirements from X are taken into account, where the X can, for example, stand for: Manufacture in DFM, Assembly in DFA, or Service in DFS. To realize this, the designer must have knowledge of many issues involved in the disciplines or, and this is more likely, the other disciplines must cooperate during design. The latter is usually meant by the term *concurrent engineering*.

The difference between *single-part modelling* and *assembly modelling* lies in the existence of several components in an assembly, together with assembly relations. Because of these additions, both the design and the analysis steps within assembly modelling are more complex than for single-part modelling. There are more disciplines involved in assembly modelling than in single-part modelling, providing more (conflicting) requirements to satisfy. Due to the existence of single parts in every assembly, assembly modelling cannot be separated from single-part modelling. The term *product modelling* is used to refer to both.

1.2 Assembly planning

Modelling alone is not enough to prepare the production of a product. The analyses within modelling check whether the requirements can be met, but do not exactly determine how they are met. This is determined within planning.

Planning is responsible for the creation of the production plans, which specify how the product can be produced.

In assembly planning, several plans are created specifying how the product can be assembled given the product model.

The difference between single-part planning or *manufacturing planning* and *assembly planning* is comparable to the difference between single-part modelling and assembly modelling. *Product planning* is therefore used to refer to both.

Assembly plans are very difficult to generate, and often this is still done manually, consuming a large amount of time. This time can directly effect the *time-to-market*, the time of a product from its preliminary requirements specified by marketing, until it finally can be sold on that market.

In particular for smaller volumes, the time needed for assembly planning becomes more and more important. Such volumes are most of the time produced on already available machines. These *flexible assembly systems* are capable of assembling families of products, in a flexible batch order. During development of the product, the time needed to generate assembly plans is here relatively important.

At Delft University of Technology, a prototype flexible assembly cell, the DIAC (Delft Intelligent Assembly Cell), has been developed (Meijer and Jonker 1991). The main goal was to develop a cell capable of assembling a large variety of product families, in relatively small batches. This research has been a collaboration between several faculties of Delft University of Technology: Mechanical Engineering, Applied Physics, Electrical Engineering and Technical Mathematics and Informatics.

One of the problems arisen in this project was the difficulty to link all developed planning modules. There was no predefined way for the modules to retrieve needed information or to store generated information. The work described in this thesis is one of the spin-off projects of the DIAC project, and is mainly concerned with this problem.

1.3 Product models

Both product modelling and planning are highly dependent on well defined *product models*. Product models can be seen as information carriers for modelling and planning. Examples of information stored in product models are the geometry of the product and the used material. By combining information, other information can be generated that can also be stored in the product model, e.g. volume, weight and center of gravity.

Nowadays, product models used in modelling and planning are hardly integrated. But also there is hardly any integration between single-part models and assembly models. This results in several different product models for a product, at least one for every discipline involved in modelling and planning.

Different product models give rise to severe problems, because of redundancy of stored data and because of loss of information due to conversions between models. This makes it extremely difficult for one discipline, to understand why certain decisions were taken by another discipline.

A possibility is the use of one integrated product model by all disciplines involved. Information is stored only once in the product model, resolving the problems due to redundant data. This makes it possible that information can be stored by one discipline, and can be used by other

disciplines. Also there is no unnecessary loss of information, because conversions from one product model to another are no longer needed.

Nowadays, in single-part models, there is a shift from storing only geometry-oriented information towards more functional-oriented information. The latter is done using *features* in feature-based models. This functional information is very useful during modelling and planning. In this thesis the feature-based concept will be applied to modelling and planning for assembly.

1.4 Main objective

The main objective of this thesis is to develop an integrated feature-based product model to be used in assembly during modelling and planning.

Therefore three issues are involved in this research:

- The structure of the product model itself. An integrated product model for assembly is needed to avoid the unnecessary loss of information during modelling and planning, and to integrate single-part and assembly models. Features are used because in single-part models they have shown their benefits, and it is expected that this will also be the case for assembly models.
- The way this product model can be used during assembly modelling. Therefore a prototype system is built, in which the feature-based integrated model can be created.
- The way this product model can be used during assembly planning. The same prototype system also contains planning functionality, in which planning activities can be activated directly from the modelling environment. It is evaluated whether the used features provide benefits during planning.

1.5 Overview

To describe the research done to realize the objective, the thesis is divided into the following parts:

Part I background, gives an overview of commonly used modelling techniques in Chapter 2 and planning techniques in Chapter 3. After this overview, the long-term goals are defined in Chapter 4, to provide a direction for future research and development in this area.

These long-term goals are the starting points for the short-term goals, which provide more clearness on the chosen solutions in the rest of the thesis.

Part II modelling with assembly features, focusses on the modelling environment. First, in Chapter 5, a new object-oriented feature-based product model is presented. This model combines elements from single-part and assembly modelling.

Thereafter, in Chapter 6, the focus is on assembly features, in the described product model to keep track of the assembly information of a product.

In Chapter 7, a prototype assembly modelling system is described. Within this system, assembly models can be created and manipulated.

Part III planning with assembly features, discusses the use of the object-oriented feature-based product model with assembly features in assembly planning modules.

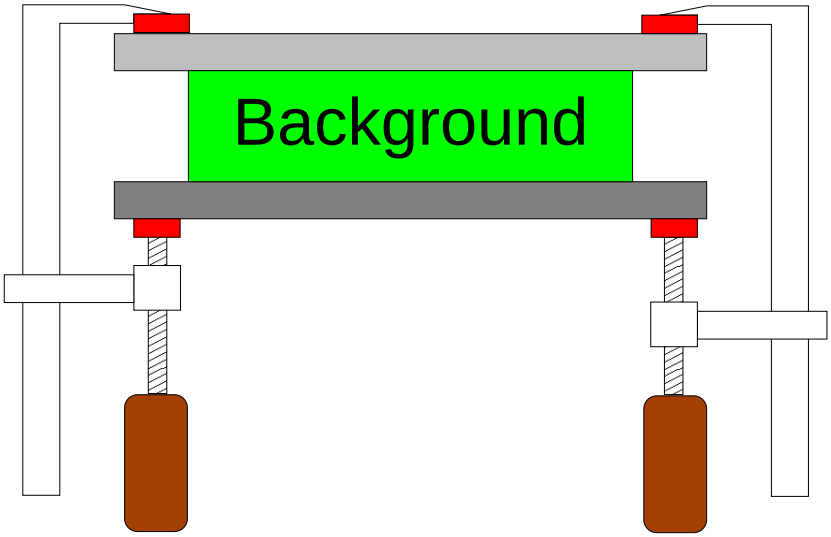
It is not the intent to give a detailed description of every planning module needed within assembly planning. Some have been chosen, to verify the concept. Besides this verification, new and extended planning algorithms are presented to show additional benefits of the product model.

First, in Chapters 8 and 9, extensive descriptions of using the product model in stability analysis and grip planning are given. This is followed by a discussion, in Chapter 10, of how the product model can be used in motion planning and assembly sequence planning.

Part IV concluding remarks, is the final part containing the conclusions and future work in Chapter 11.

To avoid confusion about used terminology, Appendix A has been included.

Part I



Part I gives an overview of commonly used modelling techniques in Chapter 2 and planning techniques in Chapter 3. After this overview, the long-term goals are defined in Chapter 4, to provide a direction for future research and development in this area. These long-term goals are the starting points for the short-term goals, which provide more clearness on the chosen solutions in the rest of the thesis.

Chapter 2

Assembly modelling

For centuries, products have been designed with the sole use of technical drawings on a piece of paper. With the "recent" introduction of computers, new possibilities became available, including the possibility to model products with the aid of a computer. This chapter will describe techniques available for this, with the focus on assembly modelling.

2.1 Top-down and bottom-up modelling

There are two main approaches in which one can create a product model, the top-down and the bottom-up approach, as was described by, for example, Libardi et al. (1988) and Lim et al. (1995).

2.1.1 Top-down approach

The top-down approach is based on the designer's point of view. The designer initially thinks in an abstract, functional manner to find ways to satisfy the requirements of the product. The product has to fulfill some specified main function. By recursively dividing the main function into sub-functions, the designer generates a product model in a top-down design or modelling way.

The initial functional — highly conceptual — information is difficult to specify in a general way, and therefore hard to store and use in a computer environment. Later on, at the sub-function level, more and more details are determined, which can be represented more simply in a general way, and therefore can be better handled by a computer. At the end, sub...sub-functions can be represented by geometry, which can now very well be

stored in a computer. Research on how to model the highly conceptual information, sometimes called *functional modelling*, is in a preliminary stage. Examples of this approach are described in Section 2.3.

2.1.2 Bottom-up approach

The bottom-up approach is based on availability of technologies. In this approach, complete — highly detailed — geometric representations of components are already available. These representations of the components are usually made with some CAD package, and later on the spatial relations between these components are modelled to specify the complete product.

This approach is now widely used because of the technical possibilities of CAD systems. In the last decades, there has been much research in *single-part modelling*, resulting in many improvements. This research was focused on how to model the geometry of a completely *detailed* component. Modelling techniques like constructive solid geometry (CSG), boundary representations (B-Rep) or hybrids of these, can be used for this. Functional intents that have led to these detailed components can, however, not be stored in these representations. In this way the "bottom", the detailed representation of geometry of sub…sub-functions, is first stored in the product model. Thereafter the relations between the components are added to the model. Examples of the bottom-up approach are described in Section 2.2.

2.2 Modelling with detailed single-part models

First the focus will be on the bottom-up modelling approach, because it is easier to explain top-down modelling after bottom-up modelling.

The assembly models used in bottom-up modelling can be subdivided into two different groups (Srikanth and Turner 1990, Requicha and Whalen 1991):

- hierarchical models
- relational models.

Both make use of already defined product models of the single parts, and combine these into an assembly model of the complete product. Therefore a brief description of product models for single parts is given first.

2.2.1 Single-part models

Compared to assembly modelling, there has been much research in the area of single-part modelling. For more details on the described models, see Requicha (1980), Mortenson (1985), Mäntylä (1988) and Bronsvoort et al. (1991).

solid models

The first single-part models in CAD were in fact computer models of 2D paper drawings. Later 3D computer models, representing the complete topology and geometry (the geometry for short) of parts, were introduced. These product models were called *solid models*, and the nowadays mostly used representations are *Constructive Solid Geometry* (CSG), *Boundary Representation* (B-Rep) or hybrid forms of these.

Within CSG, models are built from a collection of primitive solid objects, e.g. blocks, cylinders and spheres. The set operations union, difference and intersection can be applied on these objects to define new, composite objects. The data structure is a binary tree; the root represents the complete single part, and the leaves represent the primitive solid objects. The main disadvantage of this method is that there is no explicit information in the representation about the faces, edges and vertices of the single part. This information is needed during some analyses in modelling and in several modules in planning, both for single parts and for assemblies.

A product model that does have explicit information about the faces, edges and vertices in the single part is the B-Rep. The data structure is a graph structure. Every node in the graph is a boundary element of the solid object, i.e. face, edge or vertex, and arcs in the graph represent adjacencies between these elements.

Hybrid data structures of CSG and B-Rep combine the advantages of both structures: the convenience of modelling with CSG structures, and the availability of explicit information within the B-Rep structures.

feature models

A disadvantage of solid models is the lack of functional information in the data structure: only the resulting geometry is stored. The intent of the designer, i.e. why he has chosen for the specified geometry, is not stored within solid models. This information is, however, needed during analyses and planning, but also to verify whether certain changes are allowed on the model.

To overcome this disadvantage, *feature models* were introduced. Features contain, besides geometry, functional information.

Feature models represent a product by a set of feature instances, and between these feature instances constraints can be specified. In literature there are many different definitions for *features*, but a common element in these definitions is that they combine shape — geometry information — with functional significance. This functional significance is not restricted to design significance, but can be significance for any involved discipline during the product life cycle. Each discipline can have its own way of looking at a product, called a *view*. Each view may have its own set of features (de Kraker et al. 1995, Bronsvoort et al. 1996). Features used in the design view, so-called *design features*, are different from features used in the manufacturing view, so-called *manufacturing features*. In cases where functional significance is only related to a generic shape, the corresponding feature is called *form feature*.

It is undesirable that every view-specific feature model is created from scratch. Several methods have been developed to create and convert feature models, and to provide consistency between several feature models:

feature recognition The *feature recognition* method constructs feature models out of already available solid models, or even 2D CAD drawings. Several techniques have been developed to recognize features, mostly manufacturing features.

One technique uses an (Attributed) Face Adjacency Graph (*FAG*) (Joshi and Chang 1988), a graph with the nodes representing the faces and the edges representing the adjacencies between faces. For every feature to recognize, a predefined FAG is known. By using graph-based pattern-matching techniques, features in the solid model are recognized.

Another technique is decomposing the solid model into volumes, and trying to map these volumes onto known feature volumes. This technique is used by, among others, Kim (1992).

design by features When completely new product models have to be generated, one can immediately start with feature modelling. Product models are created by combining instances from a set of view-specific generic features, with constraints between them. This concept is called *design by features*, but is not restricted to the design view; a feature model for the manufacturing view can also be created with design by features.

feature conversion The method of *feature conversion*, or feature mapping, is used to create different feature models of the same product for several different views. Feature recognition and design by features are combined in this method. A feature model is created for one view, using design by features, and is converted into a feature model in another view by a kind of feature recognition, see, for example, Dohmen et al. (1996) and de Kraker et al. (1997).

More details about feature modelling, and links for further reading, can be found in Shah et al. (1991), Bronsvoort and Jansen (1993) and Shah and Mäntylä (1995).

Now that the single-part models have been described, the focus will be back on combining these models in assembly models.

2.2.2 Hierarchical models

In a *hierarchical model*, the complete product is represented by a tree structure. Nodes in the tree represent parts (the leaves of the tree), subassemblies or the complete product (the root of the tree). The position and orientation of every node is specified by a $4 * 4$ homogeneous transformation matrix. This transformation matrix can be global, i.e. with respect to the world coordinate system, or local, i.e. relative to the position and orientation of the direct ancestor in the tree.

One of the disadvantages of hierarchical models is the difficulty of specifying and calculating the transformation matrices. Another drawback of the simple hierarchical model is that the model does not contain any information about relations between the individual components.

2.2.3 Relational models

The disadvantage of having to manually provide the transformation matrices for the hierarchical model, has led to the creation of the *relational model*. The only way to automatically calculate the transformation matrices, i.e. the position and orientation of the components in the complete product, is to provide in some way the relations between these components, which resulted in a change from a pure tree structure to a more graph-oriented structure.

Ambler and Popplestone (1975) and Popplestone et al. (1980) used a product model with relations between the components to calculate the actual position and orientation of every component in the product. These relations were defined between so-called features on components. Features

were defined here as simple topological entities: planar face, cylindrical shaft or hole, edge, spherical face and vertex. They defined a set of relations, of type against — mating faces with opposite normals, coplanar — mating faces with identical normals , and fits — fitting of a cylindrical shaft and hole, and specified how transformation matrices can be calculated given these relationships.

Wesley et al. (1980) used an extended tree structure in their system called AUTOPASS (AUTOmated Parts ASsembly System). Their extended tree structure allowed additional edges between parts and subassemblies to represent certain relations. The leaves in this tree structure are not on the level of single rigid parts, but on the level of so-called *sub-parts* — primitive polyhedrals, representing shapes on a part. These sub-parts are an early form of what are now called form features.

Lee and Gossard (1985) provided relations between components in a product model with their *virtual link* concept. They use a tree structure to represent the subdivision of a product into subassemblies and parts. In this tree structure, an additional graph structure is specified; any mating pair of two components is represented by a virtual link, see Figure 2.1. A virtual link is representing the information required to describe the relation and the mating features between components needed to calculate the transformation matrices.

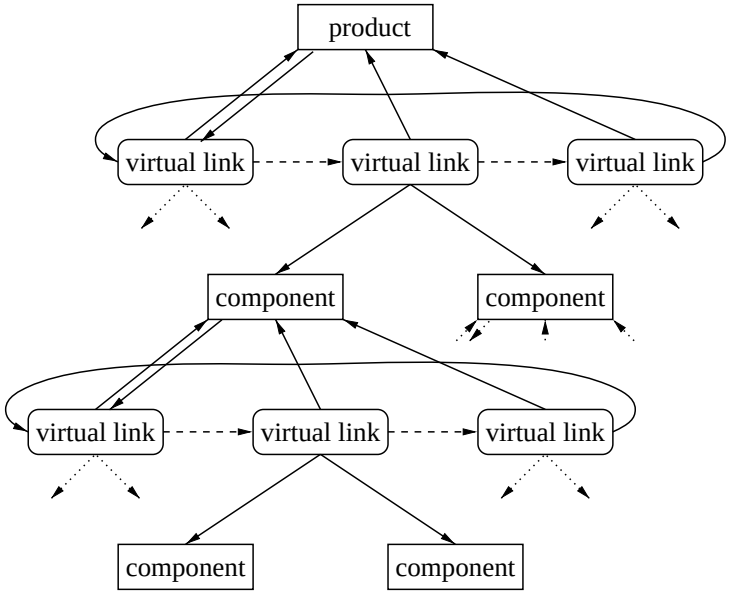


Figure 2.1: Virtual link structure for assembled products

The relations that could be represented with the virtual links were again very elementary: the *against* relation — a plane-plane mating relation, and a *fits* relation — a cylindrical shaft-cylindrical hole mating relation. Lee and Andrews (1985) provided an algorithm to automatically generate the transformation matrices from these virtual links for every component in a product. With the algorithm, the need for the difficult procedure to manually give all the transformation matrices has vanished; however, before the algorithm can generate the matrices, many of these basic relations had to be specified.

With the solution of the positioning problem of the components in the assembly, the research went more and more from only the basic creation of the models towards the use of models for analysis, and even towards the generation of assembly plans. Especially the assembly sequence problem came into focus, i.e. can we use a product model to generate a possible assembly sequence for the product.

Partially based on this, development was focused on purely graph-based product models. Within graphs, components are represented by nodes and relations between components are represented by edges. The relations between components were still very elementary. These *elementary relations* or basic relations are given by, for example, Homem de Mello and Sanderson (1989) and Srikanth et al. (1991):

contact Two components have a contact relation if there are non-rigidly attached faces, edges or vertices between them; i.e. there is some freedom of motion between two components with a contact relation.

attachment Two components have an attachment relation if there are rigidly attached faces, edges or vertices between them; i.e. there is no longer a freedom of motion between two components with an attachment relation.

assembly dimension Two components have an assembly dimension if there is a constraint between them to fully specify some other relations. These dimensions are needed, for example, when a contact relation between two faces is defined: additional assembly dimensions must be defined to uniquely specify the position of one face on the other. See Figure 2.2 on the following page for an example.

The relations are called elementary in the sense that one has to provide many of these relations to fully describe the product model.

Terminology for the elementary relations differs from author to author, but these types are commonly used. Notice, that the definition of an attachment given here differs considerably from the one given later in Subsection 6.4.3, and subsequently used in this thesis.

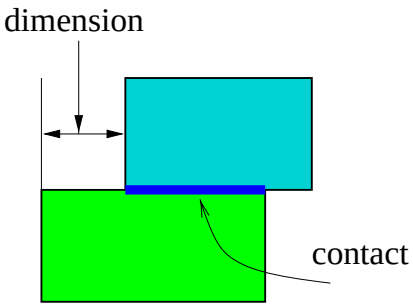
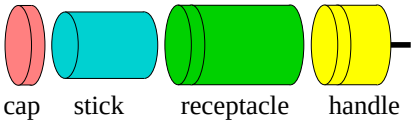


Figure 2.2: Example of a dimension relation

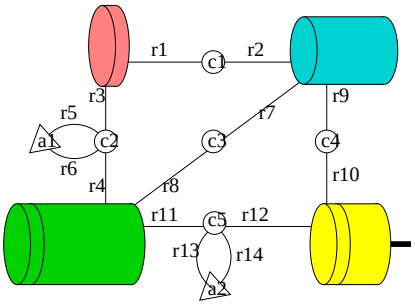
Sanderson and Homem de Mello (1990) used a graph representation to find assembly sequences. They used several representation levels for the product model:

- the solid models of the components, see Figure 2.3(a) on the next page,
- the relational model of the product, where they specified, besides the set P of components, a set C of contacts ($c_1 \ldots c_5$), a set A of attachments (a_1, a_2; in this case eliminating all degrees of freedom of a contact) and a set R of relationships between elements of the set $P \cup C \cup A$ ($r_1 \ldots r_{14}$), see Figure 2.3(b).
- the connection model of the product, a reduced relational model, showing only the connections between components, see Figure 2.3(c).

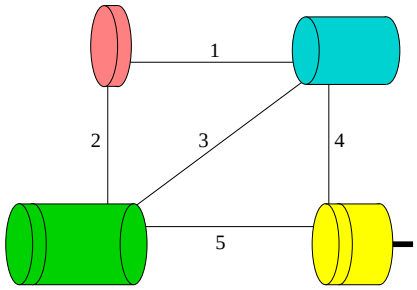
The difference between a relational model and a connection model is thus the level of abstraction. In a connection model, only the connections between components are defined; these connections can contain several relations. These connection models are similar to the *liaison graphs* described by Bourjoult; for extensions on his model see De Fazio and Whitney (1987), in which the precedence relations between connections can be given. The relational model is on a lower abstraction level; here all the elementary relations between components are specified. The connection model can be generated from the relations in the relational model.



(a) solid models



(b) relational model



(c) connection model

Figure 2.3: The product model representations used by Sanderson and Homem de Mello

There exists much variety in product models using a graph for representing the relations, but most are similar to the one described here. Some use feature models for the components, and relations are then defined between features on the components. Roy and Liu (1988) already made use of form features for their single-part components models. They use a kind of Face Adjacency Graph to represent their single-part feature models. Relations between different components are specified between these features, and are called functional relations. The generated product model is called a Functional Relationship Graph (*FRG*). Later this has been extended to the Modified-FRG (*M-FRG*) in Roy et al. (1989), where they showed how information stored in the model can be used for elementary assembly planning.

2.3 Modelling with functional information

In the previous section, we described the bottom-up approach of modelling assemblies. Now we focus on the top-down approach.

In their ”Systematic Approach”, Pahl and Beitz (1988) divided the design process into four phases:

- product planning and clarification of task,
- conceptual design,
- embodiment design and
- detailed design.

Each phase takes as input the results from the previous phase. Now the main problem of the bottom-up modelling approach becomes clear — detailed models of components cannot be generated before their functionality within the product, and their relations with other components have been specified. This can be done within the conceptual design and embodiment design phases by *functional modelling*.

A functional model represents a product structure from highly abstract to concrete, i.e. from undetailed to highly detailed (Henderson and Taylor 1993). To use a functional model in design, the flow of the work should preferably also go from abstract to concrete. This looks obvious, but sometimes the flow goes from bottom to top: specifying *a* functional structure from the detailed product description to the conceptual level.

A problem in functional modelling is how to store the abstract and undetailed information. There must be a possibility to compare stored designs, in such a way that designs or pieces of designs can be re-used in new designs. But on the highest level, the functions are represented as black boxes, describing only (textually) the task of the function. This makes it hard to find a unique way to describe a specific function in some kind of language, and difficult to search for a conceptial solution earlier used. However, some techniques developed in artificial intelligence can be used to find similarities between solutions.

Although functional models are in an early stage of development, there are some promising examples.

Mäntylä (1991) described a modelling environment for functional, conceptual and detailed design. Within the prototype environment, called WAYT (Why-Are-You-There?), a hierarchy of functions can be specified, with relations between them. Feature models of the components can be associated to these functions.

Gui (1993) described the “Δ“ system, a computer environment for top-down functional modelling. A multi-graph structure is used to represent

both functional and relational models. Every node in a multi-graph represents a (conceptual) component or a (conceptual) connector between components. Therefore every node can be a multi-graph itself. In this way, a hierarchical structure can be created from abstract to concrete. Both conceptual components and connectors can be subdivided into other (conceptual) components and connectors. On the lowest, detailed level, components represent single parts and connectors represent elementary relations between these parts. An example is given in Figure 2.4.

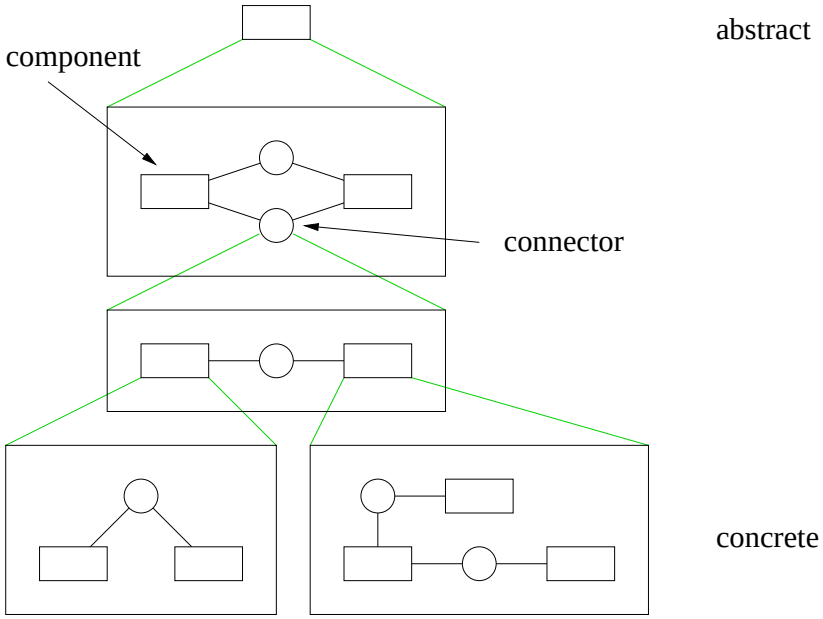


Figure 2.4: The multi-graph structure used in the Δ system; components are represented by blocks and connectors by circles

Andreasen (1992) and Mortensen and Andreasen (1996) divide a product model into domains, a way of looking at something:

functions describe the effects that the product is to create,

organs describe the entities that create the effects, and

parts are the materialization of the organs, i.e. the detailed parts.

Within this subdivision, functions, organs and parts are all described by a hierarchy; so there is a hierarchy of functions, a hierarchy of organs and a hierarchy of parts. Between these hierarchies, there exists also relations.

A function can be realized by several organs, and an organ can belong to several functions. Also an organ will normally need several parts to fulfill its realization, and a part will contain several details belonging to several organs. These interrelations with a many-to-many character make the functional models extremely complex, as can already be seen in a simple example in Figure 2.5.

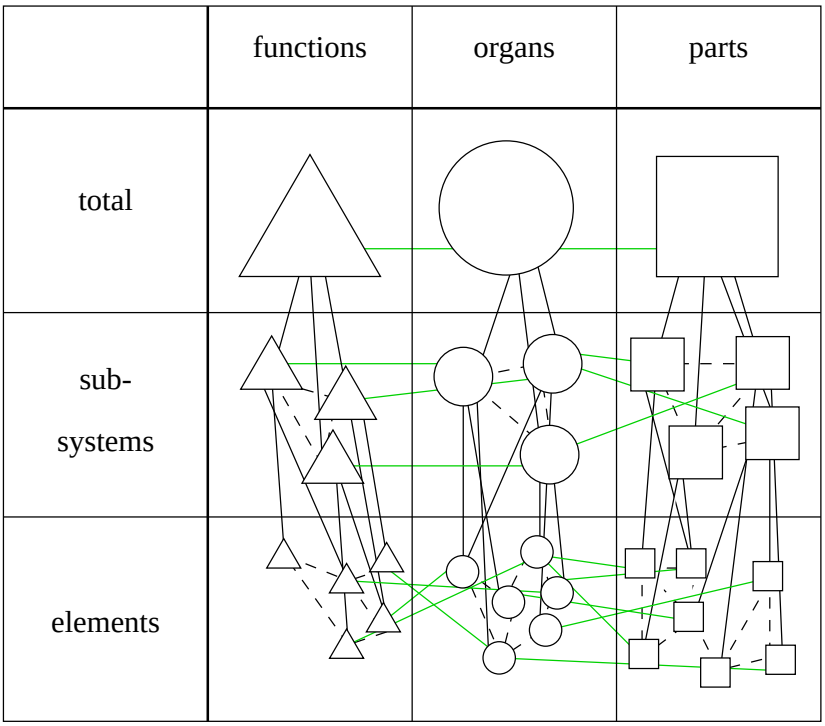


Figure 2.5: An example of function, organ, and part hierarchies, with their complex interrelations

Another point that is stated by Andreasen is the difficulty in dividing functions into sub-functions. Before the function can be divided into sub-functions, the used *means* — the technology used to realize this function — has to be defined. Without choosing a specific mean, you cannot subdivide a function into sub-functions. Every possible mean for a function, will result in a specific subdivision. This results in a *function/means tree.*

Although a good theoretical foundation is given for the top-down modelling process by Andreasen (1992), there is still a lack of proper computer tools to support this modelling concept.

2.4 Assembly features to fill the gap

In this chapter, two concepts for modelling have been described, the bottom-up and the top-down concept. The first has the problem of adding higher-level information and relations to the product model of highly detailed components. The latter has the problem of converting highly abstract information to detailed geometry information. We did not even consider the problems in automatic conversion of functions to geometry!

There is a large gap between the detailed geometry information and the elementary relations on the one hand, and the abstract functional information on the other hand. This gap is not only present within assembly modelling, but also within single-part modelling.

The way a designer has to specify a part in a CSG or B-Rep model does not correspond to his way of thinking. He prefers a more abstract concept to specify his product, and this was to some extent provided with the introduction of feature modelling. Features are on a higher abstraction level than geometric elements, and features provide the possibility to contain functional information.

Several people who noticed the possibilities of using features in single part modelling, also tried to apply the feature concept within assembly modelling.

De Fazio (1990) described a prototype feature-based DFA system, where form features for single-part modelling were used together with “features” to specify the mating relations between components. The latter features provided additional assembly-specific information — such as degrees of freedom and relative extraction directions — to the elementary relations normally used within assembly planning.

Although the additional information could be used in assembly planning, these features did not bring assembly modelling to a higher abstraction level. Still all the elementary relations had to be specified separately.

The first appropriate use of the term *assembly feature* was by Sodhi and Turner (1991). Before that the term was sometimes used, but only to specify, in fact, elementary relations (Popplestone et al. 1980, Lee and Andrews 1985).

Sodhi and Turner used assembly features for specification of relations between components on a higher abstraction level. In their opinion, assembly features served as a higher-level interface — capturing assembly relations at the functional level, and removing from the designer the burden of identifying the underlying elementary relations. See, for example, Figure 2.6 on the following page, where a *Pin Joint assembly feature* is shown. The designer can, with this feature, specify several form features

and several elementary relations in one step.

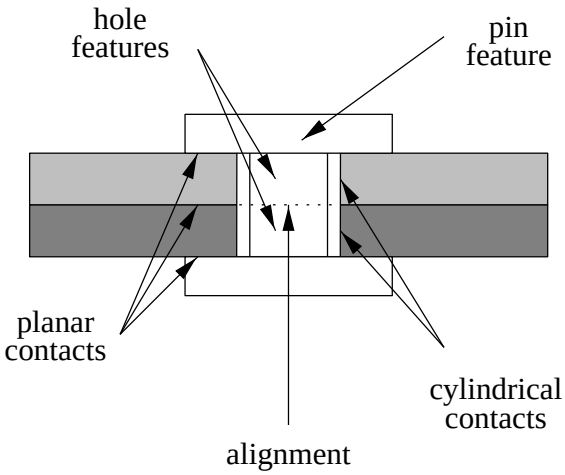


Figure 2.6: Pin Joint assembly feature

Shah and Tadepalli (1992) defined an assembly feature as an association between two form features on different parts. They used assembly features as an abstraction to specify several elementary relations.

They also pointed out that assembly features could be used within a design-by-features concept. When a designer has chosen two form features, the system can provide the designer with possible assembly features. Although this is a bottom-up approach, it can be very useful.

Later, Shah and Rogers (1993) compared assembly modelling with single-part modelling. They found that the two modelling concepts were very comparable. A single part can be thought of as an “assembly” of form features, with mutual relations. The constraints used between form features in single-part modelling, can also be used as elementary relations in assembly modelling. By combining such elementary relations, one can generate higher-level relations called assembly features. These assembly features can then be extended with additional assembly-specific information, such as degrees of freedom and fit information.

Assembly features as defined here were only used to ease modelling. They were closer to a designer’s way of thinking: with one assembly feature he could specify several elementary relations. The information stored within these features was not yet used for analysis of the model, nor creation of assembly plans.

Assembly features combine information about assembly and geometry, i.e. both abstract functional and detailed geometrical information. There-

fore they can be used to fill the gap between abstract functional and detailed geometric models. It seems easier to link pure functional models to geometric models through assembly feature models, than without them.

Chapter 3

Assembly planning

In Chapter 1, it has already been described that automation of the generation of assembly plans is needed, will the assembly process itself be effectively automated. In this chapter, modules needed within assembly planning to analyse a model and to generate assembly plans will be described.

Generally, assembly planning does not start at the moment that the design process has been completely finished. In the contemporary DFX concept, already in an early phase of the design process the designer takes into account requirements from other disciplines involved in the product life cycle, see Figure 1.1 on page 2. During the design phase, certain steps in the planning for other disciplines are done simultaneously, thus analysing the proposed product model. The generated analysis information should be stored in the product model. Modelling and planning are not completely separated, but concurrently performed by several disciplines that are closely related and simultaneously operating on the same product model.

But even within one discipline, for example assembly, there exists several kinds of planning activities. For all these activities, the product model is analysed to see whether the activity can be performed. Each different activity has its own module to perform the analysis. The next section will briefly describe some of these modules used in assembly analysis and planning.

3.1 Assembly planning modules

In literature, the term assembly planning is almost exclusively used for assembly sequence planning (Homem de Mello and Sanderson 1991*b*,

Wolter 1991, Delchambre 1992). Although assembly sequence planning — finding feasible sequences in which the product can be assembled — is a very important planning module in assembly planning, there exists many other modules (Nevins and Whitney 1989). In Figure 3.1, several planning modules are shown. It is indicated that assembly sequence planning is highly dependent on these other modules as well, especially where these modules concern about the assemblability of an assembly.

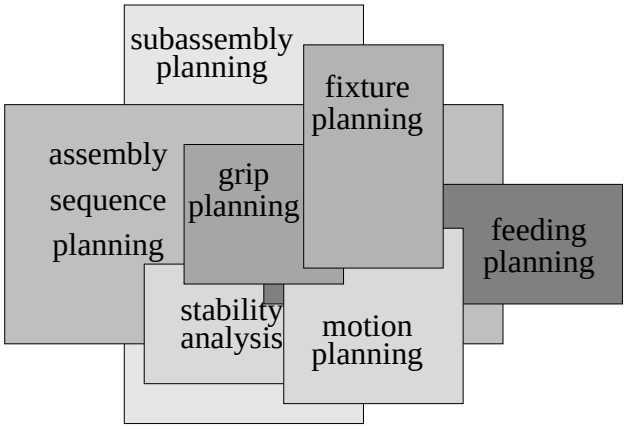


Figure 3.1: Assembly sequence planning and its relation with some other planning modules

The following list briefly enumerates assembly planning modules found in literature (Heemskerk 1990, Martens 1991, Boneschanscher 1993, Lee 1994, Gottschlich et al. 1994), with their specific goals.

fixture planning A fixture planner determines fixtures, together with base components to be used for assembling. A *base component* is the first component of an assembly that is assembled. Fixture planning determines the base components leading to a minimal number of fixturing setups during assembly and providing a maximal set of component approach directions. Another problem to solve for the fixture planner is to determine the optimal number of base components that can be placed on one fixture at a time.

feeding planning A feeding planner determines usable feeders for the components to assemble. The planner must take into account the approach directions of the components on the feeder. In connection

with the number of base components on a fixture, the feeding planner must find the optimal number of components on one feeder, to eliminate unnecessary feeding operations. Feeding is more time critical in assembly, compared to manufacturing, because assembly operations use relatively short times, comparable with feeding times. Therefore, the components to assemble must be continuously fed, otherwise the assembly cell will be idle.

stability analysis A stability analyzer checks whether a certain assembly is stable and thus can be used as partial assembly or subassembly. The analyzer takes into account different stability conditions: static conditions — only gravitational forces, transport conditions — forces due to accelerations because of transport, and assembly operational conditions — forces due to the assembly operation itself, i.e. additional forces needed to establish contacts. Chapter 8 will describe this topic in more detail.

grip planning A grip planner tries to determine appropriate tools for gripping the components, and the areas on the components where to grasp them. To find these areas, also information concerning used feeder, fixture and already assembled partial assemblies is derived. Chapter 9 will describe this topic in more detail.

subassembly planning A subassembly planner tries to divide the complete product into subassemblies. The main requirement for subassemblies is that they remain stable when manipulated, but other requirements can be added, such as that the subassembly must fulfill some functionality, or is important for service purposes.

gross motion planning Gross motion planning is the first phase of motion planning. A gross motion planner determines a collision free path from the feeding position towards a position near the partial assembly. Finding a collision free path is also known as the “piano movers” problem.

fine motion planning Fine motion planning is the second phase in motion planning, and starts at the point where the gross motion planner stopped. A fine motion planner thus determines an assembly path from a position near the partial assembly to the final assembled position. In contrast with gross motion planning, fine motion planning will often use contacts to reduce uncertainties in positioning the component. Fine motion planning uses these contacts to “lead”

the component to its final position. Section 10.1 will describe motion planning in more detail.

assembly sequence planning An assembly sequence planner determines feasible assembly sequences for a product. However, these sequences are highly dependent on output generated by almost all other planning and analysis modules. Sometimes, when problems occur during assembly, it is needed to create an ad-hoc sequence, used to finish the assembly as far as possible. Section 10.2 will describe this topic in more detail.

scheduling A scheduler determines an optimal assembly sequence for a complete batch of products. Out of a set of feasible sequences generated by the assembly sequence planner, the scheduler must choose "optimal" sequences to be used during actual assembly. Therefore resources must also be allocated by the scheduler. This can only be done when, for example, the fixture, feeding and grip planners have already selected their resources.

It can be preferable to do assembly sequence planning and scheduling simultaneously, to make use of the dependencies between these modules and to decrease the exponential complexity. Searching feasible sequences is related to available resources known already for the schedule. By exploiting this information, the search space can be restricted.

In addition to these planning modules, there can be several other modules within assembly planning, such as sensor planning, dynamical analysis, functional analysis, assembly line planning, etc. These will not be elaborated here.

The question may rise, whether there exists some fixed execution sequence for these modules, so that all required information will be available at the right moment. As can be seen in the description of assembly sequence planning, this is not always the case, because especially this module is highly interdependent and therefore very intertwined with all the other modules concerning the assemblability.

3.2 Grouping assembly planning modules

This section will describe some proposals made to group assembly planning modules, in such a way that some ordering between groups can be found.

3.2.1 On-line and off-line planning

One way to divide the planning modules is into *on-line* and *off-line* modules (Gottschlich et al. 1994).

In essence, the on-line modules are dependent on information generated during the assembly process. On-line planning modules therefore also generate their output during the actual assembly process. The output is used almost immediately within that process. On-line planning modules are therefore time critical — the longer their required planning time, the larger the risk that the assembly process, executed in parallel, has to wait.

The off-line planning modules are less time critical. They are less dependent on actual information, and therefore can generate output before the assembly process is executed.

However, it is difficult to strictly separate the planning modules into on-line and off-line modules. Some of the planning modules can be used both on-line and off-line. For example, the scheduler can make a schedule off-line for a batch of products, given some specified flexible assembly cell with resources. However, during the actual execution of such an off-line generated schedule, it must be able to generate a schedule on-line, for example when specific resources are not available or damaged (van Holland et al. 1992).

3.2.2 Using abstraction levels

Heemskerk (1990) suggests a hierarchical reference model with four levels of abstraction, and places the modules in such a level, see Figure 3.2 on the next page.

batch level The batch level generates plans for a complete batch of products to be assembled.

product level The product level deals only with one (type of) product of the complete batch at a time, and generates plans for this product.

part level The part level deals only with one part (component) from a product at a time, and generates plans for this part.

primitive level The primitive level generates plans for four primitive actions that must be executed on each part during assembly: feed, grasp, move and mount.

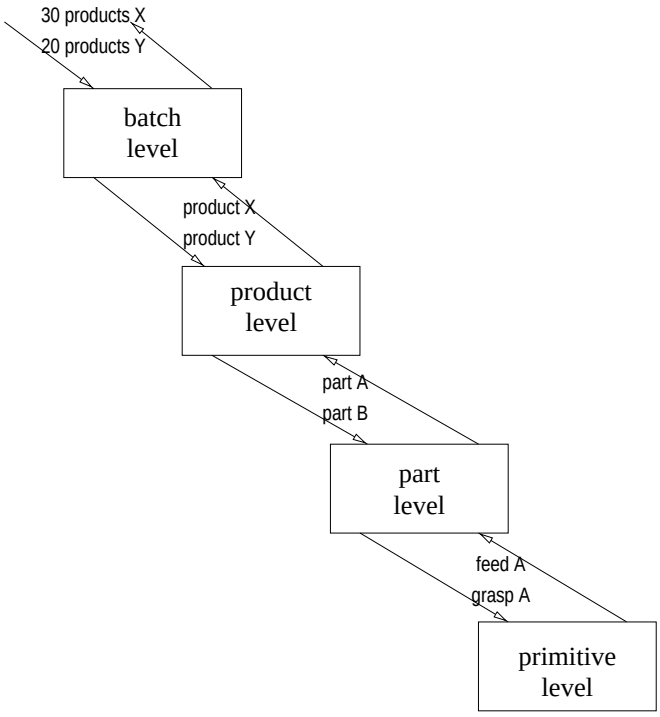


Figure 3.2: A hierarchical reference model for assembly planning modules

Although these abstraction levels provide an overview of the information needed within assembly planning, there are some disadvantages of this model (Martens 1991, Boneschanscher 1993). The main disadvantage is the restriction of placing planning modules at one abstraction level only. It will be incorrect to place, for example, assembly sequence planning, at the product level only. By doing so, the system will provide the scheduler with only one assembly sequence plan for a specific product, which may result in suboptimal schedules on batch level. Better results are retrieved when assembly sequence planning and scheduling cooperate with each other. This will decrease the total search space needed to find proper solutions (van Holland et al. 1992)

3.3 Experiences with existing assembly planners

Mostly, assembly planning modules are described independently from each other. Only a few descriptions of systems have been given that contain several planning modules. Two of these systems are the DIAC system

and the Archimedes 2 system. In the following subsections, some experiences with these systems are described.

3.3.1 Experiences in DIAC

In Section 1.2, it has already been described that the work in this thesis is a spin-off from the DIAC (Delft Intelligent Assembly Cell) project (Meijer and Jonker 1991).

Several prototype planning modules have been developed during this project, for example gross motion planning (Martens 1991), assembly sequence planning, scheduling and batch planning (Boneschanscher 1993), and fine motion planning and grip planning (Baartman 1995).

The product models used — the Product Data Model (*PDM*) and the Connection Model (Martens 1991) — were based on low-level geometric information for the components, with elementary relation information between these. The PDM contained information about:

- Insertion Point, a point near the partial assembly where the assembly of a component can start.
- Final Point, a point on the partial assembly where the component to assemble must be positioned.
- Insertion Path, the path to bring the component to assemble from Insertion Point to Final Point.
- Approach Direction Set, a set of directions available to bring the component from a place relatively far from the Insertion Point to the Insertion Point.

These attributes had to be entered manually. A disadvantage of the used product model was the lack of additional assembly-specific information required for the planning modules. So, every module had its own procedures for geometric reasoning to analyse the product geometry on specific assembly information. This information was stored locally within the module, and could not be used by other modules. In addition, there was no description of how results from a module could be stored, for use in other modules.

So, although several planning modules were provided in the system, the modules were not well integrated.

3.3.2 Experiences in Archimedes 2

At the Sandia National Laboratories, the Archimedes 2 mechanical assembly planning system has been developed (Kaufman et al. 1996)[1]. The system is able to read commonly-used CAD data files, which makes it easier to work with more complex products. The modules available within the system are: mating planning, assembly sequence planning and task planning.

Ames et al. (1995) have described some shortcomings of the planner. Some planning still has to be entered by hand: grip planning, fixture planning, fine and gross motion planning and stability analysis. Another problem is the integration of all planning modules, including the problem of resolving conflicting constraints between modules.

Further there is no possibility to store non-geometric data in the used product model, which is sometimes needed in modules. This auxiliary data has to be computed from the product model, which is very difficult and time-consuming. In a way this is also superfluous, because during product modelling this information was already, at least partially, known to the designer.

Finally, there is no possibility for an engineer to interact with generated plans during the planning process.

3.4 Experiences in manufacturing planning

The problems described are not unique for assembly planning, and some of them can also be found in a closely related area: manufacturing planning.

Here the lack of manufacturing-specific information in the used low-level geometric product models can be found. The use of higher-level feature models solved a major part of these problems. With feature modelling, see Subsection 2.2.1, the product models were converted to or even designed by manufacturing features. These manufacturing features contain, besides a description of the shape, manufacturing-specific information that can directly be used within manufacturing planning.

See Shah and Mäntylä (1995), van Houten (1991), Rosen (1992) and Jasperse (1995) for an overview of manufacturing features and their use in manufacturing planning.

[1] See their home page with descriptions of the system at: http://www.sandia.gov/2121/archimedes/archimedes.html

3.5 Features in assembly planning

As described in this chapter, most of the product models used in assembly planning are not able to directly provide the information needed for assembly planning. The promising results in manufacturing planning using product models containing manufacturing features, did not yet have much influence in assembly planning.

However, there are some authors describing features used within assembly modelling, as described in Section 2.4. Their assembly features were mainly used to provide a higher modelling level, reducing the time-consuming specification of elementary relations.

For example, De Fazio (1990) described a prototype feature-based design-for-assembly system, where features are used for modelling both single-parts and assemblies. Later more details of this system were given in De Fazio et al. (1993), and it became clear that the system could also be helpful in planning.

In their bottom-up modelling concept, the assembly features are elementary relations containing additional assembly information, such as degrees of freedom and relative extraction directions, i.e. the opposite of approach directions. This additional information could successfully be used within several of their prototype assembly planning modules, as are: the bottom-up design of the assemblies, the positioning of the components relative to each other using the features, derivation of feasible assembly sequences, and performing economic analysis on the found sequences to select one and possibly generate a conceptual assembly line for it. Needed information could be directly retrieved from the assembly model, without executing complex and time-consuming geometric reasoning procedures. This all brought the assembly models to a higher abstraction level with respect to planning.

As within manufacturing planning, information already known during design and needed during planning, could now be stored in features in the product model.

However, information generated *during* planning should ideally also be stored in the product model. Later on this information can then be used by other planning modules.

When more information is integrated in the model, also planning modules can be better integrated — information can be retrieved from and stored in the same model. Especially intertwined modules could have much profit from integrated models, for example a considerable reduction in computation time.

Therefore features could also be useful in assembly planning, both to

store required functional information together with the geometric information, and to integrate information used by several planning modules.

Chapter 4

Towards an integrated modelling and planning environment

It is not enough to investigate only modelling and planning concepts and techniques used in the past, to retrieve possible areas for improvements. When you want to know where to improve, you should know the differences between where you are now and what you want to achieve. This chapter will describe a future modelling and planning environment, and this will provide directions to work at.

The future modelling and planning environment should include concepts that at least accomplish that:

- the quality of the produced products improves due to more advanced modelling techniques,
- the time-to-market decreases due to better planning tools.

This chapter will first describe the long-term goals for a modelling and planning environment. These goals will provide the overall research directions. The derived short-term goals will be described in Section 4.2.

4.1 Long-term goals

The best way to determine the long-term goals is, of course, to investigate what is lacking in current systems, and what is needed by users. The main direction is also described by Henderson and Taylor (1993). They stated that there should be an integrated environment for physical objects — the components — and their conceptual counterparts — the intent of the components.

Key topics to realize such an integrated environment are:

- top-down and functional modelling,
- integrated single part and assembly modelling, and
- integration of modelling and planning activities.

In the next subsections, these topics will be further explained.

4.1.1 Top-down and functional modelling

One of the major disadvantages of the current modelling environments, is that the user is provided with a far too simple and limited model, which cannot handle all information already known by him or her, and that, as a consequence, information is lost. This information must often be recovered again later in the product life cycle through complex calculations, which would be unnecessary if it could be stored in the system from the start (Wilson and Pratt 1988). For example, when a user specifies a specific bolt-nut connection between two plates, he or she already knows the assembly sequence restrictions for such a connection, i.e. the bolt and nut connection must be established after the bolt and plate connections. However, this information can generally not be stored in the model, and therefore the time-consuming assembly sequence planning module tries every combination of bolt, nut and plates to find a proper assembly sequence. At the end, it will provide the user with a solution that he or she was already familiar with during modelling.

The main goal is therefore to provide a modelling environment that is able to store input on a high abstraction level, close to the way of thinking of the user. The user must be able to create a model in a top-down way, from conceptual to detailed. Therefore it must be possible to translate functional information to lower-level, more detailed information. In between the top and the bottom level, it must be possible to specify partially detailed information, such as sketches of components and functional faces, see for example Horváth et al. (1995) and Horváth (1996). The approach from Mantripragada et al. (1997) is also been going in this top-down design direction. They use the most important characteristics of a product, so-called *Key Characteristics* (KCs), to represent the design intent. These KCs are then used to identify important datums on components, and to define relationships between them.

One of the major problems in this field is to maintain the consistency of a top-down generated model. After changing something at a more detailed level of the model, higher levels should be changed accordingly. This can be extremely difficult, because changes at a detailed level can

have several effects on higher levels, and there is no one-to-one mapping of detailed information to conceptual information.

4.1.2 Integration of single part and assembly modelling

As a consequence of the top-down concept, modelling of assemblies and the corresponding single parts must be highly integrated. Creation of the component, and finally the single-part, models will be driven by the creation of the assembly model. Changes made in the assembly, directly influence the models of the corresponding components.

The relations between components partially provide the behavior of an assembly, and can also directly specify some of the geometric properties of the components. So by providing global shapes, and relations between them, elements of the detailed geometry can be generated.

These possibilities also provide the system with the opportunity to perform some planning activities on a partially detailed model. In a preliminary phase during design, analyses can be executed to see whether requirements are met. Such preliminary planning activities also require a high integration between the modelling and planning environment.

4.1.3 Integration of modelling and planning

The creation of complex products cannot be done by one person only, nor by a group of persons within one discipline. Complex products require high integration of various disciplines involved in the complete life cycle of the product (Cutkosky et al. 1992).

In fact, it is not possible to separate modelling and planning. This has already been shown in Section 1.1, describing the design for X (DFX) concept. During modelling, one has to take into account requirements from several disciplines (the X's).

Another aspect, highly related to the integration of modelling and planning, is the integration of different planning modules. Information created by one planning module, must be available for the other modules. There must not be any need for generating information within one module that already was available in another module (van Holland and Bronsvoort 1997).

4.2 Short-term goals

The long-term goals provide main directions for research in the far future. The short-term goals derived from the long-term goals are described in this section, using the same subdivision as in the previous section.

In literature, descriptions of first steps in the direction of the long-term goals can be found in Cutkosky et al. (1992), De Fazio et al. (1993), Henderson and Taylor (1993) and Bronsvoort et al. (1996).

4.2.1 Top-down and functional modelling

In Section 2.4, it has already been described how features can possibly be used to solve the gap between conceptual functional modelling and detailed geometric modelling. Current functional modellers do not properly link the functions to the final geometry, and current geometric modellers do not provide the necessary information why certain geometric solutions were used.

Features combine functional and geometric information, and therefore are a good research direction to solve the gap between them, as is also described by Henderson and Taylor (1993) and Wearring (1996). The emphasis should not only be on features describing a certain shape, but also on features describing relations between shapes, because in a top-down design not only shapes of the designs are important, but also the behavior provided by relations between shapes.

4.2.2 Integration of single-part and assembly modelling

As features are already used in single-part modelling, the integration of single-part and assembly modelling could be provided by the feature-modelling concept. However, in single-part modelling, different disciplines are involved in the creation of the model, and they all make use of different, possibly overlapping, sets of features. As a first step, it should be possible to create an integrated model for one discipline, or view, already known in single-part modelling, e.g. the design view or the manufacturing view. Thereafter, multiple views should be taken into account, as is described by de Kraker et al. (1995) and Bronsvoort et al. (1996).

4.2.3 Integration of modelling and planning

Data structures used in assembly modelling and needed in assembly planning, should, as much as possible, be combined to realize a reduction in

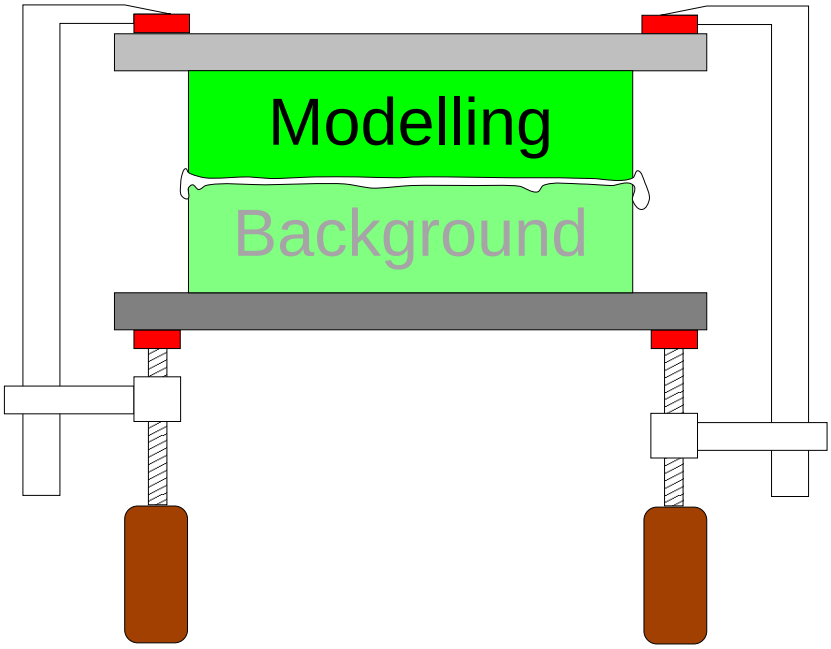
planning time. The additional functional information stored in feature models during modelling can be used in planning modules to retrieve needed information without the use of lower-level geometric reasoning procedures.

To realize a fast feedback loop of DFX analyses in modelling, it is required that there is not only one integrated data structure, but also one system in which modelling and planning activities are integrated.

4.2.4 Towards solutions of the short-term goals

In the following parts of this thesis, detailed solutions to solve the short-term goals are described. In Part II an integrated feature-based product model will be described, usable for both single-part and assembly modelling. This feature-based model will already provide some characteristics needed for top-down functional modelling. Part III will describe how several assembly planning modules are integrated in the system, making profitably use of the feature-based model.

Part II



Part II focusses on the modelling environment. First, in Chapter 5, a new object-oriented feature-based product model is presented. This model combines elements from single-part and assembly modelling.

Thereafter, in Chapter 6, the focus is on assembly features, in the described product model to keep track of the assembly information of a product.

In Chapter 7, a prototype assembly modelling system is described. Within this system, assembly models can be created and manipulated.

Chapter 5

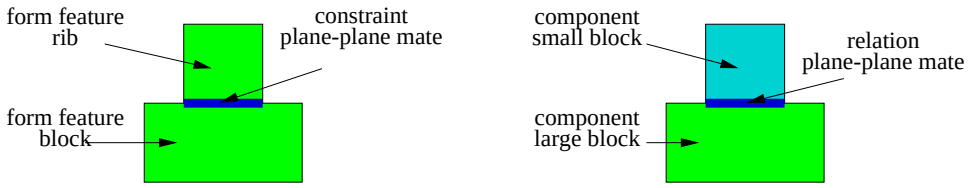
Feature-based product model

As was described in Chapter 2 about the background of assembly modelling, the modelling concept should preferably be closely related to the way of thinking of the designer, i.e. top-down. Within this concept it is also preferable that modelling single parts is comparable to modelling assemblies. This can be achieved by using the same concepts in modelling single-parts and assemblies. In other words, the feature model used for single parts must be comparable to the feature model used for assemblies. In this chapter, a feature-based product model is described, capable of modelling single parts and assemblies.

This was already mentioned by Shah and Rogers (1993). They considered a feature model for single parts as an “assembly” of form features. This concept can be seen in Figure 5.1 on the following page. Figure 5.1(a) shows a simple part containing two form features, and a constraint to specify the position between the two. Figure 5.1(b) shows the same shape, but now containing two components, and a mating relation to specify the position between these two. Both single part model and assembly model contain “building blocks” and relations between them.

Although there exists similarities between part models and assembly models, there are also differences, e.g. in building blocks and in relation types between the building blocks. Whereas in single-part modelling building blocks can be used that subtract volume, in assembly modelling mostly additive building blocks are used.

To specify a product model for both, an object-oriented approach is used. Therefore, first some concepts of object orientation are described. Then the object-oriented feature-based product models for single parts and assemblies are described. Parts of this chapter have already been published in van Holland and Bronsvoort (1996*b*).



(a) single part model (b) assembly model

Figure 5.1: Comparing models for single parts and assemblies

5.1 Object-orientation

This section will give a brief description of some issues in *object-orientation* and *object-oriented modelling*; for a detailed description see Stroustrup (1993) and Gorlen et al. (1991).

In object-oriented modelling *abstract data types* are used. An abstract data type is a user-defined data type that encompasses data elements along with the operations that can be performed on them. Most programming environments used to create models, do not support these abstract data types, but separate the data elements and the operations that can be performed on them. The main advantage of combining data elements and operations is that it is easier to change available data structures or to add new structures. An additional benefit is that object-oriented models are closer to the way of thinking of both computer engineer and user of the created models.

The power of object-oriented modelling lies in the concept of *inheritance* — start with an already developed set of object types, or *classes*, and extend these for new applications, by adding data elements and operations to form new classes. Do not write new classes from scratch, but inherit data and operations from useful *base classes*. Add new functionality by describing how the new or *derived class* differs from the base classes.

Figure 5.2 on the next page shows an example *class hierarchy* for 2D objects in a 2D modelling environment. The class hierarchy is shown in the *OMT* class diagram notation — OMT stands for Object Modeling Technique (Rumbaugh et al. 1991). Individual classes are represented in the OMT notation as rectangles with compartments. The first compartment is for the name and is always required. The other two are optional, one for

the data and one for the operations of the class. When data or operations are not directly needed for the understanding of a class, the OMT notation allows suppressing any or all elements from being displayed.

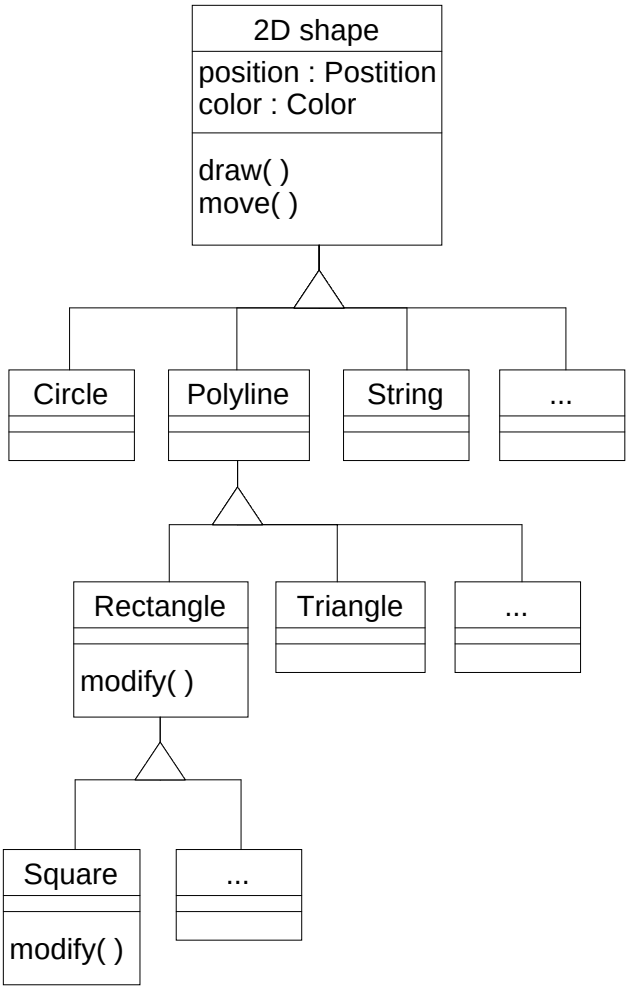


Figure 5.2: Class hierarchy for 2D objects

The base class, the 2D shape class, contains data elements used for a 2D shape, e.g. the position or color, and operations for drawing or moving a 2D shape on a screen. Derived classes inherit these data elements and operations, so it is not needed to specify the position data element and drawing operation for derived classes, if they can use the already available data elements and operations.

In the OMT notation, the inheritance between two classes is represented by a line with a triangle in between. The triangle points in the direction of the base class.

When some pre-defined operation does not fulfill all the requirements for a derived class, this operation can be *overloaded*: create for this derived class a specific operation with the same name as in its base class. For example, the difference between the Rectangle class and the derived Square class is the modification operation, where the latter class restricts the modification by defining that width and height must always remain the same. In this way only the needed additional data elements and operations have to be specified for a derived class.

It is possible for a class to derive data elements and operations from more than one base class; this mechanism is called *multiple inheritance*. Figure 5.3 shows an example of multiple inheritance. When a combined data structure for a string and a rectangle around it is needed, and data structures for a string and a rectangle have been defined, then the new class, the BorderedString class, can be created by inheriting the data elements and operations from both Rectangle and String class.

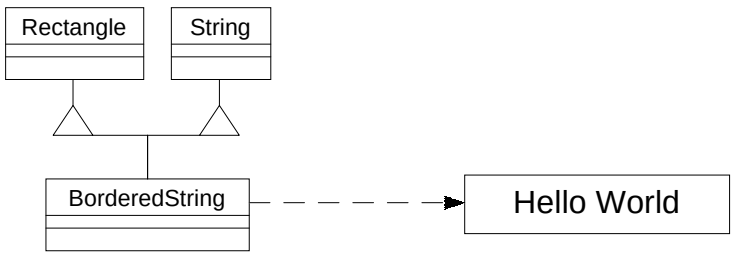


Figure 5.3: Multiple inheritance

How these object-oriented models can be used in developing data structures for feature modelling, is shown in the following sections.

5.2 Single part model

As was stated in the introduction of this chapter, the feature model for a single part can be seen as a collection of building blocks, i.e. the form features, together with mutual relations, i.e. the constraints.

The term *form feature* is used here, instead of the terms design feature or manufacturing feature, because there is no need to know whether the

used shape has some design or manufacturing intent. Only the shape of a feature is of interest here.

5.2.1 Form features as building blocks

A single part can contain several building blocks of the same type, i.e. several *instances* of form features. Each type of form feature is represented by a GenericFormFeature class. Take, for example, a single part with multiple holes in it. Each hole is represented by a specific instance, called object in object orientation, of the Hole class. Every feature inherits from the base FormFeature class. This class contains a data structure in which the geometry of the feature can be described and operations, or methods, on these data structures with some specific functionality. Each derived feature class, e.g. the ThroughHole class, derives the data elements and operations on them from its base class, and makes some modifications on data elements and operations to describe the geometry of that specific feature. Each instance, or object, of these classes, describes the exact shape, with specified attributes, e.g. a through hole with specified diameter attribute M8. Detailed descriptions of form feature classes can be found in Ovtcharova et al. (1992), and an example is shown in Figure 5.4.

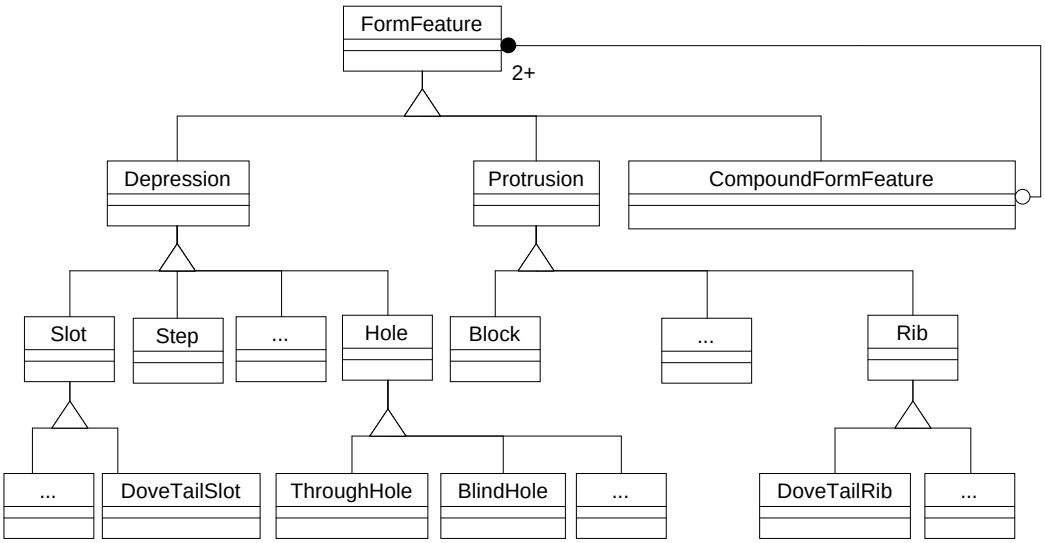


Figure 5.4: Class hierarchy for form features

A special class is the CompoundFormFeature class, where new form features can be created by taking combinations of other form features. In this way, the set of already known form features can be extended. Therefore we need an association relation between classes. In the OMT notation, such an association is represented by a line between two classes. The line terminators indicate the multiplicity of the association. A hollow ball indicates "optional", meaning zero or one. A solid ball indicates "many", meaning zero or more, or n or more, when a multiplicity of n is provided.

For the CompoundFormFeature class, the association with the FormFeature class represents: a CompoundFormFeature has always an association with two or more FormFeatures, whereas a FormFeature has an optional association with a CompoundFormFeature — it is only present when a FormFeature is in a CompoundFormFeature.

By using the term form feature, the product model is simplified: these features are not the only type of features; there are many more, see for example Pratt (1993). Of course there can be multiple views on one product model, using their own set of features. But the assumption that a product model for single parts is represented in a specific view, which contains form features and constraints, is enough for the discussion within assembly modelling. If it works with one view and conversion between views is achieved, then it will work with any view. For the discussion on multiple-view feature models for single parts and conversion between views, see de Kraker et al. (1997).

5.2.2 Constraints for mutual relations

To define the exact position and orientation of the instances, relations or constraints are defined between these instances. Therefore a base Constraint class is defined, and derived classes to describe constraints, e.g. the Mate class for the mating relation between two planes, and the Offset class for the offset relation between two planes.

Besides the geometric constraints, there can also exist algebraic constraints, to specify, for example, a relation between the area of a plane and the volume of a shape.

Details on constraints can be found in Dohmen (1995) and Dohmen et al. (1996), and a brief example of a constraint class hierarchy is given in Figure 5.5 on the facing page. Here, also, new constraints can be defined using the CompoundConstraint class by taking combinations of other constraints.

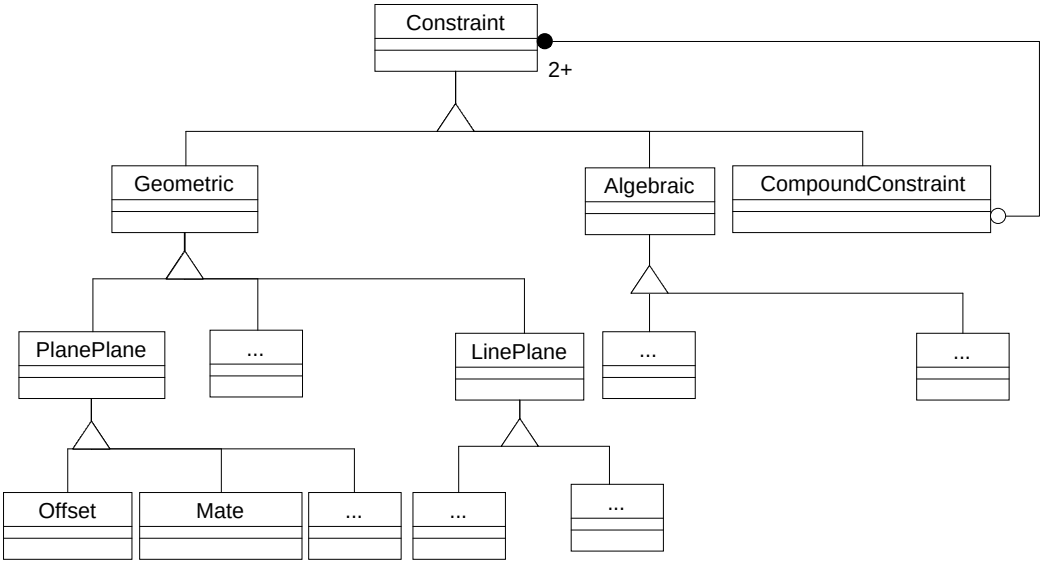


Figure 5.5: Class hierarchy for constraints

5.2.3 Feature model: combining form features and constraints

To define a complete single part, both instances of features and instances of constraints must be specified. This is done with a FeatureModel class as is shown in Figure 5.6. The FeatureModel class contains a list of feature instances, the FormFeatures, and a list of constraint instances, the Constraints on these feature instances.

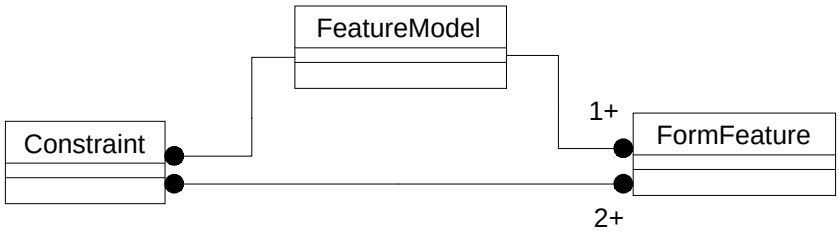


Figure 5.6: A feature model representing a single part by combining form features and constraints

A constraint solver is used to satisfy the constraints, and to calculate the resulting position and orientation of the feature instances. The FeatureModel class defines methods for adding instances of features and instances of constraints, and defines operations for calculating the actual geometry of the defined single part. In Figure 5.7 on the next page, a feature model of a single part is shown, with instances of features and instances of constraints. Associations between these feature and constraint instances are shown with solid lines, associations between the feature model and its feature instances are shown with dashed lines. Notice that the associations between the feature model and its constraints are not shown in the figure for clarity, although they do exist in the model.

5.3 Assembly model

An assembly consists of combinations of single parts, where different instances can be of the same type. These parts are not always directly assembled into the complete product, but sometimes, and for several different reasons, *subassemblies* are created as stable entities. These subassemblies can be used to assemble other subassemblies or, at the end, the complete product. Both single parts and subassemblies are stable entities (with respect to motions or transportations), and can therefore be assembled onto other entities; these stable entities are called *components*. The already assembled components are called a *partial assembly*. A partial assembly can thus be a single part (when assembly has just been started), or a, possibly motion-instable, group of components (during assembly). More about stability analysis can be found in Chapter 8.

5.3.1 Components as building blocks

In a partial assembly, the same component type can be available on several places in the partial assembly, e.g. several bolts to fasten a plate. For each different component type, we introduce a *generic component*, describing the geometry by its form features. The generic component does not describe a position and orientation in the product; this is described by an *instance* of a generic component. In this way, a product can have several instances of the same generic component. Each instance can have different relations within the product. Figure 5.8 on page 52 shows the class diagram used to describe the different components.

As can be seen in Figure 5.8, there exist two derived classes from the GenericComponent class: the Single Part and the GenericCombined class.

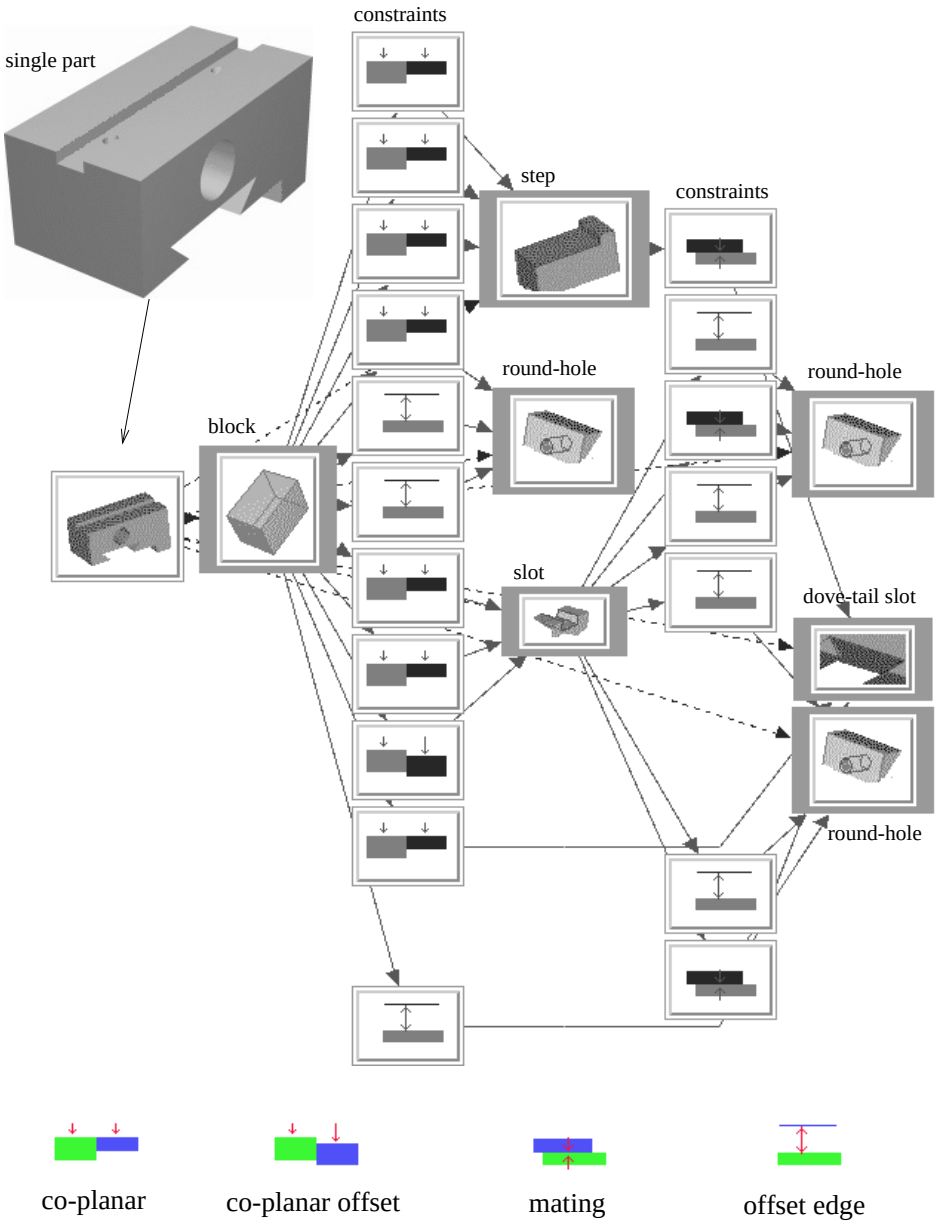


Figure 5.7: Form feature model of a single part

The difference between the two is that the SinglePart inherits data elements and operations from both GenericComponent and FeatureModel, described in Section 5.2. The SinglePart class represents the feature model of a single part, which cannot be subdivided into smaller components.

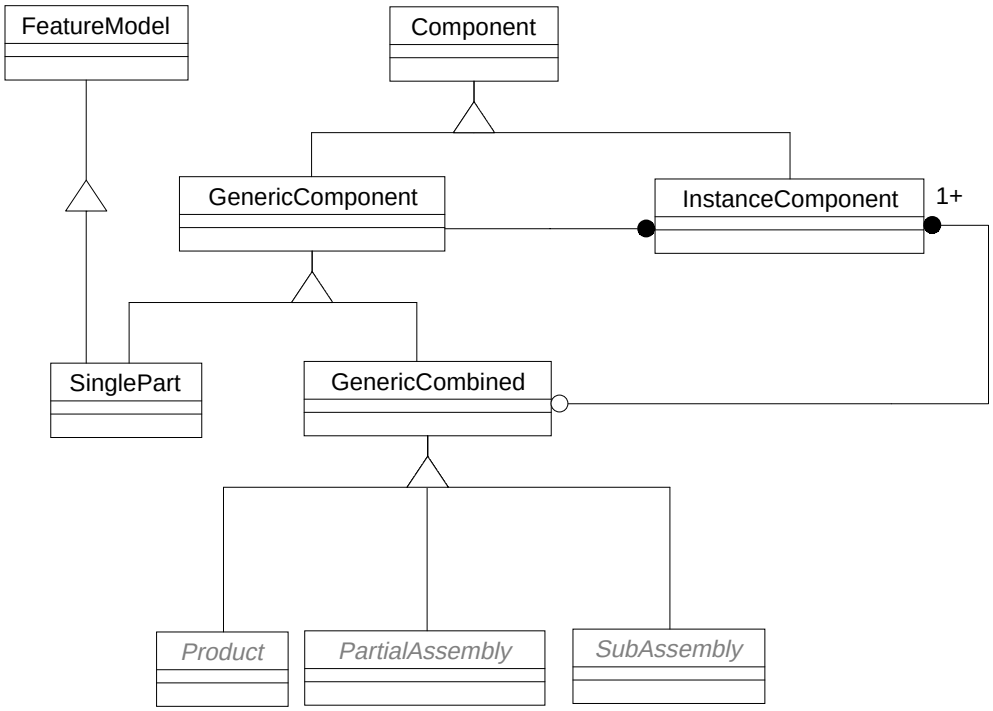


Figure 5.8: Class hierarchy for different components

The GenericCombined class represents the generic components containing more than one component. This class will be described in more detail in Subsection 5.3.3.

5.3.2 Connection features for mutual relations

As within single parts, constraints between instances of form features are used to specify the position and orientation of components. In the assembly model this is achieved by assembly relations. These can be the elementary relations as described in Subsection 2.2.3, but a higher abstraction level is preferable for the assembly relations, both in modelling and planning. Therefore *connection features* are introduced. For now it is enough to know that connection features are on a higher abstraction level than the elementary relations, and that they contain assembly-specific information. More details on the connection features are given in Chapter 6. There is a complete hierarchy of connection features; an example class diagram is shown in Figure 5.9 on the next page.

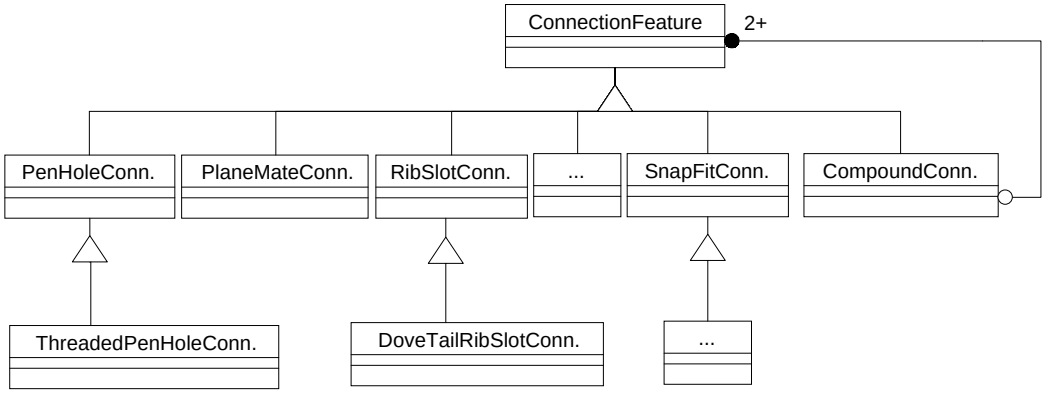


Figure 5.9: Class hierarchy for connection features

The predefined set of generic connection features can be extended by using the CompoundConnectionFeature class. With this class, existing connection features can be combined to generate new connection features.

5.3.3 Generic combined model: combining components and connection features

The GenericCombined class represents components that consists of combinations of instances of components and connection features between them. Product, subassembly and partial assembly are all assemblies that can be subdivided. It is hard to indicate the difference between these types, from an object-oriented design point of view, which is one of the reasons no different classes are introduced for product, subassembly and partial assembly (the italic classes in Figure 5.8): all are represented by their base class, the GenericCombined class.

An attribute in the GenericCombined class contains information whether it represents a stable entity, i.e. a subassembly or product, or it is not known whether the entity is already stable, i.e. a partial assembly.

In figure 5.10 an example is given of a generic combined component, consisting of three instances of two different generic components, and two instances of connection features.

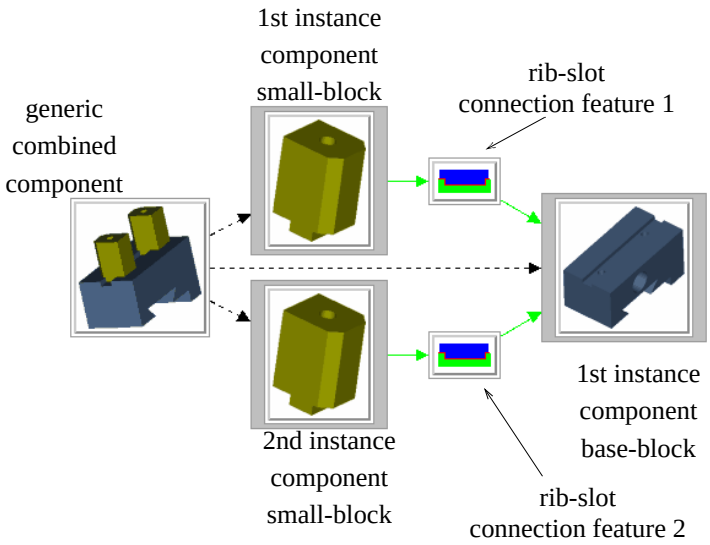


Figure 5.10: Generic combined model of a subassembly

5.4 Combined product model

As has been shown in the previous sections, in single part modelling and in assembly modelling, comparable data structures are used to represent building blocks, mutual relations between the building blocks, and the structures of the instantiated building blocks and instantiated relations. In single part modelling, these are represented by, respectively, instances of form features, instances of constraints, and the feature model. In assembly modelling, these are represented by, respectively, instances of components, instances of connection features, and the generic combined component.

Because of these similarities, new classes are defined to establish a uniform modelling environment for modelling single parts and assemblies. These new classes are: the Combined, Related and Relation class, all shown in Figure 5.11 on the facing page with a bold face.

The Relation class is introduced as a base class for all objects that represent a relation between building blocks, i.e. the constraints and connection features. The Related class is introduced as a base class for all building blocks, i.e. the instances of form features and the instances of components. The Combined class is introduced as a base class for classes in which sets of related objects and relations between them are specified, i.e. feature models and generic combined components. Using these three base classes, a uniform way in modelling single parts and assemblies is created. This uni-

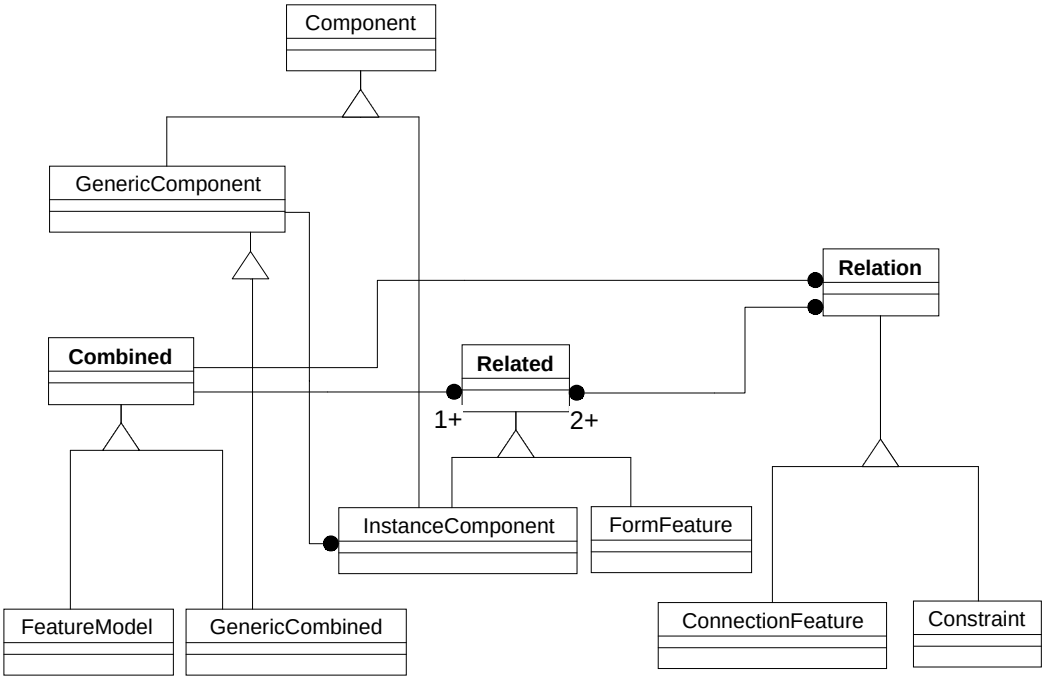


Figure 5.11: Combined data structure for modelling single parts and assemblies

form modelling concept can be used to combine the two modelling environments into one environment, in which both single parts and assemblies can be modelled.

Chapter 6

Assembly features

In Chapter 2 and in Chapter 3, assembly features were already briefly described with respect to assembly modelling and assembly planning. The assembly features were used to store assembly information not available within commonly used product models.

In the previous chapter, the term connection feature was already used to represent a higher-level relation between instances of components. This connection feature is one type of assembly feature. In this chapter, more details of assembly features will be described.

Parts of this chapter have already been described earlier in several papers, among others van Holland et al. (1995) and van Holland and Bronsvoort (1995*a*, 1995*b*).

6.1 Assembly information and assembly features

As within all disciplines involved in the product life cycle, in assembly also specific information is needed. As within the other disciplines specific information is stored within their specific features, e.g. design information within design features and manufacturing information within manufacturing features, assembly information will be stored within assembly features.

6.1.1 Generic and instance level assembly information

Looking at the assembly information needed in modelling and planning, we can distinguish two different levels:

generic level assembly information independent of the actual position and orientation of the component within the assembly;

instance level assembly information dependent of the actual position and orientation of the component within the assembly.

These levels could also be noticed in the previous chapter, where generic components contained assembly information on a generic level, and instance components contained assembly information on an instance level.

Assembly planning information is also generated for either the generic level, or the instance level. This can be seen in, for example, the lowest abstraction level of Heemskerk (1990). Within this level, four activities have to be generated for a component: feed, grasp, move and mount. The feeding, and some elements within grasping, are identical for every generic component. However, some elements within grasping, and the moving and the mounting of a component can be completely different for every instance. Therefore, assembly information can also be differentiated between generic and instance level — between generic and instance components.

Handling information, which is information mostly on the generic level, is therefore related to the GenericComponent class.

However, connection information, highly dependent of the actual specific connections between a component and other components, is assembly information on instance level. Therefore, connection information is related to the InstanceComponent class.

To store this assembly-specific information within a product model, special assembly information carriers are defined, the assembly features.

6.2 Assembly feature definitions

In Section 2.4, several definitions of assembly features were given, summarized in:

- the elementary relations between components (Lee and Andrews 1985);
- the elementary relations between components extended with some assembly information (De Fazio 1990);
- collection of elementary relations and matching form features (Sodhi and Turner 1991);
- an association between two form features present on different parts (Shah and Tadepalli 1992).

All these definitions are, in one way or another, focusing on the relation between components. Although the relation between components is very important within assembly, more information is needed. The assembly features are therefore redefined here, because the previous definitions are far too restrictive.

An *assembly feature* is defined as an information carrier for assembly-specific information. This is a broad definition, but needed to carry all assembly-specific information within modelling and planning. Assembly features are used to store assembly-specific information in a product model and to retrieve it from the product model.

Using this definition, it is not strictly necessary to directly link assembly-specific information to generic shape information, as is done within definitions for design features and manufacturing features. There exists assembly-specific information not directly mappable to some generic shape, e.g. grippers used to grasp a component and the base component to start an assembly with, and this information needs to be stored in the product model as well.

Assembly information can be divided into two types. The first type represents assembly information used to handle a component, i.e. handling-specific assembly information on generic level. The second type represents information about the connections between components, i.e. connection-specific assembly information on instance level. So the assembly features are divided into:

handling features representing handling information and

connection features representing connections between components.

It is possible that more information than described here should be included in these features, because this information is also useful within modelling or planning. Of course, this is possible by extending the data structures. Here only a framework is provided to store all required information.

6.3 Handling features

Before the actual assembly operation can take place, the components to assemble must be positioned somewhere near the area of partial assembly. Fixturing, feeding and almost all grasping information can be stored on the generic level. This is done by using the handling features.

6.3.1 Information within handling

Looking at the assembly process as a black box performing assembly activities on components, one can distinguish on the input side a large number of components entering the system, and on the output side a relatively small number of components leaving the system. All these components must be fed into and transported out of the system, which can be done using component trays or component feeders. By using component trays, the number of components and the used pattern on a tray must be known, because the system must know when a tray is empty and whether an empty tray must leave the system. Also the actual position and orientation information, or ways to retrieve this information using sensors, must be provided. It is assumed that component feeders do have an infinite capacity, and that components on these feeders are always positioned and oriented in some predefined way.

Some of the components can be used as base component, i.e. the first component of an assembly that is assembled. This base component is assembled onto a fixture tray or fixturing spot. It is assumed that a base component is always fixtured in a predefined way; thus position, orientation and fixturing tools are known for a base component.

To move a component from feeding position to fixture, or to assemble it on a partial assembly, a grasping tool or gripper is needed. It is possible that a component can be grasped by more than one gripper. Each type of gripper has its own specifications, and can grip a component in a different way. For a generic component, it is possible to generate all possible areas where a specific type of gripper can grasp the component. This can be combined with the contact areas used in the feeder or component tray, and the contact areas involved in fixturing (for a base component). For every type of gripper, usable for a generic component, a set of grip areas can already be calculated, independent of the actual position and orientation, see for example Figure 6.1 on the next page. However, the actual grip of a gripper can only be calculated on component instance level, knowing the already assembled components, which can further reduce the set of possible grip areas. See Chapter 9 for more details on how to find possible grip areas on components.

6.3.2 Handling features class

A handling feature is in some way different from other features, in the sense that there are no generic descriptions of types of handling features, i.e. there does not exist some predefined set of handling feature types with

corresponding behavior linked to these types. Therefore every handling feature instance is unique and carries information on

- feeding,
- fixturing,
- grippers and
- matching grasping areas.

Thus the HandlingFeature class provides methods to store and retrieve information about feeding, fixturing and grasping, for a generic component. This can also be seen in Figure 6.1.

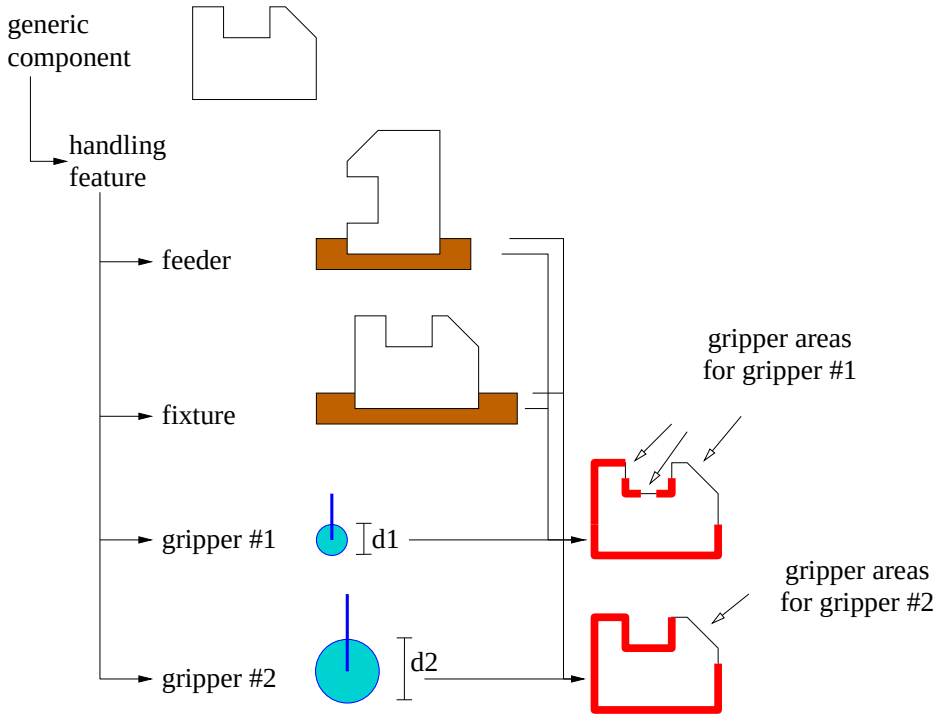


Figure 6.1: Attributes of handling feature

feeding and fixturing information

The feeding and fixturing information can be stored by providing predefined position and orientation information, together with involved contact areas.

Feeding and fixturing information can be modelled in the same way as the assembly model itself. The component together with its feeder or component tray can be seen as an “assembly”, with mutual connection information. A comparable “assembly” can be constructed for component and fixture. These “assemblies” can be represented by the GenericCombined class, described in Subsection 5.3.3: combining components and connection features. As can be seen in the next section, the connection features related to the GenericCombined class can provide information about position, orientation and contact areas, exactly the information needed in the handling features.

gripper and grasp area information

Although a gripper can be represented by a complete assembly, and an actual grip on a component can be seen as an assembly, following Baartman (1995) we have chosen for a generic gripper definition, because a gripper can be very complex due to the large number of components in the gripper. Many of these gripper components do not have any effect on the grasp areas on the grasped component, but will have an enormous time-effect on the generation of these grasp areas. Therefore we use specific information about grippers for the calculation of grasp areas and the actual grip:

- the number of fingers,
- the maximum finger width,
- the finger length,
- minimal and maximal grasp forces,
- finger tip parameters and
- possible available motions.

By using this information, combining it with the already allocated contact areas involved in feeding and fixturing, the grasp areas on a component can be calculated, and stored together with the gripper specifications. How this is done, is described in more detail in Chapter 9 on the grip planning module.

6.4 Connection features

Taking a closer look at an assembly will show that there are several kinds of relations between instances of components. These relations can, with some exceptions, be represented by elementary relations. However, there are two main disadvantages of this commonly used method. The first disadvantage is the large number of elementary relations to be specified between the instances of components to represent all relations between them. The second disadvantage is the level of abstraction of these elementary relations. A designer thinks in connection types between components, and not in elementary relations. These higher-level connection types can be structured within so-called *connection features*. These connection features are much closer to the way of thinking of a designer than the elementary relations.

6.4.1 Information within a connection

The idea of connection features is that characteristics of connection types can be incorporated in these features. By specifying a connection feature in a product model, the assembly-specific information known by the feature is also available in the model. Characteristics of a connection are, of course, the static characteristics when a connection has been established, but dynamic characteristics are even more important, especially within assembly planning. These dynamic characteristics are about how the connection can be established (during assembly) or can be broken (during disassembly).

Here an extendible list is provided, to show some information that is available when the connection type is known:

involved form feature types The type of the form features involved in the connection, e.g. both the rib and slot feature in a rib-slot connection.

final position The final position, or goal position, is the position and orientation of the assembled component relatively to the partial assembly, after the assembly operation has been completed. This position can also be specified by the relative position and orientation of the involved form features.

insertion point The insertion point is the position and orientation relative to the final position where there is just no contact between the component to assemble and the partial assembly to assemble it on.

insertion path The insertion path is a trajectory from the insertion point to the final position. This trajectory can also be a predefined algorithm for some compliant motion, i.e. a motion using contacts during assembly to eliminate uncertainties in position and orientation.

tolerances A tolerance required to establish a certain connection between the components.

contact areas Contact areas are those areas involved in the connection.

internal freedom of motion The internal freedom of motion (IFM) is the set of motions that can separate the component and the partial assembly. This can be both translational and rotational freedom of motion.

geometric refinements Geometric refinements are special refinements to ease the assembly operation, e.g. rounds and chamfers.

6.4.2 Connection feature class

Commonly used definitions for assembly features, which in fact only represent connection types, are too strict for connection features. Only the type of connection between *two* form features present on *two* different components are modelled. However, there exist connections between more than two form features and between more than two components, e.g. a bolt and nut connection connecting several plates.

The ConnectionFeature class provides methods to store and retrieve assembly information for a specific connection between several components. Every specific connection can have its own GenericConnectionFeature class, derived from the base ConnectionFeature class. In this way, connection features can be compared to form features, in the sense that connection features are also divided into a set of generic connection features from which instances can be generated.

The generic connection features hierarchy is represented by the class diagram of Figure 5.9 on page 53.

elementary and compound connection features

As with many other features, also in connection features a distinction can be made between elementary and compound features. Compound connection features can be subdivided into simpler compound features or, finally, into elementary features. The plane mating connection can be used

as the most elementary connection feature, see Figure 6.2(a). With combinations of this connection feature, almost every connection feature can be described as a compound feature, in particular the commonly used elementary relations. For example, a rectangular slot, a V-shaped slot and a dove-tail connection are all a combination of several plane mating connections, and even a pen-hole relation can be described by a number of plane mating connection features. However, it is better to have a predefined set of commonly used connection features, the *elementary connection features*, because additional assembly information can be better provided in such a way. Some examples of such elementary connection features are given in figure 6.2.

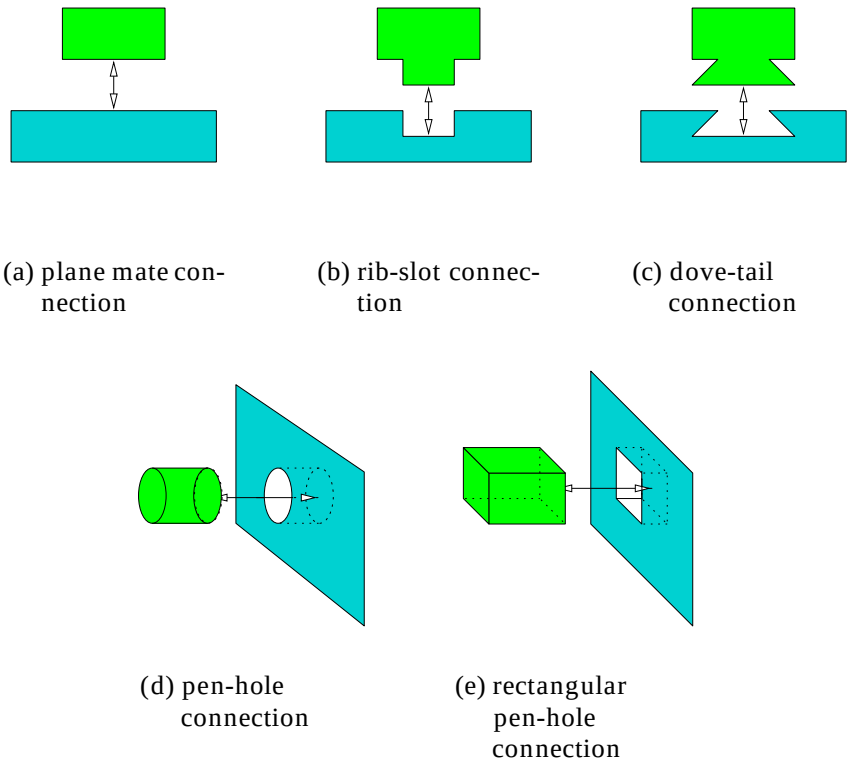
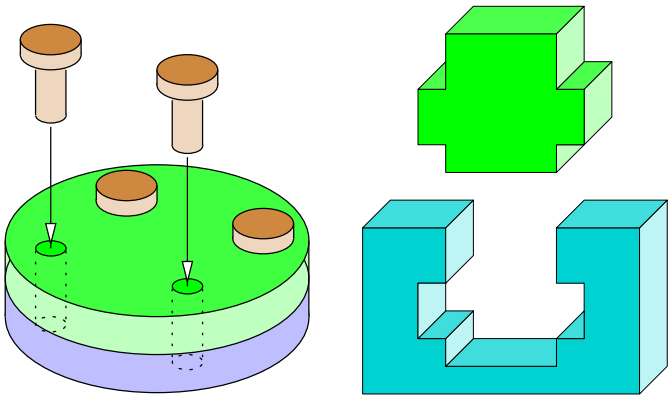


Figure 6.2: Examples of elementary connection features

Each industry has its own set of elementary connection features, i.e. a set used in the automotive industry will be different from a set used in the aircraft industry. The sets should also be extendible. Assume that there ex-

ists a set that matches the requirements for some department within some industry, then there will always be demands for, possibly new, connections not defined in the used set. It is possible to extend the used set by defining complete new ones, but it is also possible to create new connections by combining already existing ones. The latter can be done by using the *CompoundConnectionFeature* class. Examples of compound connection features are given in Figure 6.3.



(a) compound circular pattern connection feature

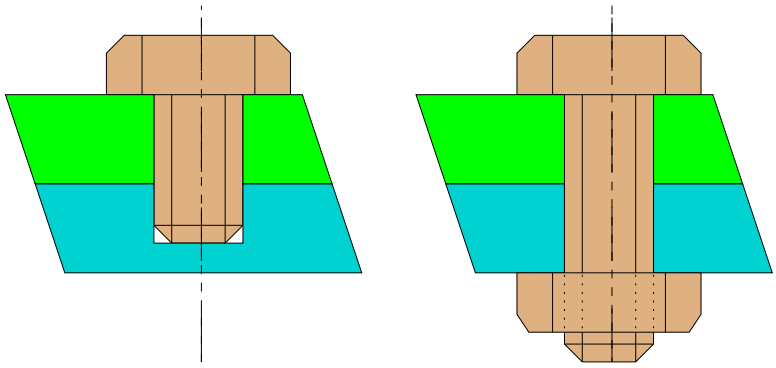
(b) compound rib-slot connection feature

Figure 6.3: Compound connection features

6.4.3 Attachments and agents

A special type of connection is the *attachment*. Attachments are connections enforced by so-called *agents*. Agents, consisting of components and connections, are only needed to enforce a connection.

Take, for example, a functional connection to fix two plates. This can be done by taking a bolt attachment connection. So, by specifying the functional connection to fix two plates, a new component is introduced with its own connections, i.e. the bolt with its mutual connections. The new components and mutual connections are called the agents — divided into component agents and connection agents. By choosing another, different, connection to fix the plates, e.g. bolts and nuts to fasten them, other components and connections are introduced. See also Figure 6.4.



(a) plates connected with single component agent (instance bolt component) and two connection agents (two threaded pen-hole connections)

(b) plates connected with two component agents (instance bolt and nut components) and three connection agents (two pen-hole and one threaded pen-hole connection

Figure 6.4: Connecting two plates with different attachment connections

The fact that a component or a connection is an agent, is important during modelling. It provides information why a certain component or connection is in the model: the mutual attachment connection is needed. Deleting an agent from a model will therefore have direct influence on the attachment relation.

Because an attachment is always related to at least one component agent, by instantiating an attachment, always one or more specific components are instantiated.

In the definition of the attachment connection, it is defined that the agents only enforce the connection. This means that there is primarily a connection behavior of the attachment, and, in addition, some component behavior. This can be found in the characteristics of the Attachment class. The Attachment class inherits directly from the ConnectionFeature class, providing the connection behavior. The attachment should also contain associations to the agents, i.e. components or connections. However, the agents themselves should also be able to tell to which attachment they belong. To solve this, two new classes are introduced, the ComponentAgent and the ConnectionAgent, which derive from respectively, InstanceCom-

ponent and ConnectionFeature. Both agent classes contain an association back to the attachment to which they belong, see Figure 6.5.

By instantiating the attachment, both the included components and connections are instantiated.

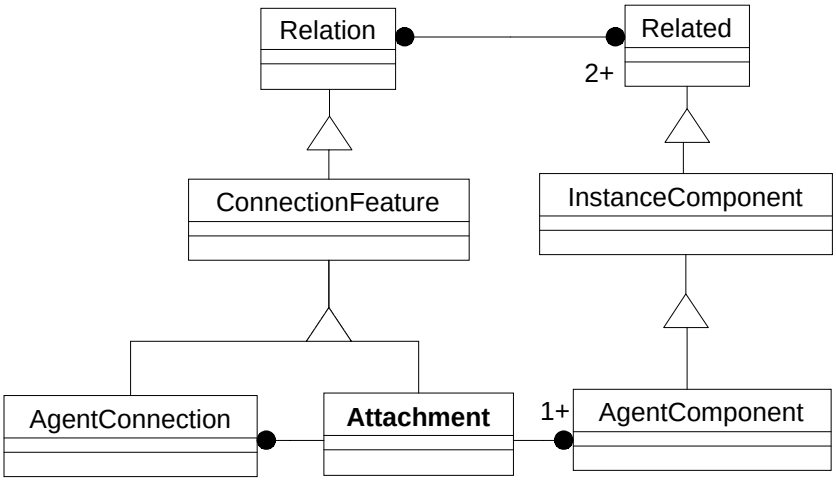


Figure 6.5: The attachment class

Chapter 7

Assembly modelling prototype system

In Chapter 5 the feature-based assembly model, and in Chapter 6 more details of the assembly features have been described. It is, however, not enough to specify only the assembly model and the assembly features. In this chapter, the way how to work with these models is described. In addition, some implementation issues of the prototype system are described.

7.1 Prototype architecture

Before more details are described of how to interact with the product models, first the global architecture of the prototype is presented, see Figure 7.1 on the following page. In this architecture, modelling and planning algorithms make use of the same product model, the feature-based product model as described in Chapter 5. There is one interface module, used for control. By using one interface module, it is possible to call planning algorithms during modelling and vice versa, which is useful for DFX analyses. Some of the functions can be accessed by a graphical user interface (gui), to provide the user with an easy mechanism to access the model. Other applications, which do not directly make use of the gui, can also have access to the interface for modelling and planning functionality.

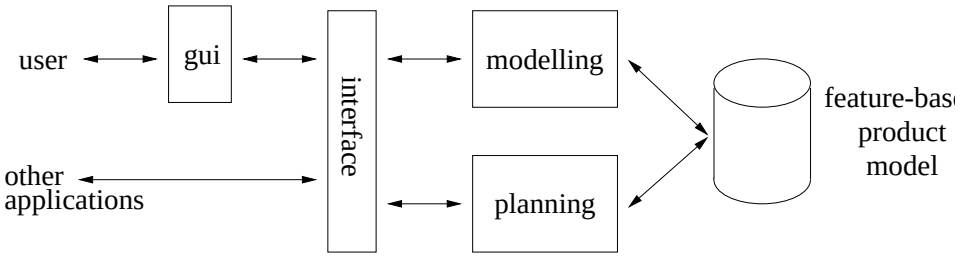


Figure 7.1: Global prototype architecture

7.2 Assembly modelling versus actual assembly of a product

It is obvious that assembly modelling and actual assembly of the product are not the same. However, sometimes modelling systems enforce that building of the assembly model takes place in the same order as the assembly sequence during actual assembly, i.e. the actual assembly prescribes the assembly modelling steps.

The obligation that a designer already provides an assembly sequence or a stable partial assembly during modelling, can have negative influence on the creative possibilities of the designer.

There is, however, no need for such a direct link between modelling and actual assembly. The only requirement is, that a modelling system should enforce that the final model representing a product or subassembly can be assembled. The modelling system must provide tools to come as quickly as possible to such a feasible model.

In some stages during design, however, the designer may want to know, whether an assembly sequence is available, or whether a certain assembly remains stable. The modelling system must provide possibilities for these checks, but only execute them on direct or indirect request of the user.

Another disadvantage of a direct link between modelling and assembly, is that the user is not allowed to model via unrealistic sub-solutions like several non-connected components, or already instantiated connections without matching components. It must be possible in some design stages to still have some undetailed elements or loose links in the model. The modelling system must, however, be able to recognize these, and remind the user if needed.

How these disadvantages can be avoided is described in Section 7.4.

First the user interface for the integrated modelling of single parts and assemblies is described.

7.3 Combined class viewers

Section 5.4 already described a combined data structure for the FeatureModel and GenericCombined classes, see also Figure 5.11 on page 55. This Combined class structure provides a uniform modelling environment for single part and assembly modelling. In this section, the user interface of this class is described.

From Combined objects, there are association relations to both Related and Relation objects. There are also association relations between Related and Relation objects themselves. All theses objects and their mutual associations can be represented by a graph; the nodes are the objects, the arcs the associations. On the other hand, the Combined object itself represents some geometry, i.e. a single part or an assembly.

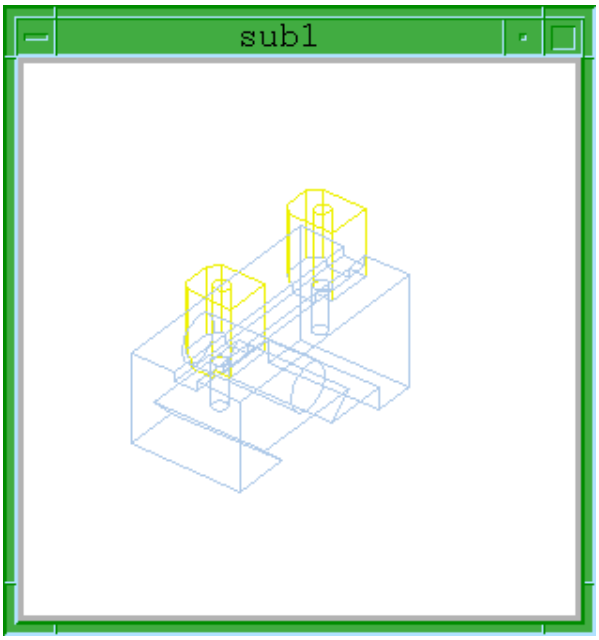
This gives two different ways of looking to the same object, as a graph or as geometry. Therefore, a *geometry viewer* and a *graph viewer* are used. Both viewers have advantages and disadvantages, and used together they can supplement each other.

7.3.1 Geometry viewer

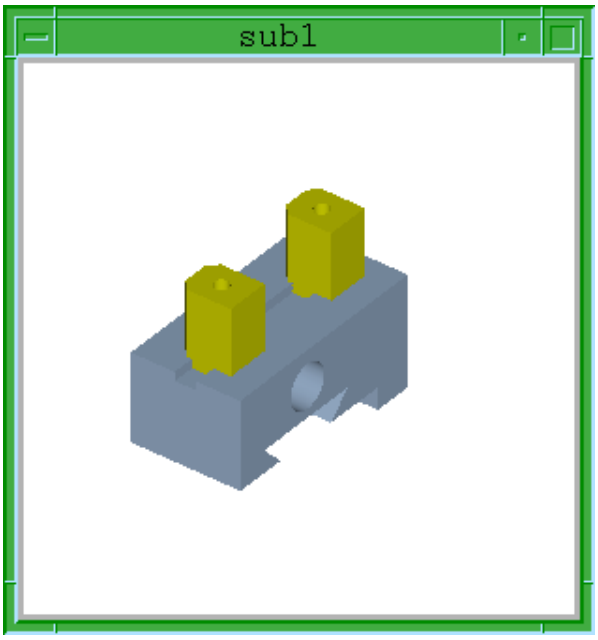
Within the geometry viewer, the geometry of the model is shown. Because we make use of the ACIS™ geometric library for the underlying geometry, we also made use of the ACIS™ capabilities to display the geometry, using line drawings, shaded drawings and mixtures of both. In Figure 7.2 on the following page, an example of a geometry view is shown. By showing the geometry to the user, a good spatial notion of the model is provided. The user can zoom in or out, and rotate the model in the view, to examine specific details in the model. With the use of mouse clicks, the user can retrieve information from faces, form features and components.

There are three main disadvantages in using a geometry viewer:

- When complex models are viewed, with many faces overlapping each other, it is very hard to select a specific face, form feature or component. This can be solved by always retrieving all relevant items under the mouse pointer, and by sequentially highlighting them, selecting the right one. However, sometimes it can be very hard to see whether the desired element is highlighted.



(a) line drawing



(b) shaded drawing

Figure 7.2: Geometry view of a combined model

- Relations between form features or instance components, represented by the Relation class, are very difficult to examine within the geometry viewer. One can show contact areas, but the type of connection cannot be shown. This can be solved by using icons within the geometry view, but within complex models, users can hardly separate the geometry from the icons.
- The structure of the whole model is hardly shown within the geometry viewer. The way the model is constructed from Related and Relation objects cannot be seen within the geometry viewer.

These disadvantages can be overcome by using the other viewer, the graph viewer.

7.3.2 Graph viewer

The Combined objects consist of Related and Relation objects with mutual associations, and these can be represented in a graph structure.

These Combined objects can also be used for building larger Combined objects, e.g. a subassembly in a product. This recursive structure can be represented by a hyper-graph structure, where nodes in the graph can be other graphs representing an InstanceComponent. This InstanceComponent is an instance of a GenericComponent, being a feature model or a subassembly.

Advantages of viewing the data structure with the graph viewer are:

- Both the Related and Relation objects can be shown within one structure.
- Both the Related and Relation objects can directly and easily be selected by following the links in the graph.
- The structure of the whole model is shown by the structure of the graph, i.e. not only the structure of the relations between related objects, but also the hierarchy of the structure.

An example of a graph view is shown in Figure 7.4 on page 75. For the actual building and viewing of the graph, the XHDG widget set — the X-toolkit Hierarchical Directed Graph widget set (Zink 1994) — has been used. This widget set is capable of representing multi-level graphs. One can hide details of a graph on a lower level, by representing only the complete graph as a single node in a higher level. By clicking on this node, the node will be expanded and the underlying graph is shown.

Displayable and displayed class

It is preferable that all objects of a product model are uniformly represented, and have the same behavior in the graph. To achieve this, a new class is introduced, the Displayable class. If an object of the model wants a representation of itself in a graph, the object must inherit this behavior from the Displayable class. Therefore, the Combined, Related and Relation classes all are derived from the Displayable class, see Figure 7.3.

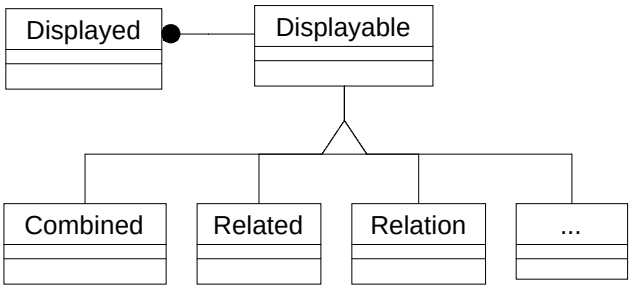


Figure 7.3: The Displayable and Displayed class

However, several Displayable objects can be viewed several times in one model. This is, for example, the case when the same generic component is instantiated several times. The instantiated component itself is unique. The associated generic component can be shown multiple times in the graph, once for each instantiation. To handle this, the same displayable object must be displayed multiple times on several places. This is performed by the Displayed class. For every *displayed* displayable, a new Displayed object is created. So, a Displayable can be associated with multiple Displayed's, and a Displayed is always associated with only one Displayable.

In Figure 7.4, an example is shown of a graph view. The model represents a product derived from the DIAC 1 prototype product (Storm 1988). Every node in the graph, represents a Displayable object: Combined, Related or Relation. The arcs in the graph represent the associations between the objects. On the left side, in the middle, you can see the node representing the Combined object, i.e. the GenericCombined object, of the complete assembly. Dashed arrow lines are drawn to represent the associations between the GenericCombined object — the assembly — and the InstanceComponent objects — the instantiated components in this assembly. In

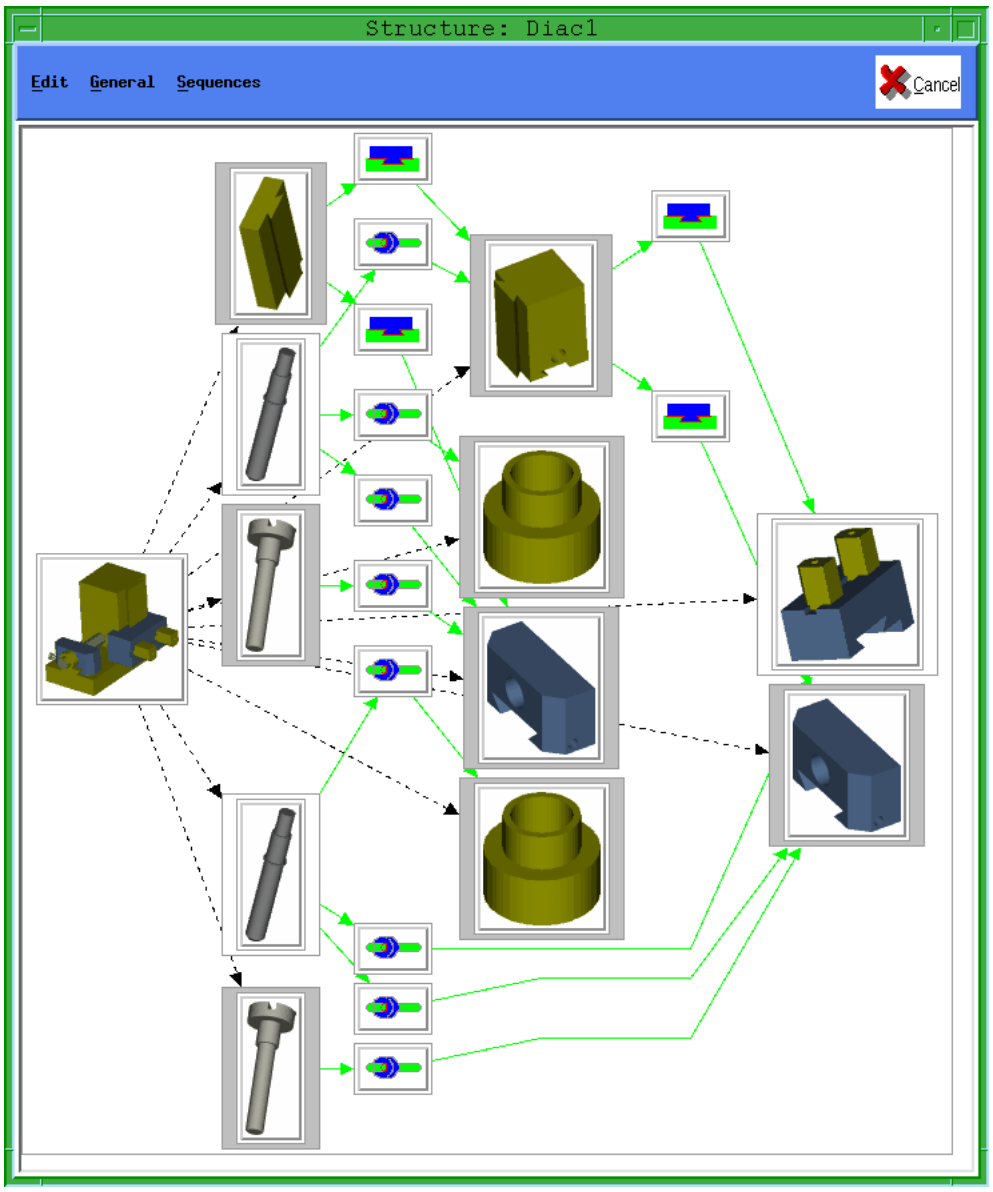


Figure 7.4: A graph view of the DIAC 1 prototype

the same way arcs could be drawn to the ConnectionFeature objects — the connection features — but to avoid cluttering, this is not shown. Associations between Related and Relation objects are represented with solid arrow lines. Nodes with a double border can be expanded to show a more detailed structure. The difference between a grey and a white border is

that grey bordered nodes represent single-part feature models, whereas white bordered nodes represent subassemblies.

To describe the possibilities of expanding nodes to show their underlying graph, a smaller example is used. This example, one of the subassemblies of the DIAC 1 prototype, is shown in Figure 7.5. The geometry of this subassembly is shown in Figure 7.2.

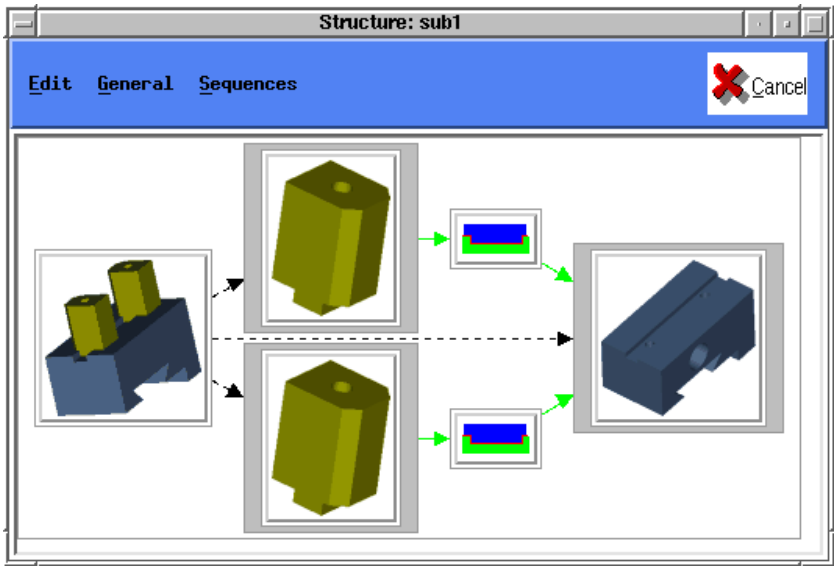


Figure 7.5: Graph representing a subassembly

By clicking one of the expandable nodes, the node shows its corresponding sub-graph, see Figure 7.6 on the facing page. This sub-graph shows more details, and therefore the associating arcs can now point to this expanded information, i.e. at first only the associations on component level are shown by the arcs, but after expanding the associations on form feature level can be shown. In this way the form features involved in a connection feature can be shown.

If a node represents a subassembly, then the underlying subassembly can be shown by expanding the node. This can go on until the node represents a single-part feature model, which is expanded to the underlying feature model. Even the individual features can be expanded to show the faces involved in a relation, see Figure 7.7 on page 78.

From showing these examples, it becomes clear that there must be possibilities to hide certain information within the graph viewer. Information that can be provided, but is not needed at some moment, is not shown,

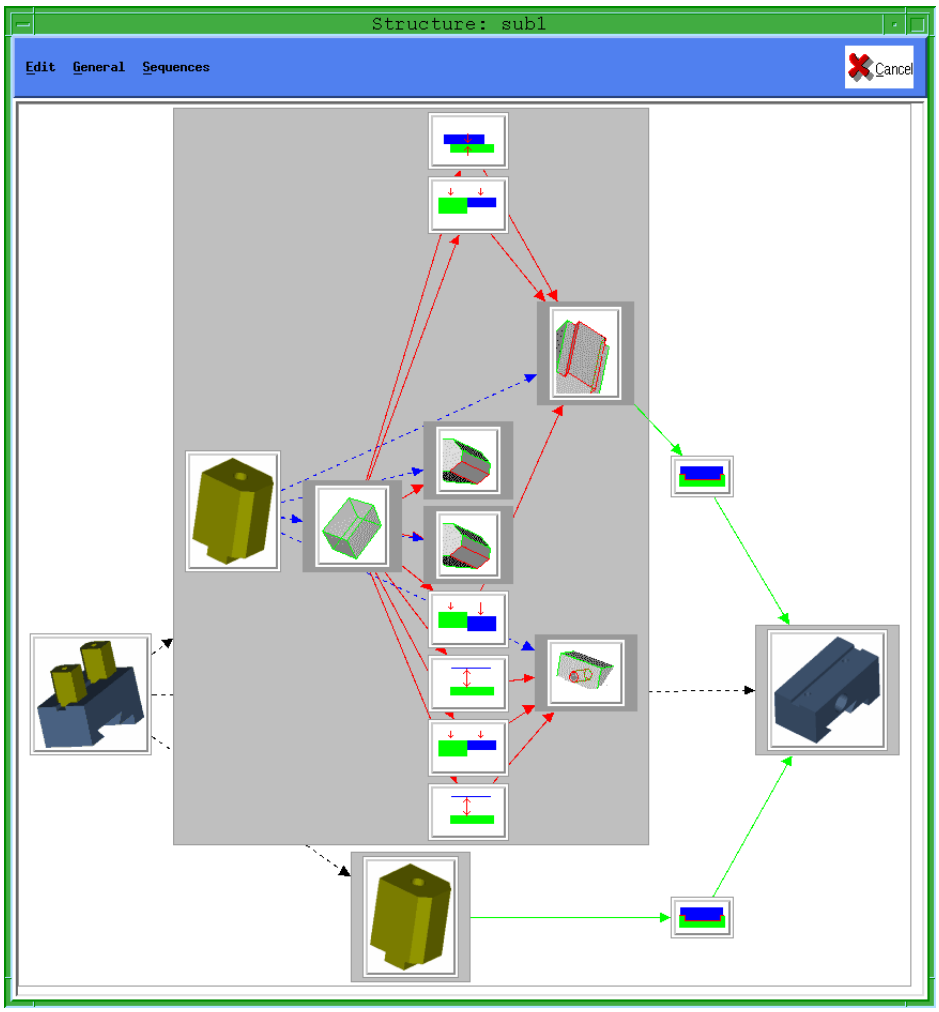


Figure 7.6: A node can be expanded to show its underlying model; notice that associations are now shown on form feature level

because otherwise the models would become extremely large. This fast growing of a graph is in fact the main disadvantage of using the graph viewer.

Of course, one can combine the geometry and the graph viewer: by selecting one of the nodes in the graph, the represented elements in the geometry viewer are highlighted.

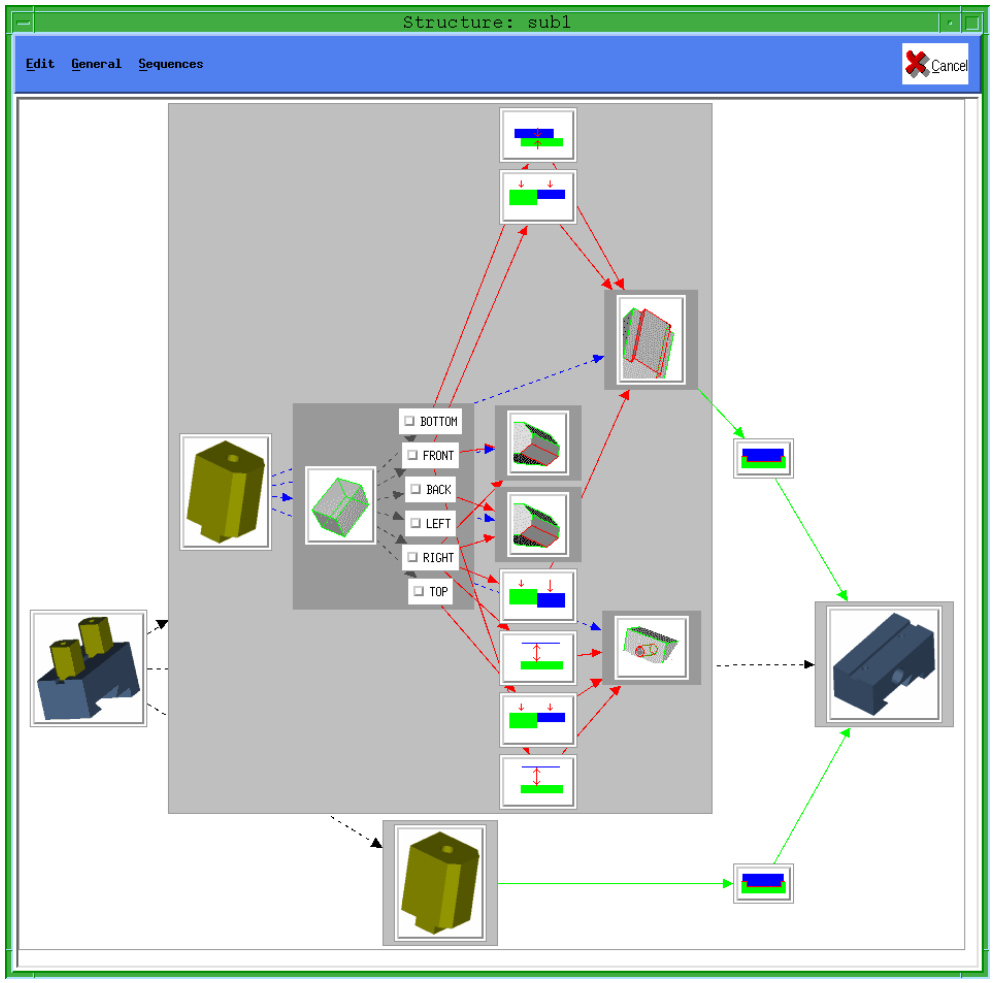


Figure 7.7: By expanding a node representing a form feature, the underlying faces are shown

7.4 Combined class modelling

In the previous section, two ways of viewing existing models were described, but nothing was said about the creation of models. In this section, possible ways of creating and maintaining the combined models are described. Some operations have actually been implemented, others have not.

Basically there exists two ways to create and modify a model:

off-line By describing the model in a text file, and parsing the file into the

modelling system. The creation and modification of the model using a text file can be done completely separately, i.e. off-line, from the modelling system.

on-line By using the user interface of the system, the user can create new and modify existing assembly models. This interface has been partly implemented.

There is no need to further describe the steps taken within off-line modelling using a text file — it works as with any other interpreter. However, some concepts of on-line modelling warrant some further explanation.

7.4.1 On-line modelling

Before the actual creation of the combined model can start, the generic elements needed in the structure must be available. These elements can be separated into the related and relation elements, and depending on the fact whether you want to create a feature model or an assembly, the form features and constraints, or the generic components and connection features are shown.

For example, before one can create a feature model, form feature descriptions and constraints must be available in some kind of library. These can be selected and added to the feature model. Then associations between the form features and the constraints must be defined in such a way that the result will be the required feature model.

An element added to the model can be an object from the Related or the Relation class. Dependent on which type you first add to the model, there can exist two types of modelling:

- related driven modelling and
- relation driven modelling.

Both can coexist within the same modelling session.

related driven modelling

In *related driven modelling*, first instances of Related objects, generic form features or generic components, are added, and then the needed Relation objects between these Related objects are specified.

This is comparable with already existing modellers, where first the entities that contain geometric information are provided (form features or

components) and then the relations between them, to create the complete combined model.

The only addition of this concept to the commonly used bottom-up concept is that both Related and Relation objects contain more knowledge about each other because of the features. For example, according to already specified Related objects, a selection can be made of possibly matching Relation objects. This can be helpful for the designer in specifying the right relations between related objects.

A disadvantage is the fact that this still is a bottom-up modelling concept: the detailed generic objects must already be available before the relations are specified.

relation driven modelling

In *relation driven modelling*, first the desired Relation objects are specified, and then Related objects are specified.

At first sight, this does not differ much from the previous way, e.g. the Related objects can be selected from a matching list dependent on the already chosen Relation objects. This is still bottom-up modelling.

However, the relation driven modelling concept can be helpful in realizing a top-down modelling concept. Suppose that the user is capable of specifying some undetailed Related objects. Then, by specifying Relation objects between these instances, and because of the knowledge available in the Relation objects, matching geometry can be created on the Related objects. In this way, more detail is specified for the Related objects. This is modelling from the conceptual towards the detailed level, i.e. top-down modelling.

mixing related and relation driven modelling

Of course, it is also possible to have both modelling techniques available in one modelling environment. In this way, the user can choose which technique he wants to use to create a model.

Some parts of the model can preferably be done by using the related driven technique, for example when using well-known standard components. Other parts of the model, which rely on more creative aspects of the model with regard to shape and preferred connections established by these shapes, can preferably be done by using the relation driven modelling technique.

7.5 Implementation issues

Most parts of the assembly modelling prototype have indeed been implemented. Because the focus of this prototype has been to validate the use of assembly features in modelling and planning, these parts have got more attention than pure user interface aspects. In the current implementation, models can be created and modified in the off-line way. For the interpreter, the packages flex and bison[1] were used. In the off-line modelling files, first the used generic components are described, thereafter the instances with their relationships. Single parts and assemblies can be described completely mixed in one and the same file.

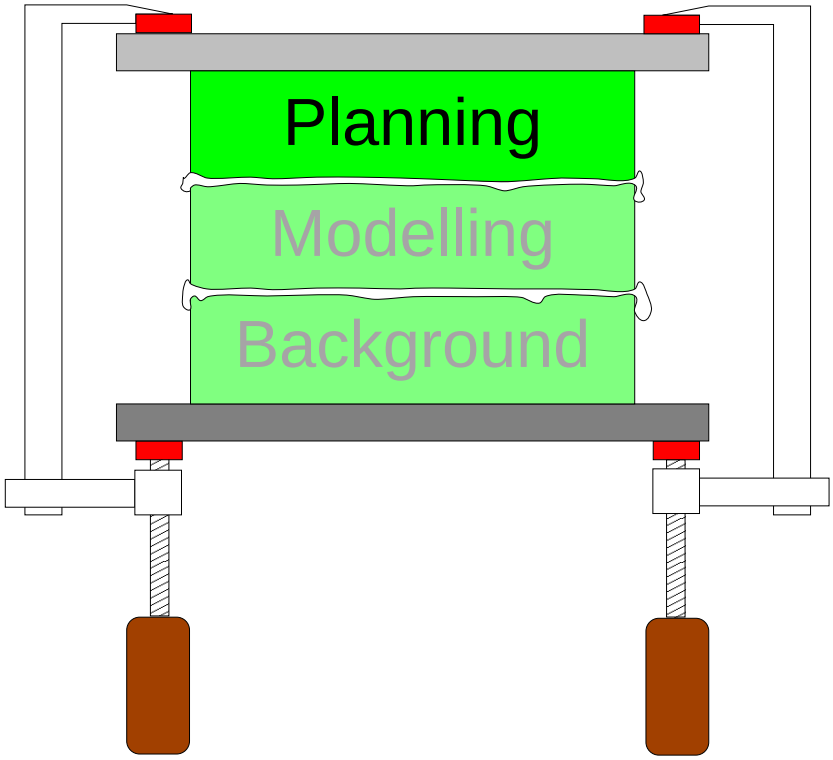
The system is able to create the complete components and assemblies from the provided instance related and relation objects. A simple constraint solver has been implemented to generate the position and orientation of every instance related object. The complete geometry of the components and assemblies is represented with the ACIS™ geometric library. Once a part or product has been created, it can directly be used as a generic component by instantiating it into another partial assembly.

For displaying and examining, the ACIS™ functionality for displaying and manipulating 3D graphical objects has been used. The objects can be displayed as line drawings, shaded images or both. For high-quality shaded images, utilities have been created to convert the model to formats used by standard rendering software. The XHDG widget set (Zink 1994) is used to create and manipulate the multi-level graphs. All the figures of windows and of shaded images in this thesis have been taken from the prototype system.

In the prototype, the planning modules can directly be accessed during modelling, both in the graph and in the graphical viewers. This makes it possible to perform DFA actions during modelling.

[1] flex and bison are the GNU variants of the commonly-used Unix tools lex and yacc

Part III



Part III discusses the use of the object-oriented feature-based product model with assembly features in assembly planning modules.

It is not the intent to give a detailed description of every planning module needed within assembly planning. Some have been chosen, to verify the concept. Besides this verification, new and extended planning algorithms are presented to show additional benefits of the product model.

First, in Chapters 8 and 9, extensive descriptions of using the product model in stability analysis and grip planning are given. This is followed by a discussion, in Chapter 10, of how the product model can be used in motion planning and assembly sequence planning.

Chapter 8

Stability analysis

Stability analysis is a very important issue in assembly. For example, in subassembly planning, an assembly is decomposed into stable subassemblies, and in assembly sequence planning, stable components must be assembled onto stable partial assemblies. So, in one way or the other, in assembly planning there must be some stability analyzing tool.

According to Boneschanscher (1993), three types of stability can be distinguished, depending on the forces taken into account:

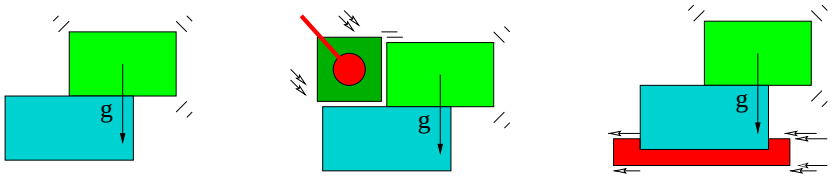
gravitational stability only takes into account the gravity. If an assembly falls apart under influence of the gravity, the assembly is said to be gravitationally instable.

assembly stability takes into account both gravity and additional forces caused by the joining operation. If an assembly falls apart during assembly of a component, the assembly is said to be assembly instable.

motion stability takes into account gravity and additional forces due to acceleration during motion of the assembly. If an assembly falls apart during moving it from one position to another, the assembly is said to be motion instable.

Examples of these types of stability are shown in Figure 8.1 on the following page.

This chapter will focus on the first stability type, gravitational stability. Therefore, where the word *stability* is used, gravitational stability should be read. The other types of stability can be checked using the same methods as for gravitational stability, by using the resulting force instead of the gravitation force. However, the exact calculation of the resulting force can be very difficult, and is not within the scope of this thesis.



(a) gravitational stability

(b) assembly stability

(c) motion stability

Figure 8.1: Different types of stability

To analyse whether an assembly is stable, two steps can be distinguished:

1. Analyse translational stability
2. Analyse rotational stability

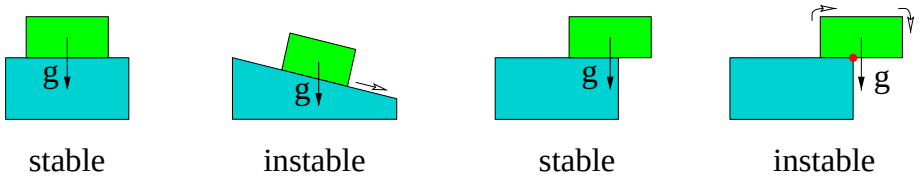
Within translational stability analysis, only translations between components are taken into account, see Figure 8.2(a) on the next page. Within rotational stability analysis, only rotations between components are taken into account, which is far more complex than the first step, see Figure 8.2(b). One advantage of dividing the stability analysis into these two steps is that the second step can be omitted when the first step already finds the assembly to be instable.

Both for translational and rotational stability analysis, first some background is given, thereafter the profits of using assembly features within stability analysis are shown.

Parts of this chapter have already been published in van Holland and Bronsvoort (1995*a*).

8.1 Translational stability

Translational stability analysis checks whether a component can translate relatively to the other components.



(a) Translational stability (b) Rotational stability

Figure 8.2: Translational and rotational stability

During assembly, each time a component is added to the partial assembly, and a relative easy thing to analyse is whether a just added component has some effect on the new partial assembly.

Taking friction into account, the just added component on a partial assembly can have stability effects on other components already assembled. However, if friction is not taken into account, only the just added component must be checked for stability. In this chapter, friction is not taken into account, so there is no need for searching other instable components in the new partial assembly.

To know whether a component can translate under influence of the gravity, one has to investigate whether it is possible to translate, or move, the component in the gravitational direction. Therefore the *internal freedom of motion* (IFM) of a component is needed, the set of motion directions that can separate a component from its partial assembly (Lee and Yi 1993). If one of the directions in the IFM can be transformed in such a way that it contains an element in the direction of the gravitational force, then the component is instable in relation with its partial assembly.

8.1.1 Internal freedom of motion

We define the internal freedom of motion between two components as $IFM(c_i, c_j)$, i.e. the set of motions that can separate component c_i from component c_j. By knowing the set of separating directions of c_i from c_j, also the opposite can be found, i.e. the set of motions that can separate component c_j from component c_i:

$$IFM(c_i, c_j) = -IFM(c_j, c_i).$$

If component c_i has both a connection with component c_j, and a connection with component c_k, then the set of motions that can separate component c_i from both component c_j and c_k, is found by taking the intersection of the two IFMs:

$$IFM(c_i, c_j c_k) = IFM(c_i, c_j) \cap IFM(c_i, c_k).$$

This rule can be generalized: the internal freedom of motion of a component c_i having connections with n other components is:

$$IFM(c_i, c_1 \ldots c_n) = \bigcap_{j=1}^{n} IFM(c_i, c_j).$$

Lee and Yi (1993) used in their definition of internal freedom of motion only the three principal axes. This is far too limited: in that case also assembly will be limited to only one of these directions. To accomplish all possible motion directions for the complete 3D space, another representation is needed. This can be done by using a method called *visibility mapping*, which is also used by Chen et al. (1993*a*, 1993*b*) for several applications in manufacturing, such as computing machinability on CNC machines, and mould and die design. Mattikalli and Khosla (1992) also used this method for stability analysis.

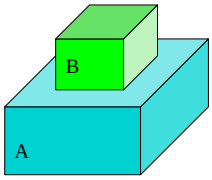
8.1.2 Visibility maps

The visibility mapping method maps the possible translation directions onto a unit sphere, resulting in a so-called *visibility map* or *VMap*.

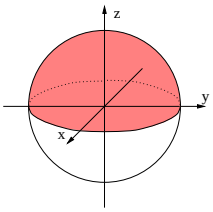
If a component c_i has a plane-mate connection with component c_j, see Figure 8.3(a), then component c_i can only move in the set of directions represented by:

$$\left\{ \vec{t} \mid \vec{n} \cdot \vec{t} \geq 0 \right\},$$

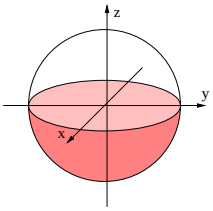
where $\vec{n}$ is the normal of the contact face. Mapping these possible translation directions $\vec{t}$ onto the unit sphere, will result in a hemisphere, shown in Figure 8.3(b) and 8.3(c). If this is done for every contact face between two components, and the intersection of all the hemispheres is taken, this will result in a convex spherical polygon representing the internal freedom of motion between the two components. This method can also be extended for non-planar faces, as described by Mattikalli and Khosla (1992).



(a) Two components with plane mate connection



(b) $IFM(c_A, c_B)$



(c) $IFM(c_B, c_A) = -IFM(c_A, c_B)$

Figure 8.3: Internal freedom of motion

8.1.3 Central projection

Computing with VMaps is difficult, because taking intersections of convex spherical polygons is possible, but time consuming. Several attempts have been made to simplify these calculations on VMaps. This can be done by transforming the 3D VMaps onto 2D maps, and perform the calculations in 2D, which is easier. Martens (1991) uses a transformation with spherical coordinates, but this results in taking intersections of sinusoid functions in the 2D domain, which is still very difficult, and not stable when the directions are close to the poles of the sphere.

A better approach is used by Oliver and Huang (1994) and Woo (1994), using the *central projection* transformation.

projecting onto one plane

With the central projection transformation, every point on the unit sphere is projected onto the plane $z = 1$. This is done by:

$$\vec{p}(\vec{t}) = \frac{\vec{t}}{t_z} = \left(\frac{t_x}{t_z}, \frac{t_y}{t_z}, \frac{t_z}{t_z}\right) = \left(\frac{t_x}{t_z}, \frac{t_x}{t_z}, 1\right) = \left(\frac{t_x}{t_z}, \frac{t_y}{t_z}\right)$$

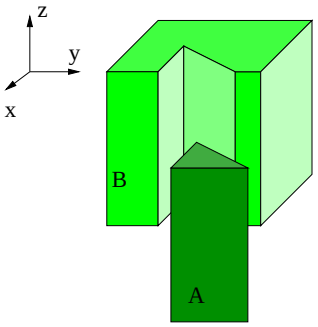
The advantage of this method is that every great circle on the sphere is transformed to a straight line in the 2D domain. A great circle is defined as the intersection of a plane with a sphere, where the center of the sphere is also a point on the plane. Every VMap is represented by a convex spherical polygon on a unit sphere, where the edges of the polygons are parts from great circles. So these edges are projected onto straight lines in 2D, see Figure 8.4(c) on the facing page. This implies that taking intersections of VMaps is reduced to taking intersections of convex polygons, which is relatively simple.

A problem of projecting the VMap onto only one plane, $z = 1$, is that *antipodal points* — (x, y, z) and $(-x, -y, -z)$ — on the sphere are projected onto the same point (p, q) on the plane. However, it can be the case that one of these antipodal points represents a possible motion direction, whereas the other does not. By projecting them onto only one point on the $z = 1$ plane, the problem is introduced that the difference between them cannot be distinguished any more.

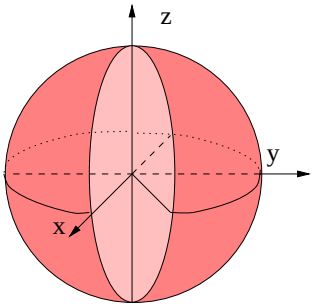
projecting onto two planes

This problem can be solved by *not* projecting onto one plane, but instead onto *two* planes. The points on the sphere beneath the plane $z = 0$ are projected onto the plane $z = -1$, and the points above the $z = 0$ are projected onto the plane $z = 1$. The sphere is thus transformed into two planes, $z = 1$ and $z = -1$; see Figure 8.4(d) on the next page for an example.

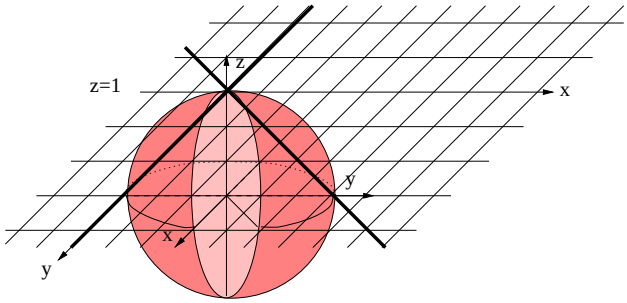
Another, major, problem both for projecting onto one and two planes is the numerical degeneracy problem. For points on the sphere with $t_z \approx 0$, the range on the projection planes becomes very large; points with $t_z = 0$ are projected onto infinity. A standard solution for this is to try to rotate the VMap in such a way that no t_z value is near zero. This can only be done if the spherical convex polygon on the unit sphere is smaller than the hemisphere, and this is not always the case. Take, for example, the plane mating connection, which has a complete hemisphere as VMap, see Figure 8.3(b) and 8.3(c) on the preceding page.



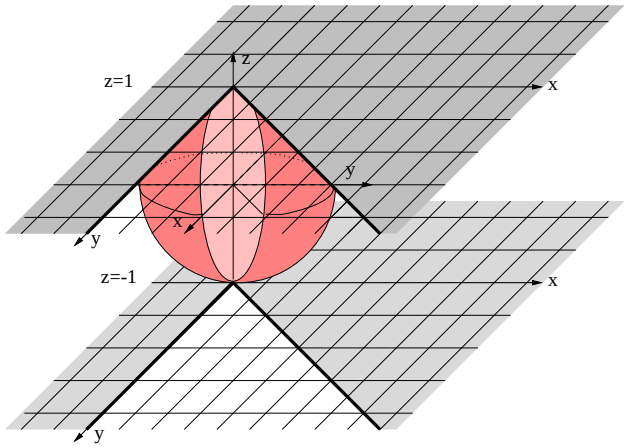
(a) example assembly



(b) VMap representing $IFM(c_A, c_B)$



(c) central projection using one plane ($z = 1$)

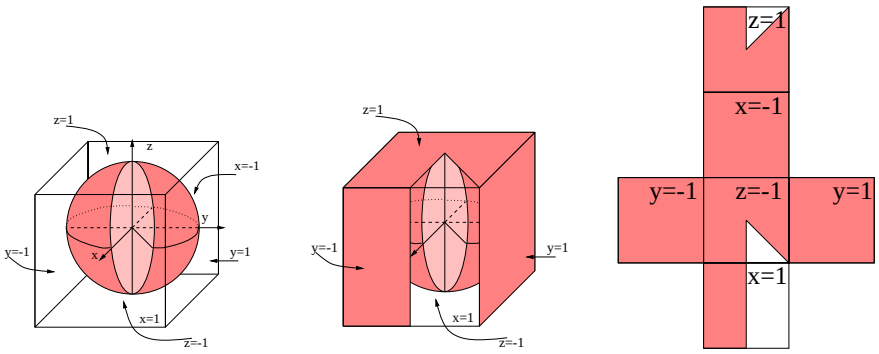


(d) central projection using two planes ($z = \pm 1$)

Figure 8.4: Central projection

central projection using the cube projection

A better solution for the numerical degeneracy problem mentioned in the previous subsection, is to project the sphere onto the six planes of a cube, instead of two planes [1]. This is done by determining for each point on the sphere which of the t_x, t_y or t_z components has the largest absolute value. If this is the t_x component, then the point is projected onto the plane $x = \pm 1$, dependent on the sign of the t_x component. Similar rules apply if the t_y or t_z component has the largest absolute value. For example, point (1,0,0) is projected onto plane $x = 1$, and point $(\frac{1}{2}\sqrt{2}, 0, -\frac{1}{2}\sqrt{2})$ onto both plane $x = 1$ and $z = -1$. As a consequence of this projection, the range (p, q) of the projection onto a plane will always be limited, within $-1 \leq p \leq 1, -1 \leq q \leq 1$, and the projection onto such a plane will always be a convex polygon. The VMap is thus transformed onto the six planes of a cube bounding the unit sphere; an example is given in Figure 8.5.



(a) VMap (b) central projection onto cube (c) unfolded cube

Figure 8.5: Central projection onto a cube

The intersection of two VMaps is now simply computed by taking for every plane the intersection of the convex polygons of the two corresponding planes of the two cubes.

For the implementation of the box projection, one can simply use the geometry classes already available for creating the assemblies, in our case the ACIS™ geometric library. One starts with a cube, representing all possible motion directions. For every face involved in a connection, one can

[1] Basically the same idea has been developed independently by Mattikalli et al. (1994)

create a halfspace. By taking the intersection of the halfspace and the cube, one retrieves the possible motion directions.

8.1.4 Using connection features for internal freedom of motion

An advantage of using connection features, is that there is already knowledge of possible motion directions available within the generic connection. Every generic connection feature already contains the generic contact areas, and contains a generic description of the internal freedom of motion. See Figure 8.6 on the next page for examples of generic internal freedom of motions for several connection features.

For every instance, this generic internal freedom of motion, can directly be used to retrieve the actual internal freedom of motion. This is shown in Figure 8.7 on page 95 for a simple assembly, where the internal freedom of motion of component D is calculated by intersecting the internal freedom of motions of the involved connection features.

To find translation stability, the resulting internal freedom of motion must be intersected with the central projection of the plane with the normal in the direction of the gravitational force. If this intersection is empty, the assembly is translationally stable, otherwise it is translationally instable.

The same calculation can be used when the resulting force during assembly or motion is known, and this force has a fixed direction during assembly or motion, which is, unfortunately, rarely the case. Then the halfspace should be taken in the direction of the resulting force. If the resulting intersection with the internal freedom of motion is not empty, the assembly is not assembly or motion stable.

8.2 Rotational stability

To decide that a component is rotationally instable, one has to find a rotation axis around which the component can rotate, given the available forces. A rotation axis differs from a translation direction in the fact that an axis contains both a position and a direction. This describes immediately the main difference between analysing translational stability and rotational stability. Translation directions can be mapped onto visibility maps, but for possible rotation axes this is more difficult.

Another difference is that when already more then two components are in the resulting partial assembly, one has to take into account that already

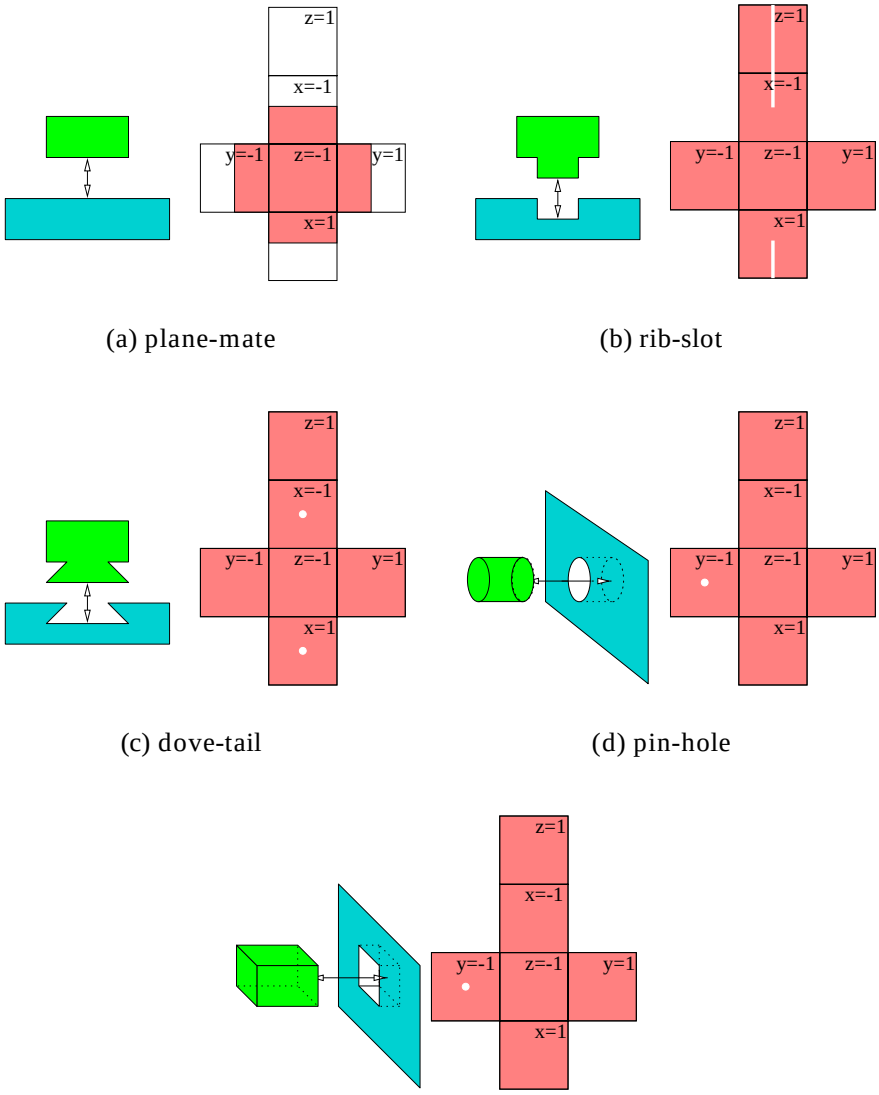


Figure 8.6: Connection features contain a description for the generic internal freedom of motion

examined contact areas can become instable by adding a new component. This can be seen in Figure 8.8 on page 96, where assembling a component C, results in a rotationally instable connection between component A and B, whereas the connection between component B and C itself is rotation-

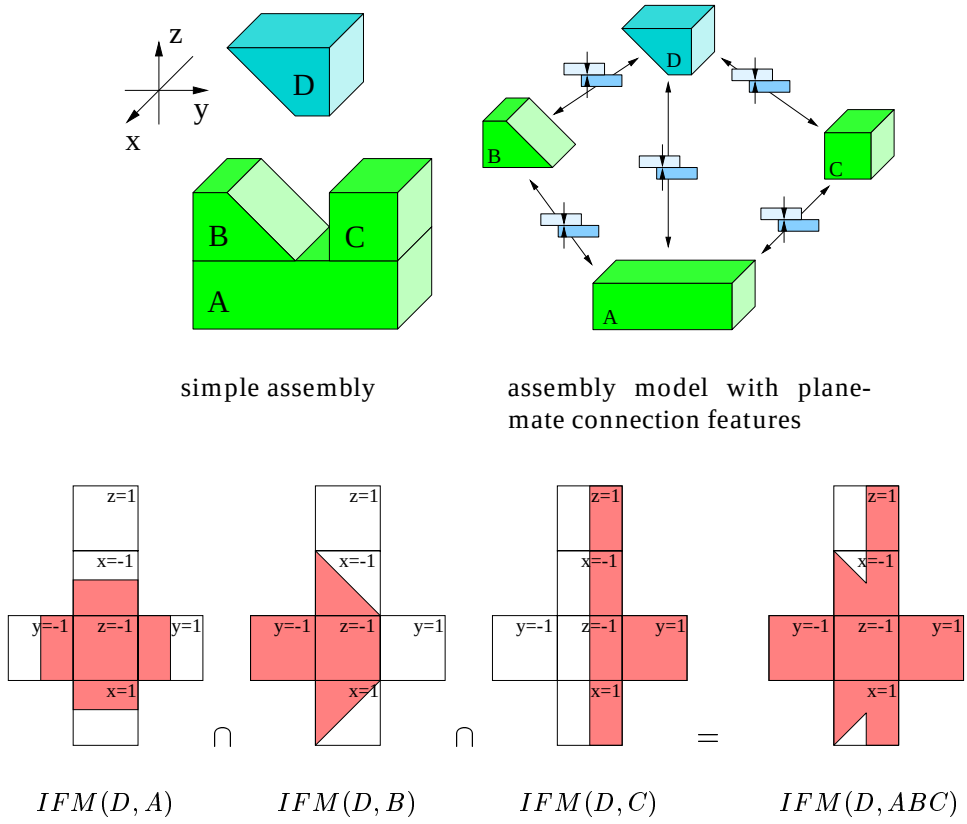


Figure 8.7: Retrieving the actual internal freedom of motion from the involved connection features

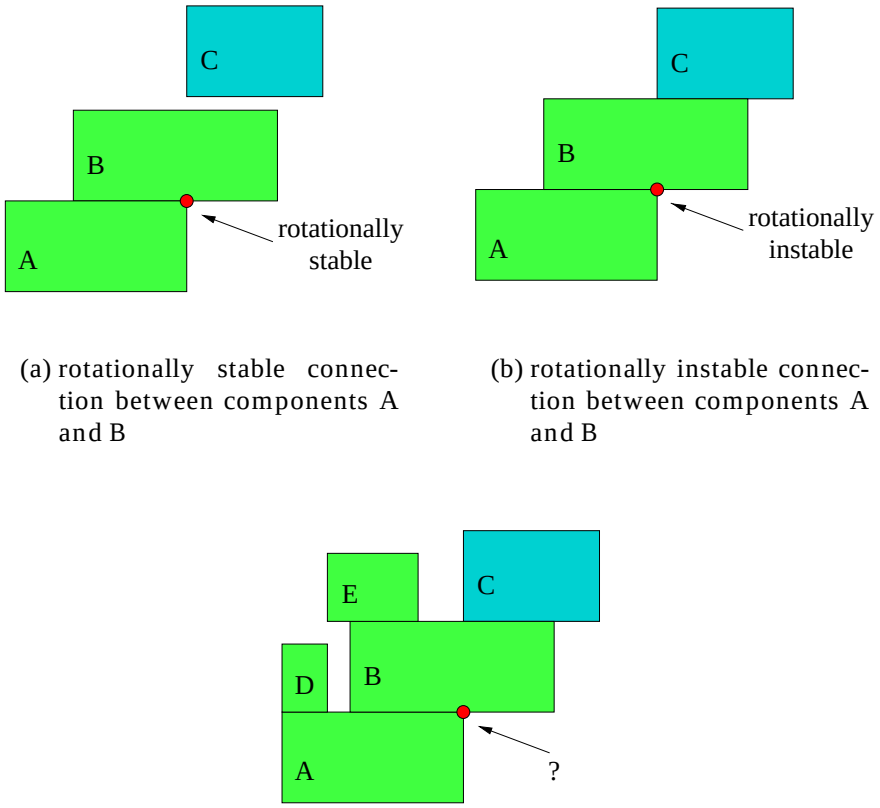
ally stable.

Also other components not directly connected to the just assembled component, can have influence on the stability of the assembly, e.g. component E in Figure 8.8(c). These components must also be taken into consideration.

8.2.1 Rotational degrees of freedom

Mattikalli and Khosla (1991) described a theoretical basis for finding all possible rotation axes; the *rotational degrees of freedom*. They also developed a structure to represent all possible rotation axes.

Every rotation axis can be represented by a direction and a position. All possible directions can be represented on a unit sphere, as was shown



(a) rotationally stable connection between components A and B

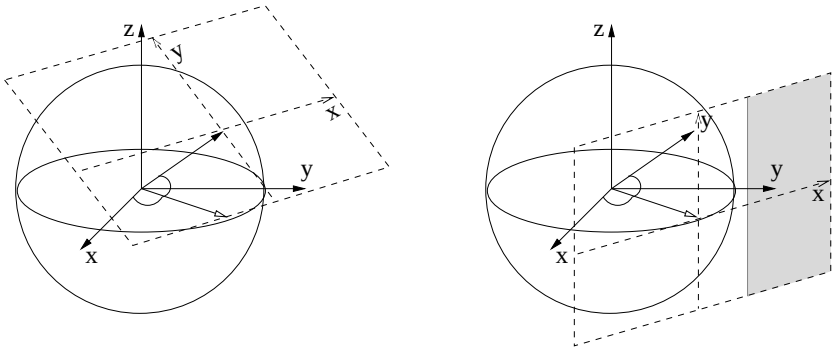
(b) rotationally instable connection between components A and B

(c) rotational stability of the connection between components A and B is dependent of components C and E, but not of D

Figure 8.8: Assembling a component can change a previously stable connection into an instable connection

in the previous section, and for a specific direction all possible axes can be represented in a plane with the same normal as that direction. Any rotation axis in a specific direction can thus be represented by two coordinates on that specific plane. Therefore there must be defined an origin of the plane, and, for convenience, this origin is chosen as the contact point between plane and sphere representing the directions. The x and y axes on the plane are chosen in such a way that the x axis is parallel with the

equator of the sphere, and the y axis is in the direction of the north-pole of the sphere, see Figure 8.9(a).



(a) represent the direction by planes and the position by positions on these planes

(b) represent only planes on the equator of the unit sphere

Figure 8.9: Representing rotational degrees of freedom

For a single contact area, they have proven that it is enough to represent only the planes around the equator of the unit sphere. These planes can be divided by a half plane representing on the one side the possible rotation axes and on the other side the stable axes, see Figure 8.9(b). Further they found that it is enough to represent only those planes that have the same direction as the edges (on the convex hull) of the contact area.

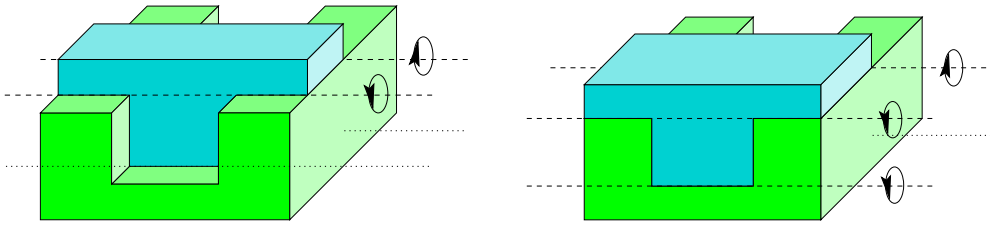
All rotation axes for every contact area can be combined in such a way that they are within one rotational degrees of freedom representation. In this representation, one can immediately see which rotation axes are possible for the given contact.

However, within stability analysis for assembly, there is no need to investigate all rotation axes in 3D space. Only the axes on the borders of the convex hulls of contact areas can generate a rotation due to available forces, and only these have to be compared with the axes in the representation. For the axes that match with the ones already in the representation, it must be calculated whether the assembly is instable under influence of gravity.

8.2.2 Using connection features for rotational degrees of freedom

To find the rotational degrees of freedom, Mattikalli and Khosla (1991) used the edges (of the convex hull) of the contact areas involved. However, by using information available in the connection features describing connections between components, there is no need to investigate all these edges. Take, for example, the rib-slot connection feature. This feature already contains the information that there are maximal four possible rotation axes. Without this information, many possible edges would have to be examined before these four axes were found, see Figure 8.10.

The possible rotation axes are stored within the generic descriptions of the connection features. Using information available on instance level, the actual possible rotation axes can directly be provided, see Figure 8.10. By combining all possible rotation axes of all involved connections, the resulting possible rotation axes are found. These can be combined with external force information, to determine whether an assembly is rotationally instable.



(a) this instance contains two of the four possible rotational axis

(b) this instance contains three of the four possible rotation axes

Figure 8.10: Connection features can give possible rotational axes, by combining available generic and instance information

Chapter 9

Grip planning

Another planning module in assembly planning is the grip planning module. The main purpose in grip planning is to find how a component can be grasped, in such a way that it can be assembled onto the partial assembly. In this chapter, the use of form features and assembly features within grip planning is shown. Before their use is described, first a brief description of grip planning in general is given. More details on grip planning can be found in van Bruggen et al. (1993) and Baartman (1995).

Parts of this chapter have already been published in van Holland and Bronsvoort (1996*a*).

9.1 Grip planning in general

In general, there are several aspects within grip planning (Baartman 1995). First, some gripper, or grippers, must be selected to grasp a specific component. Second, areas on the component must be determined to position the selected gripper on, the so-called *finger domains*. Finally, positions on these finger domains must be chosen for the actual grip.

9.1.1 Gripper aspects

A gripper can only be selected if there exists, at least, one actual grip that is capable of gripping the component. Therefore, to know which grippers can be selected for a specific component, the other aspects — determining finger domains and selecting some of them for at least one grip — must be considered. When this has been done once for a specific component, this information can be stored in the handling feature associated to a generic component, see Section 6.3.

But how do we represent a gripper, in such a way that this gripper model can be used in grip planning? The high versatility of the human hand results in a wide range of components that can be grasped by humans. Much research is done to develop a robot hand that is as versatile as a human hand, but the control of such a gripper becomes very complex. To get less complex control, the gripper must be made simpler, which means that only grips on a restricted set of components can be performed, i.e. the gripper can only be used for a very limited range of products. Most industrial grippers are made to grasp only a uniform set of components, e.g. components within one product family, which severely restricts the flexibility of the assembly system: new components require new grippers.

General-purpose grippers are developed to realize more flexibility in grasping. This flexibility is dependent on the type of the mechanical fingers, the number of fingers, their possible movements, and the contact type between fingers and the component. Here only general-purpose grippers are considered with two or three fingers. Each finger is represented by a small cylinder with a larger sphere at the end of it. Contacts with the component are represented by a point contact, where the sphere is in contact with the component. Baartman (1995) shows that, for flexibility, it is preferable to use point-contact finger-tips. These contact types are easier for computations than, for example, soft finger-tips (Cutkosky and Wright 1986), because the geometry is fixed and therefore contact forces applied to them can be computed. The component in Figure 9.1 is grasped with point-contact finger-tips.

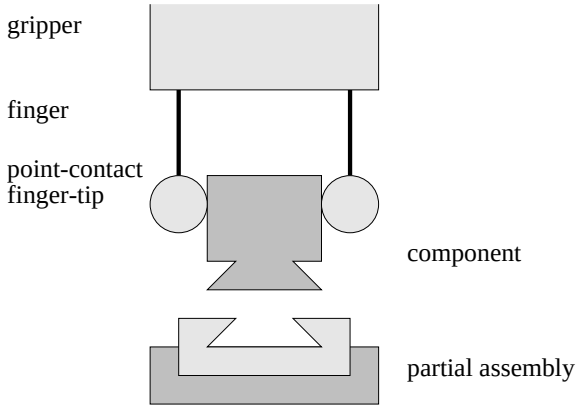


Figure 9.1: Gripper grasping a component; the contacts with the component are represented by point-contacts

9.1.2 Finger domain aspects

To find a set of grips, all areas where the fingers can be positioned must be found, the so-called *finger domains*, which is a time-consuming step described in more detail later on.

Some requirements for these finger domains, described by, among others, Nguyen (1988) and Ponce et al. (1993), are:

feasibility It must be possible to position the gripper fingers on the finger domains.

reachability It must be possible to reach the finger domains with a gripper via a collision free path from an initial position relatively far from the component.

The feasibility requirements must also take into account that the gripper is not positioned on areas of the component involved in the connection to be established and/or in contact with, or close to, the feeder from which the component is grasped. For this, among other things, the contact areas with the fixture, partial assembly and feeder must be known.

9.1.3 Actual grip aspects

When all finger domains have been determined, a selection from these must be made in such a way that the following requirements are met:

closure There must be either force closure or form closure, i.e. the contact forces respectively the location and shape of the gripper fingers must constrain the motion of the grasped component.

equilibrium There must be a balance between the weight of the component and the forces exerted by the fingers.

stability During the grip, the position and orientation of the grasped component, relatively to the gripper, must remain the same.

Most research in finding stable grips has been based on these requirements (Nguyen 1988). When the gripper has only two or three fingers, the positions to place these fingers are always on a line respectively in a plane, called the *grip plane*. This reduces the problem of finding stable grips from a 3D problem to a 2D problem, or a $2\frac{1}{2}$D problem if the orientations of the faces are taken into account.

If all these requirements are met, and several sets of finger domains remain possible, still several actual grips can be selected.

Sometimes it is enough to find only one possible grip. However, when flexible assembly is considered, and off-line planning modules are used, flexibility is increased by determining a larger set of grips on generic component level. These grips are ranked in some way to be able to select the best grip for different situations, during the on-line planning phase, on instance component level.

9.2 Finger domains and non-free regions

As was already mentioned in the previous section, finding finger domains is a very time-consuming step. More details about how to find them in general will be given in this section. Improvements will be described in the next section, where form features and assembly features are used to find them.

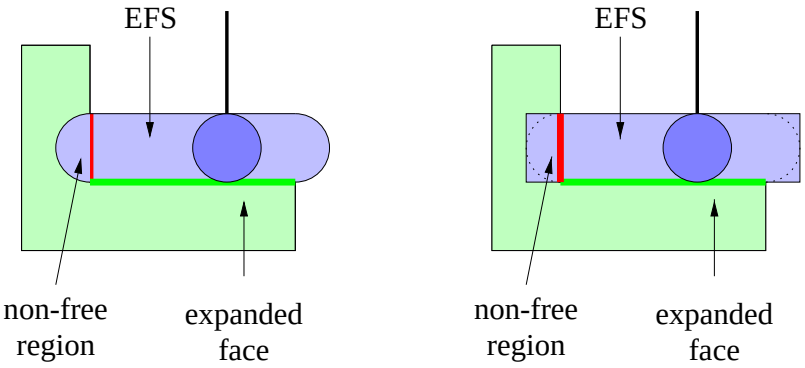
Finger domains can be found by taking the complement of all areas where the fingers *cannot* be positioned. There can be several reasons for such areas: there is no collision free path of the finger-tips to the component; the areas are forbidden, e.g. threaded or easy-to-damage areas; or the areas are in contact with the feeder, partial assembly or used fixtures. Except for the forbidden areas, van Bruggen et al. (1993) have described a method for finding them by looking for so-called *non-free regions* on the component. These non-free regions can be found by using Expanded Face Solids (EFS). Baartman (1995) has described some extensions to this method.

9.2.1 Expanded Face Solids

The basic idea of using EFSs is that every face in the product model is offset by an envelope volume dependent on the thickness of the finger-tip. The intersection of the EFS for a certain face with the model gives the region(s) where the finger-tip positioned on the face has a collision with the model, the non-free regions. The envelop volume is created by sweeping the finger-tip completely over the face, see Figure 9.2(a) on the facing page.

To get easier and faster calculations, the finger-tip is approximated with a box, resulting in a boxed-shaped EFS, see Figure 9.2(b).

To get all non-free regions from the complete model, an EFS for every face in the model must be created, and intersection of these EFSs with the model will result in all the non-free regions. Because an EFS must be created for every face, the time complexity of this method is dependent on



(a) Using the spherical finger-tip shape

(b) Using an approximation

Figure 9.2: Finding non-free regions with the Expanded Face Solids

the squared number of faces.

Although the EFS method is only given for planar faces, it is possible to extend it to other types of faces. However, offsetting a curved face is often very difficult, and is mostly done by faceting, which results in an approximate EFS of the face. Another disadvantage of the faceting step is that it generates many new faces, which is undesirable because of the time complexity of this method.

When all finger domains have been found, several finger domains must be selected for the actual grip. This is done by filtering, using several heuristics for the finger domains, in such a way that a stable grip is found.

9.2.2 Problems using Expanded Face Solids

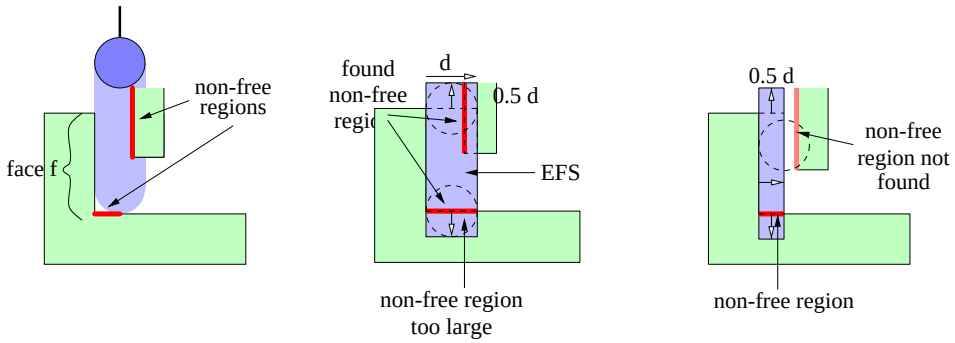
However, there are some problems with the EFS method, because the EFS of a given face does not find the non-free regions on that face, but only the areas possibly influencing that face. An influencing area is an area that is the cause for a non-free region, i.e. the finger will be in collision with the influencing area when it is positioned on the non-free region.

Suppose, we want to find the non-free regions on a model, see Figure 9.3(a) on the next page. To find these non-free regions, we must generate EFSs for all faces. By doing so, every face finds the areas it is influencing

on other faces, but this does not exactly result in all the non-free regions.

Take, for example, the model shown in Figure 9.3(a). It is not clear what the size of the envelope volume of face f should be. By choosing an offset of height d and width $0.5d$ — with d as the diameter of the finger-tip — the found intersections will sometimes be too large, see Figure 9.3(b). In this case, the method is too conservative, specifying too many areas as non-free where in fact the finger can be positioned.

By taking an equal offset of $0.5d$ for both height and width, some intersections will not be found, and some areas are supposed to be finger domains whereas they in fact are non-free regions, see Figure 9.3(c).



(a) non-free regions to be found by face f

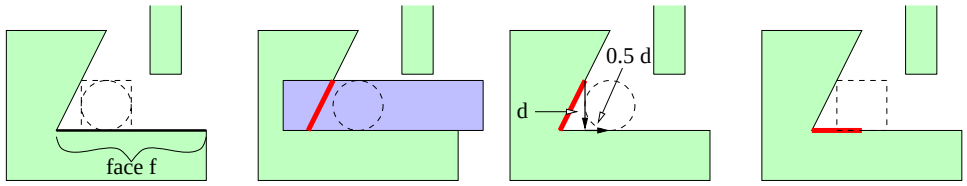
(b) some non-free regions found are too large

(c) some non-free regions are not found

Figure 9.3: Problems using the EFS method

Baartman (1995) has described a solution for these problems by using a more complex method, introducing an extra step. For the generation of the envelope volume the height d and width $0.5d$ are taken. The generated EFS of face f is used to find influencing areas, see Figure 9.4(a) and 9.4(b) on the next page. These influencing areas are used to calculate the non-free regions on face f, by *projecting* the influencing area back onto face f. Finally the area is extended by $0.5d$ to find the non-free region of the finger-tip on face f, see Figure 9.4(c) and 9.4(d). For 2D, this may look simple, but it can be very complex for 3D models.

For this enhanced method, to find all non-free regions on the model,



(a) face f, to find non-free regions on

(b) find influencing area using EFS

(c) project influencing area back onto face f

(d) resulting non-free region for face f

Figure 9.4: Enhanced EFS method, projecting the influencing areas back onto the evaluated face f

again for every face in the model, an EFS must be created to find the influencing areas for that face. These areas must be projected back onto the face to find the resulting non-free regions. Removing these found non-free regions from the complete model will result in the finger domains.

Notice that for finding all possible grips, a lot of information on the component itself, and its connections with feeder, fixture and the partial assembly, has to be taken into account. The connection areas cannot be used for grasping the component; these areas are already in contact with feeder or fixture, or they will become in contact with other components in the partial assembly. To retrieve all this information from a low-level geometric model is very time consuming, and sometimes even impossible. The next section will show how concepts from feature modelling can help to find faster and better solutions.

9.3 Using features in grip planning

The idea of using features in grip planning is to interrogate the features in a product model to see where they can or cannot be grasped. Using feature information will accelerate the finger-domain determination, therefore this method is called Feature Accelerated Finger Domain Determination (FAFDD). Which type of features are used, and in which phase of the finger domain determination, can be seen in Figure 9.5 on the following page.

In the first phase, form features and handling features can give infor-

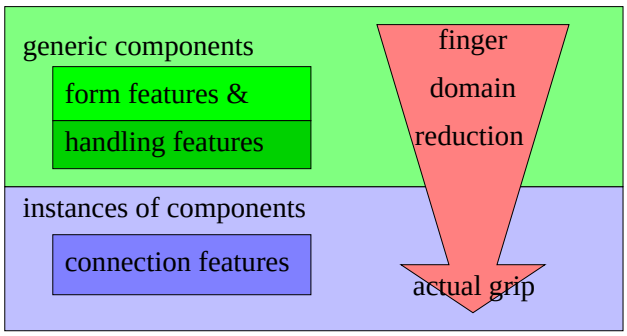


Figure 9.5: Features used to accelerate the finger domain determination, and to provide additional information for the actual grip

mation about finger domains. Every generic component is associated with a handling feature, as described in Section 6.3, which contains information on the gripper(s), feeder and fixtures to use for this generic component. The form features retrieve the finger-tip parameters for a specific gripper from the handling feature, and use these parameters to generate an initial set of gripper-specific finger domains. The areas involved in feeding and fixturing, also stored in the handling features, are removed from this initial set. Thereafter, the resulting set is stored back in the handling feature. The handling feature then contains a set of gripper-specific finger domains for a component, independent of the connections involved in the complete product.

This set, initially the same for every instance of the component, can be used in the second phase to determine actual gripper-specific grips on instance level. This can be done by combining this set of finger domains with connection information and assembly directions stored in the connection features. If several grippers can be used for the component, then for each gripper such a set of finger domains will be computed.

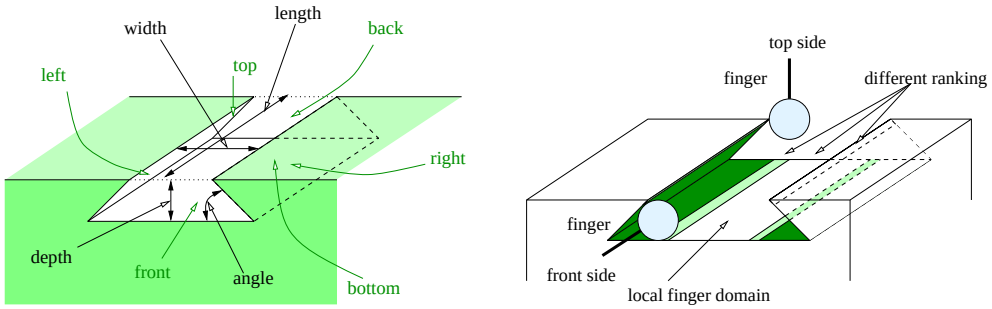
In the next subsections, these steps within FAFDD will be further explained.

9.3.1 Using form features

A feature model consists of instances of generic form features. In every generic feature definition, some information useful for grip planning can be stored. By using this information, every instance can generate potential finger domains from its parameters.

local finger domains

In Figure 9.6, information stored in the generic definition of this dove-tail slot feature is shown. The parameters for width, length and depth are defined here. On the basis of the values for the parameters of an instance, the non-free regions, areas not in a finger domain, can be computed for a specific finger-tip. The influence of other features on this feature is not yet taken into account, and therefore the found finger domains are called *local finger domains*.



(a) generic definition of form feature dove-tail slot

(b) instance form feature dove-tail slot with local finger domains dependent on gripper parameters

Figure 9.6: Generic definition of a dove-tail slot feature, and an instance of it with its local finger domain (different shadings represent different rankings: light shading for areas that can easily be grasped, dark shading for areas that cannot be grasped)

The local finger domains on each feature can be enhanced by giving some kind of ranking to it, depending on the graspability known by the feature. For instance, curved faces can be used for grasping, but it is better to use planar faces. Take, for example, the dove-tail slot feature in Figure 9.6, some finger domains can be reached by the gripper from both top, and front and back side. Others can only be reached from front and back side, resulting in a lower ranking, because the orientation of the gripper must be taken into account here. This ranking can later be used when the best grip must be chosen from all possible grips. In Figure 9.7 on the next page, some generic features and their local finger domains are given.

The local finger domains found for all features in the model can be combined to retrieve the local finger domains for the complete model.

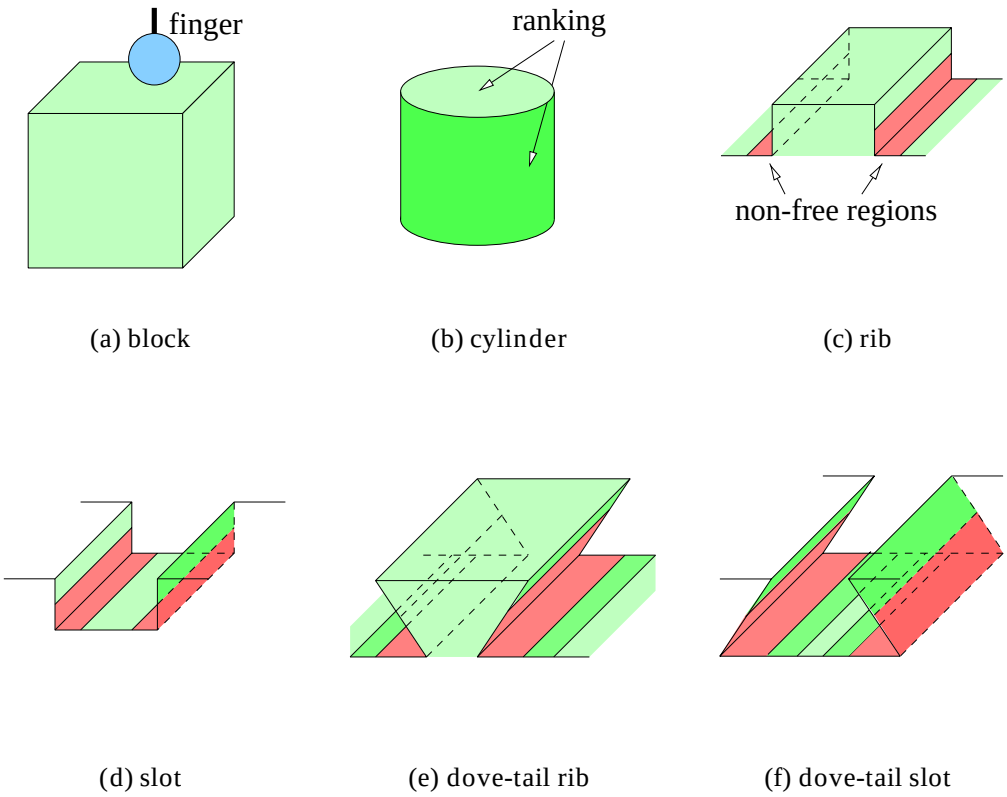


Figure 9.7: Generic form features and their local finger domains

global finger domains

Areas found to be within a local finger domain, can still be non-free regions because of the influence of other features. A feature does, in general, not know, whether there exists some influence from other features. For example, in Figure 9.8 on the facing page, the two rib features may be too close to each other to position a finger in the slot between them. Therefore an extra step within FAFDD is introduced to compute the *global finger domains* from the already generated local finger domains.

This local finger domain reduction might be done by using the EFS method. But using the EFS method, as was described in the previous section, will completely reduce the benefits provided by the features. As was already mentioned before, the EFS of a face finds the non-free regions on *other* faces influenced by the evaluated face. Therefore, to find the global

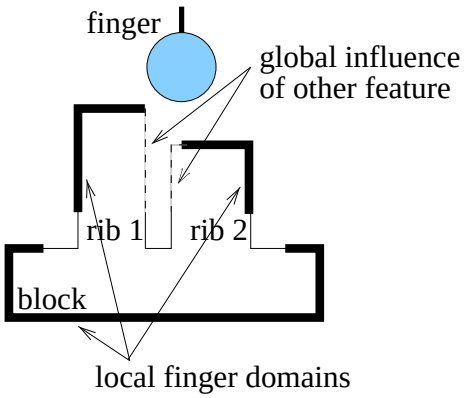


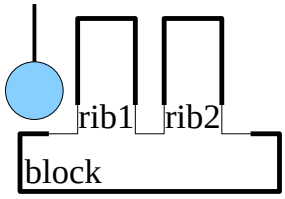
Figure 9.8: Features can have influence on other features, reducing their specified finger domains

non-free regions on the local finger domains, for every face in the model an EFS must be calculated. The number of faces to investigate would therefore be the same in FAFDD as with using the EFS method alone.

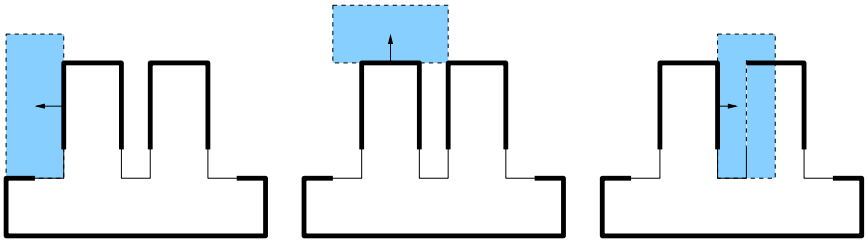
additive features influencing local finger domains

Fortunately, features themselves can give information on whether they can possibly influence the finger domains of each other. Only *additive features*, features that add volume to a model, can have influence on finger domains of other features. *Subtractive features* alone cannot have influence on finger domains of other features. More precisely, only faces belonging to local finger domains of additive features can give rise to a further reduction of the finger domains.

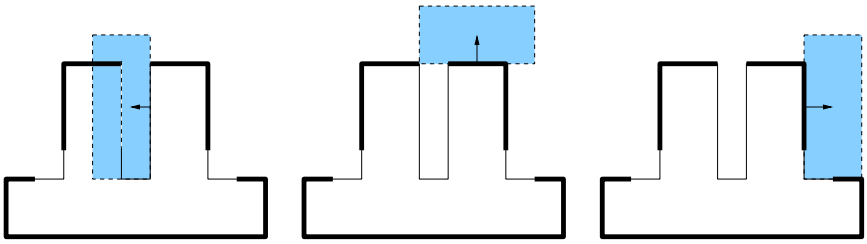
Figure 9.9 on the next page explains this with an example. We start here with the local finger domains from the three features in the model. For every face belonging to a local finger domain of an additive feature, an EFS calculation is performed. This results in a reduction of the local finger domains of the model, because the two ribs have influence on each other. There is no need to investigate non-free regions or faces not belonging to additive features (the starting block feature is neither additive nor subtractive). The number of EFS calculations performed in this example is 6, instead of 12 when the original EFS method would be used.



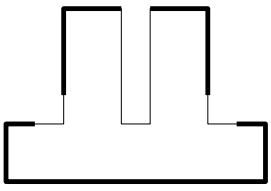
(a) start with the local finger domains



(b) check the influence of additive feature rib1



(c) check the influence of additive feature rib2



(d) resulting global finger domains

Figure 9.9: If additive features have influence only on each other, the local finger domains of the additive features are needed for EFS calculations

subtractive features influencing local finger domains

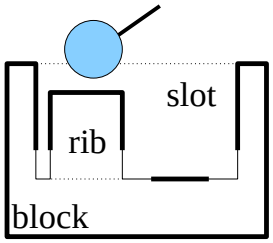
As stated before, a problem of the initial EFS method applied on a face is that it does *not* find the non-free regions on the evaluated face itself, but on the faces that this face influences. In the example in Figure 9.9 on the facing page, this was not a problem, because the faces influencing each other were both belonging to additive features. There is no need in projecting the influencing areas back onto the faces, because of the fact that the influencing areas belong also to additive features, and these are in turn investigated with the EFS method, finding the projection. Thus both, influenced face and influencing area, were handled by the EFS method, finding the non-free regions on each other.

Unfortunately, the procedure just sketched fails if an additive feature has influence on finger domains of a subtractive feature. The non-free regions on the subtractive feature are found, but the non-free regions because of that subtractive feature on the additive feature itself are not found, i.e. the projection of the influencing area. To find the latter non-free regions, the EFS method must also be performed on the just found non-free regions on the subtractive features, to perform the projection back onto the influenced face.

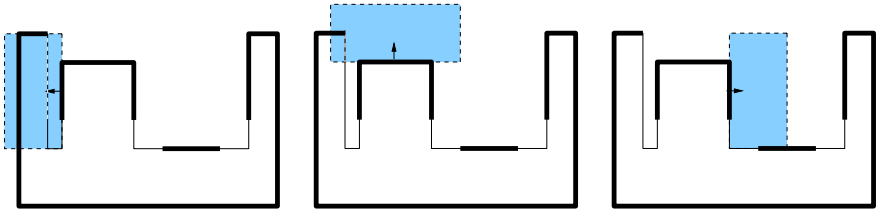
Another example explains this situation. Figure 9.10 on the next page shows the local finger domains resulting from the three features in the model. In step 1, the faces belonging to local finger domains of the additive rib feature are investigated by the EFS method. If found non-free regions belong to the subtractive slot feature, these regions are used in step 2, to find the non-free regions on the additive feature. This results in the global finger domains of the model, found with only 3 EFS calculations and 2 projections due to influencing areas, instead of 12 EFS calculations and 12 projections due to influencing areas, needed in case no feature information would have been used.

method FAFDD versus original EFS

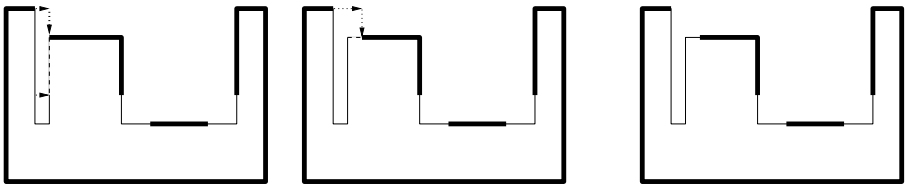
It is clear for which models the time reduction is most substantial using FAFDD. Models with only subtractive features do not require any EFS calculations at all. In the worst case, models with many additive features influencing subtractive features, many EFS calculations and projections of influencing areas are needed to find the global finger domains, but such models are rare in practice. Even then, fewer calculations are needed using FAFDD compared with using the original EFS method.



(a) start with the local finger domains



(b) step 1, evaluate the additive features, and store influencing areas of subtractive features



(c) step 2, evaluate the influencing areas of the subtractive features

(d) resulting global finger domains

Figure 9.10: When additive features have influence on subtractive features, both local finger domains of additive features and found influencing areas on subtractive features are investigated by EFS calculations

9.3.2 Using handling features

Handling features can be used to store information for a generic component. Some relevant information stored in the handling feature can be

used by the form features, e.g. the finger width of a specific gripper is used by the form features to generate the global finger domains for that gripper.

Not only the finger width is stored in the handling feature, but also other information on the grippers used to grasp the component. Every gripper can have a specific number of fingers, maximal and minimal grip opening, and possible grip forces, see also Section 6.3.

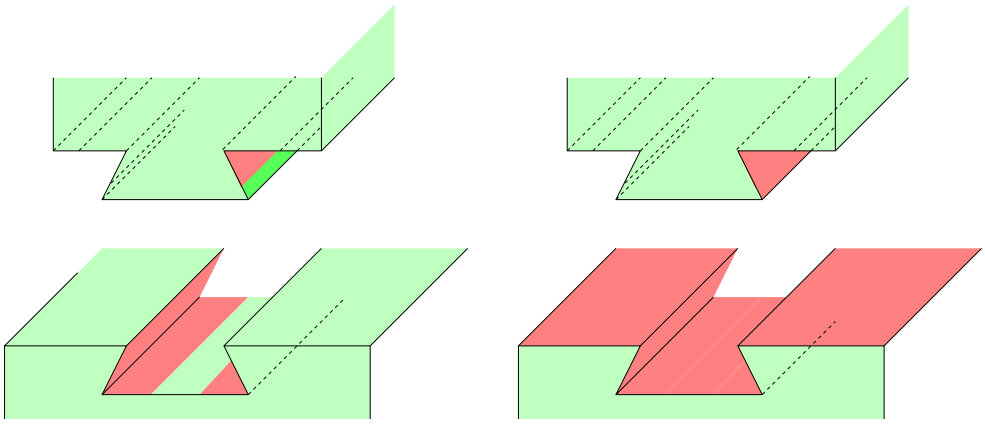
Besides gripper information, also feeder and fixturing information can be found in the handling feature. This information can be used to further reduce each gripper-specific set of global finger domains found for every generic component. All areas involved in feeding and fixturing the component are non-free regions and therefore removed from the set of global finger domains. The resulting set of finger domains is stored for every type of gripper in the handling feature. This gripper-specific set of finger domains can be used for every instance to determine the actual grip.

9.3.3 Using connection features

Until now, only information has been considered independently of the position where the component will be assembled on the partial assembly. This position is different for every instance of the component. Additional useful information can be retrieved from the connection features in the model. These features 'know' which areas cannot be in the set of finger domains because they are involved in a connection. In Figure 9.11 on the following page, this is shown for a dove-tail connection. The connection 'knows' which areas on the dove-tail slot and the dove-tail rib feature are involved in the connection. These areas are removed from the set of finger domains.

Although finding possible stable grips is not covered in this thesis, features can be used in this step as well. Possible grips can be generated by slicing the component perpendicular to the assembly direction. The slice reduces the problem of finding an actual grip from a 3D problem to a 2D or $2\frac{1}{2}$D problem. The global finger domains are intersected with the slice, and the resulting intersecting lines are used to determine possible grips that have correct values for closure, equilibrium and stability.

The form features can be further used to retrieve information on possible grip edges in such a slice. From an edge, the related face and form feature can be found. This feature can give information on the opposite face, which can be used to find a correct closure, e.g. a block feature 'knows' that its back face is opposite to its front face, and that they are parallel.



(a) local finger domains from both dove-tail slot and dove-tail rib feature

(b) further reduction because of the involved dove-tail rib-slot connection feature

Figure 9.11: Connection features can further reduce the finger domains

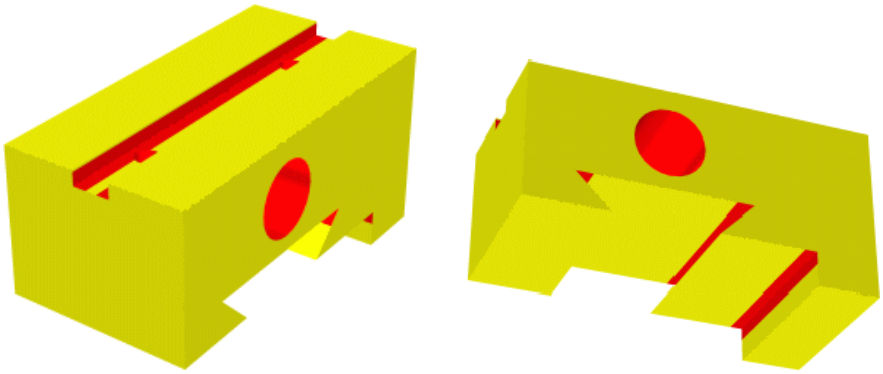
Finding opposite faces can, in general, be used as a heuristic to find possible grips. Another heuristic can be to first evaluate the finger domains available on the largest design feature in the model.

9.4 Results

In this section, some results of the FAFDD are presented.

First, the differences between using only the EFS method and the FAFDD are presented. For this, we use the feature model given in Figure 5.7 on page 51. We want to find every area on the model where a relatively small finger can be positioned. Both for the EFS method and, when needed for, FAFDD we made use of an envelope volume of both height and width of $0.5d$. Although this can generate incorrect finger domains as has been described in Subsection 9.2.2, an implemented algorithm for this method was already available, instead of an algorithm for the enhanced EFS method. Regardless which method is used, the finger domains resulting from using feature information and from using some EFS method can easily be compared. Using the enhanced EFS method instead of the original EFS method, will be more in benefit for the FAFDD method, comparing the calculation times, because the enhanced EFS method takes consider-

ably more time than the original EFS method. In Figure 9.12, a top and a bottom view are given, showing the found finger domains using the original EFS method. Dark areas on a model represent areas *not* in the finger domain, i.e. the non-free regions, light areas represent the finger domains. The model in Figure 9.12 consists of 21 faces, and the computation time was approximately 24 seconds on a HP9000/715-80 system. Figure 9.13 on the next page shows the results using FAFDD. The model consists of 9 features, and the computation time was approximately 6 seconds. No EFS calculation is needed in this model to reduce the local finger domains, because there are no additive features in this model.



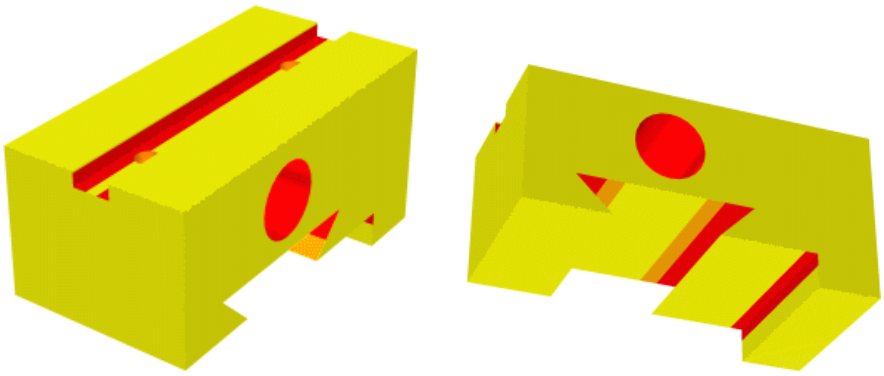
(a) top view (b) bottom view

Figure 9.12: Finger domains for a small finger found with the EFS method only

The different shadings in the dove-tail slot feature in figure 9.13 represent finger domains with different rankings. This can only be found with FAFDD using the feature information, therefore it is not shown in Figure 9.12.

Another difference can be seen in the length of the non-free regions in the dove-tail slot. The non-free regions resulting from using the original EFS method are smaller than those resulting from FAFDD. This is because of the fact that the EFS method does not take the angle between faces into account, which is incorrect. Notice, however, that the results of using the enhanced EFS method should be the same, with respect to the length of the non-free regions, as those of the FAFDD.

In Figure 9.14 on page 117, the finger domains are shown for fingers



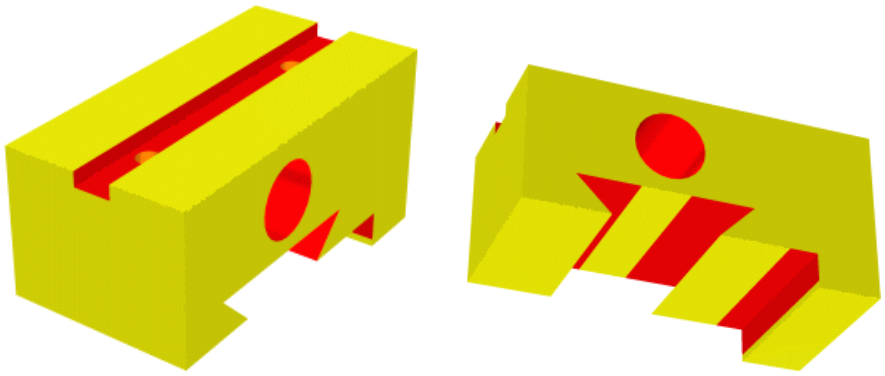
(a) top view (b) bottom view

Figure 9.13: Finger domains for a small finger determined using FAFDD

with a larger diameter using FAFDD. The finger width is larger than the width of the slot, and therefore no areas exist in the slot where the finger can be positioned. Further, the depth of the dove-tail slot is smaller than the finger width, so no areas are found with different rankings in the dove-tail slot. The computation time, using FAFDD, is the same as for the smaller finger: the same number of features has to be consulted to get the finger domains for the complete model.

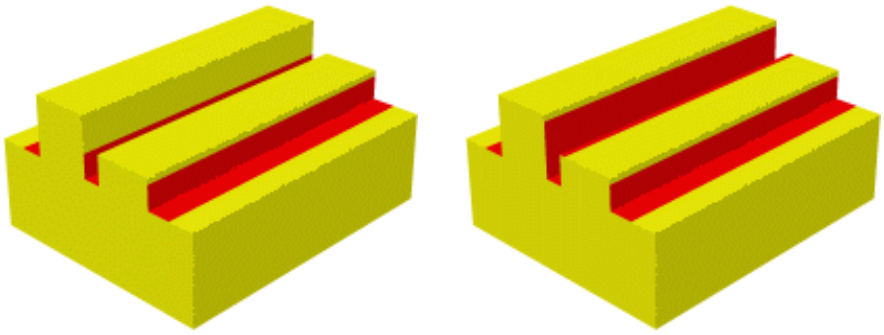
In Figure 9.15 on the next page, the differences between local and global finger domains for the same model are shown. The local finger domains are found by using FAFDD without additional EFS calculations. The global finger domains are found by executing additional EFS computations, but only on the found local finger domains of additive features. The local finger domains for this model with 3 features are found in approximately 5 seconds. The additional EFS calculations took approximately 15 seconds, resulting in a total computation time of approximately 20 seconds.

In Figure 9.16 on page 118, some other examples are given. On the left side, the finger domains are shown resulting from using the original EFS method, and on the right side those resulting from FAFDD. Some models on the right side also show different rankings, e.g. chamfers are given a lower ranking, because there is less chance of finding an opposite face for the actual grip. For every model, the number of faces and features is given, and the approximate computation time to generate the finger domains.



(a) top view (b) bottom view

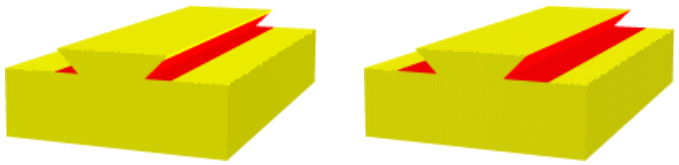
Figure 9.14: Finger domains for a larger finger determined using FAFDD



(a) local finger domains, 5 sec (b) global finger domains, 20 sec

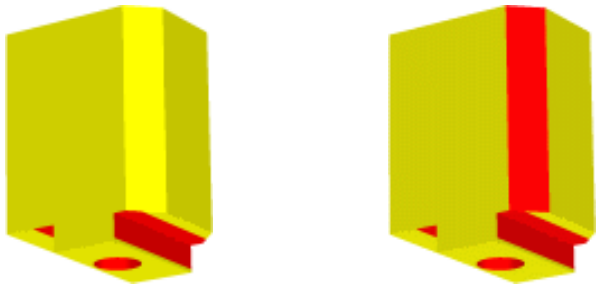
Figure 9.15: Local and global finger domains when features have influence on each other

In these results, curved surfaces have not been investigated with the EFS method. When this would have been done, these surfaces would have been faceted, generating many small faces. For each of these faces, an EFS calculation would have to be performed, which would considerably increase the computation time. Using FAFDD, the features themselves know whether it is possible for a finger to grasp on a curved surface, without

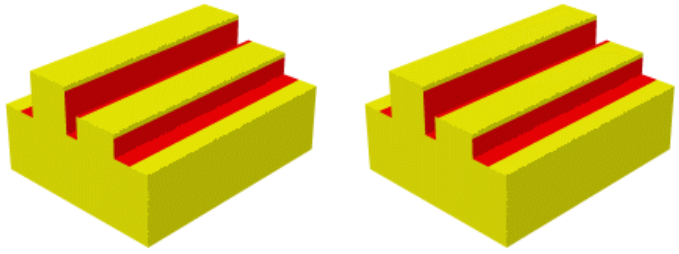


(a) 10 faces, 15 sec (b) 2 features, 5 sec

(c) 15 faces, 15 sec (d) 7 features, 3 sec



(e) 13 faces, 14 sec (f) 5 features, 10 sec



(g) 14 faces, 22 sec (h) 3 features, 20 sec

Figure 9.16: Some examples of global finger domains using the EFS method (left side) and using FAFDD (right side)

increasing the computation time.

The results of using both handling and connection features are taken together in one figure. The handling feature gives information on contact areas involved in feeding and fixturing. Connection features give contact areas and assembly directions involved in the partial assembly. For the reduction of the finger domains, it makes no difference whether the contact area is involved in a fixture or the partial assembly. In Figure 9.17, an example is given of a component, the small-block, which contains a rib-slot connection with another component, the base-block (see Figure 5.10 on page 54 for a feature model of the subassembly containing these components). The non-free regions found because of the contact areas are shown in the figure.

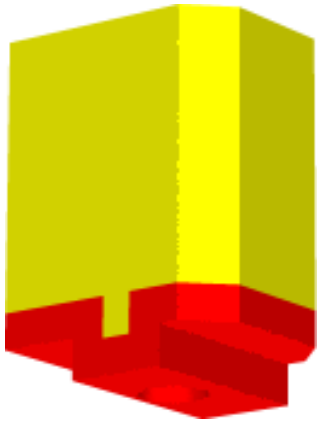


Figure 9.17: Non-free regions found on a component because of the contact areas known by the involved connection features

These results show that using assembly information available in a model in its features, in general accelerates the determination of the finger domains. Another advantage is the benefit of the ranked finger domains, which can be used as heuristics in finding the actual grip.

Chapter 10

Other planning modules

As was already mentioned in Chapter 3, assembly sequence planning is a special kind of planning module, in the sense that it is highly dependent on other planning modules, e.g. stability analysis, grip planning and motion planning (see Figure 3.1 on page 26).

In this chapter, the focus will be on motion planning and assembly sequence planning, planning modules where features, in particular assembly features, can be useful. This chapter consists mainly of concepts of how these features can be used, and not of descriptions of implementations to validate these concepts. Parts of this chapter have already been published in van Holland and Bronsvoort (1997).

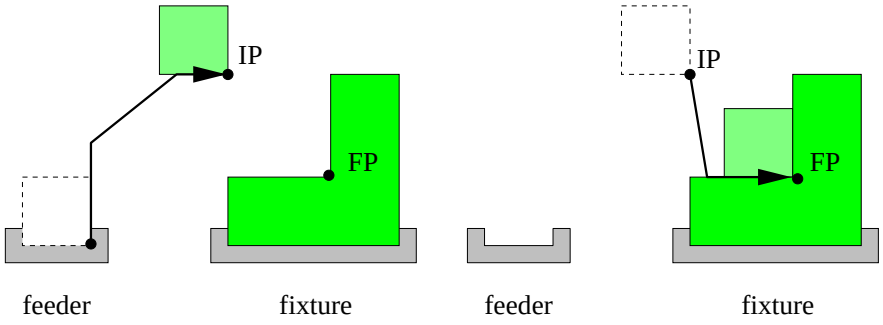
10.1 Motion planning

Motion planning is separated into two steps: gross motion and fine motion planning (Baartman 1995).

In gross motion planning, a collision free path is searched for a specific component, towards the partial assembly. In fine motion planning, the last phase of the assembly is taken into account, where contacts between components by definition *cannot* be avoided, see Figure 10.1 on the following page.

10.1.1 Gross motion planning

Gross motion planners determine a path for a component from its feeding position to its insertion position, a position near the final assembled position where there is *no* contact yet between component and partial assembly. To determine this motion path, the geometry of component and



(a) in gross motion planning, the component is moved from an initial position towards the insertion position (IP), by avoiding any collisions

(b) in fine motion planning, the component is moved from the insertion position to the final position (FP) by using contacts to reduce positioning uncertainties

Figure 10.1: Motion planning is divided into gross and fine motion planning. Notice that for clearness the distance between IP and FP has been somewhat enlarged in this example

partial assembly are taken, and an obstacle-free path is computed in 3D space.

Within gross motion planning, there is no real benefit of using assembly features, because there is no contact yet between component and partial assembly. However, for determining the insertion point, the position where gross motion ends and fine motion starts, the assembly features can be useful.

10.1.2 Fine motion planning

Fine motion planning is used to find a path from the insertion position to the final position. The contact areas between component and partial assembly are now used to “lead” the component to its desired final position.

Assembly features, and especially connection features, can be profitably used in two ways by providing:

- possible motion directions
- possible fine motion strategies

The internal freedom of motion stored within connection features, can be used to retrieve the possible motion directions. How this can be done, has already been described in Chapter 8 in the context of translational stability.

Specific connections, e.g. (threaded) pen-hole and plane-mate, often have their own specific fine motion strategies. To select the right strategy in case no connections features are used, the type of connection must be retrieved from the geometry, which is sometimes very hard, or even impossible.

When using the contact information stored within connection features, the type of contact is already available. Another advantage of using the connection features is that generic fine motion strategies can already be stored within the generic connection features. When there is a need for a specific fine motion, the specific instance of the connection feature can provide the generic fine motion strategy with parameters, resulting in an appropriate fine motion to be used.

10.2 Assembly sequence planning

There exists many publications about assembly sequence planning, and it is not within the scope of this thesis to describe all possible techniques. A brief overview will be given in this section, which will focus on the current lacks. For further reading, see Gottschlich et al. (1994) for a recent overview. Important references are De Fazio and Whitney (1987), Wolter (1988), Homem de Mello and Sanderson (1991*a*), Baldwin et al. (1991), Lozano-Pérez and Wilson (1993) and Chakrabarty and Wolter (1994).

The whole process of assembly sequence planning is complex, and can be subdivided into three main steps, of which sometimes the first two steps are taken together:

1. generate precedence relations between the components of a product,
2. generate all feasible assembly sequences,
3. find the optimal assembly sequence from the feasible sequences.

Often *assembly sequence planning* is reduced to only checking a precedence relation graph, to find the optimal assembly sequence. How this precedence relation graph is created, is, however, seldom described. But in reality the creation of the precedence graph and the search for an optimal sequence both have to be taken into account.

These three steps are now described in more detail in the following subsections.

10.2.1 Generate precedence relations

To find *precedence relations*, information relevant to both manufacturing and assembly is needed. This information is mostly not, or only to a limited extent, present in current product models, as was already mentioned in Subsection 2.2.3.

Sometimes these precedence relations are gathered by interrogating a human assembly planner, but this is extremely difficult for larger models, because of the large number of precedence relations. Therefore, mostly computer tools are used. These tools take as input a geometry description of the product, with the available relations between components. The — mostly low-level — relations are restricted to whether components mate with each other, and whether these matings are fixed or not, see also Figure 2.3 on page 17.

In the previous chapters on stability analysis (Chapter 8) and grasp planning (Chapter 9) and in the previous section on motion planning (Section 10.1), it has already been described how features in general, and assembly features in particular, can be used to determine certain specific precedence relations. Notice that in assembly sequence planning, these modules are only used to find precedence relations, and not to find all information needed for the assembly process. For example, in grasp planning it is sufficient to determine that a grip can be found, whereas later the exact grip can be determined. The latter will take more computation time.

Using features for generating precedence relations does not change the basic ideas on how to generate them, but it differs in the way assembly-specific information is retrieved. Feature models can significantly decrease the complexity and the time used to retrieve this information.

10.2.2 Generate feasible assembly sequences

Assembly sequence planning is often reduced to find the optimal — or semi-optimal sequence, because *the* optimal can seldom be found in reality, because of all kinds of competing criteria (De Fazio and Whitney 1987) — assembly sequence selected from all *feasible assembly sequences*. Feasible assembly sequences are those sequences that can result in the complete product.

The precedence relations are thus used to generate feasible sequences. The feasible sequences can be represented by an *AND/OR graph* (Homem de Mello and Sanderson 1991*b*) or an Assembly State Transition Diagram, called *ASTD* (Waarts et al. 1992). Mostly the AND / OR graph or ASTD is built simultaneously with investigating the precedence relations.

This can be done by first searching for feasible *dis*assembly sequences for the product, and then reversing them to get the feasible assembly sequences. Every component in the product is investigated in turn to check whether it is possible to disassemble it from the product or not. If the component can be disassembled, it is removed from the product, and the procedure is repeated on the components left, until all components have been taken out. Finding disassembly sequences is less combinatorial than finding assembly sequences, because every step in disassembling a product leads to a smaller product that always can be disassembled, whereas every step in assembling a product leads to a bigger product with increasing chances on constraining precedence relations for components still to assemble. The latter can result in checking many infeasible sequences.

Figure 2.3(a) on page 17 showed an example product, created by Sanderson et al. (1990), with its corresponding relation graph, see Figure 2.3(b). The resulting AND/OR graph for this product is shown in Figure 10.2, representing all feasible sequences.

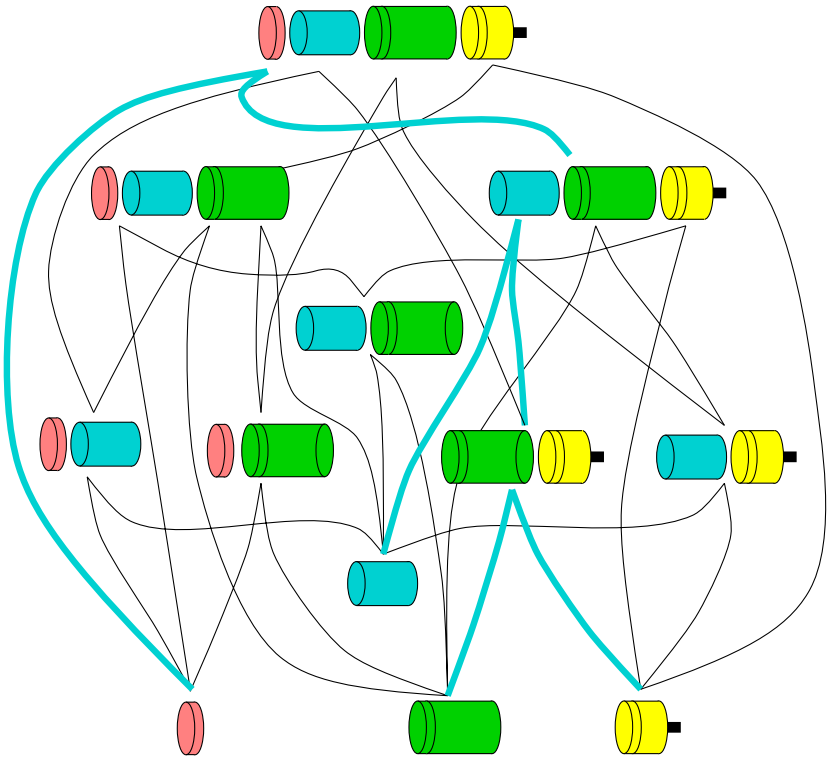


Figure 10.2: AND/OR graph representing the feasible sequences for a simple product

10.2.3 Retrieve the optimal assembly sequence

The optimal sequence is the sequence with the optimum for some kind of cost function, taking into account, for example, total assembly time, used resources, or combinations of both.

An example of such an optimal sequence is shown in Figure 10.2 on the preceding page; it is represented by the thick lines. This optimal sequence can be generated using some Branch-and-Bound method (Nilsson 1971, Barr and Feigenbaum 1981). Because of the combinatorial explosion, there are many sequences to investigate, and heuristic search methods are used for faster retrieving the optimal sequence.

Because of the need for specific resources during flexible assembly — like grippers, feeders and fixtures — it is not very useful to determine the optimal sequence off-line, and to store it together with the product model. During actual assembly, the availability of the resources can be different in time. So within a sequence of the same batch, for every batch the availability of resources can differ, and even for the same products assembled within a batch. One reason for this is that resources can also be used elsewhere by other flexible assembly cells. But another important reason is the hardly unpredictable time a resource is actually used during assembly, because of all kinds of troubles that may arise during assembly. It is therefore better to calculate the optimal sequence on-line, just before the actual assembly, so that the available resources can be taken into account, and uncertainties are eliminated as much as possible (van Holland et al. 1992).

10.2.4 Additional profits of using features for sequence planning

Usually, the selection of the next candidate to be evaluated for disassembly is done by a time-consuming trial-and-error method. One by one, every component in the partial assembly left is chosen, and is evaluated to see whether it can be disassembled.

A simple heuristic for building the search space is the following: every time you have to choose the next component to disassemble, you choose the one with the minimal number of connections between component and the remaining partial assembly. This heuristic is based on the fact that the component with the fewest connections with its partial assembly does also have the greatest chance for a successful disassembly. This heuristic can easily be used together with the feature-based product models, because every component knows exactly its involved connections.

However, more sophisticated knowledge can be used, because some features available in feature models contain already information about possible assembly sequences. By using connection feature information, the combinatorial explosion can be reduced, i.e. a reduction in search space can be realized. Connection features can often give information on the precedences of the components involved in the connection, resulting in a priority for the selection of the next component to evaluate.

Connection features with agents, can quickly give the component to select first, because they "know" something about the disassembly sequence. It is useless to try to disassemble a plate when it is still connected by a bolt, so the connection feature will present the bolt first, and after that the plate connected by the bolt.

Take, for example, the simple product shown in Figure 10.3. This example is used to show the effects in search space reduction when assembly features are used. First, in Figure 10.4(a) on the following page, the search space is shown when no feature information is used.

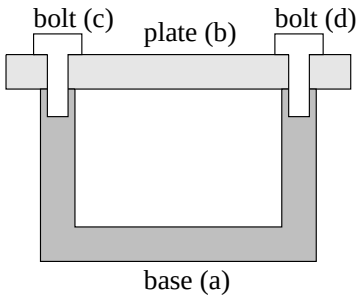
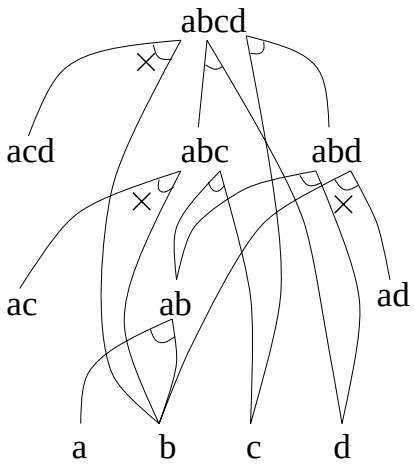


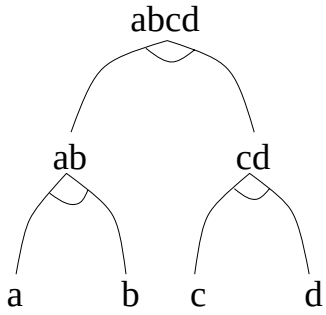
Figure 10.3: Simple product to show the effect in search space reduction by using assembly features

Between the base and the plate there exists a specific attachment, say a pattern of threaded connections. This specific attachment contains two bolts as the component agents. The attachment contains two attachments as connection agents. These attachments are the threaded connections, with each of them containing one bolt as the component agent and a threaded pen-hole and a *normal* pen-hole connection as connection agents, see Figure 10.5 on page 129.

A connection feature representing a pattern can directly indicate that it does not matter in which order to disassemble the components, e.g. that it does not matter in which order a pattern of bolts is assembled. In the example above, this means that the search space is reduced because it does



(a) without feature information



(b) with feature information

Figure 10.4: Effect on search space reduction of using feature information. Notice that the AND relations marked with a cross represent search directions initially followed but giving a negative result, because the represented (dis)assembly operation cannot be performed

not matter whether first bolt c is assembled, or bolt d.

The attachments, the threaded connections, further reduce the search

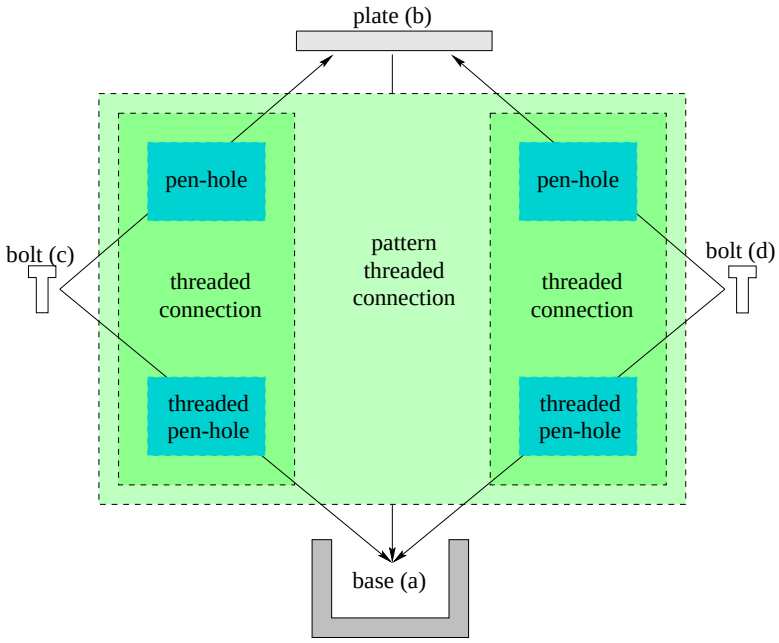


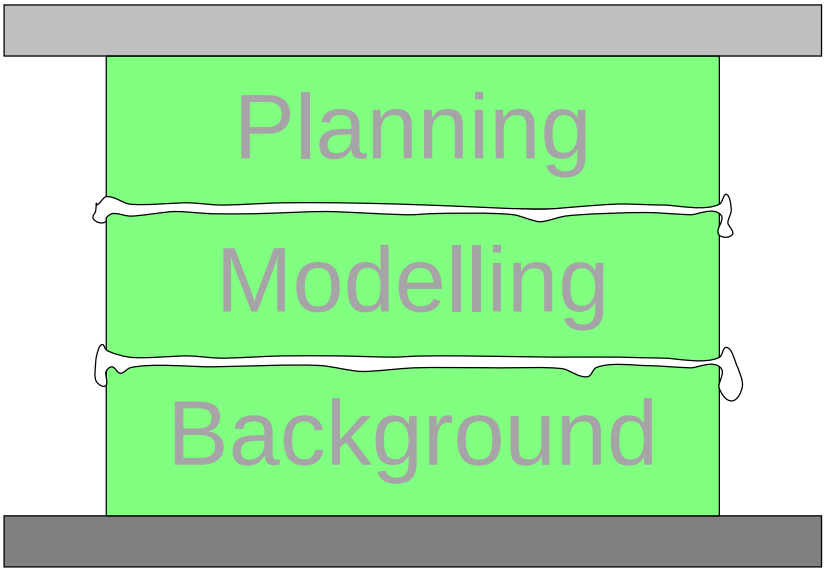
Figure 10.5: Connection features belonging to the simple product

space, because these attachments directly indicate the required assembly order for base, plate and bolt: first the base, then the plate, and finally the bolt. There is no need to search for alternative solutions here. Figure 10.4(b) on the facing page shows the search space when feature information is taken into account.

Thus, by using feature information, the AND/OR graph can shrink significantly, resulting in a smaller number of precedences that have to be checked. This will result in much lower computation times for the assembly sequence planner.

Part IV

Concluding remarks



Part IV , is the final part containing the conclusions and future work in Chapter 11.

Chapter 11

Conclusions and future research directions

As was stated in the introduction, the focus of this thesis has been on the automation and integration of modelling and planning for assembly. At the end of this thesis, some conclusions are drawn, and some topics for future research directions are indicated.

11.1 Conclusions

Just as features proved to be a good concept in manufacturing, they are also useful in assembly. Features can solve the lack of information in modelling and planning, and can provide a proper tool to integrate the two.

Modelling

In assembly modelling, assembly features provide the modeller with building blocks closer to the detailed geometrical design than the currently available prototype functional modellers, and, on the other hand, more on a conceptual level than the commonly used geometric modellers.

Assembly features

As assembly features are information carriers for assembly-specific information, more information already known during design can now be stored. Information can be stored once, and be used in many other phases. By using assembly features, it is much easier to store information needed during the complete product life cycle into one product model.

Generic and instance level assembly information

The needed modelling information is separated into information available on generic component level and information available on instance level. Information on generic component level is information that is related to the type of component, independent of the actual position and orientation of a component. The information on instance level, on the other hand, is dependent on the actual position and orientation of an *instance* of the component and the product to which it belongs.

Handling and connection features

The information about how to handle a component is stored in a handling feature. Most of this information, e.g. fixture, feeder, gripper and possible grip area information, are stored directly with the generic component.

The connection feature provides the possibilities to store connection-specific information related to instance components in the product model. This information is used both during modelling, to generate the complete model, and during analysis and planning, to enhance the planning modules.

Connection features make it possible to start with the relation-driven modelling concept. Within this concept, a model starts with raw geometry, and the relations between the geometric elements create the more detailed shapes. This is very useful in bridging the gap between functional and geometric modellers.

Integration of single-part and assembly model

The concept of combining the single-part feature model and the assembly feature model results in a very powerful integrated object-oriented product model. Modelling of single parts and of assemblies are now both based on the same structure, providing a general way of working with them together. Therefore an important step to integrate single-part and assembly modelling into one modelling system has been made.

Planning

Analysis steps within modelling can sometimes hardly be separated from the actual planning modules during assembly planning. Therefore, it is logical that in planning modules the already available assembly informa-

tion stored in the assembly features is used, and new information is stored using these features.

Integrated planning modules

Feature-based product models for assembly provide handles to integrate the different models used in all kinds of planning modules. Assembly features are very useful, or even required, using DFX and concurrent engineering.

Stability analysis

Within stability analysis, the use of assembly features shows that translational stability can now be analyzed easier using the available freedom of motion within connection features than without these features. Using visibility map projection onto the unit cube, all possible motion directions in the complete 3D space are represented, which avoids the problems of ambiguity and numerical degeneracy occurring in the projection methods using one or two planes.

In rotational stability analysis, the connection features are also very helpful, by providing already available information on possible rotational axes. This will decrease the number of possible rotational axes to analyse.

Grip planning

Within grip planning, all features available in the integrated product model are useful in accelerating the module. The FAFDD method provides ranked finger domain areas, useful for faster generation of the actual grip. These ranked areas, which cannot be created using the EFS method, are for many models generated very fast compared to the EFS method. Only models containing many additive form features influencing subtractive features, do not result in great differences between the two methods in computation times.

Motion planning

Fine motion planning makes use of the connection features to retrieve the possible fine motion strategies. This avoids difficult low-level geometric reasoning calculations, retrieving the connection type and the matching motion strategies belonging to it.

Assembly sequence planning

Assembly features are profitably used in the described assembly planning modules, and almost all these modules are used for generating the precedence relations needed in assembly sequence planning. Additionally, because of the integration of the product models used in all planning modules, it is now also possible to retrieve all the needed information from one model and to determine many of the precedence relations automatically. This is absolutely needed for assembly sequence planning, which is highly dependent on all kinds of other planning modules.

The connection features reduce the complexity of finding feasible sequences, because they contain information about possible sequences. Attachment connections specify in which order the connected components together with the involved agents can be assembled. Also pattern connections are of great help, knowing that no specific order is needed assembling a pattern of components.

Using features for generating precedence relations does not change the basic ideas on how to generate them, but it differs in the way assembly-specific information is retrieved or determined. Feature models can significantly decrease the complexity of and the time used for this. A consequence might be that the designer or engineer needs more modelling time, because he must add more assembly-specific information to the model. However, this is not that bad, because modelling will then better fit to his way of thinking, and assembly features combine large amounts of information that otherwise should be added step by step.

Assembly features in modelling and planning

In this way, the feature concept does not only integrate single-part and assembly modelling, and several planning modules, but also modelling and planning. The assembly planning modules can now make use of one integrated product model from which they can retrieve information on very detailed single-part level and on higher assembly level. Together with the benefits of using earlier generated information by other modules, the planning modules can produce their planning results faster. These planning results can, in addition, contain more accurate information, because more information was already available in the product model, which will generally increase the quality of the planning results.

All these topics have been described to make a step forward in the direction of a modelling and planning environment of the future. Several of the short term goals have been reached, especially in the integration

of a single-part and assembly model, and in the integration of modelling and planning. However, there remain many topics for future research; see Section 11.2 on the next page.

Implementation

The integration of single-part and assembly model by using the same object-oriented product model showed its benefit in the implemented prototype. With this product model both single-part and assembly models were created and manipulated in the same way.

The integration of modelling and planning in the prototype made it easy to provide direct DFA feedback during modelling.

Development environment

After so many years of development, some remarks can be made about the development environment used. The use of object-orientation has proven its benefits. However, although ACIS™ describes itself as being an object-oriented geometric library, it behaves itself as a *non* object-oriented geometric library with some object-oriented shell around it. This means in fact that the advantages of object orientation, such as combining attributes and operations in one model, and using inheritance to derive functionality can hardly be used. Another difficulty in using ACIS™ is the lack of proper documentation to explain the underlying concepts. On the other hand, commonly used concepts expected within a geometric library are provided by the ACIS™ library, and were very helpful in building geometric models.

Although in theory there is a lot of functionality available in C++, this does not mean that everything can (already) be done in practice. Maybe because of using the newest functionality *provided* by C++, there were many problems with available compilers and debuggers not capable of handling this promised functionality.

The new evolutions in, and the possibilities of, the object-oriented JAVA™ language provide many elements needed in computing science research. JAVA™ is platform-independent and provides all kinds of language-included possibilities concerning networking, multi-threading, user-interface building, etc. Besides that, the language is interpreted, and therefore provides a very useful language for a research environment building mainly prototype software.

11.2 Future research directions

In fact the future research directions have already been described in Chapter 4, but more can be said about this.

Remember first the key topics in the long term planning:

- top-down and functional modelling,
- integrated single-part and assembly modelling, and
- integration of modelling and planning.

Top-down conceptual modelling

The focus of this thesis has been relatively more on integration of single-part and assembly modelling and on integration of modelling and planning, than on the topic of functional or conceptual design. Although some first steps have been made in the direction of a top-down modelling environment, much more research must be done, especially research that tries to link the conceptual level with the detailed geometry level and vice versa. One research area in this field, still unresolved, covers the consistency checking of conceptual and detailed level. A model that is created top-down, and later changed on geometry level, should remain consistent with the higher conceptual level.

Integrated single-part and assembly modelling

In this thesis, the multiple-view concept in a single-part feature model has not been taken into account. In fact the described form features were without specific disciplinary information. Instead, they were chosen in such a way that the used connection features could use them. In reality, a feature model will have multiple views, and research is needed on the required form features for assembly modelling. Feature conversion has to be performed to derive these features, which is also still a research area with many challenges.

A special kind of feature conversion is needed to solve problems in assembly modelling not only related to form features, but also to combining components. The feature conversion should not only take place within one component, but within the complete product. By combining several components, it is possible that the partial assembly contains a shape needed in a connection. Take, for example, two components with both a step feature,

as is shown in Figure 11.1 on the facing page. Connecting the components can result in a shape comparable with a slot feature. When another component is connected using a rib-slot connection, the connection feature cannot be associated with the slot shape, because it does not yet know that two step features can result in one slot feature. Using automatic conversion of features on product level will solve this problem.

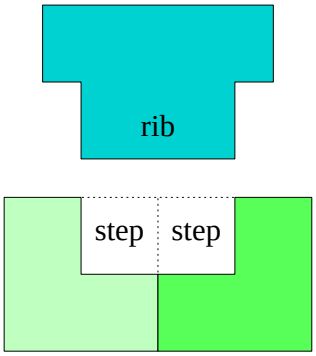


Figure 11.1: Feature conversion is needed on product level, to convert two steps into one slot feature

Integration of modelling and planning

This thesis provides a framework for storing, and using, assembly information into, and from, an integrated product model. However, research on the information itself must be extended. This research must be in the direction of searching all kinds of assembly connections with all kinds of characteristics. Questions that still remain are: which connections can be properly used in flexible assembly and when do you select them? These questions must be solved in the first place by mechanical engineers, but not alone. Computing scientists must be highly involved in this research, because only the combination of both disciplines will bring modelling and planning closer to a fully integrated environment. One provides the needed information, the other provides techniques to store and retrieve them. Both will think about practical ways to use this information.

Spin-off

As this research has been a spin-off project from the DIAC project, the project itself is not finished. We are another step forward to the “complete”

automation of the assembly process, in which products are automatically assembled from assembly plans, assembly plans are automatically generated from models, and models are automatically modified on the basis of (executed) assembly plans. For the additional steps to reach this final goal, this project itself will have to be followed by its own, multi-disciplinary spin-off projects.

Bibliography

Ambler, A. P. and Popplestone, R. J. (1975), 'Inferring the positions of bodies from specified spatial relationships', *Artificial Intelligence* **6**, 157–174.

Ames, A. L., Calton, T. L., Jones, R. E., Kaufman, S. G., Laguna, C. A. and Wilson, R. H. (1995), Lessons learned from a second generation assembly system, *in* 'Proceedings of the IEEE International Symposium on Assembly and Task Planning', Pittsburgh, Pennsylvania, pp. 41–47.

Andreasen, M. M. (1992), The theory of domains, *in* 'Workshop on Understanding Function and Function-to-Form Evolution', Cambridge University, pp. 1–10.

Andreasen, M. M., Kähler, S. and Lund, T. (1988), *Design for Assembly*, IFS (Publications) Ltd. & Springer Verlag, New York, USA.

Baartman, J. P. (1995), Automation of assembly operations on parts, PhD thesis, Delft University of Technology, The Netherlands.
URL: http://www.wbmt.tudelft.nl/pt/online/Dissertation_Baartman/dissert.bk.html

Baldwin, D. F., Abell, T. E., Lui, M. M., De Fazio, T. L. and Whitney, D. E. (1991), 'An integrated computer aid for generating and evaluating assembly sequences for mechanical products', *IEEE Transactions on Robotics and Automation* **7**(1), 78–94.

Barr, A. and Feigenbaum, E. A. (1981), *The Handbook of Artificial Intelligence*, Vol. 1, Stanford Heuristic Press and William Kaufmann, Inc, Los Altos, California, USA.

Boneschanscher, N. (1993), Plan generation for flexible assembly systems, PhD thesis, Delft University of Technology, The Netherlands.

Boothroyd, G. (1987), 'Design for assembly - the key to design for manufacture', *The International Journal of Advanced Manufacturing Technology* **2**(3), 3–11.

Boothroyd, G., Dewhurst, P. and Knight, W. (1994), *Product Design for Manufacture and Assembly*, Marcel Dekker Inc., New York, USA.

Bronsvoort, W. F., Bidarra, R., Dohmen, M., van Holland, W. and de Kraker, K. J. (1996), Feature modelling for concurrent engineering, *in* 'Proceedings of the Symposium on Tools and Methods for Concurrent Engineering', Technical University of Budapest, Hungary, pp. 46–55.
URL: ftp://ftp.twi.tudelft.nl/TWI/publications/misc/Bronsvoort.tmce96.ps.gz

Bronsvoort, W. F. and Jansen, F. W. (1993), 'Feature modelling and conversion - Key concepts to concurrent engineering', *Computers in Industry* **21**(1), 61–86.

Bronsvoort, W. F., Jansen, F. W. and Post, F. H. (1991), Design and display of solid models, *in* G. Garcia and I. Herman, eds, 'Advances in Computer Graphics VI', Springer-Verlag, Berlin, pp. 1–57.

van Bruggen, M., Baartman, J. P. and Bronsvoort, W. F. (1993), Grips on parts, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Vol. 2, Atlanta, Georgia, USA, pp. 828–833.

Chakrabarty, S. and Wolter, J. (1994), A hierarchical approach to assembly planning, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Vol. 1, San Diego, California, USA, pp. 258–263.

Chen, L.-L., Chou, S.-Y. and Woo, T. C. (1993*a*), 'Parting directions for mould and die design', *Computer-Aided Design* **25**(12), 762–768.

Chen, L.-L., Chou, S.-Y. and Woo, T. C. (1993*b*), 'Separating and intersecting spherical polygons: Computing machinability on three-, four-, and five-axis numerically controlled machines', *ACM Transactions on Graphics* **12**(4), 305–326.

Cutkosky, M. R., Tenenbaum, J. M. and Brown, D. R. (1992), 'Working with multiple representations in a concurrent design system', *Journal of Mechanical Design* **114**, 515–524.

Cutkosky, M. R. and Wright, P. K. (1986), 'Friction, stability and the design of robotic fingers', *The International Journal of Robotics Research* **5**(4), 20–37.

De Fazio, T. L. (1990), A prototype of feature-based design for assembly, *in* B. Ravani, ed., 'ASME Advances in Design Automation 1990', Chicago, Illinois, USA, pp. 9–16.

De Fazio, T. L., Edsall, A. C., Gustavson, R. E., Hernandez, J., Hutchins, P. M., Leung, H.-W., Luby, S. C., Metzinger, R. W., Nevins, J. L., Tung, K. and Whitney, D. E. (1993), 'A prototype of feature-based design for assembly', *Journal of Mechanical Design* **115**, 723–734.

De Fazio, T. L. and Whitney, D. E. (1987), 'Simplified generation of all mechanical assembly sequences', *IEEE Journal of Robotics and Automation* **RA-3**(6), 640–658.

Delchambre, A. (1992), *Computer-aided Assembly Planning*, Chapman & Hall, London, UK. ISBN 0 412 43170 X.

Dohmen, M. (1995), 'A survey of constraint satisfaction techniques for geometric modeling', *Computers & Graphics* **19**(6), 831–845.

Dohmen, M., de Kraker, K. J. and Bronsvoort, W. F. (1996), Feature validation in a multiple-view modeling system, *in* J. M. McCarthy, ed., 'Proceedings of the ASME 1996 Design Engineering Technical Conference and Computer in Engineering Conference', Irvine, California, USA. Published on cdrom, ISBN 0-7918-1232-4.
URL: ftp://ftp.twi.tudelft.nl/TWI/publications/misc/Dohmen.asme96.ps.gz

Gorlen, K. E., Orlow, S. M. and Plexico, P. S. (1991), *Data Abstraction and Object-Oriented Programming in C++*, John Wiley & Sons Ltd, Chichester, England. ISBN 0-471-92346-X.

Gottschlich, S., Ramos, C. and Lyons, D. (1994), 'Assembly and task planning: a taxonomy', *IEEE Robotics and Automation Magazine* **1**(3), 4–12.

Gui, J. K. (1993), Methodology for Modelling Complete Product Assemblies, PhD thesis, Helsinki University of Technology, Finland.

Heemskerk, C. J. M. (1990), A Concept for Computer Aided Process Planning for Flexible Assembly, PhD thesis, Delft University of Technology, The Netherlands.

Henderson, M. R. and Taylor, L. E. (1993), 'A meta-model for mechanical products based upon the mechanical design process', *Research in Engineering Design* **5**(3 & 4), 140–160.

van Holland, W., Boneschanscher, N. and Bronsvoort, W. F. (1992), Task assignment in a flexible assembly cell using and/or graphs, *in* L. Basañez, ed., 'International Symposium on Industrial Robots', Barcelona, Spain, pp. 653–658 + 642.

van Holland, W. and Bronsvoort, W. F. (1995*a*), Assembly features and visibility maps, *in* A. A. Busnaina, ed., 'Proceedings of the 15th ASME International Computers in Engineering Conference', Boston, Massachusetts, USA, pp. 691–697.
URL: ftp://ftp.twi.tudelft.nl/TWI/publications/misc/vanHolland.asme95.ps.gz

van Holland, W. and Bronsvoort, W. F. (1995*b*), Assembly features in modelling and planning, *in* M. Tichem, T. Storm, M. M. Andreasen and K. J. MacCallum, eds, 'Proceedings of the WDK Workshop on Product Structuring', Delft University of Technology, Delft, pp. 195–206.
URL: ftp://ftp.twi.tudelft.nl/TWI/publications/misc/vanHolland.wdk95.ps.gz

van Holland, W. and Bronsvoort, W. F. (1996*a*), Extracting grip areas from feature information, *in* J. M. McCarthy, ed., 'Proceedings of the ASME 1996 Design Engineering Technical Conference and Computer in Engineering Conference', Irvine, California, USA. Published on cdrom, ISBN 0-7918-1232-4.
URL: ftp://ftp.twi.tudelft.nl/TWI/publications/misc/vanHolland.asme96.ps.gz

van Holland, W. and Bronsvoort, W. F. (1996*b*), An object-oriented product structure for assembly modelling, *in* M. Tichem, T. Storm, M. M. Andreasen and K. J. MacCallum, eds, 'Proceedings of the 2^{nd} WDK Workshop on Product Structuring', Delft, The Netherlands, pp. 151–157.
URL: ftp://ftp.twi.tudelft.nl/TWI/publications/misc/vanHolland.wdk96.ps.gz

van Holland, W. and Bronsvoort, W. F. (1997), Assembly features and sequence planning, *in* M. Pratt, R. D. Sriram and M. J. Wozny, eds,

'Product Modeling for Computer Integrated Design and Manufacture', Chapman & Hall, London, UK, pp. 275–284.
URL: ftp://ftp.twi.tudelft.nl/TWI/publications/misc/vanHolland.ifip96.ps.gz

van Holland, W., Bronsvoort, W. F. and Jansen, F. W. (1995), Feature modelling for assembly, *in* W. Straßer and F. M. Wahl, eds, 'Graphics and Robotics', Springer-Verlag, Berlin, pp. 131–148. ISBN 3-540-58358-0.

Homem de Mello, L. S. and Sanderson, A. C. (1989), A correct and complete algorithm for the generation of mechanical assembly sequences, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', pp. 56–61.

Homem de Mello, L. S. and Sanderson, A. C. (1991*a*), A basic algorithm for the generation of mechanical assembly sequences, *in* L. S. Homem de Mello and S. Lee, eds, 'Computer-Aided Mechanical Assembly Planning', Kluwer Academic Publishers, The Netherlands, pp. 163–190.

Homem de Mello, L. S. and Sanderson, A. C. (1991*b*), Representations for assembly sequences, *in* L. S. Homem de Mello and S. Lee, eds, 'Computer-Aided Mechanical Assembly Planning', Kluwer Academic Publishers, The Netherlands, pp. 129–162.

Horváth, I. (1996), A workbench architecture for object oriented handling of features, *in* J. M. McCarthy, ed., 'Proceedings of the ASME 1996 Design Engineering Technical Conference and Computer in Engineering Conference', Irvine, California, USA. Published on cdrom, ISBN 0-7918-1232-4.

Horváth, I., Thernesz, V. and Bagoly, Z. (1995), Conceptual design with functionally and morphologically parametrized feature objects, *in* A. A. Busnaina, ed., 'Proceedings of the 15th ASME International Computers in Engineering Conference', Boston, Massachusetts, USA, pp. 507–516.

van Houten, F. J. A. M. (1991), PART: A Computer Aided Process Planning System, PhD thesis, University Twente, The Netherlands.

Jasperse, H. B. (1995), A Macro Process Planning System for Machining Operations, PhD thesis, Delft University of Technology, The Netherlands.

Joshi, S. and Chang, T. C. (1988), 'Graph-based heuristics for recognition of machined features from a 3d solid model', *Computer-Aided Design* **20**(2), 58–66.

Kaufman, S. G., Wilson, R. H., Jones, R. E., Calton, T. L. and Ames, A. L. (1996), The Archimedes 2 mechanical assembly planning system, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Minneapolis, Minnesota, USA.
URL: http://www.sandia.gov/2121/archimedes/papers/icra96-cons.ps.gz

Kim, Y. S. (1992), 'Recognition of form features using convex decomposition', *Computer-Aided Design* **24**(9), 461–476.

de Kraker, K. J., Dohmen, M. and Bronsvoort, W. F. (1995), Multiple-way feature conversion to support concurrent engineering, *in* C. Hoffmann and J. Rossignac, eds, 'Solid Modeling '95, Third Symposium on Solid Modeling and Applications', Salt Lake City, Utah, USA, pp. 105 – 114.

de Kraker, K. J., Dohmen, M. and Bronsvoort, W. F. (1997), Multiple-way feature conversion — opening a view, *in* M. Pratt, R. D. Sriram and M. J. Wozny, eds, 'Product Modeling for Computer Integrated Design and Manufacture', Chapman & Hall, London, UK, pp. 203–212.

Lee, K. and Andrews, G. (1985), 'Inference of positions of components in an assembly : Part 2', *Computer-Aided Design* **17**(1), 20–24.

Lee, K. and Gossard, D. C. (1985), 'A hierarchical data structure for representing assemblies : Part 1', *Computer-Aided Design* **17**(1), 15–19.

Lee, S. (1994), 'Subassembly identification and evaluation for assembly planning', *IEEE Transactions on Systems, Manufacturing and Cybernetics* **24**(3), 493–503.

Lee, S. and Yi, C. (1993), Subassembly stability and reorientation, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Vol. 2, Atlanta, Georgia, USA, pp. 521–526.

Libardi, E. C. J., Dixon, J. R. and Simmons, M. K. (1988), 'Computer environments for the design of mechanical assemblies: A research review', *Engineering with Computers* **3**, 121–136.

Lim, S. S., Lee, I. B. H., Lim, L. E. N. and Ngoi, B. K. A. (1995), 'Computer-aided concurrent design of product and assembly processes: a literature review', *Journal of Design and Manufacturing* **5**, 67–88.

Lozano-Pérez, T. and Wilson, R. H. (1993), Assembly sequencing for arbitrary motions, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Atlanta, Georgia, USA, pp. 527–532.

Mantripragada, R., Cunningham, T. W. and Whitney, D. E. (1997), Assembly oriented design: A new approach to designing assemblies, *in* M. Pratt, R. D. Sriram and M. J. Wozny, eds, 'Product Modeling for Computer Integrated Design and Manufacture', Chapman & Hall, London, UK, pp. 308–324.

Mäntylä, M. (1988), *An Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland, USA.

Mäntylä, M. (1991), WAYT: Towards a modeling environment for assembled products, *in* H. Yoshikawa, F. Arbab and T. Tomiyama, eds, 'Intelligent CAD', Elsevier Science Publishers B.V., Amsterdam, The Netherlands.

Martens, P. (1991), Cad/Cam for Assembly Planning, PhD thesis, Delft University of Technology, The Netherlands.

Mattikalli, R., Baraff, D. and Khosla, P. (1994), Finding all gravitationally stable orientations of assemblies, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Vol. 1, San Diego, California, USA, pp. 251–257.

Mattikalli, R. S. and Khosla, P. K. (1991), Analysis of restraints to translational and rotational motion from the geometry of contact, *in* A. Sharon, R. Behun, F. Prinz and L. Young, eds, 'Issues in Design Manufacture/Intergration, Winter annual meeting of ASME', Vol. 39, Atlanta, Georgia, USA, pp. 65–71.

Mattikalli, R. S. and Khosla, P. K. (1992), Motion constraints from contact geometry: Representation and analysis, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Nice, France, pp. 2178–2185.

Meijer, B. R. and Jonker, P. P. (1991), The architecture and philosophy of the DIAC (Delft Intelligent Assembly Cell), *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Sacramento, California, USA, pp. 2218–2223.

Mortensen, N. H. and Andreasen, M. M. (1996), Designing in an interplay with a product model — explained by design units, *in* 'Proceedings of the Symposium on Tools and Methods for Concurrent Engineering', Technical University of Budapest, Hungary, pp. 100–115.

Mortenson, M. E. (1985), *Geometric Modeling*, John Wiley, New York, USA.

Nevins, J. and Whitney, D. E., eds (1989), *Concurrent Design of Products and Processes*, McGraw-Hill, New York, USA.

Nguyen, V.-D. (1988), 'Constructing force-closure grips', *The International Journal of Robotics Research* **7**(3), 3–16.

Nilsson, N. (1971), *Problem solving methods in artificial intelligence*, McGraw-Hill, Inc., New York, USA.

Oliver, J. H. and Huang, H.-T. (1994), 'Automated path planning for integrated assembly design', *Computer-Aided Design* **26**(9), 658–666.

Ovtcharova, J., Pahl, G. and Rix, J. (1992), 'A proposal for feature classification in feature-based design', *Computers & Graphics* **16**(2), 187–195.

Pahl, G. and Beitz, W. (1988), *Engineering Design - a Systematic Approach*, Springer-Verlag, Berlin. Translated from *Konstruktionslehre*, Springer-Verlag, 1977.

Ponce, J., Sullivan, S., Boissonnat, J.-D. and Merlet, J.-P. (1993), On characterizing and computing three and four-finger force-closure grasps of polyhedral objects, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Vol. 2, Atlanta, Georgia, USA, pp. 821–827.

Popplestone, R. J., Ambler, A. P. and Bellos, I. M. (1980), 'An interpreter for a language for describing assemblies', *Artificial Intelligence* **14**, 79–107.

Pratt, M. J. (1993), 'Applications of feature recognition in the product lifecycle', *International Journal on Computer Integrated Manufacturing* **6**(1 & 2), 13–19.

Requicha, A. A. G. (1980), 'Representations for rigid solids: theory, methods and systems', *ACM Computing Surveys* **12**(4), 437–464.

Requicha, A. A. G. and Whalen, T. W. (1991), Representations for assemblies, *in* L. S. Homem de Mello and S. Lee, eds, 'Computer-Aided Mechanical Assembly Planning', Kluwer Academic Publishers, The Netherlands, pp. 15–39.

Rosen, D. W. (1992), A Feature-Based Representation to Support the Design Process and the Manufacturability Evaluation of Mechanical Components, PhD thesis, University of Massachusetts.

Roy, U., Banerjee, P. and Liu, C. R. (1989), 'Design of an automated assembly environment', *Computer-Aided Design* **21**(9), 561–569.

Roy, U. and Liu, C. R. (1988), 'Establishment of functional relationships between product components in assembly database', *Computer-Aided Design* **20**(10), 570–580.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991), *Object-Oriented Modeling and Design*, Prentice-Hall International, Inc. ISBN 0-13-630054-5.

Sanderson, A. C. and Homem de Mello, L. S. (1990), Automatic generation of mechanical assembly sequences, *in* M. J. Wozny, J. U. Turner and K. Preiss, eds, 'Geometric Modeling for Product Engineering', Elsevier Science Publishers B.V., Amsterdam, pp. 461–482.

Sanderson, A. C., Homem de Mello, L. S. and Zhang, H. (1990), 'Assembly sequence planning', *AI Magazine* **Spring**, 62–80.

Shah, J. J. and Mäntylä, M. (1995), *Parametric and Feature-Based CAD/CAM*, John Wiley & Sons, Inc, New York, USA.

Shah, J. J. and Rogers, M. T. (1993), 'Assembly modeling as an extension of feature-based design', *Research in Engineering Design* **5**(3 & 4), 218–237.

Shah, J. J., Sreevalsan, P. and Mathew, A. (1991), 'Survey of CAD / feature-based process planning and NC programming techniques', *Computer-Aided Engineering Journal* **1991**(1), 25–33.

Shah, J. J. and Tadepalli, R. (1992), Feature based assembly modeling, *in* G. A. Gabriele, ed., 'Proceedings of the 1992 ASME International Computers in Engineering Conference', Vol. 1, San Francisco, California, USA, pp. 253–260.

Sodhi, R. and Turner, J. U. (1991), Representing tolerance and assembly information in a feature-based design environment, *in* G. A. Gabriele, ed., 'Proceedings of the 1991 ASME Design Automation Conference', Vol. DE-Vol. 32-1, Miami, Florida, USA, pp. 101–108.

Srikanth, S., Turner, J. and Sanderson, A. (1991), Establishing part positioning in assembly modeling, *in* J. Turner, J. Pegna and M. Wozny, eds, 'Product Modeling for Computer-Aided Design and Manufacturing', Elsevier Science Publishers B.V., Amsterdam, pp. 199–226.

Srikanth, S. and Turner, J. U. (1990), 'Towards a unified representation of mechanical assemblies', *Engineering with Computers* **6**, 103–112.

Storm, T. (1988), The DIAC benchmark product specification, Internal report, Laboratory for Flexible Production Automation, Faculty of Mechanical Engineering, Delft University of Technology, The Netherlands.

Stroustrup, B. (1993), *The C++ Programming Language*, 2nd edn, Addison-Wesley Publishing Company, Reading, MA. ISBN 0-201-53992-6.

Waarts, J. J., Boneschanscher, N. and Bronsvoort, W. F. (1992), A semi-automatic assembly sequence planner, *in* 'Proceedings of the IEEE International Conference on Robotics & Automation', Nice, France, pp. 2431–2438.

Wearring, C. (1996), The functional feature model: Bridging the CAD/CAM gap, *in* J. M. McCarthy, ed., 'Proceedings of the ASME 1996 Design Engineering Technical Conference and Computer in Engineering Conference', Irvine, California, USA. Published on cdrom, ISBN 0-7918-1232-4.

Wesley, M. A., Lozano-Pérez, T., Lieberman, L. I., Lavin, M. A. and Grossman, D. D. (1980), 'A geometrical modeling system for automated mechanical assembly', *IBM Journal of Research and Development* **24**(1), 64–74.

Wilson, P. R. and Pratt, M. J. (1988), A taxonomy of features for solid models, *in* M. J. Wozny, H. W. McLaughlin and J. L. Encarnaçao, eds, 'Geometric Modeling for CAD Applications', Elsevier Science Publishers B.V., Amsterdam, pp. 125–136.

Wolter, J. D. (1988), On the Automatic Generation of Plans for Mechanical Assembly, PhD thesis, University of Michigan.

Wolter, J. D. (1991), On the automatic generation of assembly plans, *in* L. S. Homem de Mello and S. Lee, eds, 'Computer-Aided Mechanical Assembly Planning', Kluwer Academic Pub., Boston, Massachusetts, USA, pp. 263–288.

Wolter, J. D. (1995), Assembly sequence planning terminology.
URL: http://www.cs.tamu.edu/research/robotics/Wolter/asp/terms.html

Woo, T. C. (1994), 'Visibility maps and spherical algorithms', *Computer-Aided Design* **26**(1), 6–16.

Zink, R. (1994), XHDG (X toolkit Hierarchical Directed Graphs).
URL: http://www.informatik.uni-stuttgart.de/ipvr/as/projekte/grids/xhdg/xhdg-e.html

Appendix A

Terminology

Within the assembly research world, there exists no commonly accepted standard terminology. To avoid confusion about terminology used in this thesis, some frequently used terms are defined here. This terminology is closely related to a proposal given by Wolter (1995).

assemble Assemble is the process of merging or joining a component onto other assembled components.

assembly An assembly is a group of components merged together. Sometimes assembly refers to the process of assembling components.

partial assembly A partial assembly is defined as the already assembled components on which other components are assembled. Thus a component is actually assembled onto a partial assembly. This implies that a partial assembly is, at least, gravitational stable, i.e. it does not fall apart due to the gravitational force.

component A component is defined as a motion or transport-stable unit, i.e. it does not fall apart due to gravitational and transportation forces. A component can be manipulated to assemble it onto a partial assembly or, in the case of a base component, onto a fixture. A component can be either a part or a subassembly.

base component A base component is the first component of an assembly that is assembled. A base component is always assembled onto a fixture.

part A part — often called a single part — is the smallest component used in assembly. In this thesis these are assumed to be rigid. They cannot be decomposed into smaller components.

subassembly Sometimes an assembly is transport stable and can be a component itself. Such a component, called subassembly, can be decomposed into smaller components — parts or other subassemblies.

product An assembly that fulfills some specified functional requirements is called a product. The difference between a subassembly and a product is not that clear. For one department an assembly may be a product (e.g. a complete engine), whereas for another department this "product" may be a subassembly (e.g. the engine assembled in a car).

generic component A generic component is used to represent component information independent of the assembly to assemble the component in. For example, the geometry which is independent of the actual position and orientation of the component within the assembly is stored within a generic component.

instance component A counterpart of the generic component is the instance component. An instance component is an instance of a generic component, linked to an assembly. Thus an instance component represents component information dependent of the position, orientation and relations within an assembly. There can exist several instances of the same generic component within one assembly, e.g. several bolts to fasten a plate.

Index

Summary

Assembly features in modelling and planning

Computer programs can nowadays hardly be left out from the complete assembly process. Not only during modelling of assemblies, but also during planning of the actual assembly, programs are indispensable. In fact, modelling and planning can hardly be separated. To avoid problems in the assembly process, models should be verified in an early stage with plans to check whether they can actually be assembled.

Both the modelling of single-part components and assemblies, and the modelling and planning phase must be integrated.

Current generation models are highly focused on geometry. However, only geometric information is not enough in modelling and planning. Besides geometric information, there is also a need for functional information related to the geometry. In recent years, techniques have been developed for modelling and planning in manufacturing that combine geometric and functional information for single-part components. These models are called feature models, where features contain both geometric and functional information.

This feature concept is not only useful in manufacturing, but can also be used in assembly, as is shown in this thesis. Assembly features are subdivided into handling and connection features. Handling features contain assembly-specific information for handling components. Connection features contain assembly-specific information for connections between components.

Both for modelling and planning of single-part components and assemblies, an integrated object-oriented product model has been developed, combining form features with handling and connection features.

A prototype modelling environment has been developed. A model can be manipulated using a geometric or a graph-oriented user interface. The

model itself provides the possibilities for component-driven and relation-driven modelling. In the first method, one starts with completely finished components, and by adding relations to them the final assembly is created. In the second method, one starts with the relations, and from these relations the components can be created.

The product model has been verified within several analysis and planning modules.

Assembly features can profitably be used within assembly analysis. Especially the internal freedom of motion between components, stored in the connection features, accelerates the translational stability analysis. Rotational stability analysis can also be accelerated by using knowledge of possible rotation axes stored in the connection features.

In grip planning, during the determination of finger domains on components, the form features and the assembly features play a significant role. Although the time-consuming EFS method is still needed, the feature models can in many cases provide a considerable time reduction.

Further, feature models provide additional information that can profitably be used in compliant motion planning. Within connection features, the type of connection is known, which is important for compliant motions.

Also the complete search space to find the *optimal* assembly sequence can be substantially reduced, resulting in faster generation of the optimal assembly sequence. Some connection features already contain knowledge about possible, or impossible, assembly sequences themselves.

It can be concluded that feature-based product models for assembly can considerably help both in assembly modelling and planning, on the one hand in integrating single-part and assembly modelling, and on the other hand in integrating modelling and planning.

Samenvatting

Assemblage features bij modelleren en plannen

Computer programma's zijn tegenwoordig niet meer weg te denken uit het gehele assemblage proces. Niet alleen bij het modelleren van de te assembleren produkten, maar ook tijdens het plannen van het daadwerkelijke assembleren, zijn programma's onmisbaar. In feite kunnen de modellerings- en planningsfasen niet los van elkaar worden gezien. Om problemen in het assemblage proces te voorkomen, moeten modellen al in een vroeg stadium geverifieerd worden met assemblageplannen om te controleren of ze uiteindelijk wel geassembleerd kunnen worden.

Zowel het modelleren van enkel-stuks componenten en geassembleerde produkten, als de modellerings- en planningsfasen dienen te worden geïntegreerd.

De huidige generatie modelleerprogramma's zijn vooral gericht op de geometrie. Geometrische informatie is echter maar beperkt toepasbaar tijdens het modelleren en plannen. Naast de geometrie is ook de functionele betekenis van de geometrie noodzakelijk. Bij het modelleren en plannen van enkel-stuks componenten voor fabricage, heeft men de afgelopen jaren een model ontwikkeld waarbij men naast geometrie ook de functionele betekenis van de geometrie op kan slaan in het model. Deze modellen worden feature modellen genoemd, waarbij features zowel geometrische als functionele informatie bevatten.

Dit feature concept is niet alleen bruikbaar voor fabricage, maar kan ook gebruikt worden voor assemblage, zoals wordt aangetoond in dit proefschrift. Assemblage features kunnen worden onderverdeeld in handling en connection features. Handling features bevatten assemblage specifieke informatie die zich richt op het hanteren van componenten. Connection features bevatten assemblage specifieke informatie die zich richt op verbindingen tussen componenten.

Voor zowel modelleren als plannen van enkel-stuks componenten en assemblages is een geïntegreerd object-georiënteerd model ontwikkeld waarin naast form-features, ook handling en connection features zijn opgenomen.

Voor het modelleren van assemblages is een prototype modelleeromgeving ontwikkeld, waarmee modellen op een geometrische en op een graaf-gebaseerde wijze kunnen worden gemanipuleerd. Het model biedt de mogelijkheid om zowel component gedreven, als relatie gedreven te modelleren. Bij de eerste methode start men met de componenten en voegt men daarna relaties toe. Bij de tweede methode start men met relaties en vanuit deze relaties kunnen dan de componenten worden gecreëerd.

Het gebruikte model is toegepast op een aantal analyse- en planningsmodulen om de bruikbaarheid te verifiëren.

Bij stabiliteits analyse kunnen de assemblage features goed worden gebruikt. Vooral de mogelijke bewegingsvrijheden tussen componenten, opgeslagen in de connection features, versnellen de analyse van translatiestabiliteit. Ook het analyseren van rotatiestabiliteit kan worden versneld door de aanwezigheid van informatie over mogelijk rotatieassen in de connection features.

Bij greepplanning spelen tijdens het zoeken naar mogelijke grijpgebieden op componenten form features en assemblage features een belangrijke rol. Hoewel het volledig uitbannen van de tijdrovende EFS methode niet is gelukt, kunnen feature modellen in veel gevallen toch een aanzienlijke tijdwinst opleveren.

Verder leveren de feature modellen extra informatie op die zeer nuttig gebruikt kan worden om compliante bewegingen te berekenen, dit omdat nu juist het type verbinding tussen componenten bekend is.

Ook het totale zoekdomein voor het vinden van een *optimale* assemblagevolgorde kan aanzienlijk worden beperkt, hetgeen resulteert in het vlugger genereren van een oplossing. Sommige connection features bevatten nu zelf al kennis over eventueel mogelijke, of onmogelijke, assemblagevolgorden.

Geconcludeerd kan worden dat feature modellen voor assemblage aanzienlijk kunnen helpen in zowel het modelleren als het plannen van assemblages, aan de ene kant bij de integratie van het modelleren van enkelstuks componenten en geassembleerde produkten, en aan de andere kant bij de integratie van modelleren en plannen.

Curriculum Vitae

Winfried van Holland was born October 12, 1968, on his brother's birthday, in Maarssen, the Netherlands. He married Nelleke Post in 1991, and they now have two boys, Job (1994) and Menno (1997).

He obtained his VWO certificate in June 1987 at Niftarlake College in Maarssen. In June 1992, he received his master's degree in Computing Science at Delft University of Technology. His master's thesis described the design and implementation of a scheduler for a multiple-robot, flexible assembly cell, called the Delft Intelligent Assembly Cell (DIAC). The research was carried out at the Faculty of Mechanical Engineering, Laboratory for Flexible Production Automation, at Delft University of Technology.

In July 1992, he joined the Computer Graphics and CAD/CAM group of the Department of Computing Science, at Delft University of Technology, as "assistent in opleiding" (AIO), for a PhD-research project on assembly features, which resulted in this thesis.

In November 1996, he joined the Baan Research Department of the Baan Company in "Klein Zonneoord" in Ede, as a Software Architect, where he is involved in the development of the Baan's future Enterprise Resource Planning (ERP) software.

Feel free to contact him via e-mail: WvHolland@baan.nl