

540670

20172547

TR diss 2014

TR diss  
2014

stelling

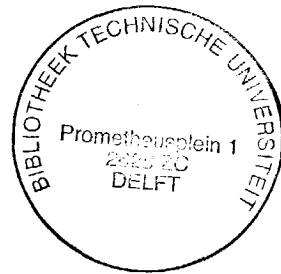
# CAD/CAM for assembly planning

## Proefschrift

ter verkrijging van de graad van doctor  
aan de Technische Universiteit Delft,  
op gezag van de Rector Magnificus,  
prof.dr.s. P.A. Schenck,  
in het openbaar te verdedigen ten overstaan  
van een commissie aangewezen door  
het College van Dekanen  
op 9 januari 1991 te 16:00 uur door

Peter Martens

geboren te Zutphen,  
Werktuigkundig ingenieur



Dit proefschrift is goedgekeurd door de promotor:  
Prof.ir. L.N. Reijers  
en de toegevoegd promotor:  
Dr.ir. T. Storm

© 1990 P. Martens, Berkel en Rodenrijs  
vormgeving en omslag door J.W.A. Martens-de Jong

CIP-Gegevens Koninklijke Bibliotheek, Den Haag

Martens, Peter

CAD/CAM for assembly planning / Peter Martens. - Delft :  
Faculty of Mechanical Engineering and Marine Technology,  
Delft University of Technology. - Ill.  
Proefschrift Technische Universiteit Delft. - Met index,  
lit. opg. - Met samenvatting in het Nederlands.  
ISBN 90-370-0057-6  
Trefw.: CAD/CAM / productieplanning.

# Contents

<b>Acknowledgements</b>	<b>7</b>
<b>Samenvatting</b>	<b>8</b>
<b>Summary</b>	<b>10</b>
PART 1	
INTRODUCTION, ANALYSIS AND RELATED RESEARCH	12
<b>1 Introduction</b>	<b>13</b>
1.1 Justification of assembly CAD/CAM	13
1.2 The DIAC project	13
1.3 The project CAD/CAM for assembly planning	14
<b>2 An architecture of a Flexible Assembly Cell</b>	<b>15</b>
2.1 Introduction	15
2.2 Production functions	17
2.3 Hierarchical levels of control	17
2.4 Architecture of the FAC	18
2.4.1 FAC functions	18
2.4.2 Operation of the FAC	19
2.5 Product Design	19
2.5.1 Triads	19
2.5.2 Design in three phases	20
2.6 Process planning and concurrent engineering	21
2.7 From product development to process planning.	22
2.8 PDM context and function	24
<b>3 Product modelling</b>	<b>25</b>
3.1 Introduction	25
3.2 Product data requirements	25
3.3 Architecture of product models	26
3.3.1 The use of standards for product data exchange	26
3.3.2 STEP	26
3.3.3 PDM entities mapped on STEP	27
3.4 Analysis of product data objects	30
3.4.1 Semantic datamodelling and logical and geometrical objects	30
3.4.2 Proportional relations in the product model	31
3.4.3 The semantic model	32
3.4.4 OO versus Relational database and implementation considerations.	37
<b>4 Assembly Modelling</b>	<b>39</b>
4.1 An assembly process model	39
4.2 Connection description requirements	41
4.3 The connection model	43

4.3.1 Description schemes alternatives	43
4.3.2 Properties of the connection model	44
4.4 Overview of the connection model	46
4.4.1 Connection model attributes	47
4.4.2 Description of a frame	48
4.4.3 Description of a set of directions	49
<b>5 Assembly analysis</b>	<b>51</b>
5.1 Introduction	51
5.1.1 Specification of products and connections to be analysed	52
5.1.2 Additional system requirements	54
5.2 Related Research in assembly analysis	54
5.2.1 Collision avoidance and path finding	54
5.2.2 Combination of contact conditions	55
5.3 Contact finding	55
5.4 Feature recognition	57
5.4.1 Feature based design	57
5.4.2 Syntactic pattern recognition	59
5.4.3 Topological pattern recognition in 3D	59
5.4.4 Features for assembly	59
5.5 The obstruction concept	60
5.5.1 Freedom of movement	60
5.5.2 Classification of obstructions	61
5.5.3 Combination of obstructions	62

## PART 2

### THE CAD/CAM SYSTEM

	64
<b>6 CAD interfacing and product model synthesis</b>	<b>65</b>
6.1 Overview of subsystems	65
6.2 2D and 3D product definition	69
6.2.1 2D: The mechanical engineering drawing	69
6.2.2 3D part definition	69
6.2.3 Product structure and instancing	70
6.3 Overview of the PDM	71
6.4 The CAD system	72
6.5 Implementation of the CAD connection	73
6.5.1 Reading CAD sheets	73
6.6 Generating geometry	75
6.7 User interface to the PDM	77
6.7.1 Editable data	77
6.7.2 Interactive specification of transformations	78
<b>7 Obstruction theory</b>	<b>81</b>
7.1 Relocation set representation of a point	81
7.2 The relocation direction set of an object	85
7.3 The relocation direction set of obstructions	87
7.3.1 Introduction	87
7.3.2 Selective relocation direction set	87

7.3.3 The imperative twist axis	88
7.3.4 Validity rules for relocation direction diagrams	88
7.4 The obstructions of an object	90
7.5 Combination analysis	91
7.5.1 Translation analysis	91
7.5.2 Rotation analysis	91
<b>8 Obstruction Recognition</b>	<b>93</b>
8.1 Definition and classification of obstructions	93
8.1.1 Obstruction parameters	93
8.1.2 Obstruction prototypes	94
8.2 Determination of contact	95
8.2.1 COG bubble check	95
8.2.2 The relation network	95
8.3 Contact analysis	97
8.3.1 Boundary points	97
8.3.2 The compare criterion	99
8.3.3 Boundary point generation	100
8.3.4 Contact situations	100
8.4 Resume of the obstruction recognition	101
<b>9 Connection Synthesis</b>	<b>103</b>
9.1 The connection model attributes	103
9.1.1 Applying the relocation direction set	103
9.1.2 The insertion point	103
9.1.3 The approach direction set	106
9.2 Performance and limitations of the connection modelling procedure	106
<b>PART 3</b>	
<b>EPILOGUE</b>	<b>110</b>
<b>10 Evaluation</b>	<b>111</b>
10.1 Overview of the implementation	111
10.2 Results	113
10.3 Comparison with related research	114
10.4 Conclusions	115
10.5 Recommendations for further research	115
<b>11 Future CAD/CAM techniques</b>	<b>117</b>
11.1 Introduction	117
11.2 The product model	117
11.3 Data based product modelling	119
11.3.1 Database management	119
11.3.2 Data working levels: a proposition	119
11.4 Assembly design	122
11.4.1 Design for assembly	122
11.4.2 Assembly design support	122
<b>References</b>	<b>125</b>



APPENDICES

	130
<b>A. The DIAC specification products</b>	<b>131</b>
A.1 Design criteria	131
A.2 Prismatic product: DIAC-1	131
A.3 Cilindric product: DIAC-2	134
A.4 The special connection product: DIAC-3	135
A.5 The Cranfield Benchmark	136
<b>B. Obstruction prototypes</b>	<b>139</b>
B.1 Plane	139
B.2 Cylindrical face	140
B.3 Cylinder	141
B.4 Thread	142
<b>List of figures</b>	<b>145</b>
<b>Index</b>	<b>149</b>

## Acknowledgements

The research described in this thesis has been done at the Laboratory of Manufacturing Systems of the Delft University of Technology, in the context of the Delft Intelligent Assembly Cell project.

I would like to thank to following persons for their support, technical contributions and discussions: Nick Reijers, Ton Storm, Jan Peter Baartman (also for the proof-reading), Nico Boneschanscher and Paul Friederichs. Especially I would like to thank my PhD-colleagues for their elation and encouragement over the years.

Finally I would like to thank Mr. van Houwelingen, former secretary of state for the Dutch Department of Defence, for his remarks regarding the support for civil research; without the arrangement with the Department of Defence about my military draft, I would not have proceeded.

This research was partially sponsored by SPIN.

Peter Martens,  
Berkel en Rodenrijs  
30 October 1991.

## Samenvatting

Flexibele assemblage wordt in de industrie nog maar mondjesmaat toegepast. Flexibele assemblage behelst het toepassen van robots voor het automatiseren van montage handelingen, waarbij de robot wordt toegerust met automatische aanvoerapparatuur en mechanische grijpers om hem geschikt te maken voor het te assembleren produkt. Een van de problemen daarbij is dat dergelijke apparatuur nog zeer duur is, terwijl de betrouwbaarheid van het assemblageproces te wensen overlaat. Zowel betrouwbaarheid als flexibiliteit varen wel bij de toepassing van sensoren, die de robot terugkoppeling kunnen geven over wat hij aan het doen is. Echter, naarmate meer en complexere sensoren worden toegepast (denk aan vision, krachtterugkoppeling), wordt het programmeren van een dergelijk systeem moeilijker. Om aan dit bezwaar tegemoet te komen, kan het programmeersysteem op basis van een produkt beschrijving veel routine programmeerwerk uit handen nemen. Immers, door het gelijktijdig aansturen van de sensoren en de robotbewegingen worden zelfs routine handelingen complex.

Ervan uitgaande dat de produkt beschrijving voorhanden is in een CAD systeem, moet een CAD/CAM koppeling worden ontwikkeld die het produkt in een verteerbare vorm aan een assemblage-werkvoorbereidingssysteem aanbiedt. Met name de informatie over interacties tussen onderdelen is in eerste instantie nog niet voorhanden, maar is essentieel voor bijvoorbeeld het bepalen van de volgorde van monteren. Een redelijk gedetailleerd beeld van de verbinding tussen twee onderdelen levert tevens een goed uitgangspunt op voor de bepaling van de montage bewegingen (**fine motion planning**). Dit proefschrift bevat een studie naar beschrijvingswijzen (*modellen*) van zowel produkt als onderdeel-relaties en naar manieren om deze samen te stellen, uitgaande van een beschikbaar CAD systeem.

Het Produkt Data Model (**PDM**) bevat de entiteiten, noodzakelijk en gebruikelijk in een produkt beschrijving aanwezig: identificaties en namen van het produkt, samenstellingen en onderdelen. Verder een beschrijving van de vormen van de onderdelen en additionele geometrische attributen, zoals toleranties op dimensies en verwijzingen naar onvolledige geometrie, denk aan schroefdraad. Het PDM kan worden opgebouwd met informatie die aan het CAD systeem wordt onttrokken. Echter de mogelijkheden van commerciële CAD systemen laten te wensen over aangaande de mogelijkheden om informatie te onttrekken. Verder is een deel van de informatie alleen aanwezig in de traditionele 2D werktuigbouwkundige tekeningen, terwijl het eigenlijk eigenschappen aangeeft van 3D geometrische elementen.

Het verbindingen model (**Connection Model**) beschrijft de relaties tussen de onderdelen, twee aan twee. Uitsluitend een typering, zoals 'cilindrische pen/gat', geeft te weinig informatie en is bovendien niet erg flexibel. Niets geeft zoveel informatie over een verbinding dan de beweging waarmee hij wordt gerealiseerd. Daarom bevat het verbindingen model enkele karakteristieken van deze beweging: het startpunt van waaruit de montage handeling moet beginnen (**Insertion Point**), hoe we daar moeten komen vanuit de vrije ruimte zonder botsingen te krijgen (**Approach Direction Space**) en de lijn waarlangs de assemblage beweging *kan* verlopen (**Insertion Path**).



Het verbindingen model moet worden gezien als een uitbreiding van het produkt model en kan als zodanig een verbetering van bestaande produkt informatie betekenen.

Het bepalen van deze parameters uit de geometrie is niet eenvoudig. Daarom zoeken veel gelijksoortige onderzoeken hun heil in vereenvoudigingen zoals de aanname dat assemblage handelingen altijd langs een van de hoofdasen van het produkt-assenkruis verlopen en dat de beperkingen in graden van vrijheid ook altijd zich afspelen in die richtingen. Dit levert duidelijke beperkingen, dus een andere aanpak is gewenst. De voor dit project ontwikkelde algoritmen gaan ervan uit dat de contactvlakken die onderdelen gemeenschappelijk hebben, een bedoeling hebben. Bij voorbeeld, een cilindrisch vlak aan een onderdeel is waarschijnlijk bedoeld als 'geleider' van een rotatie; andere rotaties zijn veel onwaarschijnlijker. Een zeker contactvlak beperkt de graden van vrijheid van een onderdeel ten opzichte van zijn opponent, maar laat daarbij uiteraard een beperkte bewegingsruimte wél vrij. Door nu de beperkingen van elk van de contactvlakken te combineren, ontstaat een resulterende bewegingsruimte van het onderdeel.

De contactvlakken worden eerst opgespoord en ingedeeld in een categorie (**Obstruction**). Door de geometrische situatie te onderzoeken worden de karakteristieken van elk contactvlak bepaald; bijvoorbeeld van een cilindrisch contact worden bepaald de positie van de rotatie-as en de hoek waarover het contact zich uitstrekt. De oriëntatie van het geheel is daarbij willekeurig.

De gevonden contactvlakken worden gecombineerd door hun afzonderlijke invloed op de rotaties en de translaties van het totale onderdeel te berekenen en daarna de doorsnede van de bewegingsruimte te bepalen. Als de bewegingsruimte vervolgens wordt geëffectueerd, kan het onderdeel worden weg-bewogen van zijn contacten. Op het moment dat het onderdeel geheel los komt van zijn opponent, is de positie bekend waarop de assemblage moet beginnen.

De ontwikkelde algoritmen voor verbindinganalyse zijn in staat de meeste gewone verbindingen in een produkt te beschrijven. De beperkingen betreffen vooral vervormbare onderdelen. Als de verbinding gemaakt moet worden door delen van de geometrie elastisch of zelfs plastisch te deformeren, zal de hierboven beschreven procedure geen oplossingen vinden. De hulp van de gebruiker (i.c. werkvoorbereider) zal dan moeten worden ingeroepen. Echter veel voorkomende verbindingen in diverse configuraties en oriëntaties zullen automatisch kunnen worden geanalyseerd.

CAD systemen zullen in de toekomst meer en duidelijker informatie beheerssystemen voor produkten moeten worden, terwijl de gebruiker een scala aan gereedschappen ter beschikking heeft om informatie te manipuleren en ontwerpen te evalueren. Buiten de toepassing in assemblage werkvoorbereiding, zouden de gepresenteerde algoritmen ook kunnen worden gebruikt in het ontwerpproces als evaluatie van ontworpen produkten op hun montage-eigenschappen.

## Summary

Flexible assembly does not find large scale implementation yet, because small series production suffers from reliability problems as well as the complexity and price of sensor integration. Applying more and complex sensors requires a great deal of skill to integrate and use, while preparation time is critical in small series production. Programming assembly actions that are supposed to be routine jobs, become tedious. Therefore, research is directed towards computer support for planning of flexible assembly cells.

The product description in a CAD system can be made to serve as basis for an automatic product analysis. In order to reason about assembly sequences and fine motions concerning the assembly, a model of the product is required that offers more information on the product's assembly properties. The connections between pairs of parts need to be described in a form, so that the planning system has a starting point. This thesis presents models to describe a product including connections, and algorithms to geometrically reason on connections.

The Product Data Model (PDM) contains the entities that are commonly present in product descriptions, such as names and identifications of the product, its assemblies and the parts. Several attributes are attached to the geometry, that may specify geometry (such as thread), or contain tolerances for the dimensions. Any attributes that originate from annotation in the 2D drawing, are stored as 3D geometrical attributes. The PDM is built from data extracted from a commercial CAD system and stored in an accessible way.

The Connection Model links parts two by two, attaching characteristics to the connection. A connection can be characterized in detail by the motion that is needed to establish it. Therefore, the connection model described in this thesis contains some main parameters of this motion. First, the point where one part is to be assembled relative to its counterpart in the connection: the Insertion Point. Secondly, the directions from where this point can be reached: the Approach Direction Space and finally the Insertion Path, that shows a possible route to assemble the part.

Analysing a connection requires extensive geometric reasoning. Previously, it has been done by assuming that there is no other motion than along the main axis of the frame of the product. Of course, this seriously limits the usability of the analysis. The algorithms developed for this research project look at the limitations, imposed by existing contacts, which are in any orientation and shape. Contact area's are supposed to have a meaning: for instance, a cylindrical contact is very likely to be the centre of a rotation. Every contact is classified into a so-called obstruction and is assigned properties. The combination of obstructions shows which possible directions can still be moved in, as the impossibilities imposed by the obstructions add up. When the found free directions are applied, a way out is found for the part in this position. As soon as all contacts cease to exist, the insertion point is found.

The developed algorithms are able to analyse most of the common connections. Limitations involve primarily deformation. On any connection that has to be

established by elastic or plastic deformation, the procedure will fail. A user will have to provide the information manually.

This research project learns that the interfaces of a commercial CAD system are not very effective for use in a CAD/CAM system. A standard product model such as STEP could overcome this problem. Apart from providing information on products, the standard product model could be extended with a connection model, in order to give basic information about assembly properties of the product.

Future CAD systems will contain much designer support and will behave as a data management system. Apart from assembly planning, a promising use of the presented algorithms could be as a design-evaluation tool to judge a product's assembly properties.



PART 1  
INTRODUCTION, ANALYSIS AND RELATED  
RESEARCH

# 1 Introduction

## 1.1 Justification of assembly CAD/CAM

Using robots for flexible automatic assembly is a promising area of rationalization in manufacturing. However, economy and technology do not justify a large scale implementation yet. At this moment, flexible automatic assembly is mainly a promising area of research.

A robot applied in large scale manufacturing is merely a pick and place unit, its flexibility usually hardly used: it is doing the same thing over and over again, optimized for its task and the environment optimized for the robot. The peripheral devices for a robot assembling large batches of products are well adjusted to this product and the robot program is well tested, running smoothly. All possible causes of errors are eliminated or dealt with upfront.

When the environment, the product and the production equipment are not optimally in harmony, errors may occur. Equipping the robot with eyes and ears may solve part of that problem, if adjusting the environment is not feasible. In general, adding sensors enlarges the possibilities and flexibility of the robot.

Assembling small batches of products with a robotic cell is only feasible if sensors can adjust the robot's behaviour and if the peripheral devices are flexible as well. However, programming such a setup is time-consuming and requires a great deal of skill of the programmer. The sensors are usually not integrated in the robot system and the robot programming system. Since sensor equipment can consist of very complex electronic devices such as vision and force measurement, the interface to the robot and control via the robot's I/O can be very difficult.


The programming of numerically controlled production equipment such as lathes, machining centres and robots, requires skilled employees with both manufacturing knowledge and some computer knowledge. When the programming and the equipment becomes more complex, this requires even more skill. If batches are small, this scheme is not justifiable.

## 1.2 The DIAC project

Research in the area of automatic assembly addresses the problems mentioned above: making programming easier or obsolete, developing better sensory equipment, developing more flexible peripherals and actuators (robot and other). This highly multidisciplinary research requires many new techniques to be merged and interfaced.

At the Delft University of Technology, the research project **Delft Intelligent Assembly Cell** is a cooperation between four university departments, Mechanical Engineering, Applied Physics, Electrical Engineering and Computer Science. Eight research groups each provide the project with the newest technology in their respective specialities.

The Manufacturing Systems laboratory of the department Mechanical Engineering provides for the process knowledge of assembly and manufacturing. Product design,



process planning and production control are the main contributions to the DIAC project.

The DIAC project is a means to focus the on-going research in the various research groups towards a common goal: the realisation of a working Flexible Assembly Cell. The designed architecture is generally applicable to a FAC, and has been used to design a prototype implementation.

### 1.3 The project CAD/CAM for assembly planning

The goal of the project described in this thesis is to study techniques to automate and facilitate assembly planning. The project focuses on product modelling and product model based assembly analysis. The research covers the area of preparation preceding the actual process planning, from product design down. A new model is developed for product and assembly description, and the algorithms to automatically generate this model.

Within a group of researchers involved in the DIAC project, an architecture has been developed for the process planning of FAC's. This architecture is generalized here, in order to create the context of the project CAD/CAM for assembly planning (chapter 2). Part 1, chapters 1 through 5, of this thesis investigates the requirements and available techniques for CAD/CAM with respect to assembly planning. New techniques are developed as available techniques do not suffice.

Part 2 (chapters 5 through 9) introduces a CAD/CAM system based on the basic techniques founded in Part 1. Especially the connection analysis theory, which is an innovation developed in the context of this project, will be expounded in Part 2. A prototype implementation has been made to demonstrate and evaluate the results of this project. This prototype will be incorporated in the final DIAC prototype.

In Part 3, the evaluation and a vision of the future of CAD systems are presented (chapters 10 and 11). The appendices provide an overview of the products used for evaluation and specification, and an overview of definitions used for connection modelling.

DIAC has provided for a testbed and context to develop a CAD/CAM implementation for a FAC. The interaction with DIAC as related research has been an inspiration and a source of specification. The concepts of some of the developments however, soar over the target context of the implementation described in chapter 2. The theory of geometric reasoning on interacting, moving parts may provide a basis for future studies on assembly.

## 2 An architecture of a Flexible Assembly Cell

### 2.1 Introduction

This chapter develops an architecture for a Flexible Assembly Cell. A useful architecture is compiled partially from general reference models and the conditions and goals of research on a FAC for small series production. It is shown that product development (CAD/CAM) is an integrated part of the architecture because it is specific for assembly.

Some of the concepts in this chapter are the joint effort by the members of the DIAC

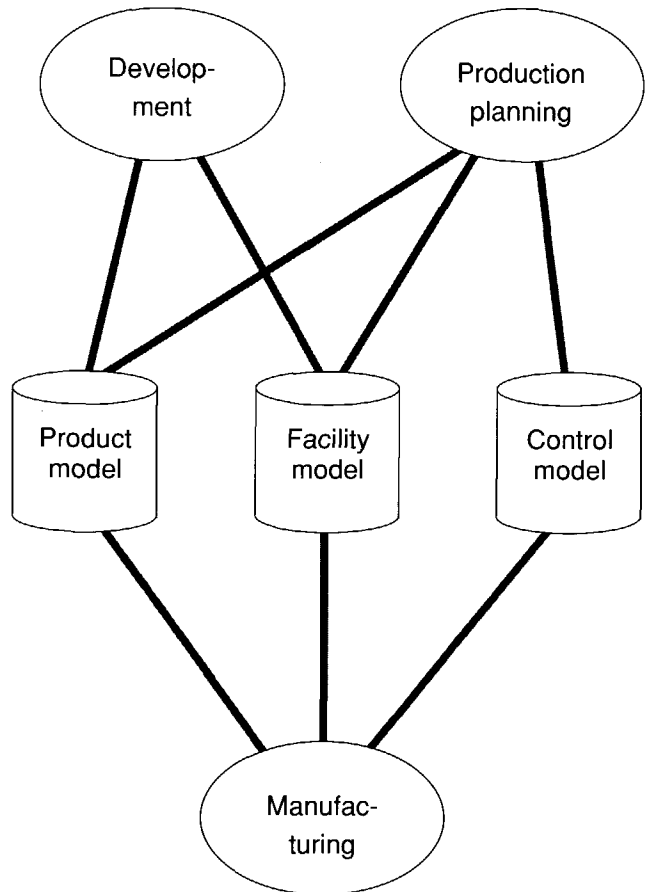


figure 2.1 Major facility functions and interfacing databases

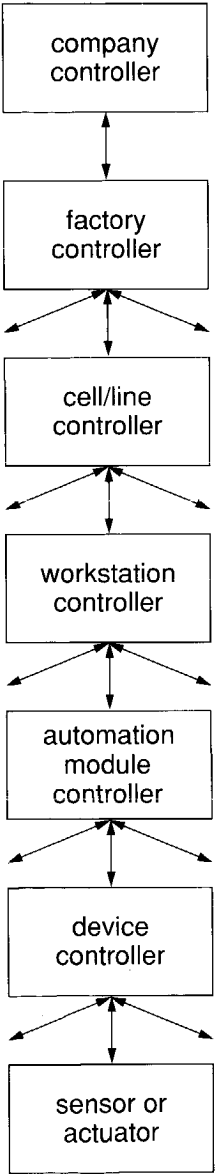
CONTROL LEVEL	CONTROL SERVICE	TASK OF CONTROLLER	EXAMPLE COMMAND
 <p>company controller</p>		determine firm and forecast production targets	
<p>factory controller</p>	dispatch products at due dates	plan production -which items to process in which time slot-to dispatch products timely	dispatch radios d at day e to f
<p>cell/line controller</p>	process item in allotted time slot	schedule when, where, which operations are executed on parts	process boards g between day h and i
<p>workstation controller</p>	execute operation on part	coordinate the execution of processing steps on objects to realise operations	put components h,i,j on
<p>automation module controller</p>	execute processing step on object	determine required paths of joint variables, which describe the state of effectors	move object m to n, max. pressure o
<p>device controller</p>	follow joint path	issue control signals so that joint variables are servoed by physical parameters	change joint angle p to q
<p>sensor or actuator</p>	modulate physical parameters		(control signal to motor)
sensing tasks, services omitted			

figure 2.2 reference model of MPCSS [Biemans 89]



project [Jonker 88].

## 2.2 Production functions

A production facility can be roughly split up into three major activities [Reijers 90]:

- Manufacturing
- Production planning, scheduling and control
- Product Development and -design and process planning

Manufacturing is the actual production facility: the computer controlled machines and the people directly involved in producing products. The production planning concerns itself with product orders, material flow (logistics) and control of the production facilities. Control can be split up in hierarchical levels, some of which usually reside in manufacturing itself (see section 2.3 on page 17). Product Development produces the description of the product to be manufactured, and also *how* it is to be manufactured (process planning).

Within a production facility, three major collections of data can be recognized:

- Product Model
- Control Model
- Facility Model

The Product Model contains primarily the product description (the results of Product Development), including the Process Planning results. The Control Model holds all Production Planning results, such as orders for specific machines, due-dates etc. The Facility Model contains information on the manufacturing equipment and tools.

The three major production functions can be connected via these models, see figure 2.1.

## 2.3 Hierarchical levels of control

The **Manufacturing Planning and Control System** (MPCS) is a formal, strictly hierarchical view on a production facility divided into seven levels, see figure 2.2. This is a reference model, meant to identify functions conceptually. It will serve as a reference for the FAC architecture.

Production Planning concerns itself with the logistics and organisation of the production. This complies with the factory controller and maybe the company controller in the MPCS reference model [Biemans 89]. Techniques such as MRP apply here, in order to make a plan for the entire production facility, based on company strategies and customer orders.

From the cell/line controller down, the control levels are assumed to be part of the Manufacturing function. The cell/line controller typically matches actions to physical machines. The workstation controller controls the tasks of a single machine.

Product development as such is not part of the control hierarchy; it provides information for modules at several levels in the control hierarchy. One can argue that Product Development takes at least orders from the highest controllers (company/factory), but functionally it does not map onto the presented control system. The factory controller takes product information such as the Bill of Materials to plan production of parts and products. The workstation and automation module controller need more detailed information on the product and its manufacturing process.

Concluding, two of the major production functions map to the control model: Production Planning to the (company and) factory controller and Manufacturing from the cell/line controller down. The third, Product Development, takes a separate place besides the control system.

## 2.4 Architecture of the FAC

So far, only the general reference model has been presented and no distinction between machining and assembly has been made (both *manufacturing*). The reference model identifies conceptual functions and an architecture will select the functions that are to be realised. How does the MPCS translate into automated flexible assembly and what functions would be realised in a Flexible Assembly Cell?

### 2.4.1 FAC functions

Clearly a FAC is part of Manufacturing. Production control is assumed to be outside the system borders, so the highest level of control is the cell/line controller. As stated

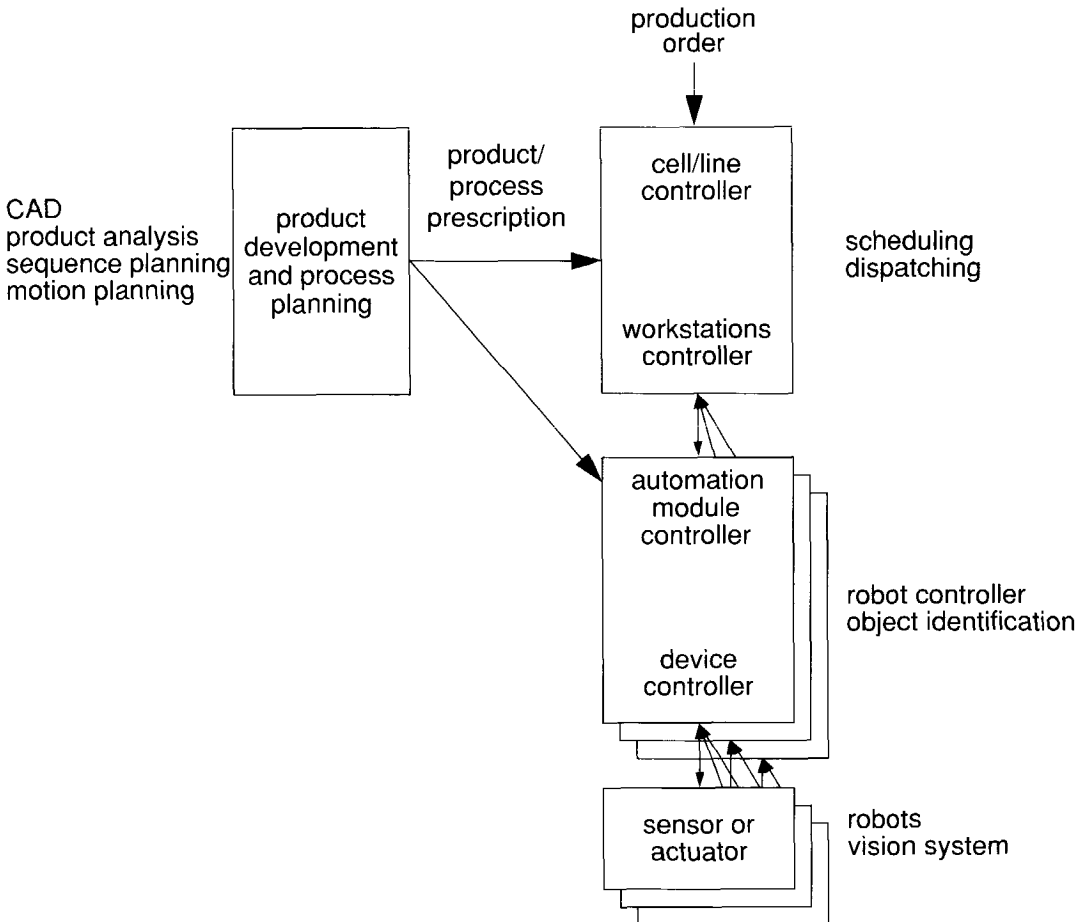


figure 2.3 System borders and elements of the FAC planning and control.

in the first chapter, for smaller batches of products, more sensing equipment and more flexible actuators are needed. The process planning of such a complex integrated manufacturing system is very specific to the various components in the cell.

*Therefore, it is considered necessary in this stage of research and development of advanced assembly systems, that the process planning is an integral part of the system. It must be stated here that research in process planning for assembly might be independent of equipment development for assembly, but any working integrated prototype will show many dependencies.*

The functions that are taken into account for the FAC are drawn in figure 2.3. The process planning and product development are part of the architecture, and provide for the product description and the process prescriptions. The cycle of a product through the system starts with a CAD system, which models the product geometrically.

As said before, a FAC fit for small series must have sensing equipment. Computer-vision for instance is a powerful sensor that can help perform tasks better and provide more flexibility and reliability. The figure 2.3 shows what the various controllers of the reference model comprise of in the FAC. There is hardly any interaction between process planning and the cell controllers other than assembly scripts (the process prescriptions). Some of the general product descriptions are used, such as geometry data for the vision: e.g. the vision control system matches observed objects to the known profiles of the product's parts. The process planning however uses knowledge on the configuration and capabilities of the system, in order to make a process plan that is geared to it.

### 2.4.2 Operation of the FAC

The general control commands to the cell are production orders. The higher level controller will issue batches to be produced and due-dates. Scheduling and dispatching of actions are done by cell/line level controllers and at workstation controller level. A Batch of products is scheduled in time and the appropriate actions are assigned to the various workstations in the cell. The scheduler is provided with assembly scripts, which contain the assembly actions needed to assemble the product. These scripts are the result of the process planning, which is not contained in the control hierarchy, but merely a source of information. The process planning itself is therefore not scheduled.

Examples of the controlled automation modules are the robot controllers and the vision controller. The vision module is capable of processing the images it receives from the actual vision system, into qualifications such as *object recognized* or *position/orientation is...* In a preparation phase (off-line), the vision controller takes object and product geometrical descriptions, from which properties are analysed and stored. During production (on-line), the vision system uses the prepared product data to recognize the objects in the scene, their locations and properties (inspection) [Buurman 90].

## 2.5 Product Design

Product design is a crucial function in the presented architecture (figure 2.3 on page 18). The concurrent product and process development is not a feasible scheme for the presented system. This section explains why.

### 2.5.1 Triads

The design of a product has several phases. The first matches specification to

functionality. The second phase translates functionality into design (shapes and structures) and in the final phase the product is detailed and laid down in engineering drawings and specifications, ready for manufacturing. To illustrate the fundamental differences between these phases, the following table mates concepts into triads that show similar evolution in three phases. Please note that the list is just a vehicle to give an impression of evolutions which come in three concepts; there is no relation in *vertical* direction between the concepts of the rows.

- |                             |                           |                                |
|-----------------------------|---------------------------|--------------------------------|
| • Conceive -                | design -                  | detailing                      |
| • Research -                | development -             | engineering.                   |
| • Artist's impression -     | calculation model -       | mechanical engineering drawing |
| • Inventor -                | engineer -                | technical draughtsman          |
| • Function -                | shape -                   | dimension                      |
| • Product -                 | part -                    | feature                        |
| • Reference model -         | architecture -            | implementation                 |
| • (Geometric) model entry - | internal representation - | external representation        |

For instance, engineering may require a design process all by itself, only starting from existing knowledge. Research concerns itself with creation of new techniques. Development is the intermediate to translate new techniques into tools and knowledge for improved engineering.

## 2.5.2 Design in three phases

### First phase

The creation of the principles of an apparatus is a creative process. Alternatives must be generated and judged. The result of this first step is the function and concept of the product. The main tools are creativity, experience and a design-methodology.

### Second phase

Development of the concept into shapes and structures requires knowledge of mechanical engineering and of materials. Calculation of static and dynamic properties of parts and constructions must provide inputs for the shaping and may give feedback that the concept does not work. In most cases this phase is performed with engineering knowledge and executed by hand with calculators. Sometimes there may be computer programs available for special cases. Lately there are more sophisticated software packages available that support the engineer at common techniques such as finite element analysis. The program takes a solid model, material characteristics and boundary conditions (e.g. applied forces and attach points) as input and calculates properties (e.g. place and value of highest stresses). The (sometimes integrated) solid modeller must be used to produce a rough-cut model of the relevant parts of the product.

### Third phase

The design department usually presents its output in the format of mechanical engineering drawings, that contain standardized two-dimensional representations of three dimensional entities. Many CAD systems are available for this purpose, which support the process of making the drawing.

Three-dimensional modelling is not yet used much as a means of specifying the designed product. A common usage is making nice pictures of the product and, as

stated above, as input for design-support software. CAD/CAM and CAPP software requires 3-D, so a solid model should be output as well. It is more likely that a draughtsman will produce such an elaborate detailed 3-D model, than the designer himself. The rough models created in the second phase may provide a head-start, but the product needs to be specified in detail. So what we need is a CAD system that is able to support 2-D and 3-D detailing, in order to produce all design output.

It may be so that sometimes the boundaries between these phases are not distinguishable, and/or they are performed by a single person. However, structured design requires some sort of phase-division.

## 2.6 Process planning and concurrent engineering

There are two basic philosophies to do process planning: batchwise analysis of the product description afterwards or integrate process planning in the design phase, sometimes referred to as **concurrent engineering**. However, concurrent engineering is more: design of special production tooling, production planning and more company functions, that are better integrated and apply concurrency in their tasks concerning a product. In that sense, concurrent engineering is a organization change, in order to reduce the design-to-market time. Therefore, concurrent engineering itself is not discussed in this section, but just whether the batchwise analysis can be skipped if the planning is integrated with the design phase.

The designer knows what he is developing and will devise at least one way to manufacture and assemble the product. Why wouldn't we 'extract' these thoughts right away and use them for manufacturing process planning instead of analysing the product afterwards?

There are five objections to this procedure:

### **Missing of optimal solutions. One way vs. the best way.**

The designer thinks of at least of one way to manufacture/assemble things. This may not be the best way. Assembly sequence planning and the use of (non-functional) subassemblies in production are usually handled by the production planning.

### **The designer may not be the one detailing the product.**

This could be a more basic problem. As shown above, the final model of the product is only created in the last phase of design. In this phase all exact features of the product are drawn, from the directions created in the previous phase. It is very well possible that the designer himself is not the one detailing the product to engineering drawings. The system the designer uses to do calculations, needs a geometric model as well, but it need not be exactly the same as the final one. This poses problems on the conservation of input-features: they do not represent the actual product anymore.

### **Design features vs. manufacturing features.**

If a system chooses to use features to enter product data, the user will need a type of feature that is related to design: the design feature. Manufacturing may have a different view on aspects of the product and define its own manufacturing features (either assembly or machining features). This is a practical obstacle that may produce erroneous results. E.g. interference of two parts that are not really connected, but are so close that they should be regarded for the assembly process planning as having a connection. Or: two slots close together (take-away volume feature) yield another feature: a fragile rim that needs special care. When the process planning is done

afterwards, there is no doubt in which features to recognize and extract, indifferent of any design features.

### **Response time for interactive systems will grow.**

Performing complex calculations with an interactive system can cause the response time to become unacceptably long. A minor point, that may be solved in the future by optimal software and fast hardware.

### **Change of working method and more tasks for the designer.**

Full concurrent engineering is before anything else, an organizational improvement, which affects large parts of the company. For process planning integration only, a change of methods for the designer may delay acceptance by designers.

The advantages of concurrent engineering are obvious: reduce product-to-market time, reduce errors and improve manufacturability. But to my opinion, there is no real alternative to separate process planning, but it may be better integrated with design and benefit from more detailed, process related product data produced by the designer.

The creation of a CAD system is a complex matter and research shows that incorporating tools that support concurrent engineering is even more complicated [Veltheer 89, Cutkosky 89]. Since the main stream of this research is process related and aims at automatic process planning, the CAD system itself is not subject of research here. However, product design is closely related to the product data model, so attention for CAD principles is unavoidable. An interface to a commercially available CAD system will be presented. This thesis will focus on batchwise interpretation of product design afterwards.

## **2.7 From product development to process planning.**

The planning of assembly actions is performed on the basis of a product description, a system (cell) description and process knowledge. For every product, the cell would have to be configured by adding the right peripheral equipment and modifying the fixtures and transport system. A cell configuration for (mixed) families of products is referred to (here) as an *application*.

For each application, the system (cell machinery and capabilities) data is entered once and maintained manually. The product information, however, is one of the two main inputs to the manufacturing system; the other input is simply production orders (refer to figure 2.4). Gathering, structuring and (pre-)processing of product data is therefore a substantial part of product development and process planning. This is the task of the system built around a Product Data Model (PDM) and is referred to as *Product analysis system*. This chapter is intended to create the context for the Product analysis system.

According to Heemskerk [Heemskerk 90], Assembly Process Planning can be performed at four levels (within parentheses the most important activity on that level):

- Batch
- Product (sequence planning within a product)
- Part (coarse motion planning)
- Primitive (fine motion planning)

To my opinion, batch level should not be taken into account in the process planning. If there is any planning at this level to be done at all, it is part of the scheduler:

scheduling for several products in a row. However, the concept of the hierarchical planning steps product-part-primitive is obvious: one does not do motion planning if the conditions defined by the sequence planning are not yet laid down.

Assembly sequence planning relies on more product information than is directly available from the product drawings as can be derived from [Heemskerck 90]. For instance the relations between parts and the relative degrees of freedom of interacting parts.

*Therefore there is a missing link between the product design phase and the required product model. This pre-planning step is the actual CAD/CAM interface, which upgrades the product drawing data to a usable product model for assembly process planning.*

With the three levels of process planning and the product design link, the function 'Product development and process planning' is outlined in figure 2.4.

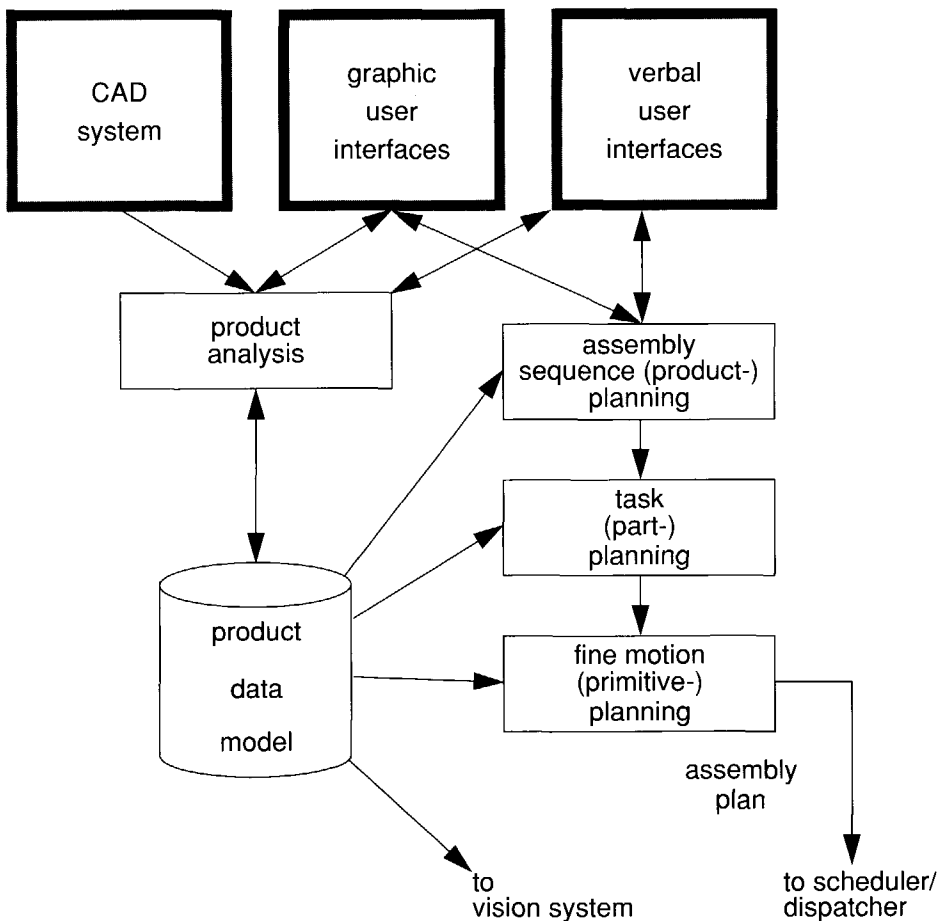


figure 2.4 Product development and process planning



## 2.8 PDM context and function

The function of the PDM is to provide information to various subsystems of the FAC on products as defined in the CAD System (see figure 2.4). Besides plain geometrical data for functions such as the vision and object identification, the assembly-related product information for the planning system is most important. The assembly characteristics that can be derived from the geometrical description are done in the Product analysis system, all system related reasoning is done in the planning. The Product analysis system interfaces to a CAD System, interpreting the various data, and can therefore be considered to do *product-related pre-planning*.

The Product analysis system does geometrical reasoning without placement-uncertainties (it assumes a perfect actuator). Any tolerance information in the dimensioning, provided by the designer is stored for later interpretation. The system judges the design on its functional features for assembly. It does the product-related data-management and produces geometrical data, a relation network and assembly-connection parameters.

The sequence (product level) planning [Heemskerk 90] uses the data on located connections (freedom before/after assembly) in order to determine an assembly sequence with its alternatives. The task (part level) planning and fine motion (primitive level) planning [Baartman 90] take the detailed connection parameters and reason about uncertainties in order to generate a strategy to assemble the parts.

The FAC needs interfaces to the user in several ways. It is considered impossible to automate the planning process completely, so a user (the process planner) can give directions and can answer questions from the system in a dialogue. The designer of course uses the CAD system for entering primary product data, but can also give verbal directions in the drawings. In order to resolve problems analysing the product data automatically, a user can enter or modify derived product data, which is normally the result of interpretation. A graphic user interface provides this functionality.



## 3 Product modelling

### 3.1 Introduction

The contents of the product model for the flexible assembly cell are dictated by the applications that use the information. A quantitatively large part of the required product data is similar to the requirements of manufacturing process planning. Therefore, general product model standards are reviewed for inspiration for the assembly model.

The elements of a product model and the structure or relationships between those elements form the actual model. A study is presented of ways to represent these (verbally expressed) relations. This will lead to the structure of the product model for assembly. Chapter 5 discusses the database derived from the study in this chapter and the database's contents, the products.

### 3.2 Product data requirements

The functions in the automatically planned assembly cell that require product data are the 'data consumers' to the Product analysis system. They are (also see figure 2.4 on page 23):


- User interaction modules (graphic user interface, natural language interface)
- Assembly sequence planning (Product level)
- Task planning (Part level)
- Motion planning (Primitives level)
- Vision (recognition and determining of position/orientation)

Three categories of product information can be distinguished that are required by the mentioned functions.

- Product structure data.
- Geometrical and other physical data
- Assembly (process-) related data

The product structure data contains the necessary identifications, names, subassemblies, comments and so on. A common subset is the **Bill Of Materials (BOM)**: a list of all different parts in the product, some comment and material specification. A mechanical engineering drawing often contains, beside name and material specification, some comments on manufacturing or assembly of the product. This information could be processed by a language interpreter [v. Rijn 91].

Geometrical descriptions are also required by most data consumers. Position and orientation of parts in the product, along with a volumetric description of shapes are most important. Physical properties of an object such as **Centre Of Gravity (COG)** and inertia properties are taken into account as well. A mechanical engineering drawing contains annotation, specifying the dimension of (sets of) geometric elements. The nominal sizes are assumed to be given correctly in the geometric model, but additional information is valuable: tolerances and thread specification. Since these



are traditionally placed in the 2D views, they must be interpreted to translate to 3D properties.

Assembly related data is specific for this application, and merely the result of an analysis, a *pre-planning* phase. It concerns assemblability of the product and classification of connections used in the design. This information is part of the product data for assembly. Standard product models usually contain no or very little assembly information.

### 3.3 Architecture of product models

#### 3.3.1 The use of standards for product data exchange

Standard product definitions aim to provide a neutral format that makes a smooth data exchange possible. A commercial system (either a CAD system or process planning) should conform to these standards in order to be able to communicate with other vendor's systems. The continuity of a company's drawing library is only guaranteed if drawings can be interpreted by a next generation of CAD systems, not necessarily of the same vendor.

Manufacturing systems need product data in various forms. For a production planning system such as Manufacturing Resource Planning (MRP), the BOM will do, if it includes information on subassemblies that act as intermediate products in the shop. Manufacturing process planning, where the cutting data is produced, needs a detailed geometric description of parts. For the purpose of a product model for manufacturing as an interface to CAD systems and in between CAD systems, there are several evolving standards [Vergeest 89, Grabowski 89]. STEP, the ultimate product model standard, is discussed separately in the next section.

At this moment, IGES is the most widely used and accepted standard for geometry transfer. Version 4.0 handles solid geometry as well. The German VDAFS (VDA Flachen Schnittstelle) provides extensions to IGES primarily for complex surface modelling.

The European CAD\*I [Schlechtendahl 87] is a definition for a neutral file for CAD geometry and product structure, for the exchange of information between CAD systems. The results of this project have contributed to STEP (see subsection 3.3.2).

The French SET is a standard that is currently probably the most competent standard currently in use (IBM's Catia, Prime's CADD4X and ProEngineer have developed interfaces for SET). It has good geometrical description facilities, including Brep (Boundary Representation) and CSG (Constructive Solid Geometry) for solid modelling. It provides a more compact data exchange form than IGES does.

The American PDES (Product Data Exchange using STEP) project has been merged into the international STEP standardization effort.

Several other standards exist, some of which are developed by vendors of CAD systems and accepted by others. For example, the DXF (Data eXchange Format) format by AutoDesk is a very popular interchange format for drawings between PC-based CAD systems.

#### 3.3.2 STEP

STEP, **ST**andard for the **E**xchange of **P**roduct model data, is an international effort to combine all existing product modelling experience into one standard. When

comparing existing standards with STEP, one can see that STEP indeed covers most, if not all of its predecessors.

At the conceptual level (architecture), the **Integrated Product Information Model (IPIM)** is divided into two parts: the *Resources* (e.g. geometry) and the *Applications* (e.g. Finite Element Method data). The idea is that all applications refer to the uniform resources and that implementing new applications does not affect any of the existing models. The IPIM is specified in the special STEP-datamodeling language **Express**. An implementation of the (conceptual) STEP-IPIM is a structured file with standardized format (like IGES); another implementation would be a database. Current standards such as IGES cover part of STEP's resources (see figure 3.1).

Resources:

- Geometry
- Topology
- Shape
- Tolerances
- Material
- Presentation

Applications:

- Mechanical Products
- Architecture engineering construction
- Electrical
- Finite Element Method
- Drafting

The Geometry model contains the usual 2D and 3D primitive entities such as points, surfaces and lines and coordinate systems. STEP defines the curve and surface primitives to be parametric, and allows B-spline for complex forms.

The Topology model uses and combines the entities in the geometry model in order to create an entity with a certain meaning or a *set*. For instance, two points are defined in the geometry model and used in the topology model to create two vertices and an edge (a vertex is a point used to define an edge, such as in the ends of the edge). These are the typical Boundary Representation entities, used in the Shape model to create solids, surface models and wire frame models.

### 3.3.3 PDM entities mapped on STEP

Most of the standards provide a definition for storage of a variety of geometrical data. Solid models are stored in Brep or CSG and combinations; there are surface models to make complex curved surfaces, even wire frame models are provided. Additional data is found in so-called **applications**. This would be the way to add process data to the product model. For manufacturing process data there are separate standards such as **Cutter Location File (CL-file)**.

What we are looking for is a Boundary Representation for the geometry. Our main interest lies in the assembly application. The product definition standards provide, not surprisingly, very little to go on for description of assembly actions. STEP at least recognizes product structure in the form of (sub)assemblies. The limited BRep uses a subset of the entities provided in STEP (see figure 3.2 on page 29). There is no need

STEP-IPIM

Resources

Geometry	
Coordinate system	Surface
Point	plane
Curve	cylindrical
line	B_spline_surface
circle	B_spline_curve
Topology	
Vertex	Shell
Edge	open_shell
Face	closed_shell
Shape	
Wireframe_model	Solid_model
Surface_model	CSG_solid
	Facetted_Brep
	Swept_area_solid
Material	
Tolerances	

Applications

Mechanical Products
Architecture engineering construction
Electrical
FEM
Drafting

figure 3.1 (incomplete) Overview of the STEP-IPIM entity hierarchy

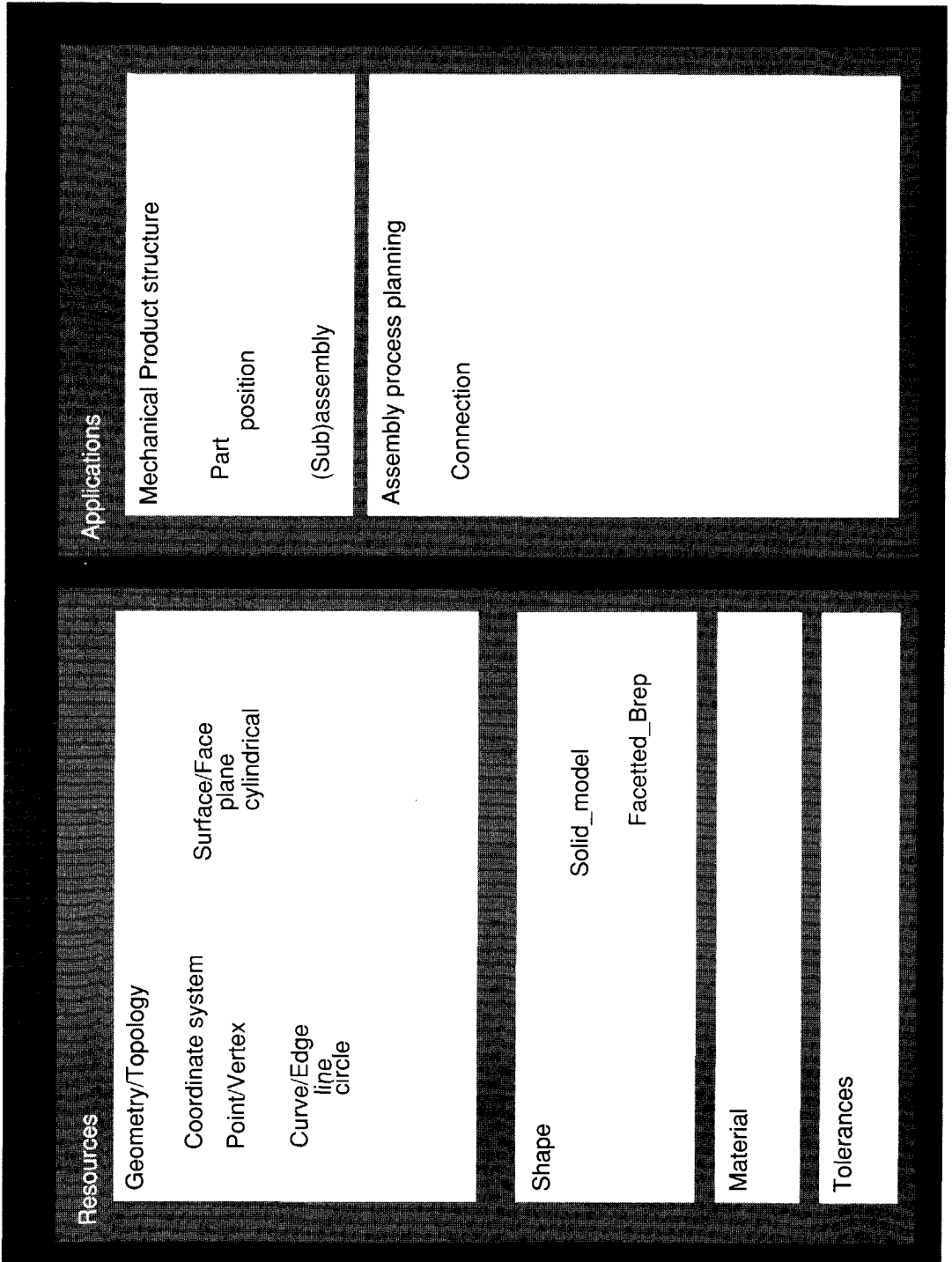


figure 3.2 Mapping of the main PDM entities on STEP

to make a distinction between points and vertices, curves and edges, etc., because the only users of these resources are BRep entities. Thus there is no difference between geometry and topology as modelling structures.

For this particular application, CAD/CAM for assembly, we are not specifically conforming to a standard. We are interested in doing assembly process planning and need some kind of product modeller, i.e. a CAD system. The research focuses on analysis of the geometry and not how to define it. Therefore a standard CAD system is chosen, which has limited data exchange capabilities. The used CAD system, Medusa, provides some 3D data exchange capabilities, so a custom interface can be built.

The interest in product definition standards in the context of this chapter is to see how the data that is required for CAD/CAM in assembly is organized and to copy that wherever possible. It is however possible to add some of these extensions to a product definition standard (see chapter 10). In the context of STEP, clearly assembly process planning must be seen as an application that uses (part of) the resources. In the next chapters, assembly (process) modelling is discussed.

### 3.4 Analysis of product data objects

The entities in the PDM as discussed above, have relations linking them. Since all entities of the product model are defined explicitly for the application of CAD/CAM for assembly, the relations have to be investigated in order to structure the data. The results are summarized and used in section 6.3 on page 71.

This section is devoted to the study of the properties of entities and their relations. For studying purposes, a semantic datamodel is constructed. This thesis cannot provide a detailed introduction to object-oriented and semantic datamodelling, so the reader is referred to the literature references.

First, a number of concepts concerning datamodelling are introduced. These concepts will be used to clarify and describe the relations between various entities in the product database. Please note that the attributes of the entities are not defined here.

#### 3.4.1 Semantic datamodelling and logical and geometrical objects

The label *Object Oriented (OO)* is used for several common computer techniques, such as programming and databases [Stroustrup 87, McConalogue 91]. An object is an autonomous functional entity with attributes attached. One can define operations on a **class** of objects, which are called **methods**.

The concepts of classification, aggregation and generalization are used in OO languages, and also form the basis of semantic modelling [Ter Bekke 91].

**Classification** is the definition of a class of data objects, a type. The word 'object' has been used in this thesis for physical, concrete items. Therefore we make a distinction between **data objects** and **physical objects**. A physical object is something in the real world, which is touchable. A data object is an abstract *concept* of something in the real world, which can be assigned attributes that inform us about some properties. A physical object, when represented in a datamodel, is therefore one of the so many *data* objects of the model. Examples of data objects in the context of this thesis are a product, a subassembly, a part and a physical object.

**Aggregation** is the joining of several objects or attributes to form a new object. An object *has* a certain attribute. For example, a person has a name and an address.

figure 3.3

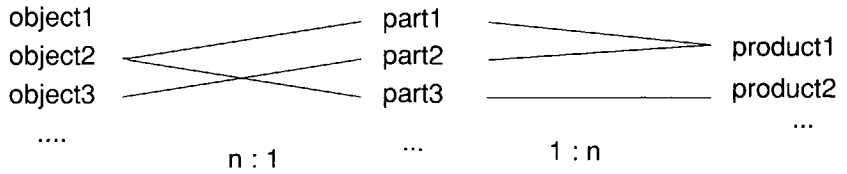
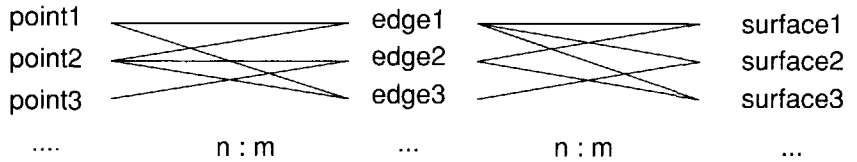


figure 3.4



When a class of objects resembles another (has attributes in common), but is a more general description, it is called **generalization**. For example, a van *is a* car, and a car *is a* vehicle. The van *inherits* the attributes of the car: the car *has* a manufacturer, an engine power, a colour, etc., and so has the van.

An **instance** is *an* object of the type 'class'. When a class of objects is defined (e.g. van), a representative of this class is called an instance (e.g. my brother's blue van, manufactured by Peugeot).

For an elaboration on semantic modelling, refer to the relevant literature [Ter Bekke 91].

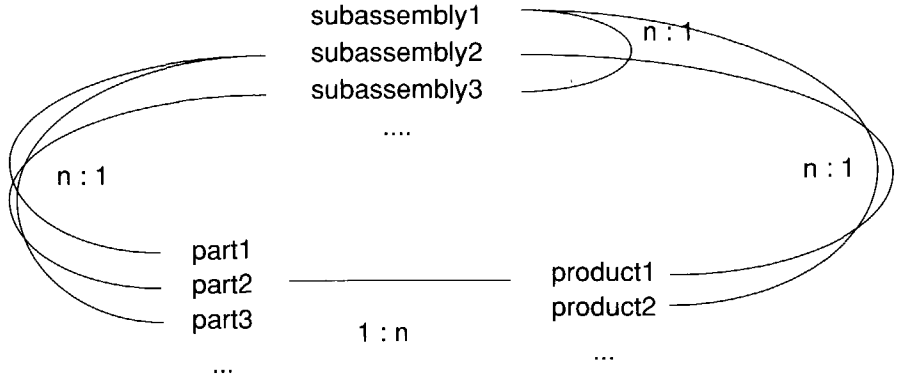
### 3.4.2 Proportional relations in the product model

The representation of the product data in a data model has several problems, which make it hard to express in a standard data modelling technique and a standard database. Relations between objects have a *proportion*. The proportion of a relation appears to play an important role in product modelling and accounts for some of the complexity. We start with examining the proportional relations in product modelling.

As an example, figure 3.3 shows a possible relationship between two products, three parts and three objects. It is obvious that a product can have multiple parts, but a part belongs to only one product (part : product = 1 : n). On the other hand, a part refers to a physical object. An object may be used by multiple parts (object : part = n : 1). For instance, there exists an object 'Screw M4x20', which is the object of a part 'Fasten screw' in a product 'Box'; In another or the same product, there is a part 'Adjustment screw' using the same object 'Screw M4x20'. This means that the indirect relation product : object is n : m, for an object can be used in different products. In figure 3.4, two examples of direct n : m relations are shown: edges use multiple points, and points may be shared among edges.

A subassembly is a peculiar case: it has a relation with members of the same class: components of a subassembly are either parts or other subassemblies (1 : n). A product consists of one or more subassemblies (1 : n), so the schematic overview looks like figure 3.5. Connections between two parts make up the relational network (also see figure 4.2 on page 41 and figure 8.5 on page 96), so a 2 : 1 relation would exist. Since a part can be connected to one or more other parts, there exists a 2 : n relation.

figure 3.5



First, we will translate the relations described above for the product model problem in the semantic datamodel. Next, the mappings on respectively the relational database and the object oriented database are presented.

### 3.4.3 The semantic model

In figures, object classes are rectangles, an aggregation is shown as lines from the centre (such as in figure 3.6) of the blocks and a generalization is shown as lines from the corners of the blocks (such as in figure 3.9).

Concerning objects in the product model, we can put forward the following statements:

- a product has a number of  $n$  parts.
- a part belongs to a specific product: a part has a product. (analogous to the manufacturer and the car).
- a part is a physical object with a relative position in the product.
- a part has a geometry.
- a product is a subassembly.
- a subassembly consists of parts and other subassemblies.
- a connection exists between two parts.
- a part can have multiple connections.

Obviously, figure 3.6 is semantically wrong: the product shown here is not a class but a certain product, which contains an enumerated number of parts. The number of parts would have influence on the model of class 'product'.

The correct semantic representation of the part-product relation is shown in figure 3.7: a part has a product. However, 'product' must have its own properties such as its name, so a tree of attributes must be attached to product as well.

Next, figure 3.8 shows a part having a geometry in the form of a physical object (see explanation in section 3.4.1 on page 30). However, a part is a physical object, so figure 3.9 shows the relation between part and object as a generalization. This cannot be correct however, because a specific part refers to one object, it does not simply copy or inherit the attributes.

So an object must be an attribute to part as well, along with the position in the product. Finally figure 3.10 shows this construct: there is nothing left of the intuitive hierarchy



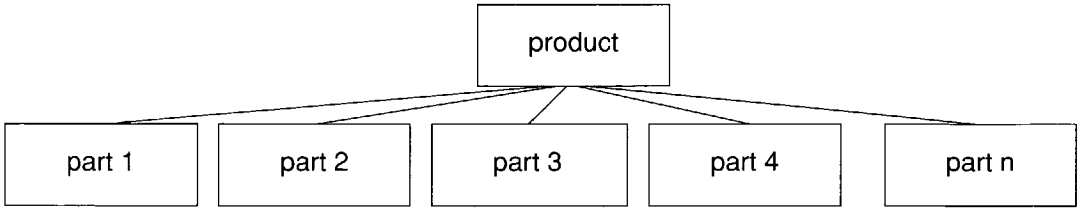


figure 3.6

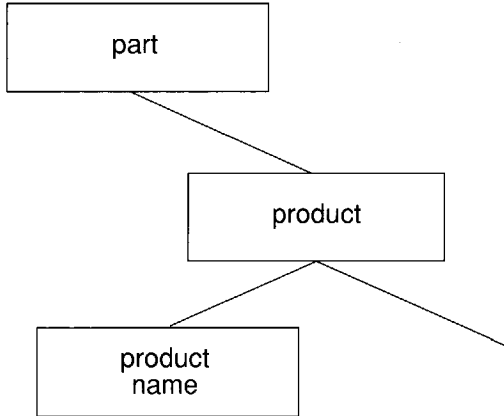


figure 3.7

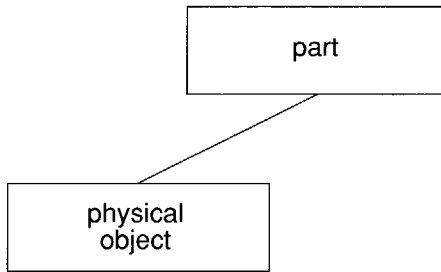


figure 3.9

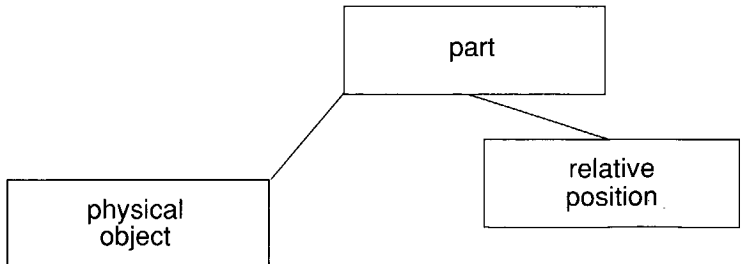


figure 3.8

of product-part-object, but this is the correct semantic model of the relationships.

Now the concept of subassembly must be built in. The type subassembly has relations with part and with itself: a subassembly consists of entities which can be subassemblies again or parts. The product is also a subassembly. Since a number of parts belong to the same subassembly, the same relation as between product and part must exist. The figure 3.11 shows how this works out: subassembly is an attribute to part and an attribute to itself. But: figure 3.12 also holds, since product is a special case of a subassembly, while a subassembly belongs to a product. In that case, it might be better to have a part refer to its product via the subassembly it belongs to, no matter whether this is a real subassembly or actually the product itself. This yields figure 3.13.

The  $n : m$  relation between object and product could be modelled by recognizing the entity 'part'. The same technique holds when modelling the relations between edges, points and surfaces. A link between one edge and one surface needs to be made explicit by a new entity, say, **surface-edge**. An edge can be thought of as a number of lines, say **edgeline** between *two* points. In fact the edge can be a polyedge: a (bent) edge built up of two or more vertices. The figure 3.14 shows the linkage between point, edge, surface and physical object. Finally, figure 3.15 combines all and shows all main objects of the productmodel and their relationships. Note that the object's attributes are not shown and that there are many auxiliary objects to describe the properties of the main entities (for example, a part has a position, which consists of the object 'frame', elaborated later, in the next chapter). The description of the connection also contains motion description, which is a complicated model by itself, but not elaborated in this chapter.

Discussions on engineering databases sometimes refer to the need to have a changing *database structure* in order to store different products. This must be the result of incorrect modelling. If the proportions in the relations are not recognized (as in the example of figure 3.6), objects themselves are mixed up with the classes they belong to. In that case, dynamic creating and structure changes are necessary to store the items.

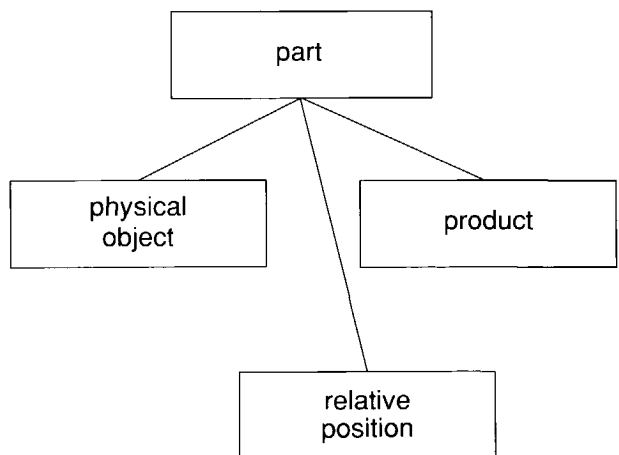


figure 3.10

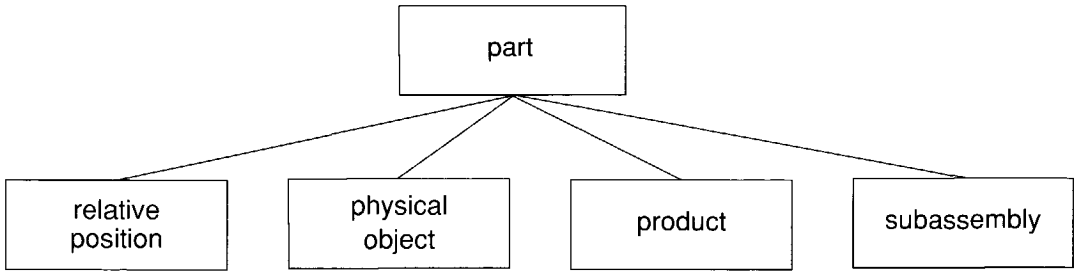


figure 3.11

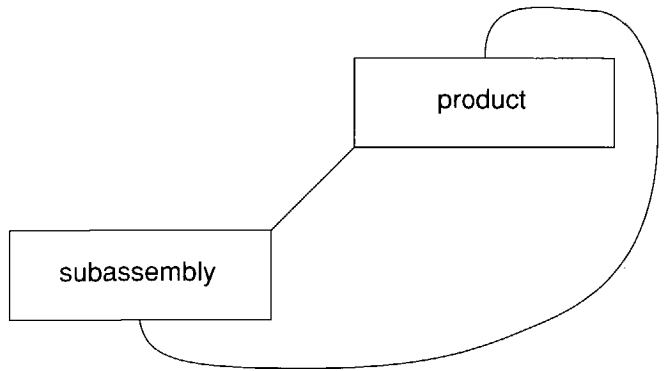


figure 3.12

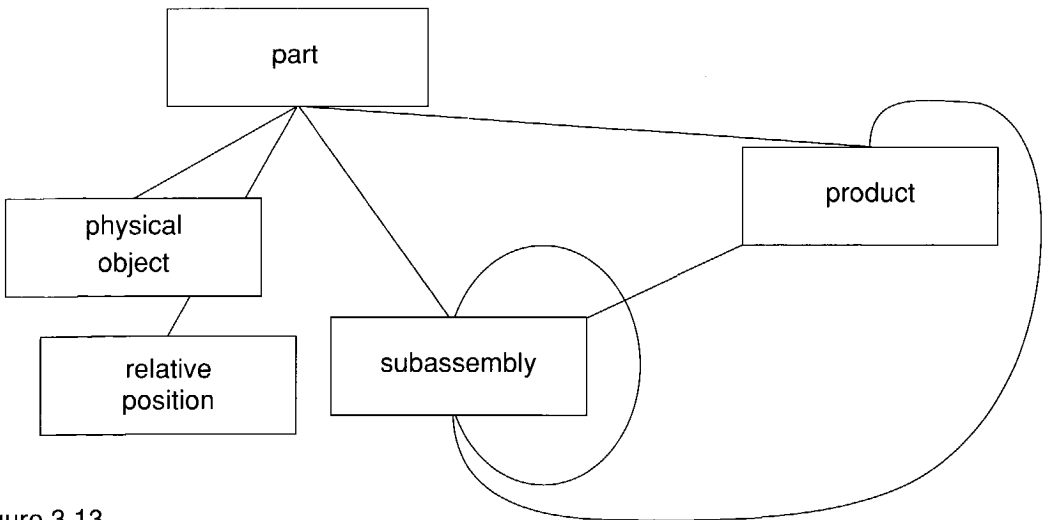


figure 3.13

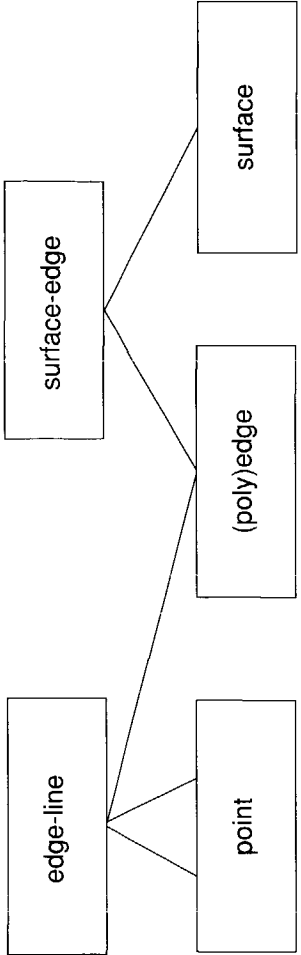


figure 3.14

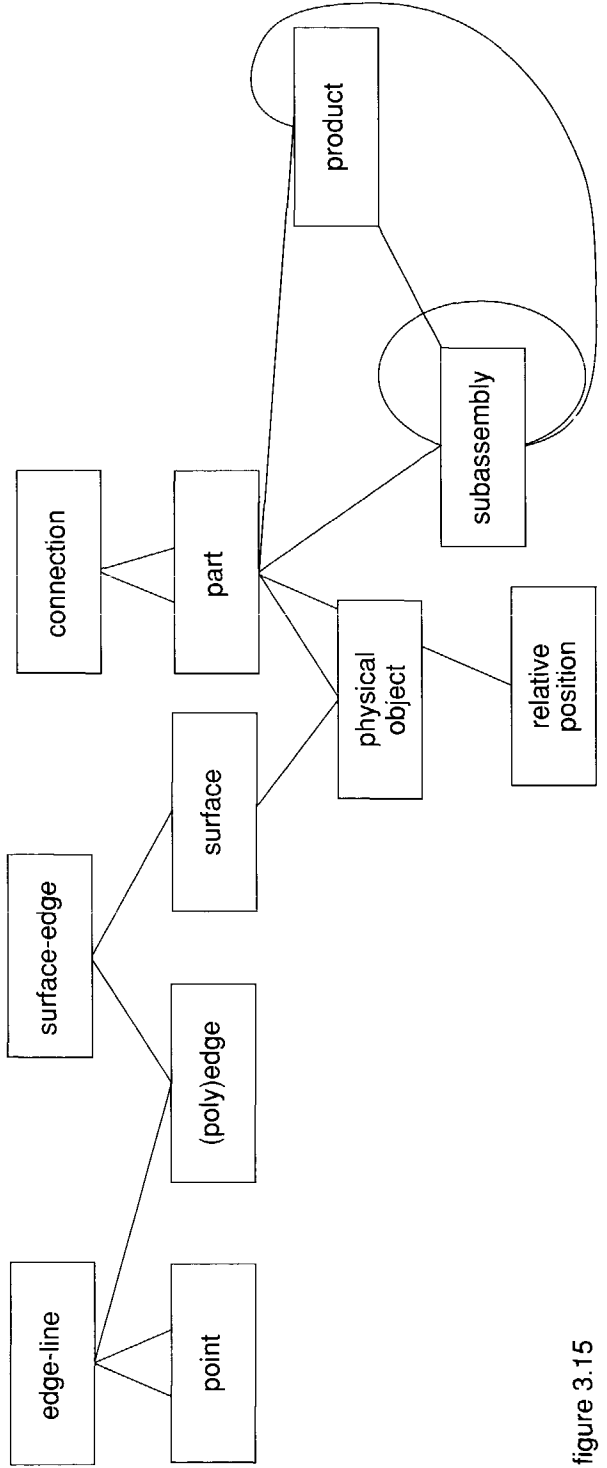


figure 3.15

### 3.4.4 OO versus Relational database and implementation considerations.

Semantic modelling can provide a clear understanding of the entanglements of the model's entities. The implementation of the model into the database (with its own data-definition or datamodeling language) is a matter of translation. OO databases and Relational databases theoretically have no difference in modelling power [McConalogue 91]. However, a semantic model maps better to OO than to the relational model.

The product model has a strong coherence: there are many links, many data objects refer to other objects. In fact, with only one object: the product identification, a whole tree of related data can be found. In a relational database, this requires the extensive use of *keys*, which uniquely identify an object. The key is but an attribute to an instance, the relational system itself does not identify the record by the key. This means more data to keep track of, and extensive searches. Especially the joining of tables, searching for coupled data is time consuming. This is a drawback that makes product modelling very slow with a relational database. In OO systems, this identity is inherent when creating an instance of the object class.

A particular problem form the points-edges relation and edges-surfaces. With its proportion of  $n : m$ , a relational implementation requires three tables: one to hold edges and one for the points, both uniquely defined among physical objects and within the object; the third implements the linkage 'edgeline' (see above). The edges also refer to surfaces, so a reference to a surface must be added too. A typical physical object contains a few thousand points and two to three times that number of edge-point combinations. When the database contains a few hundred objects, tables can get up to millions of records, that must be searched for each relational join. This fact will slow down object reading dramatically: estimated search time will be in the order of hours for a complete product. In the final implementation of the PDM, this is solved by using a **structured file** for the geometric information of each physical object. This is a data file with a pre-defined format, only meant to be read from, directly into memory. With the Brep in a file, the database constraints are relaxed, because a special description can be much more efficient. Because of the use of the structured file that contains points, edges and surfaces and their mutual references, the objects 'surface-edge' and 'edge-line' do not have to exist. The special relation is easily solved by a format of nested data structures.

The main advantage of OO over conventional techniques in databases and programming, is the clarity and fitness for coherent datastructures. There is almost a one-to-one translation possible from the semantic model to the OO database [Weerd 90]. In this respect, the relational model is least fit to express complex relations comprehensively. But OO databases and semantic databases are still under development and not available on a wide scale on different computer architectures. The main subject of this research is assembly, so it is wiser not to use experimental tools for development, if not necessary. Therefore, for this implementation the use of a relational database management system (RDBMS) is chosen, in conjunction with structured files.



## 4 Assembly Modelling

Assembly modelling can be seen as an extension to the product model. This chapter explains why and how it is advantageous to already have information available on the connections between parts in the product, before the actual process plans are created.

The enhancement of the product description with connection characteristics creates the need of a description scheme for the connections between parts of a product. This models the connection in such a way that an efficient and effective assembly process planning is possible. This chapter presents the model and the techniques developed to synthesise the connection model.

### 4.1 An assembly process model

The assembly process has very little theoretical basis. The systematics of assembly are merely a collection of design rules, manufacturing rules of thumb based on experience and a commercial catalogue of connection means. Boothroyd [Boothroyd 82] has done work on the design side, based on manufacturing experiences. Lozano-Perez [Lozano-Perez 83] has done much work on motion of objects in free space and in contact with other objects. Van Brussel [Brussel 84] describes the compliant motion in an accurate way, providing for a comprehensive motion data model.

Heemskerk [Heemskerk 90] uses the sequence *Feed-Grasp-Move-Mount-Check* to express the actions for an assembly and the sequence *batch-product-part-primitive* for assembly process planning (also see chapter 2). Neither model clarifies the assembly process itself. For a closer look at the connection, the *Mount* needs more detailing. The anatomy of the *Mount* is dictated by the connection characteristics. Following is a model of the assembly process and the introduction of definitions, which will be used to express and analyse connections.

The product frame is a virtual position in the product, relative to which all other positions are defined. All motions and positions in the following chapter are relative to the product frame. In other words, the product is the reference frame.

Establishing an assembly, doing the *Mount* manually or automatically, usually comprises of three phases (see figure 4.1 on page 40). First, the *Move* action of object A must end close enough to object B to provide a take-off position for the fine motion: the **approach phase**. The main conditions of the approach phase are dictated by collision avoidance, maybe combined with collision safeguards.

During the next phase, *contact* is established between the two objects: the **contact phase**. This requires a controlled motion that responds to touching the surface: a guarded motion. With force/torque sensors (as is the human operator or a sensor-equipped robot) this contact phase can be performed with a *strategy* to find the correct assembly configuration. For instance, sliding over the surface until a peg 'feels' the hole. The strategy can be pre-arranged; for a number of situations a ready made strategy can be called. Without sensors, this phase is based on fixed positions, passive compliance (in the design and/or the robot) and the accuracy of the robot.

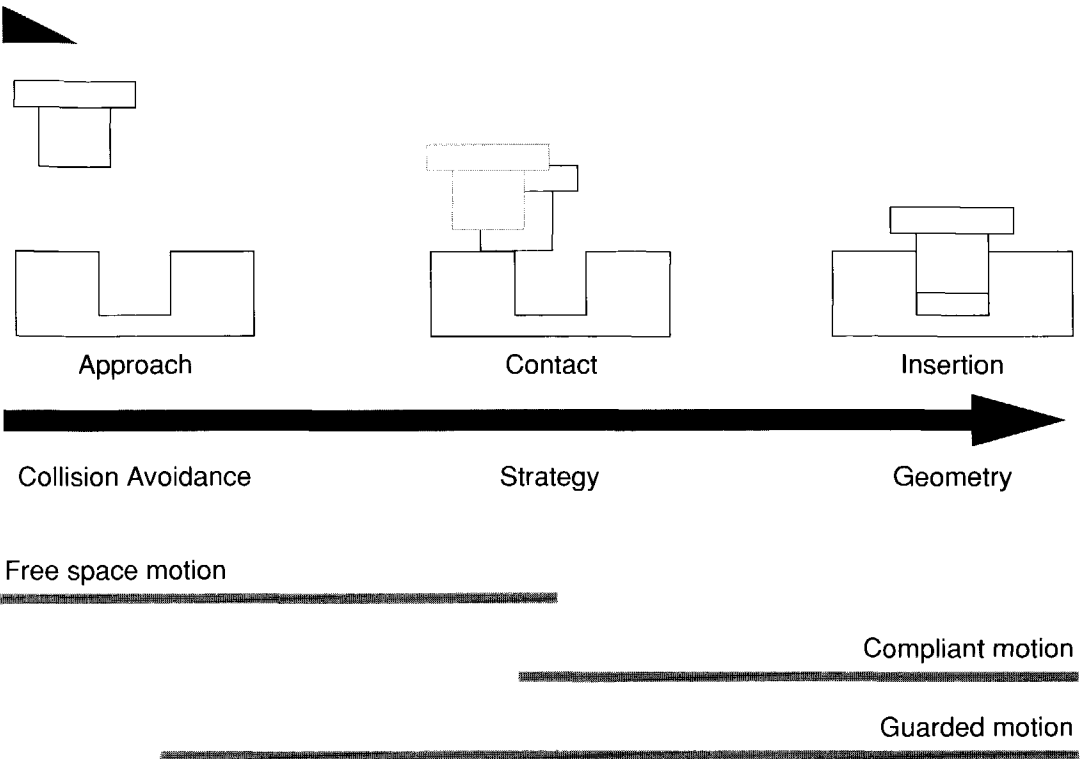


figure 4.1 Phases of an Assembly Action

The **final position** is the position/orientation that is defined by the product model as the assembled position of the part, relative to the product. Referencing figure 4.4 on page 45, this will be discussed in detail later.

The final phase is the actual *insertion* of the part under study into the counter part: the **insertion phase**. This is primarily based on the geometry of the connection, with little freedom. The motion of the insertion is a compliant motion, sliding the one object along the other. For the sequence planning, the most interesting information of this exercise is the point where the insertion starts: that point must be reachable at least in order to make the connection possible. For the selection of a fine motion strategy, information can be used such as connection shape and type, tolerance (slide fit, loose fit, press fit etc.) and material characteristics (frictional resistance).

The position/orientation where the insertion starts, is defined as the **insertion point**. The specification of the insertion point conditions follow later.

The **assembly path** is the description of the translations/rotations required to reach the final position from the insertion point to the final position. The part, when in the insertion phase, should travel along this path. The assembly path itself is not a motion, for it does not yet define speed or contact forces.

This simple model of an assembly action is the starting point for the connection model.



## 4.2 Connection description requirements

The need for a description of a connection arises when separating process planning from the CAD/product modelling.

As explained in chapter 2, the assembly process planning comprises of three stages (see figure 2.4 on page 23). The first stage in assembly process planning is the determination of the assembly sequence [Heemskerk 90]. The sequence planning problem is typically one of finding a way to reduce the number of solutions: the unrestricted, theoretical number of ways to assemble a product of  $n$  parts is  $n!$ . Hard criteria such as accessibility are not sufficient. One needs rules and heuristics based on product characteristics to reduce the complexity. The product information derivatives used for sequence planning are:

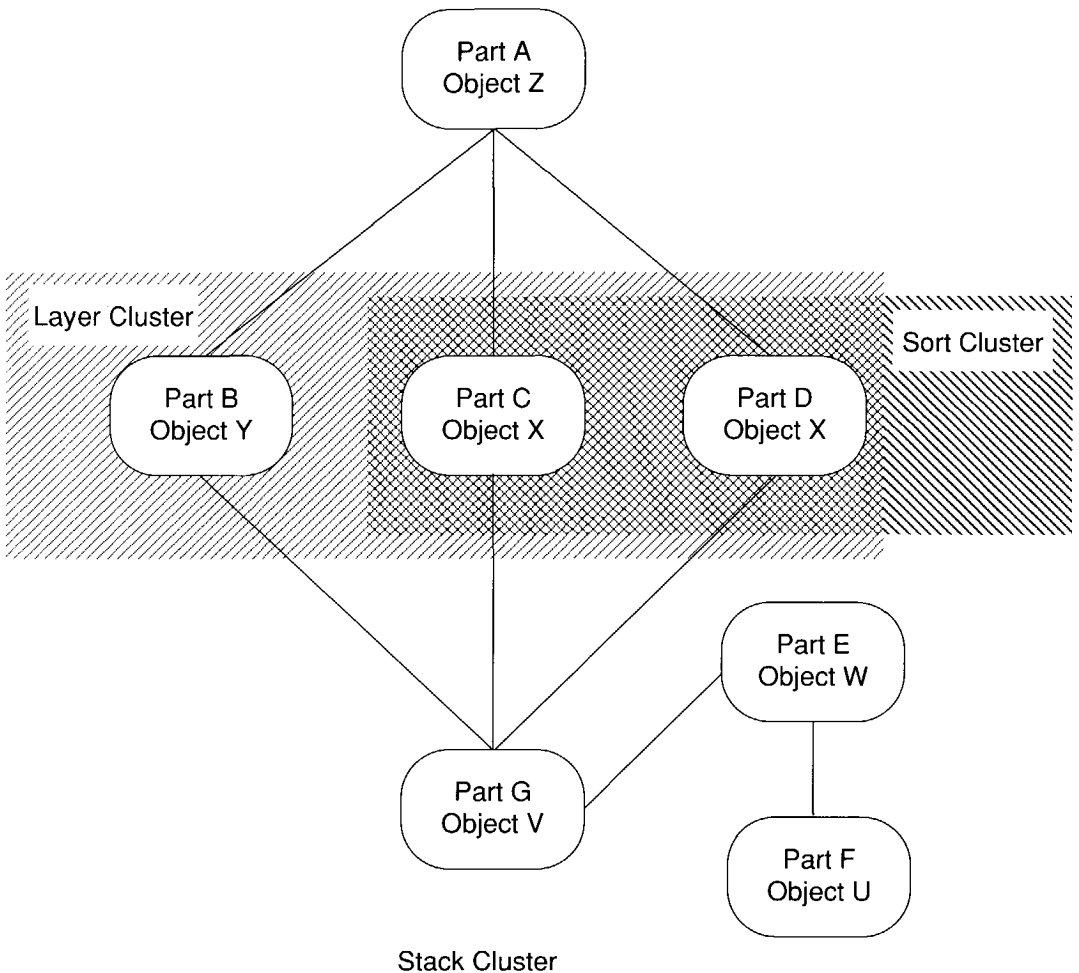


figure 4.2 Example relation network and clusters

- The **relation network**. Every pair of parts in the product that physically makes contact is stored as a **connection**. The relation network is a graph of all connections in the product. See for examples figure 4.2 on page 41 and figure 8.5 on page 96.
- groups (cluster) by sort: parts that are instances of identical objects are candidates to assemble uninterrupted consecutively (no gripper change needed).
- groups by layer and stack: objects that have contacts with the same objects (layer) or are part of a chain of objects (stack) might be assembled consecutively as well. These classifications are patterns in the relation network and can be recognized as such. See the examples in figure 4.2 on page 41.
- stability: the existence of a situation depends on whether parts will be kept in place. This must be derived from the geometry and is a very complex exercise [Boneschanscher 88]. Usually simple algorithms are used that ensure stability, but they are not capable of examining the limits or extremes.
- accessibility: a part can only be assembled if the assembly path to its final position is free in a given situation. Especially the approach and contact phase are critical. The insertion often concerns only the interference between the two parts being connected. Of course the basic product and part information must be available such as identification and geometry. Groups by sort are easily found in the Bill Of Materials. The next most important information is the relation network: which parts have a connection with eachother.

Especially stability and accessibility require extensive geometric reasoning. It would be very convenient if there is information available in the product model that facilitates evaluation of these criteria at a higher level than pure geometry. For instance, a simple stability check could examine if a connection between parts would fall apart, given the main assembly direction.

The second stage in assembly process planning is task planning. Much research has been done in the field of task level programming [Lieberman 77, Lozano-Perez 83]. A system would keep track of parts in an assembly cell and plan the coarse motions itself, while avoiding obstacles (collision avoidance). Automatic grasp planning is assumed to be done in the fine motion planning. The product model must provide the position to move to. Since this concerns the approach phase of the assembly, this position is the *starting point* for contact/insertion, not the final position.

The last stage is the fine motion planning, which defines the strategy with which to assemble a part in the assembly [Bartman 90]. In this stage, uncertainties in positions of actuator, fixtures and parts are used to derive the best strategy, using the available sensors. The current situation (state) of the assembly is known, so interference with the environment can be determined and used. An outline of the insertion provides an efficient basis for determination of the strategy. This means that an insertion must be known nominally, which is detailed later into a true compliant fine motion. The region of interest (connected parts and contact situation) and the insertion outline can be given by the connection model of the product model.

## Conclusion

The approach taken in this project is to *decouple* the CAD/CAM interface from assembly planning by providing a connection description as an enhancement to the product model. The requirements of the assembly planning therefore are normative.

Additional information on the connection is necessary: from which direction can the part be assembled, where does the insertion start, what is the main assembly direction,

what does the insertion look like. One would not have such information unless some geometric reasoning has been done already. However, the detailed geometric reasoning with uncertainties and environment interaction is not done until the process planning. A connection model is therefore a useful enhancement of the product model, available before the planning stages execute.

The connection model can confine to examining parts two by two, because the sequence is not determined yet. To cope with a given configuration however, the algorithms to analyse and reason with interfering geometries are available to the fine motion planning as well.

## 4.3 The connection model

### 4.3.1 Description schemes alternatives

The connection characteristics that are provided to the planning stages, need to be simple yet complete enough to reason with. From the previous section we learn that the connection model should contain characteristics that describe the assembly action on a relatively high abstraction level. Literature on the subject of assembly planning and modelling describes research projects that usually do not model the connection explicitly. Most projects aim to generate assembly plans directly from the geometry or ask a user to provide this information.

Known description schemes are:

- the compliant motion itself or a subset of motion parameters
- the C-space of the insertion situation
- classification
- assembly features
- freedom matrix

A very detailed model of a connection would be the full **compliant motion**, required to establish the connection (insertion phase). This description does not show any approach towards the final assembly (approach/contact phases), so the planning system lacks necessary information. Also, the system at this time (pre-planning), has no knowledge of the assembly cell and therefore can not compose an assembly plan. Sequence alternative generation, uncertainty reasoning and action scheduling will have great influence on the definitive assembly cell commands (much more than motion alone). This means that most generated motions will be incomplete or inaccurate or both. The compliant motion must be known at the end of the planning as a final result [Weule 89, Ko 87].

Another detailed description would be to provide a **C-space** [Brady 82] description of the insertion problem. A Configuration Space extends every object with the sizes of the moving object, so that the moving object shrinks to a moving point. In a six degree of freedom problem, this means that the obstacles are complex objects in a six-dimensional space. However, this kind of information would be too complex to do sequence planning with, it is meant for (coarse) motion path planning.

A **classification** is the simplest and most widely used method. Most known projects use some sort of classification; solely [Sekiguchi 83, Ko 87] or as an attribute [Weule 89, Zussman 90]. For instance if the connection type is found to be 'round-peg/hole', the appropriate motion strategy can be applied right away. If all connections could be classified, motion planning would be nothing more than a set of macros defined for

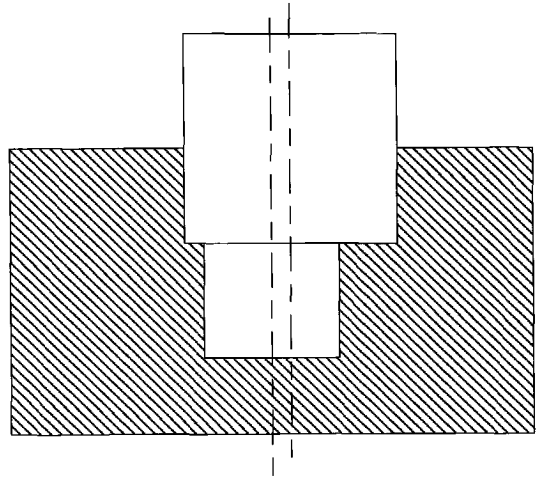


figure 4.3 Eccentric multi-stage peg

each connection in the list. There are however a large number of parameters required for motions and it would be impossible to list all existing (and not yet existing) connections. The classification can be used as an additional feature to activate specific pre-cooked strategies, but can not serve as a connection model by itself.

One of the problems with classification is the inability to model connections made up of several standard connections. For example, see figure 4.3. In order to overcome this problem, Zussman [Zussman 90] models connections with several classifications. These are called **assembly features**. A new problem then is determination of the consequences of these combinations, so the advantage of classification (directly attachable attributes) is lost.

The **freedom matrix** [Jasperse 88, Heemskerk 90] is very limited. It describes the degrees of freedom of the assembled part at the final position (assembled). This will not work for a situation where the motion required to assemble the part is more complex than a straight line.

Concluding, evaluation of existing research learns that there is no good connection description scheme available for the PDM. Since there are no standards in this area either, a proprietary solution will be presented. A compilation of properties of existing schemes, extended with some properties to provide data needed for the applications (such as fine-motion), yields a new connection model. The full compliant motion is too detailed, but the applications would be satisfied with a selection of properties of the compliant motion. The used connection model is therefore a simplified and abstracted version of the assembly motion. The next section discusses its properties.

#### 4.3.2 Properties of the connection model

For the assembly planning, the point where the insertion *begins* is a relevant position which marks the transition from contact to insertion phase. Starting from that insertion point, the insertion motion would be fully determined by the geometry of the assembling parts. This can be modelled by the motion or the path from insertion point to the final position. The **insertion point (IP)** is defined loosely as a position and orientation for a part being disassembled, where the part loses its *direct* contact. The **insertion path** is a sequence of translations and rotations to reach the final position

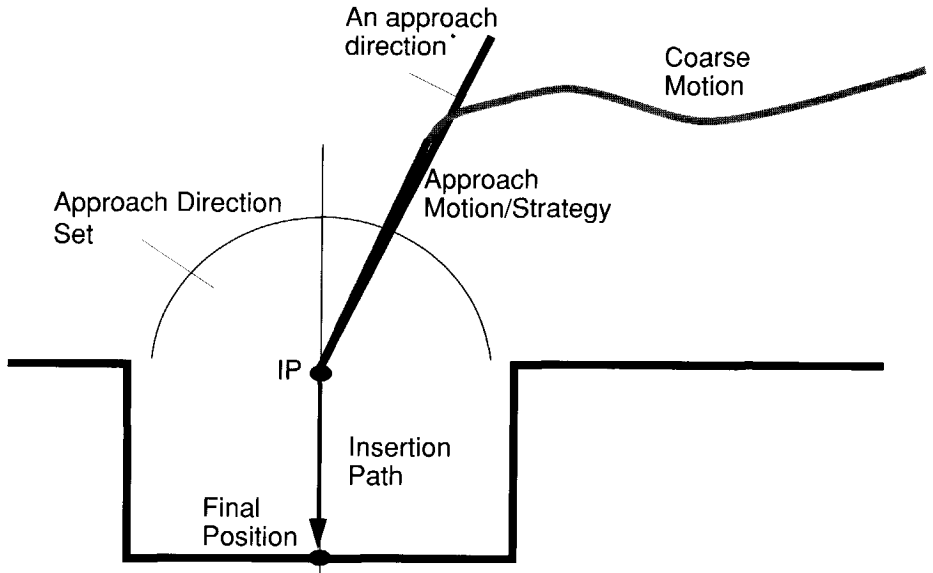


figure 4.4 Connection model definition

from the insertion point. If available, indication of insertion force will be added. The geometry usually does not give such information, but the drawing annotation could provide information on fits.

The approach and contact phase need only be modelled roughly: a few alternatives for the direction from where the insertion point can be reached suffices for assembly planning. Also, as the insertion point is free of contact, the obstacles are assumed to be out of reach; so no rotation needs to be considered. The **approach direction set** is defined as a (set of) translation(s) towards the insertion point that guarantee a collision free path, with regard to the connected parts (see figure 4.4).

With regard to the two parts being examined, the approach direction is collision free and therefore a useful characteristic to the coarse motion planning (definition of the Mount task: 'bring part into final position'), fine motion planning (guaranteed starting situation) and sequence planning (approach must be free: do not assemble parts that block the way). In most cases, the insertion motion directly after the insertion point will yield the most usable approach direction. If the connection is simple, this motion is the only one, directly leading to the final position.

So a good connection model would be to provide the insertion point along with the approach direction set and the insertion path, supplemented with a classification if applicable and available. It is a very brief description with sufficient information for both sequence and motion planning. It is not said that the approach direction is the *only* valid way or even a good way to approach the insertion point. It is just *guaranteed* to be valid. The fine motion planning may have alternative strategies which fit the situation better, especially for the contact phase. The approach direction and insertion point are only aids in the planning process. An important advantage is that it will work for almost any situation, not just peg/hole or plane/plane contacts. A situation where this description fails, is where the insertion should start at a point to which no direct

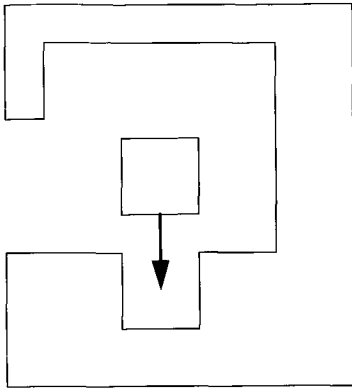


figure 4.5 A potential problem

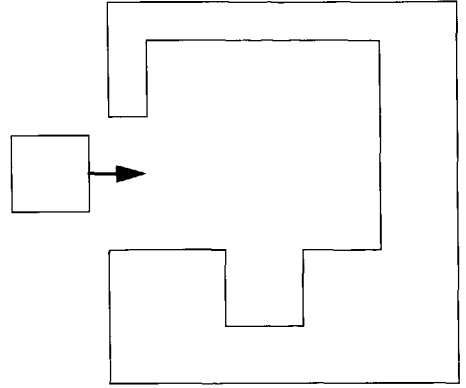


figure 4.6 By changing the Insertion point and direction, the approach route is collision free.

approach is possible or a rotation is required. A solution would be to move the insertion point further away, outside of the part, putting a larger burden on fine-motion planning (see figure 4.5 and figure 4.6 on page 46). This situation is rare however: the shown part is odd-shaped. Because parts are examined two by two (explained in subsection 5.1.1 on page 52), there is no interference with other parts assumed, so normally the approach motion appears to be straightforward.

During fine motion planning however, collision with other parts of the assembly must be taken into account. The next chapter will discuss the concept of *clumps* for this (5.1.1 on page 52).

#### 4.4 Overview of the connection model

The following defines the components of the connection model in more detail. We need two general definitions for the connection model: a frame and a direction set. Because these definitions are not trivial and important for the rest of the thesis as well,

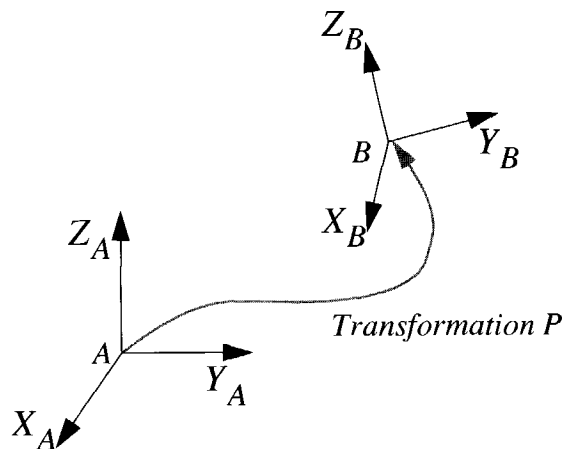


figure 4.7 A transformation of a frame

they are elaborated in subsections 4.4.2 and 4.4.3.

#### 4.4.1 Connection model attributes

The connection model comprises of six main entities:

- A connection identification (The two parts connected).
- A connection classification (if available).
- The Insertion Point (IP).
- The Final Point (FP, actually part of the product model).
- The Insertion path.
- The Approach Direction Set (ADS).

Force and friction information is only presented in the form of attributes (e.g. tolerance fields and material characteristics) to the geometry in the product model. As

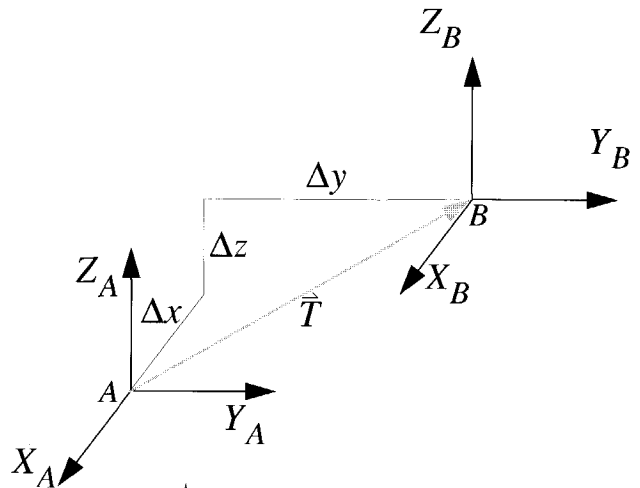


figure 4.8 Translation

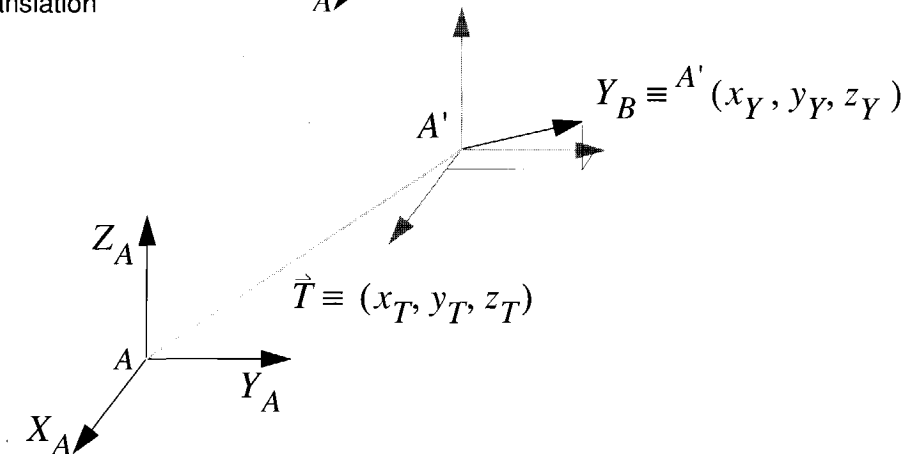


figure 4.9 Rotation Matrix: unity vector notation

a facility to the fine motion planning, the surfaces in contact are marked as well.

The IP and FP are *frames*, meaning that they define position as well as orientation of the part (referenced by its part frame). The frame is already mentioned when discussing the product model, but not yet defined; the next subsection gives the exact definition.

The connection classification can only be provided if the connection appears to be classifiable. It is not mandatory, so will be omitted if the classification fails. The other attributes are sufficient to build a connection.

The insertion path is a finite sequence of translations/rotations, leading from IP to FP. A combination of translation and rotation is only supported in a constant proportion, for instance the pitch of a screw-thread.

The approach direction set gives a number of alternatives to move away from a contact situation, starting from the IP. The description method could be a number of discrete directions, given by points in space or directions, defined in the IP. However, the ADS is in fact a continuous set of direction vectors  $\vec{p}$ , which can be described by a prescription  $P$ :

$$\vec{p} \Rightarrow \{ \vec{p} \in \mathcal{R}^3 \mid P(\vec{p}) \} .$$

Subsection 4.4.3 defines the general way to describe a set of directions.

#### 4.4.2 Description of a frame

The description of a frame in space is always relative to another. So, in fact, it is the description of a motion from the reference frame  $A$  to result frame  $B$ , defined by a **transformation** (a translation and a rotation, see figure 4.7). Of course,  $B$  can also be a reference frame to subsequent transformations. A transformation is not a motion, because the path between  $A$  and  $B$  is not defined, just the end-positions are. Especially in robotics, the description of transformations is important: the links (limbs) of the robot itself represent a transformation, starting from the robot's base. There are several methods to make the description.

A translation in 3D is very simple: independently, the X, Y and Z coordinates can be defined for translations in the respective directions (see figure 4.8). Subsequent translations can be combined just by adding, unless a rotation was involved in one of the steps of the transformation.

The rotation has three degrees of freedom as well. Description of the rotation is possible, using three angles of rotation. They can be defined around axes of the original (mother) frame, or around axes of the transforming frame (subsequent rotations, also called Euler angles). It can be shown that either method suffers from singularities for certain rotation situations. Also, reverse calculation (transformation from  $B$  to  $A$ ) requires trigonometric functions that cause singularities.

In order to avoid singularities, the degrees of freedom could be expressed by special parameters, that depend on the actual rotation axis through a trigonometric function. These parameters are called **quaternions**. There exists a description with three quaternions (Hamilton quaternions), that uses complex numbers, based on so-called Euler parameters. There is also a quaternion description with four parameters, based on the concept of the twisting axis (see section 7.1 on page 81) [Bottema 79, Asea 87]. An advantage of the latter is the ability to describe multiple revolutions in the translation path; another advantage is that combined twistings are simply the product



of two quaternions. A main disadvantage is the complex description, that makes non-automated processing and user interaction extremely difficult.

A very suitable description method is that of the **Rotation Matrix**. Each unity-vector of the new frame  $B$  is expressed in coordinates of the reference frame  $A$  (see figure 4.9). Condition for the length of a unity vector is:  $|\vec{Y}| = 1$ . Repeating this for each of the three unity vectors in a frame, yields nine parameters. Since it is an orthogonal frame, conditions apply. Together with the three translation coordinates and four dummy parameters, the unity vector coordinates form a  $4 \times 4$  homogenous matrix, holding all parameters necessary to define a transformation, and with pleasant calculation properties:

$$\begin{aligned} \vec{X} &= \vec{Y} \times \vec{Z} & |\vec{X}| &= 1 \\ \vec{Y} &= \vec{Z} \times \vec{X} & |\vec{Y}| &= 1 \\ \vec{Z} &= \vec{X} \times \vec{Y} & |\vec{Z}| &= 1 \end{aligned} \quad R_{P(A \rightarrow B)} = \begin{bmatrix} x_X & x_Y & x_Z & x_T \\ y_X & y_Y & y_Z & y_T \\ z_X & z_Y & z_Z & z_T \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{aligned} R_P^{-1} &= R_P^T \\ R_{PQ} &= R_P \times R_Q \\ A_{\vec{k}} &= R_P \times B_{\vec{k}} \end{aligned}$$

The main disadvantage is that there are twelve parameters for three degrees of freedom, giving redundancy and more storage requirements. The redundancy is specified with the dependencies in the formulae above. Advantages are:

- Easy user interaction: it is clear what each of the twelve parameters stand for.
- Simple subsequent transformation calculation: matrix multiplication.
- Simple calculation of reverse transformation: matrix transpose.
- Simple expressions to calculate twisting axis and quaternions and back.
- Easy geometric calculation with vectors, points and geometric elements, expressed in a local frame.

For research and development of geometry related software prototypes, it is clear that rotation matrices are the best to use; as for production, quaternions are better.

#### 4.4.3 Description of a set of directions

A direction in 3D can be defined by two parameters, the so-called sphere coordinates. A common notation takes the three coordinates of a unit vector (with length 1, like the unit vectors of the frame notation). However, again this is redundant with one parameter and also rather hard to use for a continuum of directions.

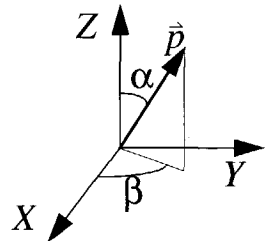



figure 4.10 Definition of a direction



Expressed in a frame, a direction can be defined by two angles (see figure 4.10). The disadvantage of using angles, experienced with transformations, does not apply when expressing a direction: the angles are independent and the singularities are limited. When no restrictions apply, directions could be expressed by two different sets of parameters:  $(\alpha, \beta) \equiv (-\alpha, \beta + \pi)$ , so we demand that  $-\pi < \beta \leq \pi$ . For the singularity  $\alpha = 0$ ,  $\beta$  is meaningless, which means that if  $\alpha$  is close to 0,  $\beta$  can get unstable. However, calculating back from a given direction to the angles will appear to be rare.

These *sphere coordinates* allow description of directions without redundancy and reasonably easy expression of continuous sets (e.g.  $\alpha < \frac{\pi}{2}$  means 'only up',  $\alpha = 0$  means along Z axis).

## 5 Assembly analysis

### 5.1 Introduction

The previous chapter described a way to model connections, so that they can be used as input for the assembly process planning. This chapter discusses the possibilities and requirements to generate that model. A major issue here is the ability to handle a true mechanical engineering design, drawn and annotated as is common to the engineering practice, with few restrictions. First we will discuss the characteristics of the products that the system must be able to handle and other requirements of the system. This gives the context for the assembly analysis problem.

In search for ways to perform the analysis, literature on related research projects is discussed. A common analysis technique, feature recognition, is discussed separately in order to investigate the possibilities of this technique for our problem.

Finally, the approach to the assembly analysis problem is discussed. This is the

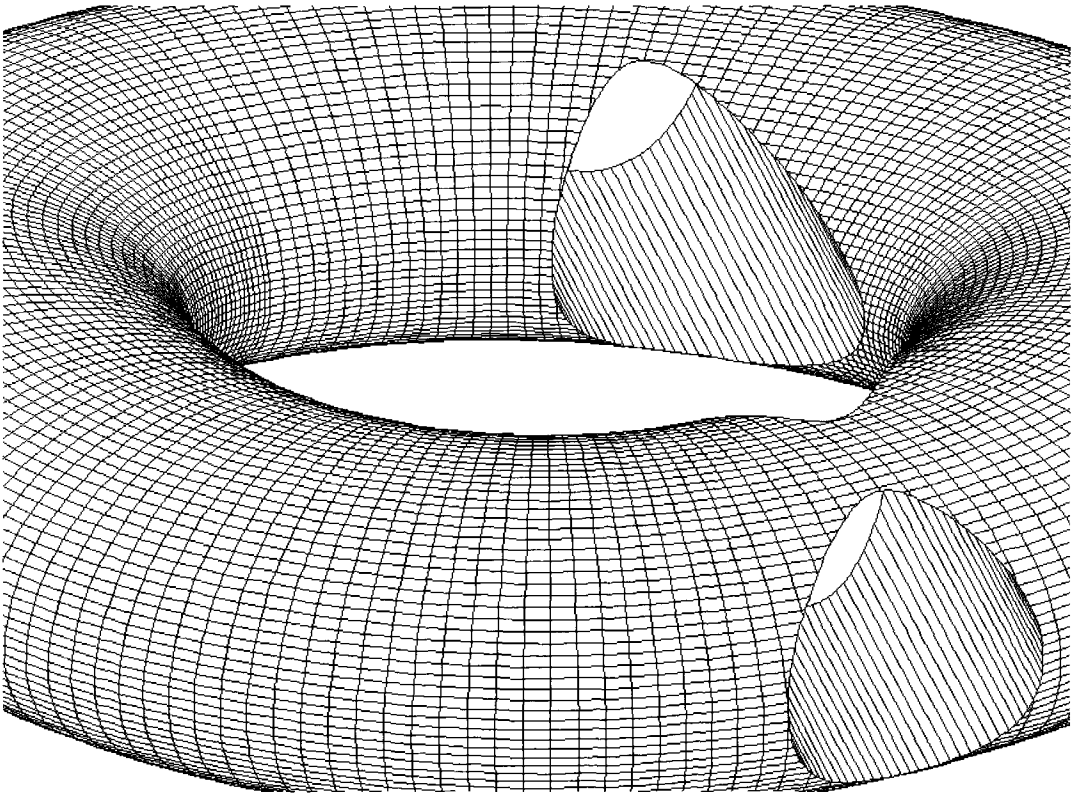


figure 5.1 An object with curved surfaces in tiles and inner profiles

starting point for part two of this thesis, giving a solution for the analysis problem and the implementation of the CAD/CAM connection.

### 5.1.1 Specification of products and connections to be analysed

Essential to the development of a connection analysis algorithm is the specification of the connections it must be able to deal with. In the previous chapter it is stated that classification of a connection does not suffice; a catalogue poses constraints and combinations of (individually classifiable) connections would not be allowed.

For the research project DIAC, a number of specification products have been developed ([Storm 88], see appendix A. on page 131). Since this research project complies with DIAC, the specifications are conformed to and extended ([Donselaar 90] Diac-3 product, see appendix A.). Apart from the most common peg/hole and screw connections, there are several more complicated ones. Some connections require rotations (other than screw) and some require several consecutive motions to assemble. The directions in which to assemble vary, although most are vertically down. Connections that combine individually simple connections exist as well.

The shapes used for the parts of the products, do not conform particularly to any special restriction. This means that common shape elements such as flat surfaces and cylindrical surfaces occur, but also more general, curved surfaces and surfaces with an

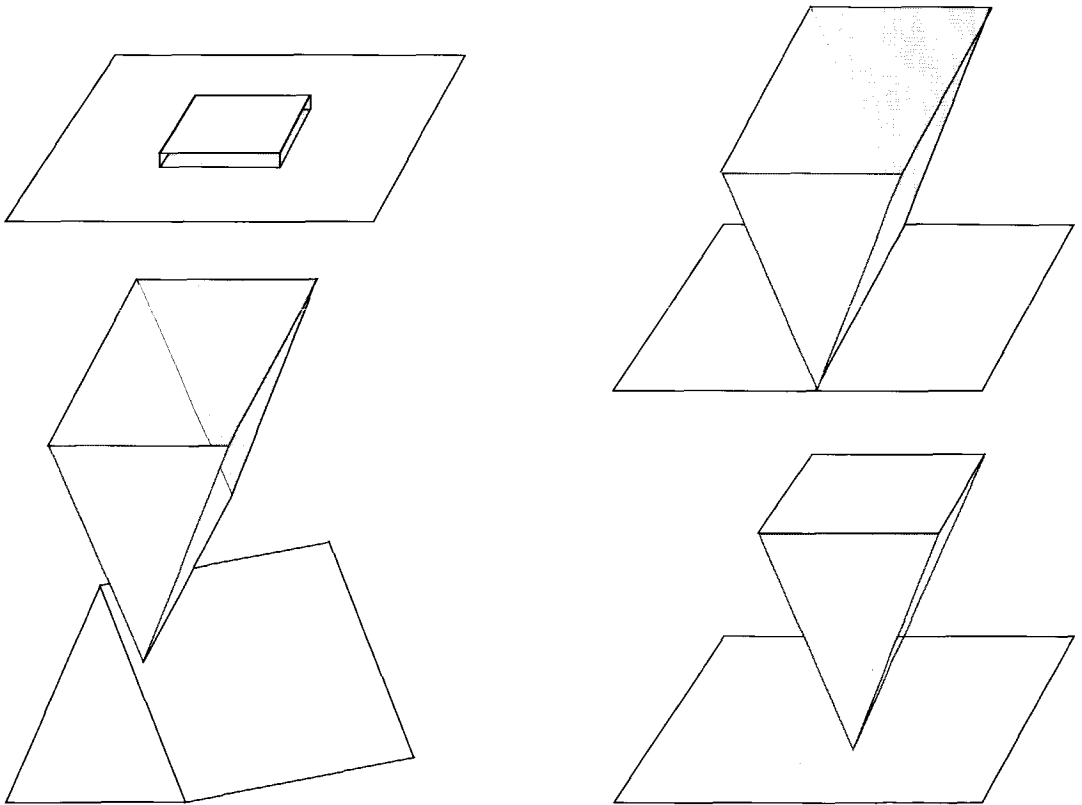


figure 5.2 Contact forms

inner profile are considered (see figure 5.1). Curved surfaces are approximated with a number of **tiles**, small flat surfaces. The geometric elements that are in contact with each other may comprise of any combination: not only surface to (curved) surface, but also surface with edge or vertex, and even edge with edge (see figure 5.2). The combination vertex to vertex would violate the design-for-assembly rules, for it is a very unstable situation. Therefore vertex-to-vertex is not taken into account.

In mechanical engineering, some geometries are not fully specified but abstractly refer to a true geometry by agreement. A common example is a screw. The thread is only annotated on the drawing with a pair of lines and a dimension of the form 'M6'.

Another example is tolerances, which are only shown in the dimensioning (e.g. 10H6

or  $10^{+0.1}_{-0.1}$ ). The geometry consists of elements in the nominal dimension, so the tolerance information is lost if not kept in special attributes. This concept is referred to as incomplete geometry.

Parametric objects carry parameters to specify geometry (In this context, *object* is a logical subsystem of the geometry, but therefore part of a physical object). Depending on the CAD system, the object call is translated into true geometry (Macro substitution), or the object continues to live as an editable parameterized part of the geometry (Instance creation, see also subsection 6.2.3 on page 70). If the original parametric information is kept, it would be possible to assign manufacturing and assembly data to every parametric object. However, the user may combine primitive features into newly defined compound features [Emmerik 90]. The manufacturing and assembly data would have to be redefined for the new feature, unless there exists a reliable algorithm to automatically determine the effect of the combination.

Especially for derivation of assembly information from mechanical designs, the ability to incorporate incomplete geometry is essential. After all, screws and tolerances usually are used for parts of the geometry that have a function in assembly.

Subassemblies sometimes are used by the designer to group parts and structure the product. The product data model passes the information on the designer subassemblies, for these may be usable as subassemblies in production as well. However, sometimes parts are not introduced to the assembly department as separate objects, but in fact as one. For example, press-fitting may be considered to be manufacturing and performed elsewhere, so the press-fitted peg is one with its counterpart. We will call such siamese twins **clumps**, and they will be handled as one part.

The concept of clumps can also be used to apply the connection modelling algorithm for other compound geometries, such as an intermediate state of the assembly. The set of parts present can be processed as one and the connection analysis will be valid for the current assembly situation.

## Conclusion

An assembly analysis system must be able to handle:

- General solid geometry.
- Rotations along the assembly path and translations in arbitrary directions.
- General contact forms, except vertex-vertex.
- Combinations of assembly features.
- Assembly paths that consist of several consecutive motions.

- Incomplete geometry, in particular dimensioning.
- Clumps of compound geometries.

### 5.1.2 Additional system requirements

Where the previous section discussed the input that the assembly analysis system must be able to handle, other requirements apply as well.

It is very unlikely that an automatic assembly analysis system is able to handle all possible connections and generate a correct assembly path. Also, it may occur that the result is not satisfactory and the designer or planner demands to make changes to the assembly path. Furthermore, the exclusions made in appendix A. on page 131, are in fact frequently used methods: gluing, floppy parts, plastic deformation, elastic deformation (snap-fit). Therefore, the designer and the production-planner must be able to interact with the system and/or the assembly data. Preferably, the connection analysis system should be expandible: add special characteristic connection configurations to be recognized.

The assembly analysis as part of the CAD/CAM connection has been defined as a pre-planning phase, preparing for sequence planning, task planning and fine-motion planning. Especially the fine-motion planning could use several parts of the assembly analysis again. When an assembly sequence has been established, the configuration of parts is known. So a given set of parts may be assembled, while a next is to be added. At this point, a new analysis of the assembly action may be required, with the set of assembled parts regarded as one, in a clump.

Since the system is meant as a pre-planning stage, the main objective is to analyse connections between two parts at once. This poses no restriction since parts can be joined, but this will only be done at request. Initially, the system automatically determines connections two by two.

#### **Conclusion**

An assembly analysis system must:

- Allow designer specials.
- Provide a user interface for intelligent data access.
- Be able to join parts at request (handle temporary part clumps).

## 5.2 Related Research in assembly analysis

Several research projects are concerned with automatic assembly planning. However, most proposed systems do not interface to a CAD system, but assume assembly information available or ask a human operator to enter the data concerning the assembly. A common approach is to use the available CAD data (a basic product model with geometry) and make a classification of the connections based on simple heuristics. The focus of most related projects is either determination of the assembly sequence or path-planning/fine motion-planning.

### 5.2.1 Collision avoidance and path finding

The systems based on configuration space [Lozano-Perez 83, Mazon 89] are task planners and path planners, aiming to solve the general case of an object moving through a space with obstacles. Path planners based on other principles such as the bubble technique [Verwer 91] are also solutions to the general case of free space motion.

Generation of an assembly path by trial and error can be done with a collision detection algorithm [Mazon 89, Krishnan 91]. However, first one must find a direction to move in. This could be generated at random or with an educated guess, but it is not certain that a solution will be found. Especially when there are several discrete directions to move in (all of which must be found!), trial and error with collision check does not yield proper results.

A collision detection or interference algorithm can be used to determine the initial conditions of the contacts. Also, when a direction to move in is found, the algorithm can determine to what length this path can be followed (necessary if the assembly path is not a single translation).

Concluding, to some extent these techniques are applicable for (part of) the connection analysis (see chapter 7 and 8). As a solution to assembly path finding they fail, when contacts are involved and friction applies. Very loose fits and directly accessible surface-to-surface connections could be analysed.

### 5.2.2 Combination of contact conditions

All following assembly analysis systems use some kind of contact combination method, that is based on the discrete directions along main axes. Rotations are only handled if they are part of a screw-fit. Of course this results in simple combination algorithms, for these directions are mutually exclusive (orthogonal, so independent). No system handles an assembly path that consists of more than one translation.

One method [Sekiguchi 83] classifies contacts between parts and determines from combinations which part can be disassembled from the assembly. For all three main axes, relations between parts are determined and thus can be analysed which part obstructs others, resulting in an assembly sequence condition (which is the main objective of the project described). The classification is derived directly from attributes in the drawing, such as tolerance annotation and plane contact (it is not clear whether this is done manually).

Combination of surface contacts, not just between parts, provides more detail [Ko 87]. The designer has to specify manually which surfaces are in contact and what kind of contact it is (against, fit, tight-fit, etc.). The combination method resembles that of the previous project.

KOMPASS [Weule 89] determines the contact surfaces automatically, allowing a clearance that is still to be considered a contact as well. In addition, the system allows (manual) specification of special connections, including deformation during assembly and tight fits.

The last project [Shpitalni 89, Zussman 90] employs graphs to express the connections between objects. The search for contacts is based on comparison of edges and vertices of both objects. A contact is combined into an assembly feature according to rules. Each edge in contact dictates a limited number of directions to move in. The intersection of these directions yield the set of allowed directions, one of which is taken to be the assembly path.

## 5.3 Contact finding

The first step in assembly analysis is to find the geometric elements that are in contact. The contact-surfaces, edges and vertices together form a contact topology (see figure 5.3). A small clearance between the elements must be allowed, for the model has small

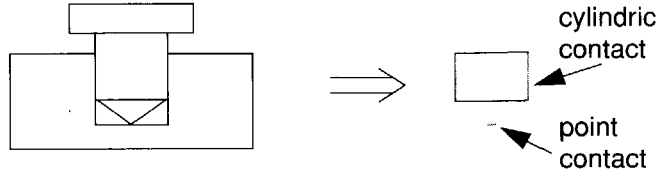


figure 5.3 Contact topology

discrepancies and a near-contact can pose the same effect as a normal contact. In figure 5.4, a number of contact situations are drawn that must be recognized correctly.

Methods for finding surfaces and other geometric elements in contact are:

- volumetric enumeration and comparison
- intersection calculation (mathematical surface/edge equations)
- edge and vertex comparison
- Configuration space calculation
- bubble comparison

Volumetric enumeration divides the space in *voxels* (3D pixels), and keeps record of the ones occupied by an object. Adjacent voxels occupied by different objects indicate a contact. Volumetric enumeration has the well known disadvantages, resolution errors and/or memory usage; advantage is the simplicity. The resolution is a problem for small contacts (such as vertex to surface).

Theoretically, intersection calculation is exact and can detect all kinds of contacts. However, the algorithm can be very complex and it can still be confused by small modelling errors. It is also very difficult to recognize the examples drawn in figure 5.4, where a vertex touches in the middle of a surface. The same holds for edge and vertex comparison as contact finding technique.

Configuration space is very usable to find out whether there is contact at all, but it does not identify the elements in contact. Also, it is not able to handle parametric geometry.

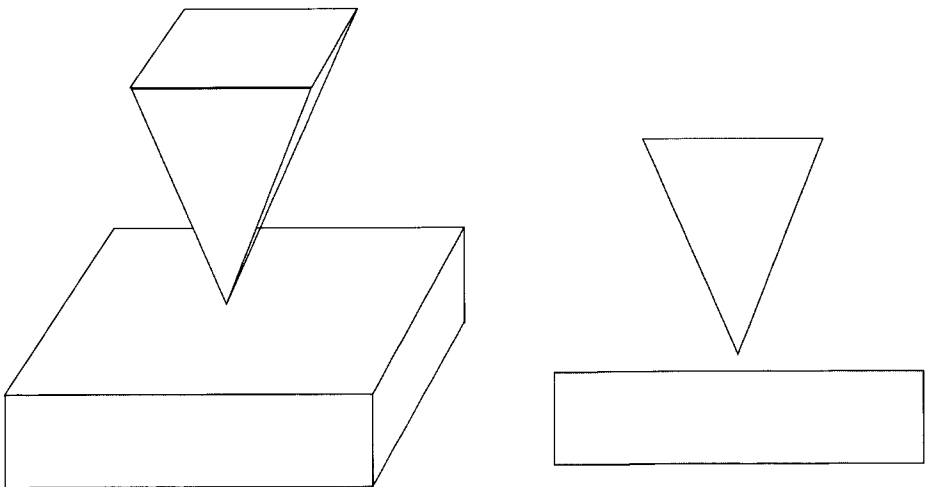


figure 5.4 The contact situation vertex-surface



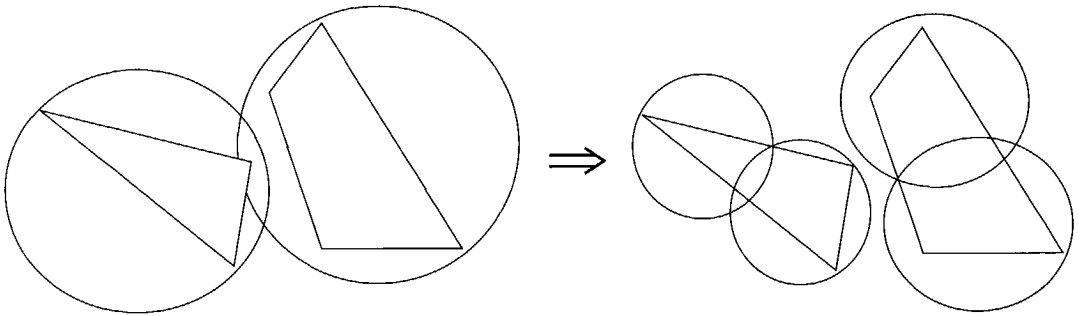


figure 5.5 Bubble collision check. left: initial bubbles around objects intersect. right: after splitting the conclusion 'non colliding' is drawn

The bubble technique [Verwer 91, Bouts 90] is efficient and not sensitive to small errors and clearances (see figure 5.5). Disadvantages are the memory usage, for the resolution of bubbles must be better than the clearance, and the time consuming comparison of all bubbles of two objects. However, this technique offers the best starting point for recognizing complex situations. Therefore, the algorithm described in chapter 8 inherits the basic idea of the bubble technique: distance check of points on the two objects.

## 5.4 Feature recognition

Many systems that interpret CAD information are based on **features**. A feature is an aspect of the geometry that is considered interesting to identify. An example is a hole or a slot. Systems that aim to produce manufacturing data define features concerning certain machining actions: a hole can be manufactured with a drill.

This section will discuss the feature concept and the usability for assembly planning.

### 5.4.1 Feature based design

In order to avoid having to recognize features, one could adapt the CAD system, so that it works with features as design elements (design features). This is referred to as feature based design. If the designer is to specify process related data as well, it is called concurrent engineering [Cutkosky 89]. As discussed before in section 2.6 on page 21, this concept is considered not capable for the general design, process planning and manufacturing practice. The discrepancies between design features and

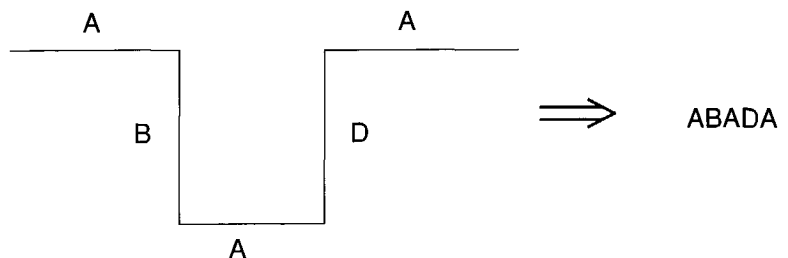


figure 5.6 Syntactic pattern recognition: grammar of a flat bottom hole

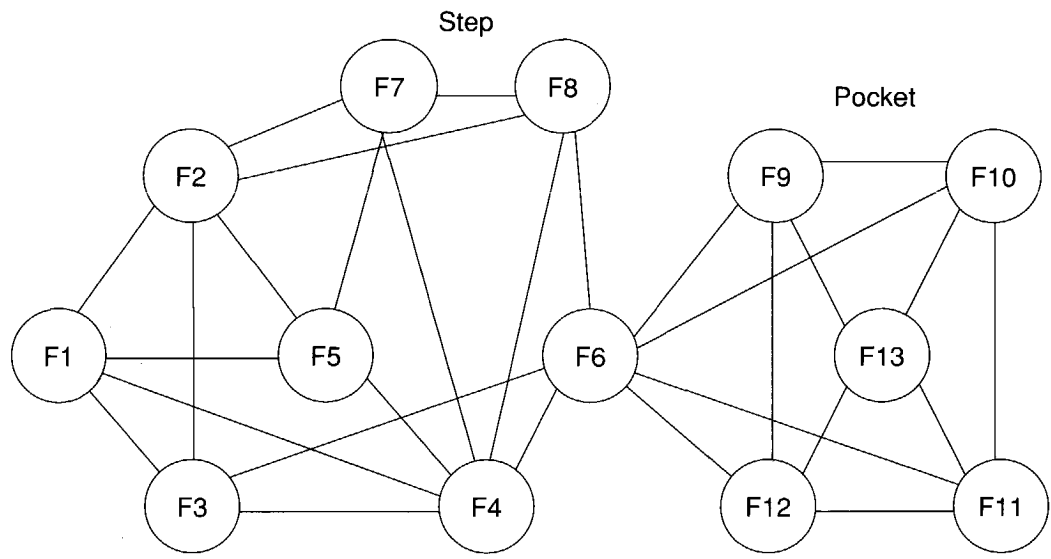
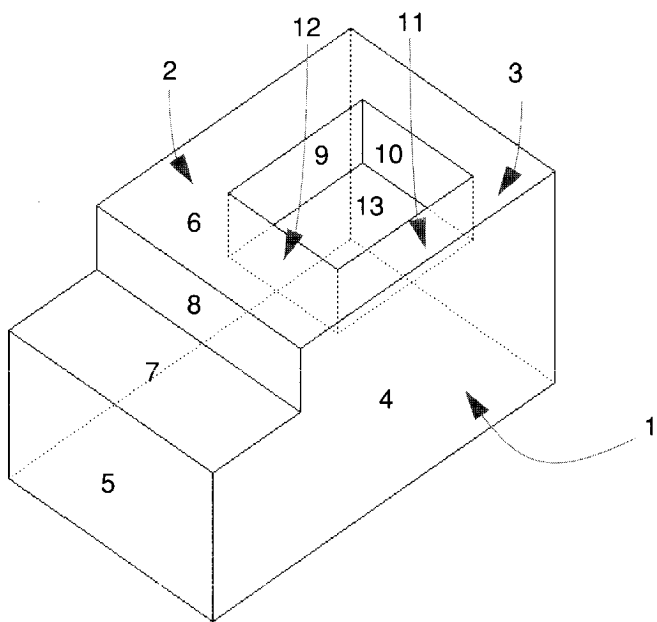


figure 5.7 Attributed Adjacency Graph of an object

manufacturing features cause problems, but the integration of assembly features and concurrent assembly process design as well leads to logical impossibilities [Veltheer 89].

#### 5.4.2 Syntactic pattern recognition

Syntactic pattern recognition [Rong-Kwei 88] is a technique that uses sequences of geometric elements to determine certain features. The ordered geometric elements make up the profile of the part. This will only work in two dimensional drawings that can be decomposed into such geometric elements as 'edge up', 'arc left', etc. The sequence of elements, represented by a string of codes, can be searched for patterns that match a feature definition ('Grammar'). For example, a feature 'Flat bottom hole' (see figure 5.6), can be recognized as a substring in the string representing the object.

#### 5.4.3 Topological pattern recognition in 3D

A derived technique in three dimensional drawings concerns the recognition of configurations in a three dimensional representation. The recognition takes place on the basis of neighbouring faces, thus creating a network of relations between elements. This network is called Attributed Adjacency Graph [Friederichs 89, Joshi 87], see figure 5.7. Patterns in this AAG (figure 5.8) can be identified as predefined features. "A pattern is described as a series of geometric entities, which have to satisfy geometrical and topological constraints and need to have pre-defined relationships with adjacent entities" [Houten 91].

The configuration of geometric elements can also be used for assembly feature recognition. The set of elements in the geometry of two parts that make up the connection (the contact topology) can be analysed. If the feature 'square hole' was identified as the mating configuration, a classification of the connection could be established.

#### 5.4.4 Features for assembly

Analogous to manufacturing features, an *assembly feature* can be defined; it may

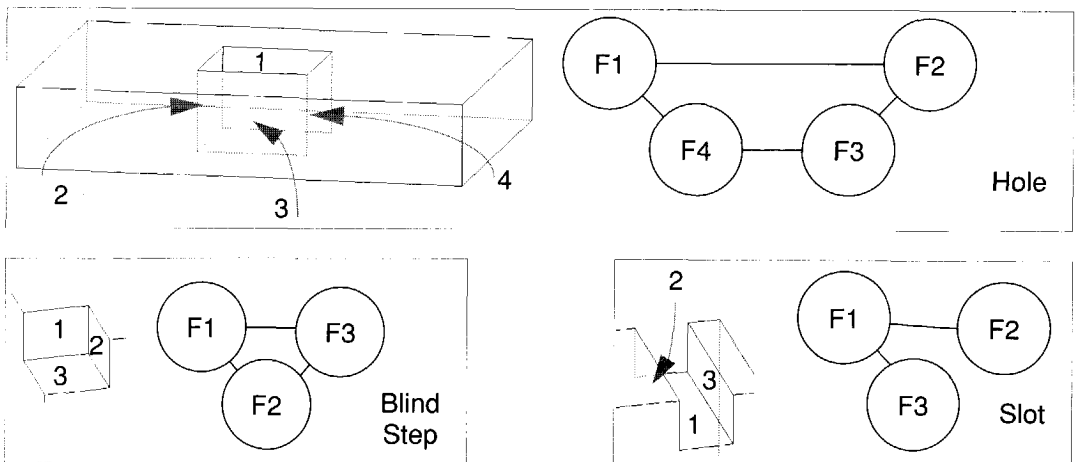


figure 5.8 Examples of AAG features

consist of the configuration of geometries in contact, appended with attributes of that configuration. Recognized and classified assembly features are sometimes referred to as *mating conditions*.

Due to the two dimensional limitation, 2D syntactic pattern recognition is not usable for assembly planning.

3D pattern recognition seems more a promising technique. The concept of direct translation of an assembly feature into a connection classification has been elaborated [Veltheer 89], but not with success. The same problems as with classification of connections (see section 4.3 on page 43) apply: not everything can be classified and combinations of assembly features give unpredictable results. Another problem is that a certain classification, e.g. asymmetric-peg/hole, can have innumerable manifestations, such as square peg or triangle peg or eccentric multi stage peg. A simple chamfer creates a new feature definition, but leading to the same classification. Thus, the recognized contact topology needs to be translated to a higher level of abstraction before processing into assembly information.

## 5.5 The obstruction concept

The previous sections have shown available techniques to tackle the assembly analysis problem. No really satisfactory solution has come to the attention, although parts may be usable. Especially combination of contact configurations *in 6 dimensions* (translation *and* rotation) pose problems. The remainder of this chapter will present a way to solve this combination problem more fundamentally, based on original work done in the context of this project [Friederichs 91]. The mathematical elaboration follows in later chapters.

### 5.5.1 Freedom of movement

Each (part of a) contact surface limits the degrees of freedom of the other object. The object under study experiences the contact with its environment as an *obstruction*, that limits the freedom of movement. The shape of the contact area determines the consequences for the limitations. At every (imaginary) point  $i$  on the contact area, a normal vector  $\vec{n}_i$ , perpendicular to the contact-plane, can be placed. The allowable directions to move in, may make a maximum angle with the normal of  $90^\circ$  (see figure 5.9). Theoretically, a given direction  $\vec{p}$  can be checked against the normals in all points in contact. The condition can be formulated as:

Three problems arise when applying the above for assembly analysis:

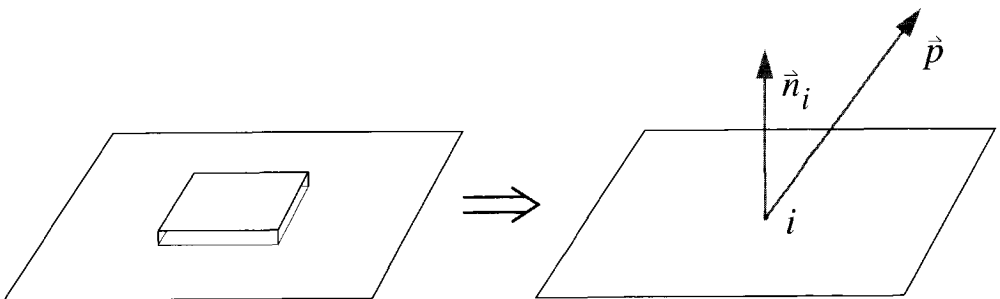


figure 5.9 Normal and direction vectors for contact point  $i$

$$\vec{n}_i \cdot \vec{p} \leq 1$$

- How do we select a direction to move in? The number of directions to be checked is infinite, if we do not want to limit ourselves to a number of discrete directions (e.g. main axes of product frame).
- How do we check the direction validity? The number of points in contact is usually infinite as well.
- How do we handle incomplete geometric information (parameterized geometry, for example the thread of a screw).

The answer to the problem of the infinite number of points is of course to group them into surfaces (on which they lied in the first place). However, the shape of the surface must be modelled somehow in order to incorporate the effect of all possible points. If the number of contact area shapes can be classified without losing essential information, the obstructing effect of this particular shape of contact area can be predetermined. Every contact area that is modelled this way is a geometric feature of connections: the *connection feature* or **obstruction**.

### 5.5.2 Classification of obstructions

A limited number of obstructions can be formulated and are available to reason with. Each obstruction can be equipped with attributes and parameters, that give information on the occurrence in the connection under study. This is knowledge inherent to the assembly analysis system, and can be extended to a set of *obstruction prototypes*. Each obstruction will have to be mapped to one of the prototypes.

The limitation that is yielded by the classification set of obstructions is not severe. If the obstructions are not limited in number and orientation, then most contacts can be modelled directly or via an approximation (such as edge-edge, in subsection 8.3.4 on page 100). The various obstructions can be combined without limitations in position and orientation, to determine the joint effect on the degrees of freedom. By far, most contact situations can be described this way.

The product model is not very detailed on curved surfaces, they are approximated with small flat surfaces and circular forms are recognized as such. Free formed, 3D curved surfaces are rarely used as a *functional* surface for contact with other objects.

Obstructions must be recognized from the geometry, raising the level of abstraction of

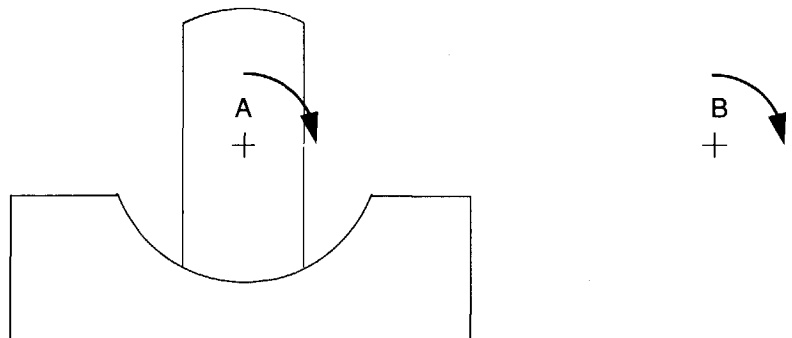


figure 5.10 Cylindrical obstruction: two rotation possibilities, A is more likely

data (interpretation of low-level data). The problem is analogous to feature recognition, but obstructions are of a simpler shape. The obstruction is an abstract description of a contact topology, offering a way to handle abstract information. Thread of a screw can be recognized not from the geometry, but from attributes to the geometry. Therefore, the obstruction classification of parameterized geometry is not obtained by *interpretation* but by *assignment*.

The designer of a mechanical product will create a geometry with a function, the shapes of objects are not arbitrary. The contact area was designed to perform a specific kind of obstruction and this often gives clues to the direction the part was meant to move in. For example, a cylindrical surface allows rotations around many axes, but the one through the centre of the cylinder is a likely candidate (see figure 5.10). These simple heuristics can be stored as attributes to obstructions in the classification, thus helping to solve the problem of the infinite number of directions. Chapter 8 elaborates on the recognition of obstructions by geometry interpretation.

### 5.5.3 Combination of obstructions

As each predefined obstruction allows a set of directions to move in and prevents from moving in other ways, they must be combined for an object with several obstructions. Determining the combination of translations in a certain direction is not difficult. Combining both allowable translations and rotations, however, demands an extensive mathematical exercise (as will become clear in chapter 7).

All possible translations and rotations of an object at a certain position is called its **direction set**. Besides independent translation and rotation, there exists a simultaneous motion: the screw motion. Determining the direction set means joining the partial direction sets of the obstructions the object is subjected to. A mathematical theory for direction set calculus will have to be developed to use the obstruction concept [Friederichs 91]. For illustration, the easiest combination is that of two pure translation direction sets, simply by determining the intersection of the direction sets (see figure 5.11).

#### Usage of the obstruction concept

The concept discussed in this section has led to an algorithm that can determine the main assembly directions of a connection. It works backward from the desired assembled-position to a position where the two parts collide and then restarts the process, searching for the next free direction. The process ends when a path is found where the parts do not meet anymore. This will be the approach direction of the

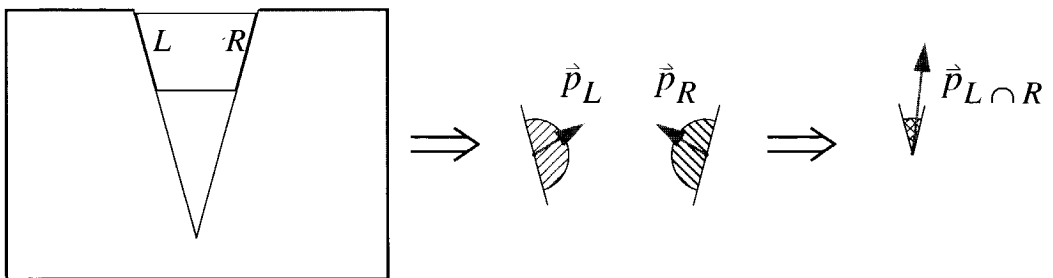


figure 5.11 Combination of two pure translation direction sets L and R

connection model. The insertion point is found by calculating a minimum clearance between the two parts.



PART 2  
THE CAD/CAM SYSTEM



## 6 CAD interfacing and product model synthesis

This first chapter of part 2 describes the CAD/CAM system. This is a rather practical one, whereas following chapters go into the theory developed for the connection analysis. However, logically the CAD interface connection has to be established before any geometrical reasoning can be done.

The chapter starts with an overview of the whole system and an overview of the main data definitions of the system. Next, the CAD connection is elaborated. Finally, the possibilities to access the PDM for the user are discussed.

### 6.1 Overview of subsystems

This section presents an overview of all subsystems of the *Product analysis system* shown in their mutual relations. This creates the context in which the subsystems operate. Section 2.8 on page 24 shows the place of the PDM and the product analysis in the architecture of the FAC. Since the functions can be translated into data-processing subsystems with specifyable interfaces, the system overview is presented in a **Data Flow Diagram** [Yourdon 88]. The figure 6.1 introduces the implementation of the architecture.

#### Generation

The process of generating a product and connection model from a set of CAD sheets<sup>1</sup> roughly takes four steps:

- (1. *Read CAD sheets*) Reading and interpreting the 2D CAD sheets of (sub)assemblies and part sheets. Building a basic product description.
- (2. *Generate geometry*) Reading and interpreting the 3D CAD models of the appropriate objects. Generating properties and matches 2D annotation to 3D entities.
- (3. *Analyse contacts*) Determining contacts between parts. This results in a relation network and a geometric model of the (partial) surfaces in contact (the obstructions).
- (4. *Build connection model*) Combining the obstructions and building the connection characteristics.

The connection analysis (3 and 4) will be elaborated in later chapters. The CAD connection (1 and 2) is especially developed for one particular CAD system, Medusa. A *graphic user interface* (6) allows viewing and editing some of the PDM data.

#### PDM access

The PDM is implemented as a set of tables in a relational database, which is meant to be accessed only by special tools. Building a memory-model of one product is done by a process (5 in the figure). It will combine and structure all known information on a product, so that other applications can use it. In this context, the contact analysis subsystem and the user interface are applications too.

1. As a convention in this thesis, a **sheet** refers to a CAD file with drawing data; a **drawing** is the piece of paper onto which is drawn. Sometimes there is no distinction, however.

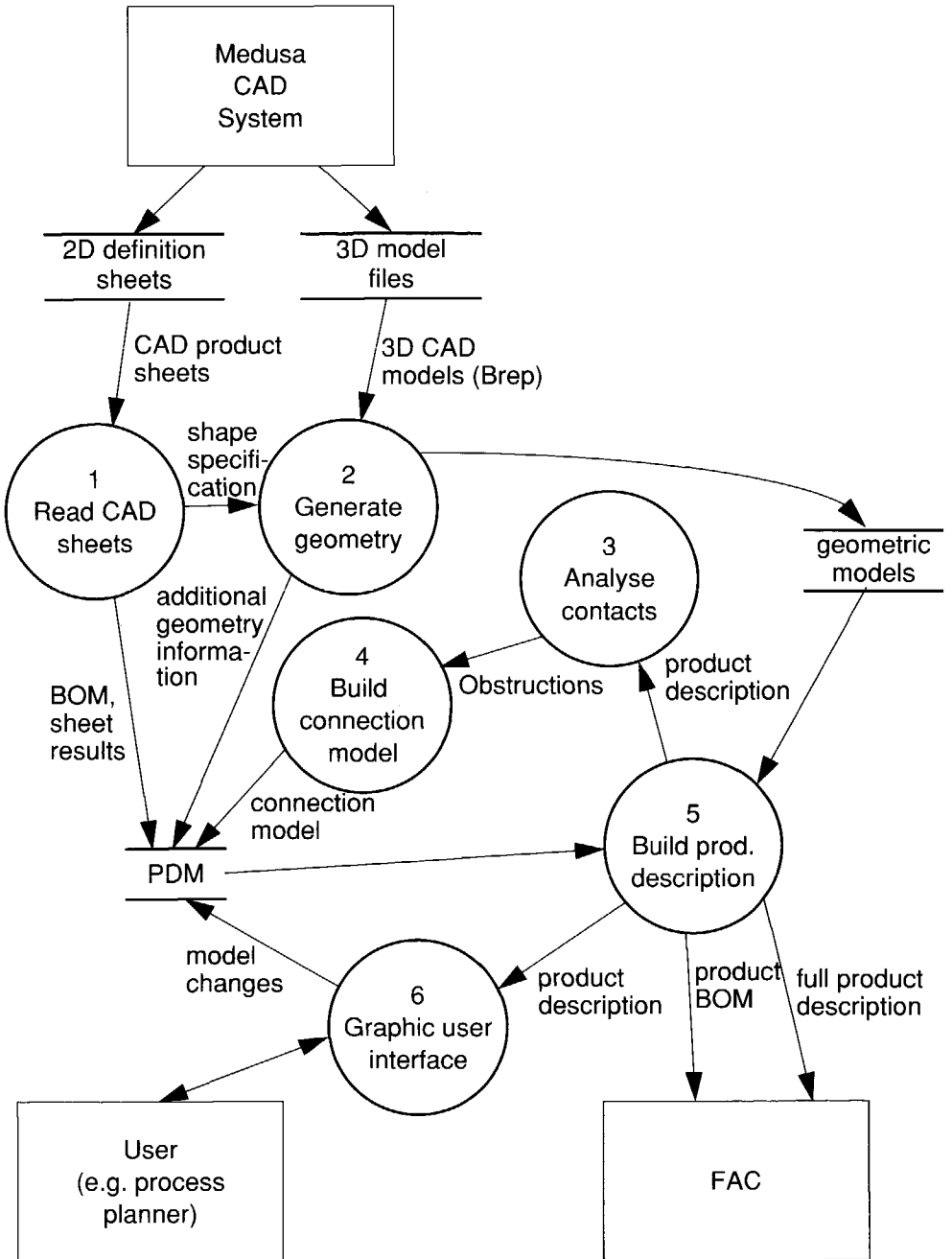


figure 6.1 Process oriented overview of the Product analysis system

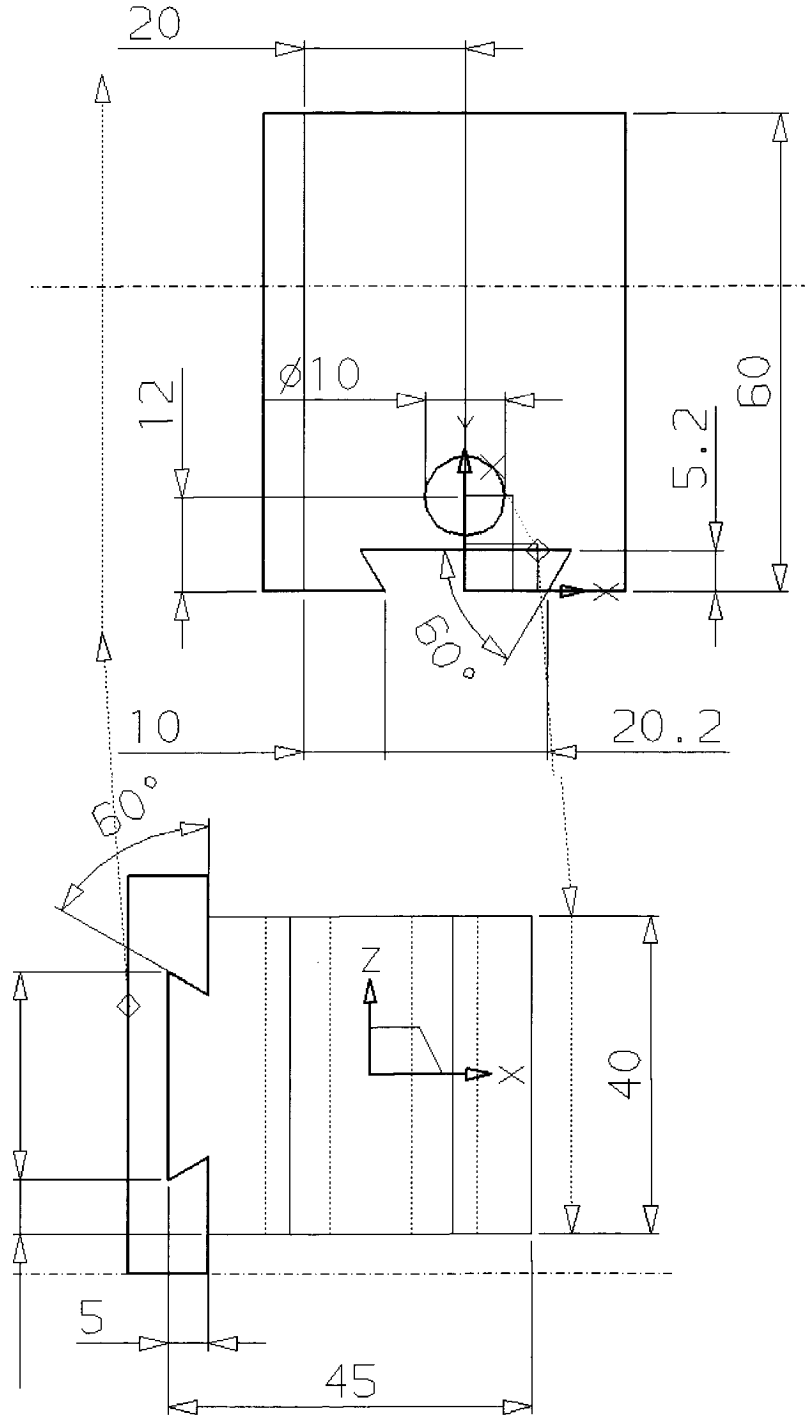
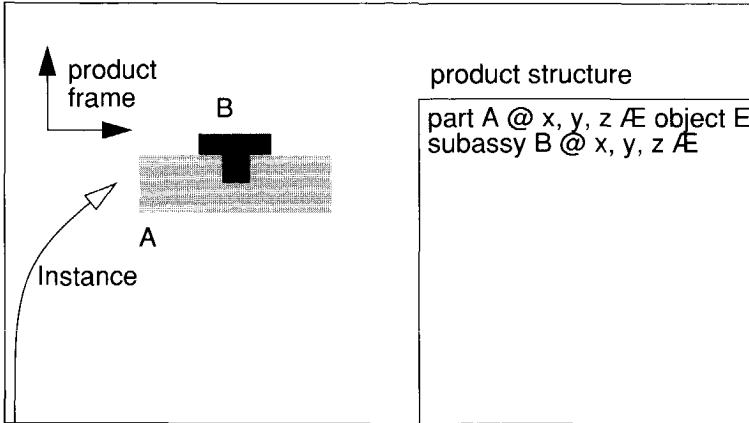


figure 6.2 Example of dimensioning in a mono sheet (3D Medusa definition sheet)



*Assembly sheet, defining the top-level assembly*



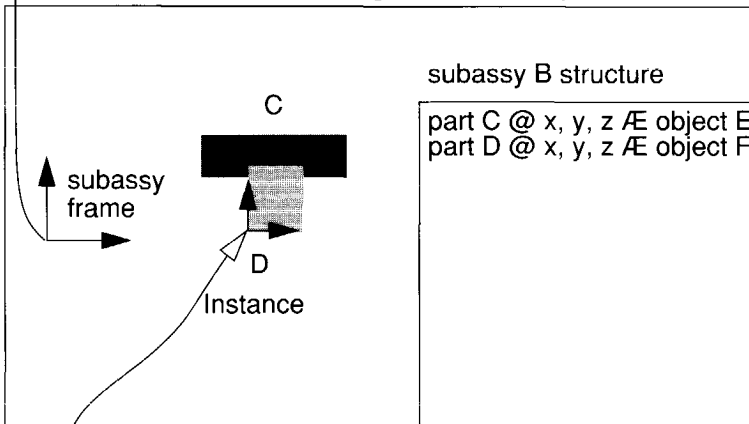
Attributes defined

Product name  
Product comment

Subassembly position

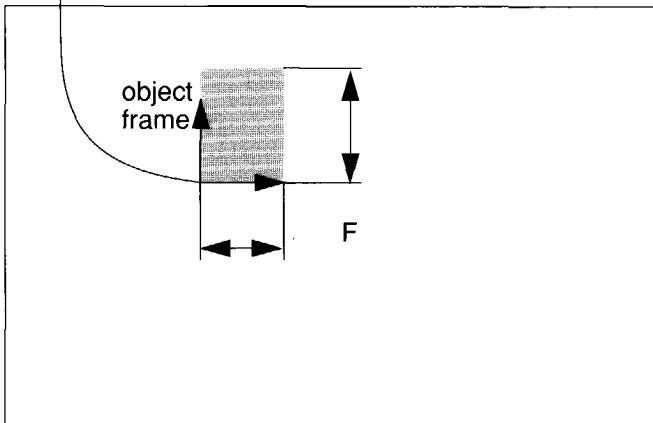
Part name  
Part comment  
Part position  
Part clump

*Assembly sheet, defining a subassembly*



Part name  
Part comment  
Part position  
Part clump

*Mono sheet, defining an object*



Object name  
Object material  
Object comment

Cylinder properties  
Dimensions/tolerances

figure 6.3 Mono sheets and instancing

The complete product data model or specified parts of it, is accessible for applications in the Flexible Assembly Cell through the *Build Product Description* process (5). Besides sequence- and motion planning, the vision systems and collision avoidance / path planning use the product data (see section 2.8 on page 24).

## 6.2 2D and 3D product definition

The product structure is important product data for manufacturing and especially for assembly. A Bill Of Materials will contain some information on parts of a certain kind and how many there are in a product. E.g., a product may contain 20 parts, but only 10 different kinds of parts (also see section 3.4.2 on page 31). The description of a kind of part is called a *physical object*. Each object is defined by a separate single drawing, the **mono sheet** or object sheet.

### 6.2.1 2D: The mechanical engineering drawing

The traditional mechanical engineering drawing contains a wealth of information for the skilled eye. It offers a 3D geometrical definition along with attributes for surfaces, tolerances, assembly information etc. (see figure 6.2 on page 67). The 3D information is created through a number of different views and cuts, along with standardized agreements or rules for interpretation of lines on the drawing. The engineer must have a good understanding of the three-dimensional properties of all parts in order to make such a drawing.

Each part of a product must be specified on a separate sheet, the mono sheet. This defines a physical object in detail: its name, shape, material, dimensions, etc. The designer may make remarks here too; usually they refer to the manufacturing of the part. Some objects are not defined in a mono sheet: they are purchased parts, with standardised properties and taken to be known. A well equipped CAD system will have such catalogue parts in a library, so that they can be called in order to be drawn.

An assembly drawing shows all parts together in place, with cuts to disclose the inside of the product. The assembly drawing contains the Bill of Materials, which shows the use of catalogue parts and parts on mono sheets. Overall dimensions may be annotated here, but also remarks and tolerances concerning the assembly.

### 6.2.2 3D part definition

A three-dimensional model of an object may be produced in several ways, using techniques such as sweeping or CSG. When defining a part for a product however, the 3D model of the object is not the only result: the information that traditionally was available from the mono sheet is also needed, such as a name, material and dimensioning. An engineering system (second phase in design process, see section 2.5) will only need a 3D model and maybe material characteristics, but a product definition needs to be fully detailed (third phase).

An exception is the incomplete specification of geometry such as the thread of a screw. Through special attributes, the exact shape is specified (also see 5.1.1 on page 52). The thread is stored in the database as an attribute to geometric elements, typically a cylinder, of the Brep.

Concluding, the definition of an object that can be used as a part in a product includes mono sheet information as well as a solid model.

### 6.2.3 Product structure and instancing

A 2D CAD system cannot help to make a 2D assembly drawing, for it does not know about visible and invisible aspects. If a mono sheet is copied into an assembly sheet, all lines will remain visible and only the user can determine which lines must be made invisible in order to make a correct drawing. In case of alterations to the originating mono sheets, the system will not be able to automatically make the right modifications to the assembly sheet. A 3D CAD system however would be able to help to make an

Relation name	Attributes	Description / refers to
Connection	partid1 partid2 ctype ads apa ip	FK: partid, refers to a PART FK: partid, refers to a PART FK: ctype, refers to a connection type (classification) NM FK: refers to a direction set FK: refers to an assembly path FK: the insertion point, refers to a frame
Part	partid prodid pname objectid relpos special clump comid	key field, defined internally FK: prodid, refers to the product this part belongs to. an identification name for human interaction NM FK: objectid, refers to specification of the physics FK: frame_id, refers to the relative frame in the product dataset for special device use and instructions. NM identifier for compounding parts NM FK: comment. An optional comment line. NM
Object	objectid shapeid oname matid weight catnum comid cog volume	key field, defined internally FK: shapeid, refers to a describing geometric model an identification name for human interaction NM FK: material that the object is made of. NM the total weight of the object in kg. NM if a standard part, catnum is the catalogue number. NM FK: comment. An optional comment line. NM FK: frame_id. center of gravity and main axes (relative). NM volume in order to determine weight. in mm3. NM
Product	prodid prodname comid	key field, defined internally. an identification name for human interaction NM FK: comment. An optional comment line. NM

Notes: the underlined attribute(s) make up the primary key of the relation mentioned in the first column. FK means Foreign Key and NM is Not Mandatory.

figure 6.4 Definition of the main entities in the relational version of the PDM

assembly sheet. It can be calculated which parts block the visibility of certain features from any viewpoint. Usually the mono sheets (actually only the 3D models) are not copied, but just referred to. In case of modifications to the mono sheet, the system can determine the consequences for the assembly sheet.

This technique of referencing is called **instancing**, see figure 6.3 on page 68. The word *instance* is taken from the OO world (see subsection 3.4.1 on page 30) but has conceptually the same meaning here. An instance uses a frame which represents an object defined in a mono sheet and places the object at a certain position and orientation. Every instance has its own position, but can refer to the same object; each instance of an object is a separate **part**. An assembly sheet combines several objects and may be used by itself in another assembly sheet (subassembly hierarchy).

Therefore, an assembly sheet represents a geometric model as well, but the primitives consist only of predefined (instanced) objects.

### 6.3 Overview of the PDM

The discussion in section 3.4 on page 30 was an analysis of the relations between the entities in the product model. For practical reasons, for the implementation the use of a relational database system is chosen in conjunction with structured files. Therefore, the database definitions can be derived from the relations of the identified items. In a relational implementation, identification numbers have to be attached to entities, in order to refer to the items (e.g. productno, partno). Refer to the table in figure 6.4.

The table **connection** combines two parts, assigning attributes. The properties of the connection such as discussed in section 4.4 on page 46, are implemented as separate entities (e.g. the **IP** is in fact a frame, defining position and orientation with a rotation matrix; see subsection 4.4.2 on page 48). The connection analysis will be discussed in more detail in following chapters. Table **part** contains the identifier **clump**. If several parts in one product have the same clump number, then they are assumed to be compounded. The **comid**, which is present in the tables **object** and **product** as well, allows attaching comments. The CAD sheets can contain remarks on several levels, such as in an assembly sheet, a mono sheet or attached to an object instance (= a part); they are stored as an attribute to the corresponding item.

The item **shape** is used as a link to refer to what is in the structured file, that contains the BRep for the item **object**, which is stored in the database. The format of the file is directly adopted from Medusa, the CAD system that produces it. The shape in the file contains a definition of all surfaces, edges and points used in this geometry. In the definition, these items refer to each other:

shape: surfacelist edgelist pointlist

surface: surfaceid type profiles

profile: edges

edge: edgeid type points

point: pointid (x,y,z)





- commercially available.

The standard mechanical engineering drawing is desired to produce drawings from which the parts can be manufactured. In spite of 3D views, the shop-floor still requires the familiar 2D drawing. Annotations such as dimensions and tolerances must be attached to the geometric element that it refers to. At this time, this is only standardized in a 2D drawing, but there are already CAD systems which are able to place 3D annotation. In order to model an assembled product, the separate parts must be merged into an assembly sheet. The position of each part (in 3D) must be available, and it must be clear which parts are used. For each part in the product, a geometric model must be available.

A software product that is commercially available is supposed to be more stable and contain less bugs. Even so, with a demanding application such as this one, the chosen CAD system showed flaws. Another advantage of using a commercially available (and successful) product is the credibility of the application.

Without going into details, the Medusa CAD system was taken for the project. Medusa is one of the larger CAD systems, widely used in industry. The CAD connection is very specific to Medusa, but many of the problems would have arisen with another system as well. Especially the mix of 2D and 3D capabilities causes inherent problems. See for an example of a Medusa mono sheet, figure 6.5.

## 6.5 Implementation of the CAD connection

This section presents a brief overview of the CAD connection (processes 1, *Read CAD sheets* and 2, *Generate Geometry*, see figure 6.1).

### 6.5.1 Reading CAD sheets

Once all mono sheets and assembly sheets for a product have been created, the product can be entered into the CAD/CAM system. This process starts with the reading of the top-level assembly sheet, which contains the references to subassemblies and parts. The system will follow the references and will eventually have accessed all assembly- and mono sheets that belong to the new product. For each mono sheet, an *object* entry is created and for every instance of this mono sheet, a *part* entry is created. Several products may share mono sheets, the system will detect duplicates. If the product appears to be already existing in the PDM, all sheets will be checked for updates.

The sheets are scanned for interesting elements, which are stored in the PDM. For (sub)assembly sheets these are:

- name of the assembly.
- references of instances (sheet identification).
- position and orientation of each instance.
- explicit names of instances, if any.
- directions regarding clumps (grouping of instances).
- remarks (comments) made by the designer regarding the assembly.

An instance is either a part or a subassembly, depending on the type of sheet found when the sheet reference is followed. The position and orientation of a subassembly is projected onto all instances found in it (positions of parts in a subassembly are relative to the assembly's reference frame). Names of instances may be given to distinguish between identical objects; for example, an object *screw\_M4x20* is used as



figure 6.6 The chord tolerance

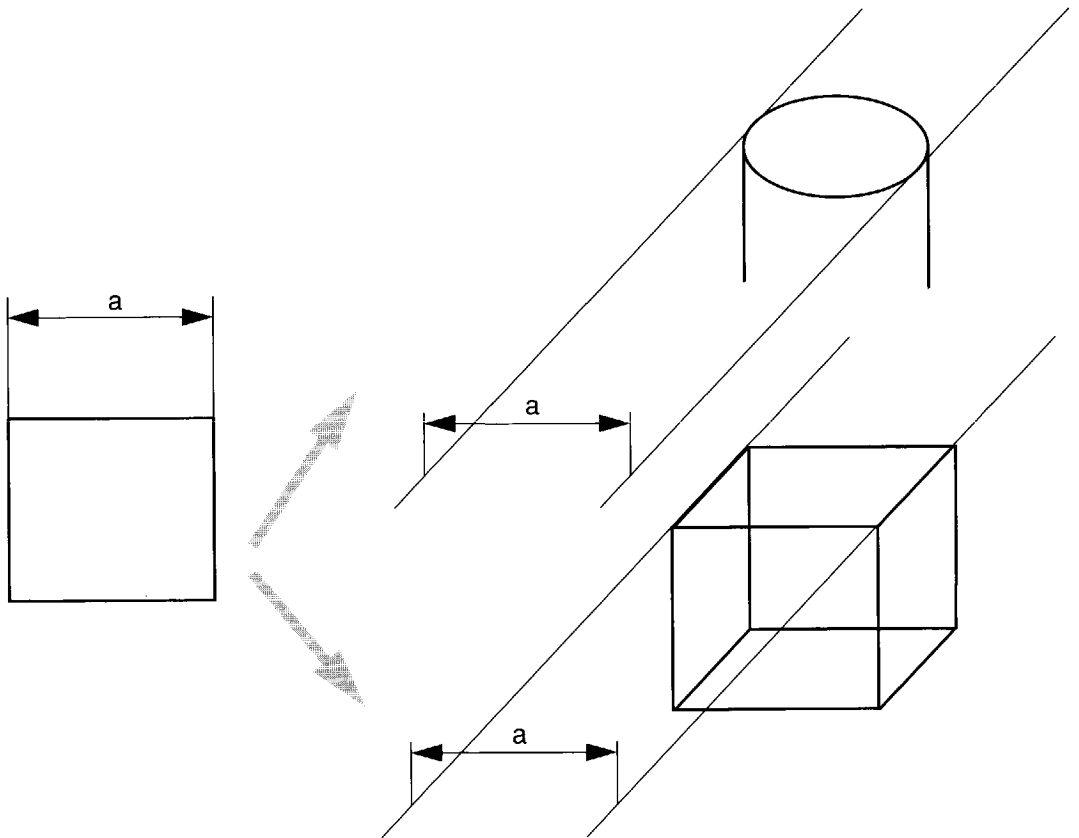
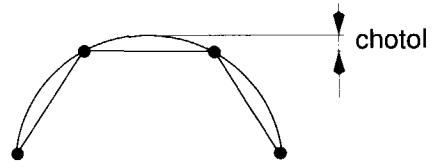


figure 6.7 Mapping of ambiguous 2D dimensioning to 3D reality

a part called *fasten\_screw* and as an *adjustment\_screw*. Clumps are groups of parts that come unseparated to the assembly (also see subsection 5.1.1 on page 52).

The information on the assembly-hierarchy is inherently available and stored when descending through the sheet reference tree. Note that parts in the first place belong to a product, and that the product structure (a part belonging to a subassembly) is just an additional arrangement.

The elements that are searched for in mono sheets are:

- name of the object.
- chord tolerance.
- dimension/tolerance strings and attachment positions in a 2D view.
- circle references.
- remarks (comments) made by the designer regarding the assembly or manufacturing.

The chord tolerance is used when the 3D model is created. Curved surfaces and edges are approximated by flat and straight pieces and the chord tolerance specifies the maximum deviation (see figure 6.6). This value is used as a standard for comparison in geometric calculations.

Circle references point at circular elements in the geometry, that can be used to qualify edges and surfaces in the Brep as circle-shaped and to assign dimensions to geometrical elements. It is stored in auxiliary database tables, to be used when generating the geometry description.

## 6.6 Generating geometry

### **Brep interpretation**

The Medusa CAD system is able to store its internal Brep to a file with the discussed format (section 6.3). It contains definitions of points, edges and profiles/surfaces. As can be seen in figure 6.1 on page 66, the geometric models are stored separately from the PDM. This is because the Brep file does not have to be changed by the system and can be read efficiently. The equivalent of the Brep in a relational model would be very inefficient (also see chapter 3). Additional information that can refer to geometrical elements, is stored in database tables.

### **Assignment of 2D annotation to 3D elements**

The annotation elements such as dimensioning on a mechanical engineering drawing in 2D, refer to features of the actual 3D geometry. This poses an interpretation problem on the processing of sheet items. All dimensions and tolerances are text strings, that are attached to the geometry by leader lines, in a 2D view (see figure 6.7). By projecting the 2D view on the 3D reality, a line is created that must be tangent to the subject element. The line, perpendicular to the 2D view, can be compared to geometrical elements. The dimension text already discloses the fact whether it

concerns a circle or not (e.g. 8H6,  $10^{\frac{+0.1}{-0.2}}$ , or M6), which can help finding the right element in case of ambiguity.

### **3D properties**

The PDM contains some physical properties of the objects:

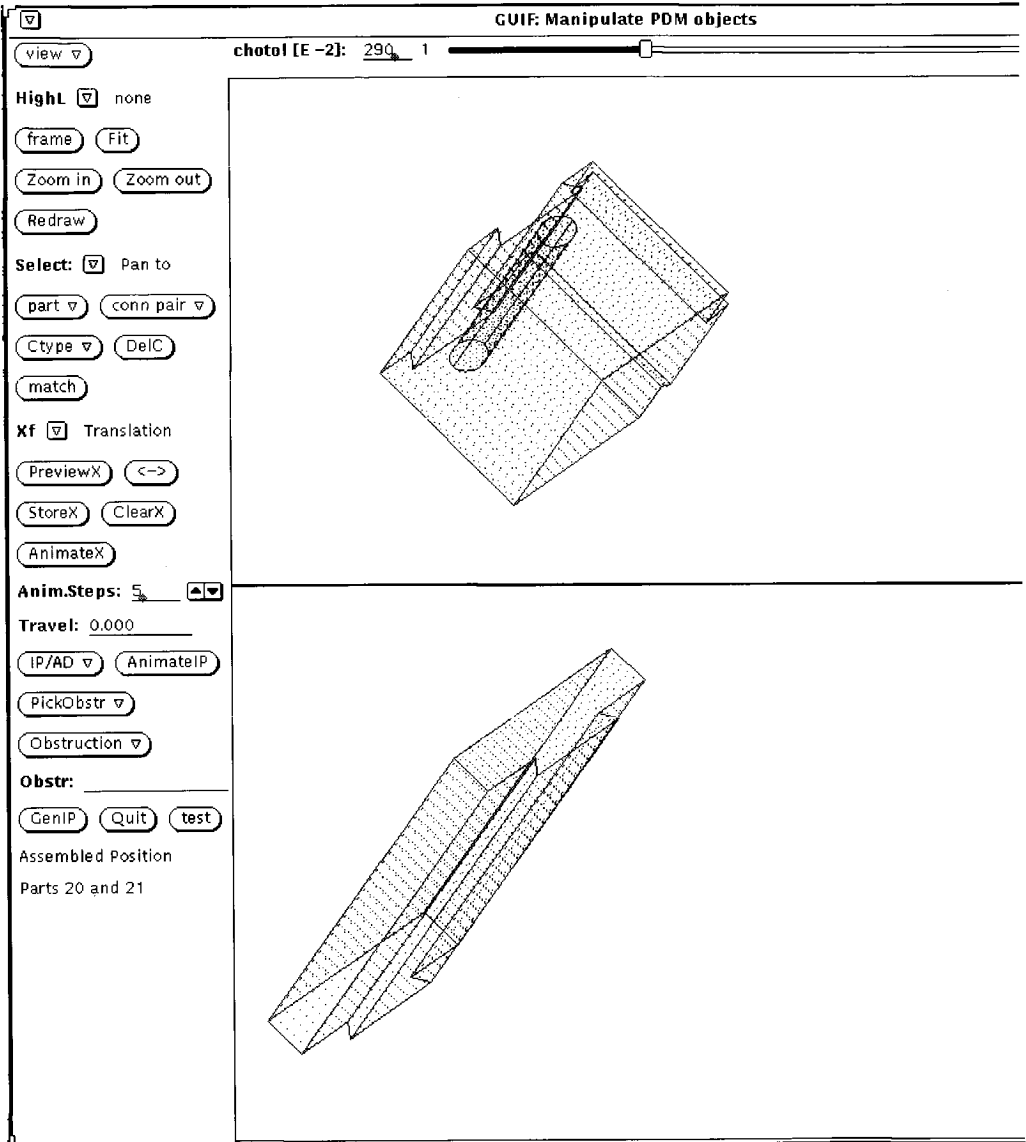


figure 6.8 The GUIF  
Partial screendump of the program guif, showing a session on two parts in the DIAC-1 product.

- material
- colour
- volume
- weight
- Centre Of Gravity
- inertia

The material specification can be found in designer's remarks, e.g. "Material: Aluminum" or "material=St70". While parsing the remarks, the material specification is recognized and looked up in a table. There exists a table with various material specifications and their properties, such as density and colour. The object is assigned a material identification, heterogeneous objects are not supported.

From the Brep, a volume can be calculated. Together with the material density, a weight can be determined. The COG and inertia around the object frame can be determined from the shape as well. These are stored in the database as properties to the shape of the object.

## 6.7 User interface to the PDM

The purpose of the Graphic User InterFace (GUIF) is to enable the user to change attributes which' values appear not to be suitable for assembly planning, or which the assembly analysis failed to generate. Also there is a desire to be able to verify and study the results of the pre-planning. There is no intention of allowing the user to specify a complete product with the GUIF, for the amount and complexity of data would be too large. See for an impression of the GUIF, figure 6.8.

### 6.7.1 Editable data

Only a number of selected data items of the PDM may be hand edited. In particular the attributes that are the result of interpretation and reasoning, derived from the basic product data, are subject to editing. It is possible to manually change data in the database itself, but this is difficult and sensitive to errors.

A general relational database system is not able to enforce integrity (relational integrity and application-implied rules) on the various data. So for guarding data integrity as well, it is undesirable to have users access the data directly. It is better to always provide access by means of an application program, which can be user-friendly and takes care of the complex data structures. The GUIF is such a program, made for editing:

- Connection relations (exists / does not exist, type)
- Insertion Point (a 3D frame with position and orientation)
- Insertion Path
- (Discrete) Assembly Directions

The presence of a connection is usually handled correctly by the automatic system, but there may be occasions where the user wants to specify a connection between two parts that need special attention with respect to each other while assembling. Only if a connection exists, a path may be specified with which one part should be fit onto another part.

## 6.7.2 Interactive specification of transformations

One of the problems when creating a GUIF that uses 3D models, is how to specify motions of objects, or actually, transformations of objects. It is not desirable that the user manually has to enter positions and orientations in 3D.

Initially, when loaded into the GUIF, two parts are positioned relative to each other according to their appearance in the assembly sheet (final, assembled position). The positions, translations and rotations are not arbitrary in an assembly environment: there are surfaces that slide, edges that move along another, etc. The GUIF allows the user to specify transformations by using the existing, real geometric elements of the parts.

Translations can be specified by (see figure 6.9, figure 6.10 and figure 6.11):

- picking an edge (running edge); translation vector equal to the edge, length is changeable.
- bringing boundary points or vertices together (joining vertices).

Several translations may be combined into one. The user may pick two boundary points (in vertices, on edges or surfaces) from both or from one of the parts displayed, thus creating a vector. Boundary points (used for the assembly analysis, see 8.3.1 on page 97), are displayed by the GUIF in specifyable density.

Rotations are made by specifying a twisting axis analogous to the translation and manually specifying a twist angle.

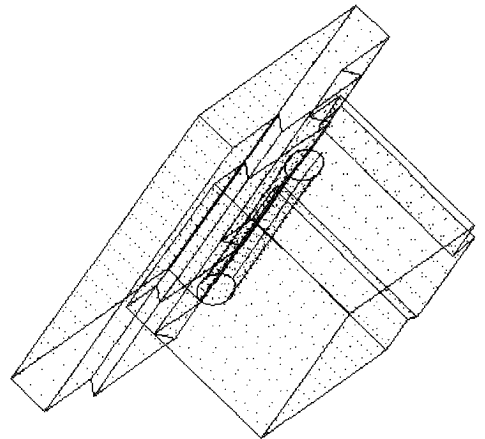


figure 6.9 The parts in the final position (as read from the CAD assembly sheet)  
Partial screendump of the program guif

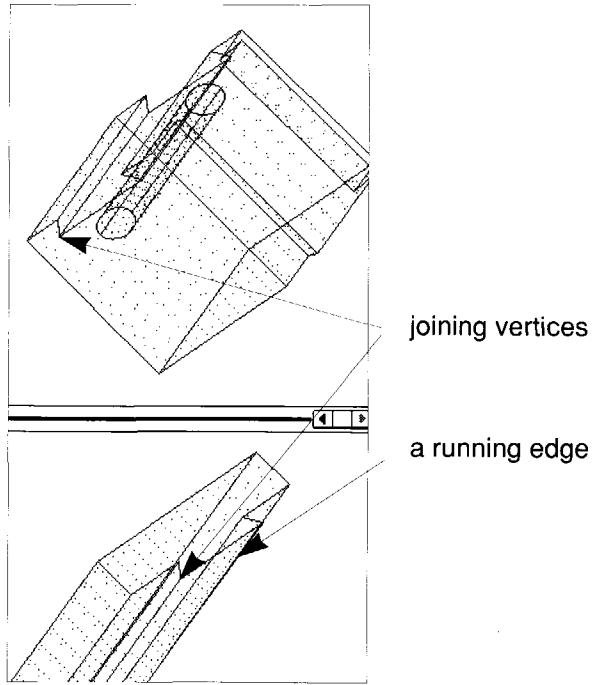


figure 6.10 Specifying translations by using geometrical elements of the individual parts  
Partial screendump of the program guif

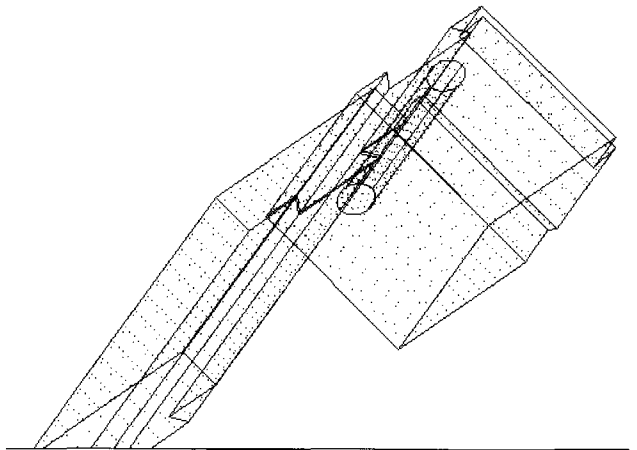


figure 6.11 The insertion point, interactively specified  
Partial screendump of the program guif





## 7 Obstruction theory<sup>1</sup>

The contacts within a connection between two parts in a product are to be described with obstructions: features of the connection at a higher abstraction level (see discussion in section 5.5 on page 60). The next chapter discusses the methods how to build an obstruction model of a connection.

The joint obstructions determine the actual freedom of movement for the object under study. This chapter introduces the obstruction concept in detail and presents the mathematics, developed to deal with obstruction combination. It elaborates on the spatial properties of objects and implications of obstructions found in a contact situation between two parts.

### 7.1 Relocation set representation of a point

A free moving object has 6 degrees of freedom: 3 rotations and 3 translations are required to locate a position. A direction (rotation and/or translation) to move in has only 5 dimensions, for we do not account for the distance travelled. The number of directions the object can move in are infinite: its **relocation direction set** has no limitations. If the object is limited in its freedom by another object in its vicinity, the relocation direction set will be cut down with certain areas. A 'point' in the relocation direction set represents a five-dimensional **relocation vector**, describing an allowed combination of rotation and translation *for the current reference point of the object*. Another point would show different values for the same motion of the object.

If the object would be moved with a certain distance along a relocation vector, a new situation arises with respect to the object's environment, and the relocation direction set adjusts. A set of subsequent relocation vectors and distances is in fact a *path*, along

1. Acknowledgement: Many thanks to Paul Friederichs for his valuable contribution to the techniques described in this chapter [Friederichs 91].

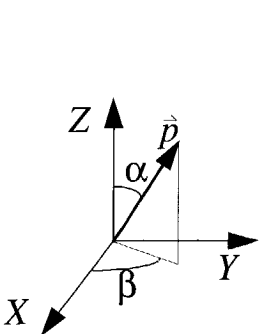


figure 7.1 Definition of a direction  $\vec{p}$

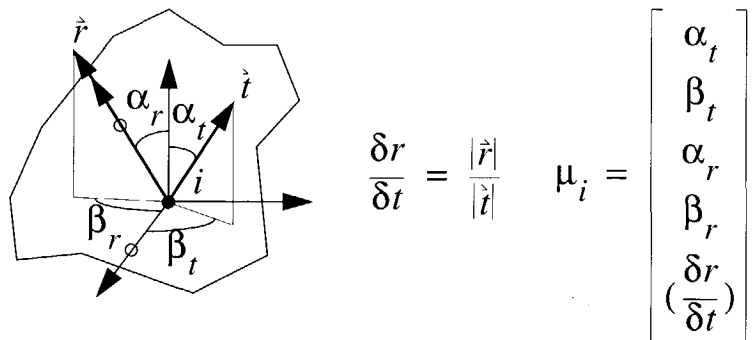


figure 7.2 Relocation vector  $\mu_i$  in a point  $i$  in an object

which the object can move.

The relocation vector  $\mu_i$  in a point  $i$  can be written with 5 parameters:  $\alpha_t$  and  $\beta_t$  for the translation direction,  $\alpha_r$  and  $\beta_r$  for the rotation direction and  $\frac{\delta r}{\delta t}$  for the proportion between translation and rotation, see figure 7.2 (figure 7.1 recalls the definition of subsection 4.4.3 on page 49). The lengths of translation and rotation vectors are in the proportion of  $\frac{\delta r}{\delta t} = \frac{|\dot{r}|}{|\dot{i}|}$ . The distance of travelling can be expressed, if needed, with either rotation or translation, as the proportion is stored with the relocation vector. The following constraints apply, due to the definitions:

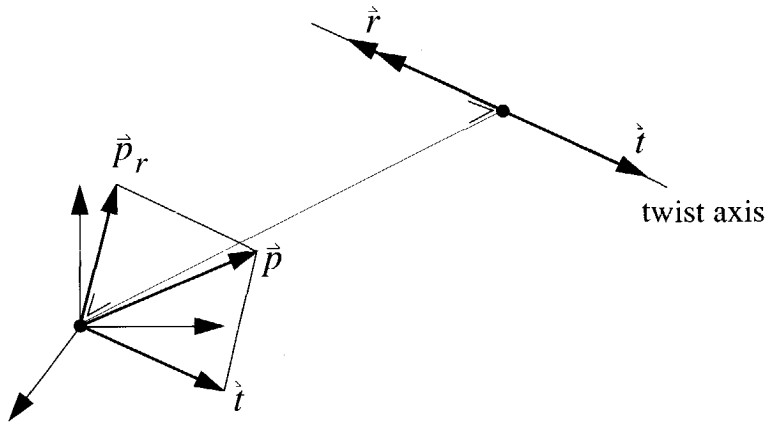


figure 7.3 The twist axis and the direction in a point

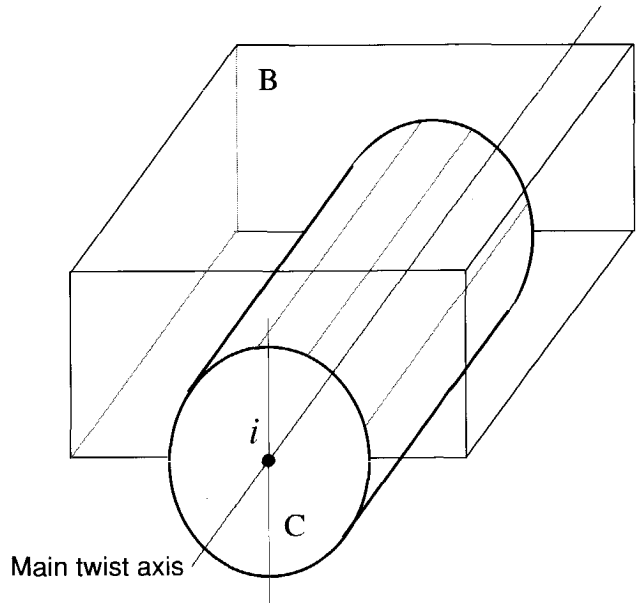


figure 7.4 A contact with a cylindrical face

$$0 \leq \frac{\delta r}{\delta t} \leq \infty, 0 \leq \alpha_{t,r} \leq \pi, -\pi \leq \beta_{t,r} \leq \pi.$$

For a screw being mounted, the  $\frac{\delta r}{\delta t}$  is constant, and equal to the pitch, but only if *the point lies on the centre axis*. It can be shown that at a single moment, there exist a line of points in an object, for which the rotation vector as well as the translation vector lie along this line: the momentary twist axis, see figure 7.3. Theoretically, the location of the twist axis can change continuously along a motion path. In the mechanical engineering reality however, the twist axis remains the same over a span of time.

When there is a pure translation,  $\frac{\delta r}{\delta t} = 0$ , the twist axis is meaningless (lies in infinity). When  $\frac{\delta r}{\delta t} = \infty$ , there is a pure rotation ( $|\dot{t}| = 0$ ). See for more information on twist axis theory [Bottema 79, Asea 87].

A set of relocation vectors in the relocation set represents an area in the five-dimensional space defined by  $\alpha_t$  and  $\beta_t$  for the translation direction,  $\alpha_r$  and  $\beta_r$  for the rotation direction and the proportion  $\frac{\delta r}{\delta t}$ . For virtually every motion from the mechanical engineering practice that has to be represented in the relocation set, the translation, rotation and their proportion can be regarded as independently within one area. In other words, we may say that for example the translation direction may vary within its allowed set, without influencing the rotation and the proportion. This means that an area in the 5D space can be represented by three area's in 2D and 1D, respectively  $(\alpha_t, \beta_t)$ ,  $(\alpha_r, \beta_r)$  and  $(\frac{\delta r}{\delta t})$ . However, there may have to be more than one of such area's, so there may exist several independent sets of rotation, translation and proportion.

### Example

Refer to figure 7.4. The cylindrical face contact allows of course the rotation of Block B around the main twist axis (in the centre of Cylinder C). Simultaneously, the translation along this axis (in the same direction) is allowed, in any proportion. Also, a set of translations is allowed that move the block B directly away from the cylinder C. So, for  $(\alpha_r, \beta_r)$  representing the main twist axis,  $(\alpha_t, \beta_t)$  may be a range of values as drawn in the figure 7.5 and  $\frac{\delta r}{\delta t}$  can be anything. If rotations are considered not important, the range of translations is still available.

A common example is a screw in a hole with thread. Obviously,  $\frac{\delta r}{\delta t}$  equals to the pitch of the thread and can have no other value, so translation is only possible with the appropriate rotation. The diagrams for the relocation vector are drawn in the figure 7.6 on page 84. Note that when  $\alpha = 0$  or  $\alpha = \pi$ ,  $\beta$  may be anything (singularity in the representation); translation and rotation vectors are in the same direction, so the diagrams are equal. However, mounting the screw requires a different translation/rotation combination than unmounting, for the proportion cannot be negative. Since the rotation of mounting cannot be combined with the translation of *unmounting* and

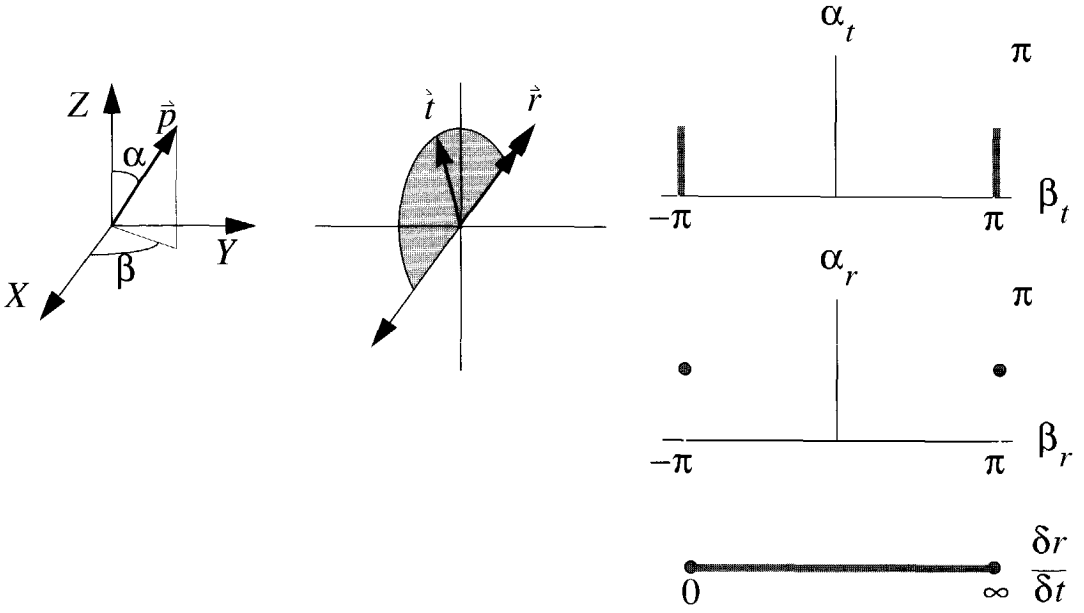


figure 7.5 Relocation direction set representation

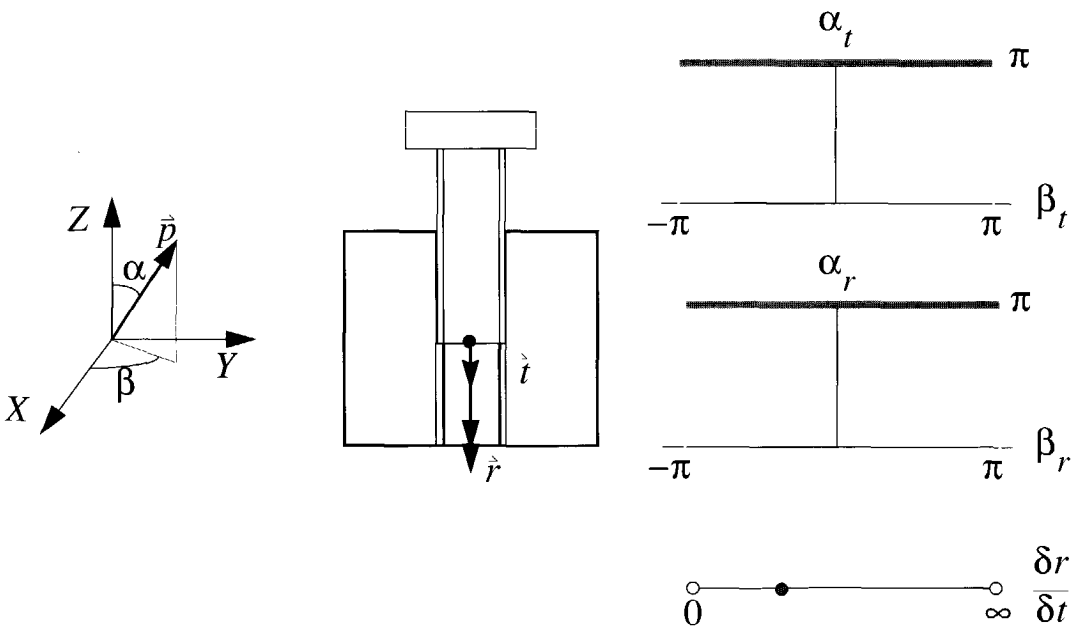


figure 7.6 Mounting a screw

vice versa, we need a second relocation direction set area, as shown in figure 7.7. Together these two sets of relocation vectors make up the total relocation direction set for this part in this position.

## 7.2 The relocation direction set of an object

The relocation vector and relocation direction set are valid for a single point in an object. Rotation of an object can be described if the current twist axis is known; it can be calculated with the relocation vectors of a few points in the object. The twist axis can be represented by a point on the twist axis, a **support point** (located somewhere on the axis), and the rotation vector for the direction. Translation of a point is dependent on the distance to the current twist axis; only for points on the twist axis the translation equals that of the entire object.

In the examples mentioned earlier, the point  $i$  was always on the obvious twist axis. The meaning of  $\alpha_r$  and  $\beta_r$  in that case is the orientation of the current twist axis.

Theoretically, there could be more twist axes in one point, but normally the  $\alpha_r, \beta_r$  diagram will only contain one or two values. If two twist axes intersect, the support point can be placed there (since a support point may lie anywhere on the twist axis) and there will be more than one value in the diagram.

This means that for every non-intersecting, non-movable twist axis, a separate set of relocation direction set diagrams is needed, each with a different support point. Also, if the  $\alpha_r, \beta_r$  diagram is empty and/or  $\frac{\delta r}{\delta t} = 0$  (in fact equivalent, as shown later), the translation diagram is valid for every point in the object; it is a pure translation. The only case when an infinite set of rotation directions exists for one support point, is the sphere: no matter what line is drawn through the centre of the sphere, it is always a

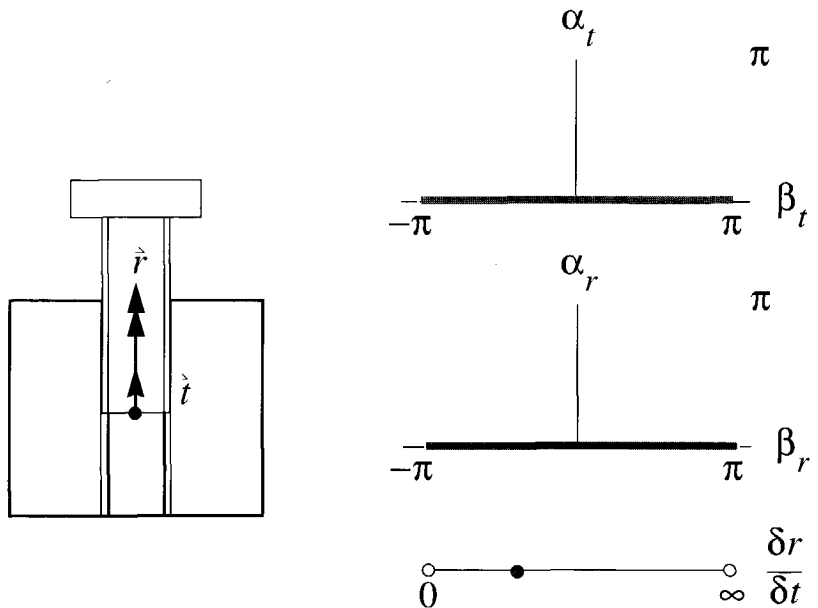


figure 7.7 Unmounting a screw

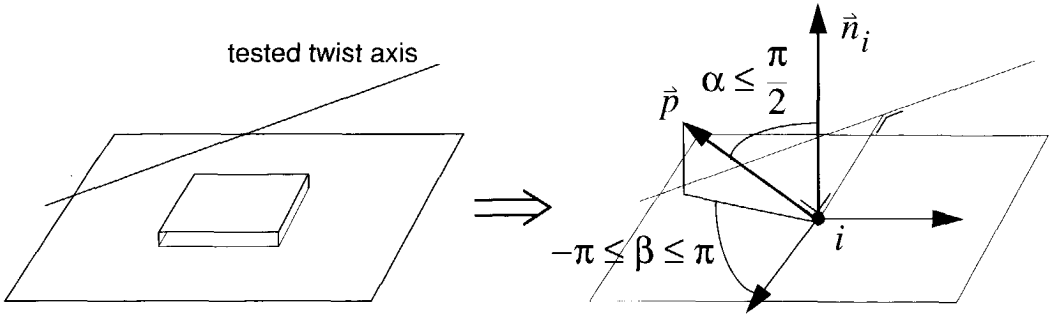


figure 7.8 Normal and direction vectors in contact point  $i$ , for rotation around twist axis.

twist axis. In all other cases, we will see one or two points, except when  $\alpha = 0$  or  $\alpha = \pi$ , in which case  $\beta$  is undefined and the diagram contains lines (see for instance figure 7.6 and figure 7.7 on page 85).

In order to test whether a certain relocation vector  $\mu$  is valid for an object with contact points and surfaces, local directions (relocation vector in a point) must be calculated.

All points in contact experience a direction  $\vec{p}$ , as a result of the translation and rotation of the object. As long as this direction agrees with the local normal vector

( $\vec{n} \cdot \vec{p} \leq 1 \Leftrightarrow 0 \leq \alpha \leq \frac{\pi}{2}$ , see figure 7.8), this point agrees with the proposed

relocation of the object. Theoretically, if all points in contact could be tested with all possible relocation vectors, we can find all valid directions to move in for a given contact situation.

As explained earlier (Chapter 5), this runs into computational infinity problems. The

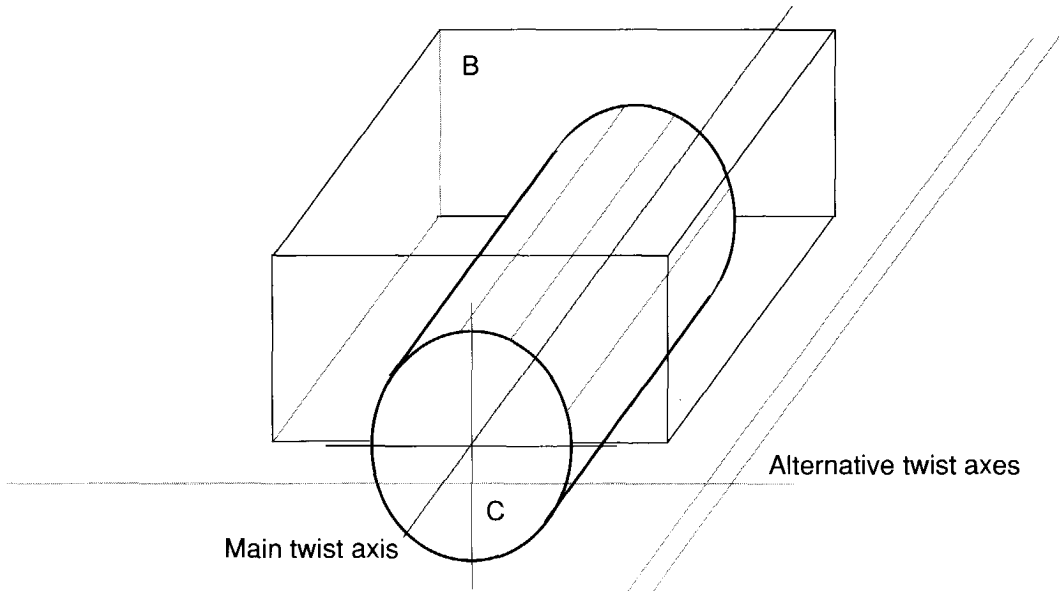


figure 7.9 Alternative twist axes for a contact with a cylindrical face

concept of obstructions will manage with this problem by guiding the search for meaningful and valid relocation vectors heuristically. Obstructions have defined configurations and therefore known properties. For each obstruction, a (limited) relocation direction set can be set up.

## 7.3 The relocation direction set of obstructions

### 7.3.1 Introduction

An obstruction is a contact, experienced by an object, described with a few parameters. This contact is therefore an entity, which is recognizable and classifiable. It obstructs the part in its freedom of movement in a defined and predictable way. The obstruction is a *feature* of the connection which is significant for studying the effect of the connection. In analogy, geometric objects can be assigned manufacturing features (for example, a 'slot'), which are used to highlight certain aspects of the object with respect to its manufacturing.

The obstruction is defined with a local frame, the *obstruction frame*, that defines the orientation and position of the contact area, relative to the part frame. Usually the obstruction frame will be chosen such that its origin is a support point as well. However, the origin of an object's object frame is defined elsewhere, and specified in the geometry (Brep). Later in this chapter, a method will be discussed to transform the frames of the relocation set diagrams.

### 7.3.2 Selective relocation direction set

The translation part of several local relocation direction sets can be tested and combined together easily. The problem concentrates on an infinite number of possible twist axes. The solution to this problem is to reduce to a limited number of rotations within each obstruction. Refer to figure 7.9. An obstruction of type cylindrical face allows not only rotations around its centre line, but also rotations around axes that lie outside of the object. These secondary rotations use the fact that the cylinder may move out of the gap with a translation. The points on the contact area use different direction, but it is allowed. There seems no good cause to examine these rotations, because the designer probably meant to rotate around the cylinder, if at all. If any other rotations have to be examined, it suffices to check a few significant points on the contact area whether a certain rotation is allowed.

The considerations above have led to the concept of *selective* relocation direction set: Only store and review a limited number of likely rotations and as many of the allowed translations as possible. These prominent twist axes do not have to be determined automatically (predefined obstruction prototypes). As a rule of thumb, we can say that the prominent twist axis of an obstruction can be found by intersecting the normal vectors of all points in contact (cylinder/sphere contact), or it is parallel to the normal vectors, perpendicular to the contact plane (plane contact).

If a rotation is required to (dis)assemble a part, but the rotation is not invoked by any obstruction of the part, the algorithm may fail. This happens when there is very little room in an assembly to mount a part and it needs to be tilted to bring it into assembly position. In that case, the contact situation in assembled position has little to do with the approach, in which a complex path is required to avoid collisions. The algorithm will use the existing obstructions to move out of the assembled position and perhaps run into a new obstruction that will guide the rotation after all. For exceptions where even this strategy does not work, the program has to ask for user assistance to resolve

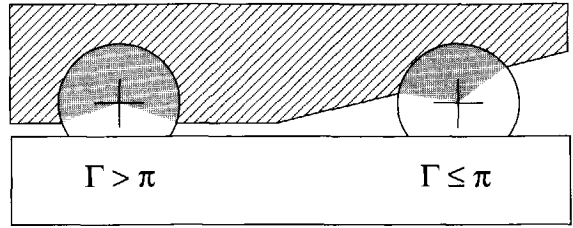


figure 7.10 An imperative and an optional twist axis

the problem.

### 7.3.3 The imperative twist axis

In the example of the cylindrical face, alternative twist axes are allowed, which can be checked with a few contact points and their normal vectors. However, if the angle  $\Gamma$  over which the contact extends is larger than  $\pi$ , no alternatives are allowed: this is an **imperative twist axis**. This can occur for all curved contact forms, which then are promoted to a special obstruction type (currently cylinder, cone and sphere). The actual angle over which the contact extends is no longer interesting (see figure 7.10). In contrast with the potential twist axis a cylindrical face invokes, the imperative twist axis allows no other rotations of the object. The support points of this imperative twist axis can be moved, so they may coincide with support points of other twist axes. If the support points of two imperative twist axes cannot be made to coincide, no rotation is possible (see figure 7.11). The sphere knows an infinite number of imperative twist axes that may run through the centre of the sphere (therefore the support point may not be moved at all).

Imperativeness is stored as a property of the support point. Note that no other rotations can or need to be defined for this obstruction.

### 7.3.4 Validity rules for relocation direction diagrams

The diagrams of rotation, translation and proportion can be screened on their physical

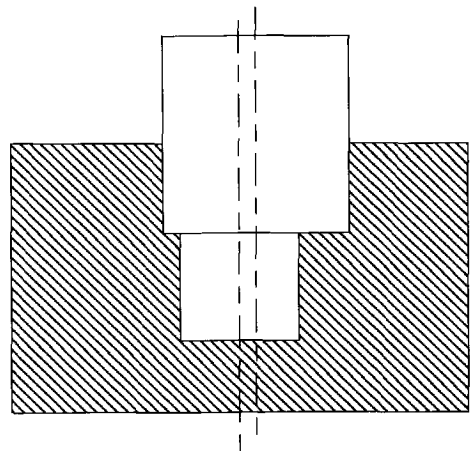


figure 7.11 Irreconcilable imperative twist axes



properties. Some combinations of diagrams have no meaning in the reality, so they can be filtered out each time new diagrams are built (especially in the combination process). The validity rules for the diagrams are:

1.  $(\frac{\delta r}{\delta t} = 0) \Leftrightarrow (\alpha_r, \beta_r) = \emptyset$ , because  $|\dot{r}| = 0$ . This is a pure translation.
2.  $(\frac{\delta r}{\delta t} = \infty) \Leftrightarrow (\alpha_r, \beta_r) = \emptyset$ , because  $|\dot{r}| = 0$ . This is a pure rotation.
3.  $(\frac{\delta r}{\delta t} = \emptyset) \Rightarrow (\mu) = \emptyset$ , because no valid directions exist.
4.  $(\alpha_r, \beta_r = \emptyset) \wedge ((\alpha_r, \beta_r) = \emptyset) \Rightarrow (\mu) = \emptyset$ , because no valid directions exist.
5.  $(\frac{\delta r}{\delta t} = C) \wedge (\alpha_r, \beta_r) = \emptyset \Rightarrow (\mu) = \emptyset$ , because there must be a rotation for  $\frac{\delta r}{\delta t}$
6.  $(\frac{\delta r}{\delta t} = C) \wedge (\alpha_r, \beta_r) = \emptyset \Rightarrow (\mu) = \emptyset$ , because there must be a translation for  $\frac{\delta r}{\delta t}$

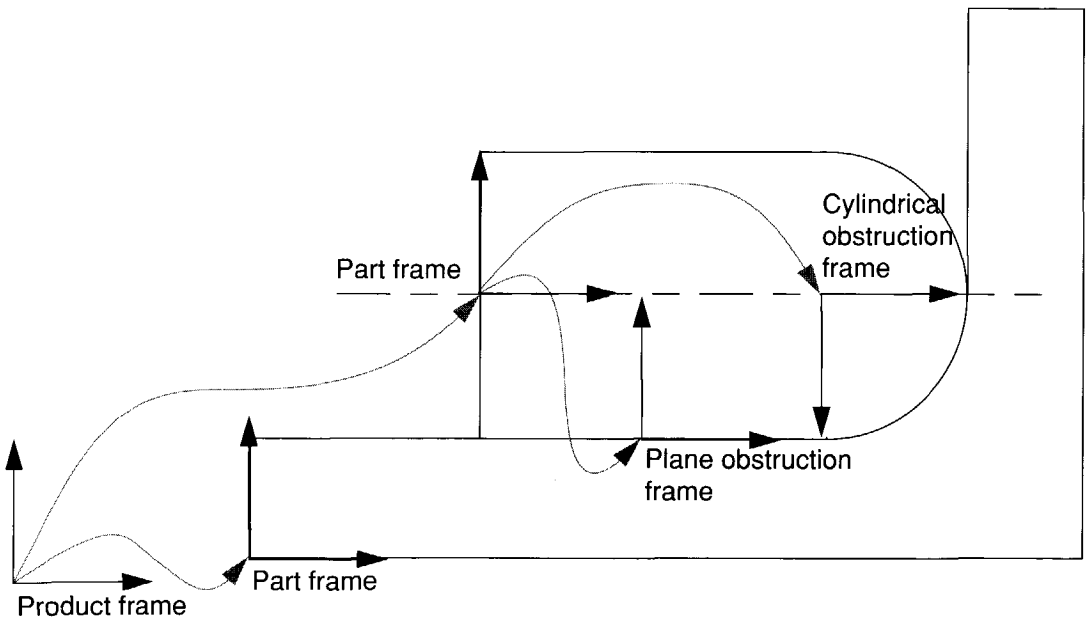


figure 7.12 Using two obstructions to describe a contact configuration

Combining obstructions means combining the relocation sets together (e.g. see figure 5.11 on page 62). The remainder of this chapter is devoted to techniques to do that.

## 7.4 The obstructions of an object

An obstruction prototype is defined with reference frames, over which the potential twist axes run. The actual obstruction in a part will be specified with its position and orientation relative to the part frame (see figure 7.12). The obstruction is identified and located (described in the next chapter), and the parameters are calculated. In the example, the cylindrical obstruction frame had to be rotated, since the obstruction is defined with the start of the contact area at the X axis (pointing down). The angle over which the cylindrical contact area extends is the parameter  $\gamma_{cf}$ , so here equal to  $\pi/2$ , for the contact extends to the Y axis over the full first quadrant of the cylindrical obstruction frame. The potential twist axis, invoked by the existence of the cylindrical face obstruction, runs perpendicular to the view in the figure, through the origin of the cylindrical obstruction frame. As usual, the position and orientation of an obstruction frame relative to the part frame is expressed with the transformation matrix R (see 4.4.2 on page 48).

The relocation direction sets of the obstructions, defined relative to the local obstruction frames, must be transformed to one frame, in order to combine them and calculate the resulting relocation direction set of the part. Parameters of the obstruction that are defined locally (support points, contact points) or vectors (normals) can be transformed with a simple matrix multiplication, using the properties of the rotation matrix that defines the local obstruction frame:

$\vec{p}_p = R^{-1} \cdot \vec{p}_o$ , with  $R^{-1} = R^T$  and  $\vec{p}_o$  a local vector and  $\vec{p}_p$  its representation in the part frame.

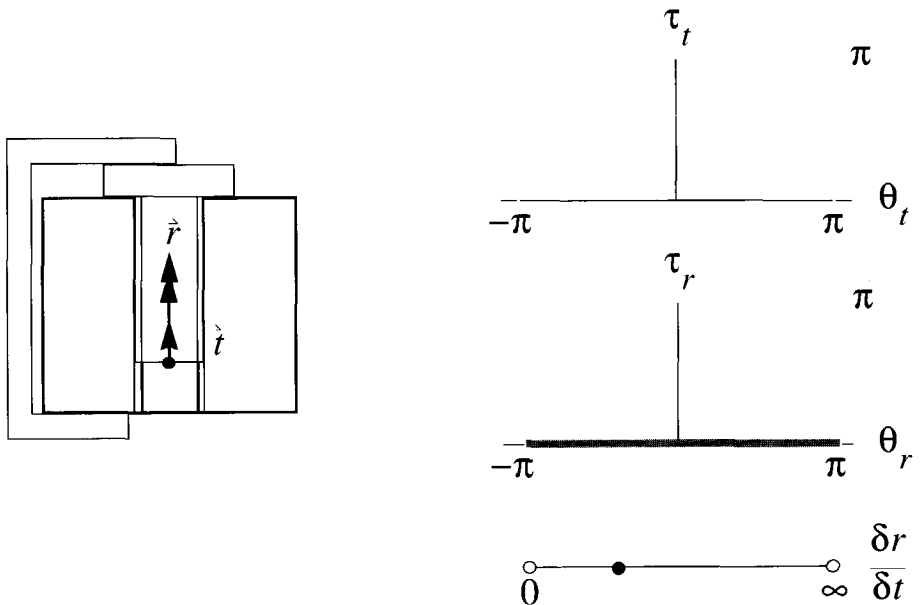


figure 7.13 A separate rotation and translation analysis

The directions will change in the case that the obstruction frame has a different orientation than the part frame, see for example the cylindrical face in figure 7.12 on page 89. For clarity,  $\alpha$ ,  $\beta$  are considered local to the obstruction, while its transformed counterpart is represented with  $\tau$  and  $\theta$ . Each time a new set of diagrams  $((\tau_r, \theta_r)$ ,

$(\tau_r, \theta_r)$ ,  $\frac{\delta r}{\delta t}$ ) is created by evaluation of rotation and translation sets, they must be checked against the validity rules described before. For example, the screw in figure 7.13 on page 90 has no resulting translations, so the  $(\tau_r, \theta_r)$  diagram is empty. Since  $\frac{\delta r}{\delta t} = \text{Constant}$  also, we learn from rule 6 that  $\mu$  must be empty, so there is no valid direction to move in for this part and all diagrams should be empty. If it had been a plain cylinder instead of threaded, the proportion would range  $0 \leq \frac{\delta r}{\delta t} \leq \infty$ . Rule 2 says that because there are no translations:  $\frac{\delta r}{\delta t} = \infty$ . Obviously, the part may only rotate around its centre, which will lead nowhere.

## 7.5 Combination analysis

Translation and rotations of objects are assumed only to appear in a fixed proportion. Therefore, the analysis of translations and rotations can be done separately, but considering the rules in the previous section. The obstructions have a local frame, with a known relative position in the part, that has to be converted before determining the intersection. See for an overview of the combination process for two translation sets, figure 7.14.

### 7.5.1 Translation analysis

Combining obstructions of parts is now a matter of transformation of the parameters to one common frame and determining the intersections of relocation direction sets. If the proportion is not a single value, the transformation of the relocation direction sets can be done separately for the translation and rotation. If  $\frac{\delta r}{\delta t}$  is one single value between 0 and infinity, we have the case of a fixed screw thread, which will be handled separately. The extremely rare condition of a range of values for the proportion (would be something like a variable pitch) will not be considered here.

It is indifferent where the translation vector of a part is attached, so several  $(\alpha_r, \beta_r)$  diagrams can be transformed and joined into the  $(\tau_r, \theta_r)$  diagram just by determining the intersections (see figure 5.11 on page 62). The validity rules must apply however, so one constraint is that neither  $\frac{\delta r}{\delta t}$  diagram contains only infinity. This would, combined, result in  $\frac{\delta r}{\delta t} = \infty$ , in which case the translation diagram would be empty.

### 7.5.2 Rotation analysis

For rotations several complications must be handled, in order to determine whether rotations are valid. The support points are to be combined and checked whether they agree in such a way that rotations are possible. In the case of non-imperative rotations,

there could be more rotations possible, which have to be checked explicitly with the properties of the obstruction: normal vectors on the contact area and contact points.

If none of the obstructions has preference for certain rotations, this does not mean the object cannot rotate, but there is just no contact that creates the need or the preference to rotate. In that case, it is assumed that no rotation is needed and therefore the rotation set is empty ( $(\tau_r, \theta_r) = \emptyset$ ).

As explained before (section 7.3), imperative support points can only be combined if they can be moved such that they lie on the same line. The  $(\tau_r, \theta_r)$  diagram will contain the direction of the allowed twist axes through the support point. The result can only contain rotations around the line through the support points, and the intersection of the directions. This line can be regarded as a  $(\tau_r, \theta_r)$  diagram with two points, with which must be intersected as well.

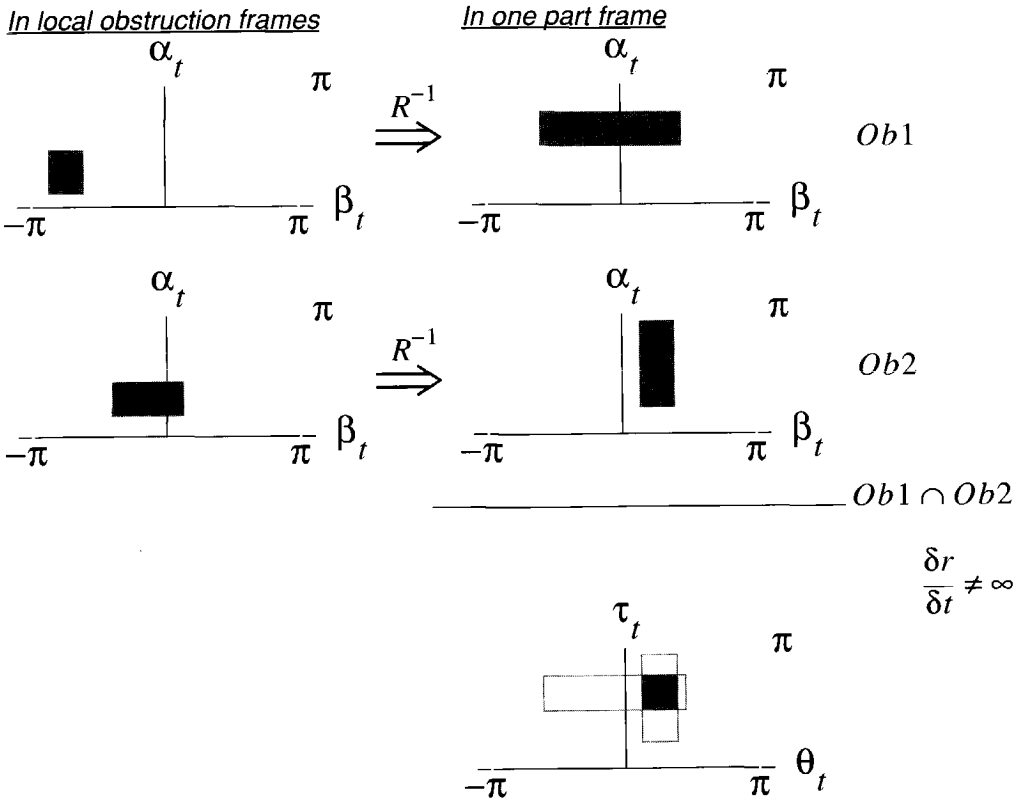


figure 7.14 Translation transformation and combination

## 8 Obstruction Recognition

As explained in chapters 5 and 7, analysis of all contacts between two parts will not easily yield a (dis)assembly path. The theoretically infinite number of directions to move in and the infinite number of points in contact make it practically impossible to derive an assembly path. The solution to this problem is to compile the contact analysis information to a more compact description: the obstruction. This is information on a higher level of abstraction, that allows reasoning about degrees of freedom and motion directions. This chapter is about the transformation of a given contact situation into a set of obstructions.

Recognition of obstructions is in fact determination and analysis of geometric elements in contact. For a given combination of two parts, this procedure must yield a number of obstructions, classified and complete with their properties. If contact exists, the part-relation can be added to the list of connections, thus creating the relation network.

### 8.1 Definition and classification of obstructions

#### 8.1.1 Obstruction parameters

In sections 5.5 on page 60 and 7.3 on page 87, the concept of obstructions was introduced. An obstruction is a description of a (partial) contact between geometrical objects, classified according to an *obstruction-prototype*. Classification therefore consists of choosing one out of the collection of obstruction-prototypes. The obstruction-prototype is a predefined configuration of contacts, with a number of parameters, specific to that prototype. As parameters, at least the frame must be stored, relative to which the obstruction is defined. So, if a contact situation is recognized, the appropriate prototype is instanced and the obstruction parameters are assigned their values.

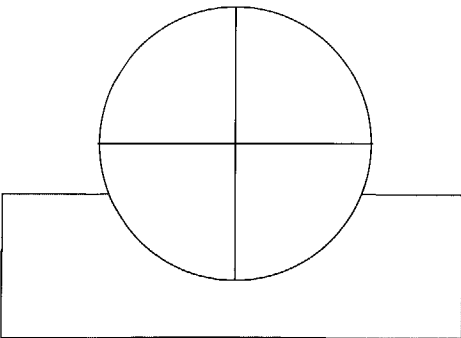


figure 8.1 A cylinder in sections has a single continued obstruction.

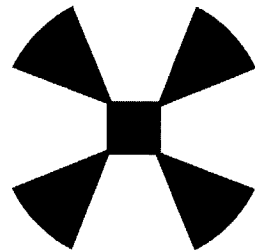


figure 8.2 A discontinuous obstruction must be evaluated into separate ones.

For example, a cylindrical contact is recognized as such and the parameter list of the obstruction-prototype *cylindrical face* asks for the angle over which the contact extends (see appendix section B.2 on page 140). The frame of a cylindrical-face obstruction is defined with the Z-axis along the centre of the cylinder. The angle is defined starting from the X-axis, around the Z-axis. Notice the fact that the radius of the circle does not have influence on the obstruction: the limitations of the relative motions of parts with this kind of contact are determined only by the position/orientation of the frame and the extend-angle. The length along the Z-axis is of minor importance for this obstruction and is given implicitly by the significant points, see below.

Usually, the obstruction consists of only one (continuous) contact area, but the contact analysis may show that several contact areas indubitably form one obstruction. For example, the Brep may divide a cylindrical surface into tangent sections (see figure 8.1 on page 93) cylinder in sections), while in fact the sections are part of one cylindrical contact. When in doubt, the obstructions would have to be separated, see figure 8.2. The obstruction combination algorithm would filter and combine the obstructions into a resulting obstruction/motion set after all.

The obstructions are used to derive the most likely directions to move in. In the process of combining obstructions, a certain direction to move in must be tested as well. In order to test whether the contact area of an obstruction does allow this direction, the configuration and parameters may not provide sufficient information for this particular direction. For example, a rotation around an axis, away from the contact area (see figure 8.3). In that case, all (points) of the contact area will have a slightly different direction vector, but not significantly different. Testing only a few representative points will do. These are called *significant points* and are provided along with the parameters of the obstruction. The normal of the contact in each significant point is given by a function, provided with the obstruction prototype. For instance, the normal of every point in contact of a cylindrical obstruction is always directed towards the centre of the cylinder: the Z axis of the frame of the obstruction.

### 8.1.2 Obstruction prototypes

The most common ways to connect two parts should be contained in the obstruction prototype list. The list presented hereafter is not very large, but can be extended (it can be regarded as a knowledge base). Especially because several obstructions can be combined in one contact situation, this limited list can handle very complicated configurations. In fact, all connections of the test products (see appendix A. on page 131) can be modelled with these obstructions, except for the snap fits. Whenever the obstruction can not be recognized, the user must be called upon.

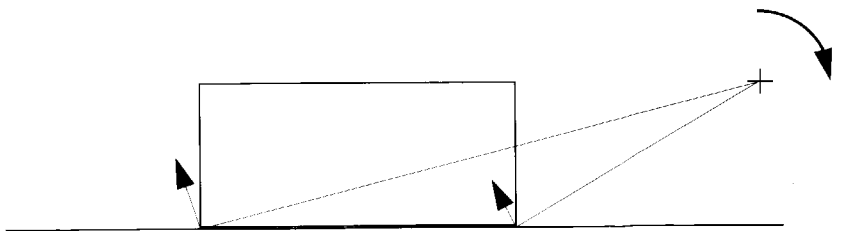


figure 8.3 Distant twistaxis checked with significant points of the obstruction

- Plane.
- Cylindrical.
- Conical.
- Spherical.
- Thread, left or right.

For cylindrical, conical and spherical, if the angle over which the contact extends is larger than  $\pi$ , this obstruction can be seen respectively as a full cylinder/cone/sphere in contact, with different properties. See for the definitions of some of these obstruction prototypes appendix B. on page 139.

## 8.2 Determination of contact

### 8.2.1 COG bubble check

In order to avoid going through the contact check for parts that are far apart, a fast contact check is done. For every part, a bubble is created around its Centre Of Gravity, which is known in the product model. It is likely that the COG is physically somewhere in the middle of the part. The radius of the bubble is the maximum distance of any geometric element to the COG. If these bubbles of two parts intersect, a more detailed analysis is required (see figure 8.4).

The first stage of the contact analysis (as described below) is performed with a low accuracy. If there still appears to be contact, a high accuracy contact analysis is done. The example in the figure 8.4 will require one step of contact analysis, after the bubble check has shown intersection.

A faster method would be to make a bubble-refinement step inbetween COG bubble check and contact analysis. If all surfaces of the two parts were to be surrounded with a bubble, fewer (time-consuming) contact analyses would be required. However, since calculation time is not a real constraint in this off-line application, this refinement is not elaborated.

### 8.2.2 The relation network

A relatively simple result of the contact analysis is the fact whether parts are connected at all. However, this is important information for the product model. As soon as it is

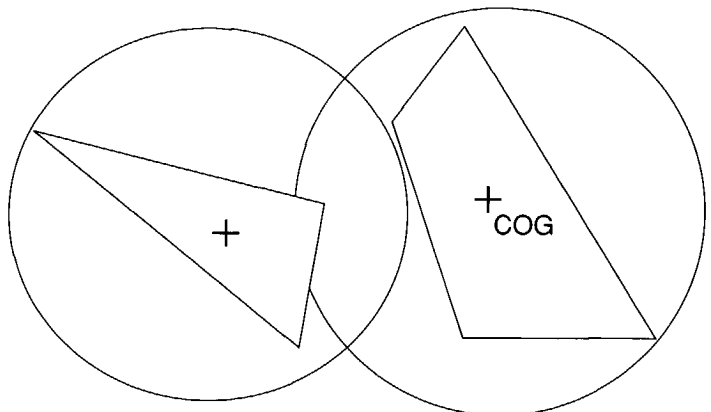


figure 8.4 Initial check of contact: intersection check of bubbles around COG

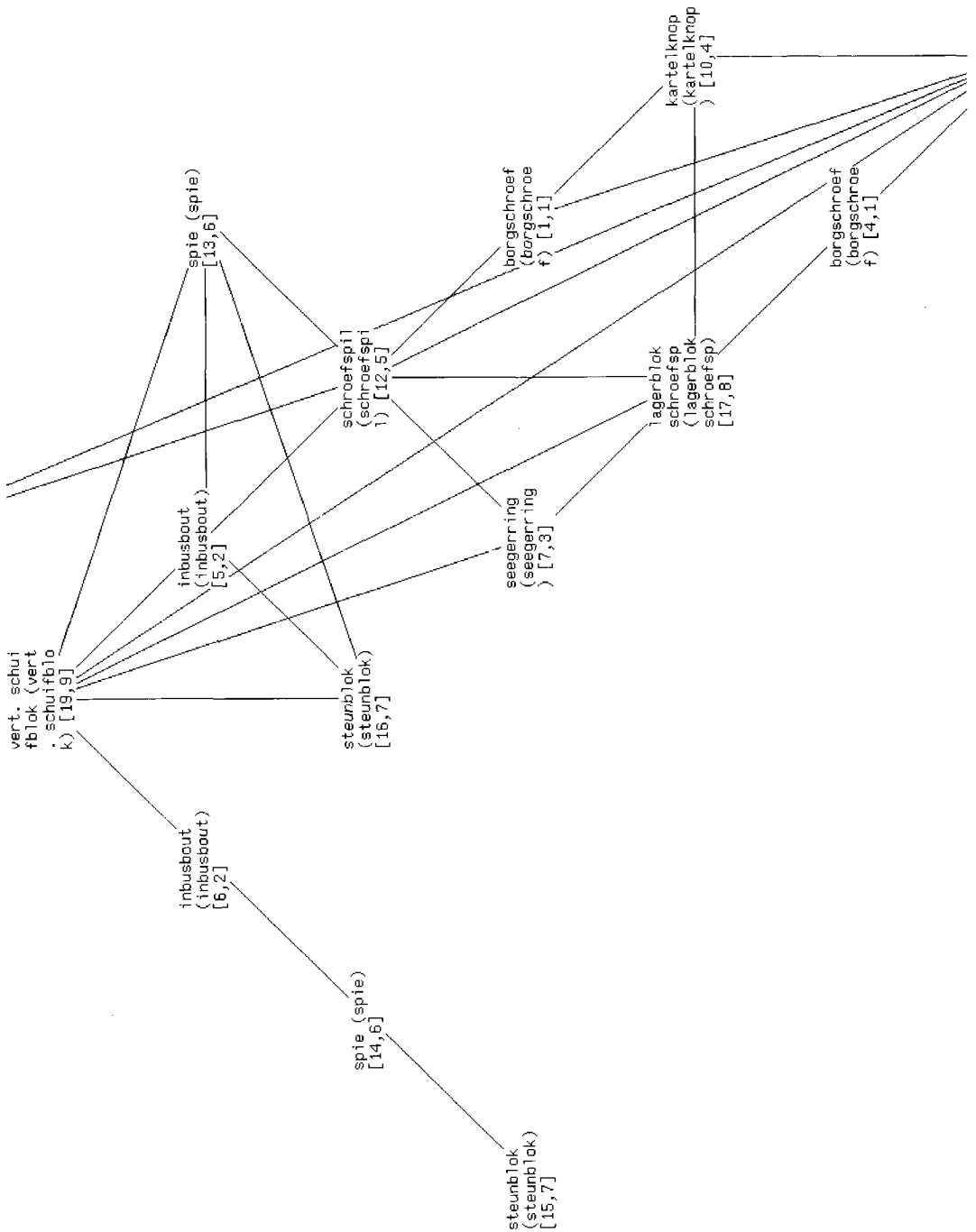


figure 8.5 A part of a relation network.  
 Screen-snapshot of program **relat\_netw** analysing product Diac1. The program queries the database for connections and determines recursively how to place the parts, so the connection lines will be shortest and least crossing



assessed that two parts have contact, an entity of type **connection** is created. This entity serves as repository for the results of the contact/connection analysis.

Assembly sequence planning algorithms (see also section 5.2 on page 54) often use the connection information as main input. Graphically, the set of connections in a product is represented as a **relation network**. An example of a relation network is shown in figure 8.5.

### 8.3 Contact analysis

The contact analysis will have to determine which elements (vertices, edges or surfaces) of the two parts have contact, and what kind of contact topology can be recognized. As stated before, some tolerance must be allowed: a very small clearance between two surfaces must be regarded as contact as well. The most basic requirement is not to miss any contacts and the bubble method is very suitable for that. Furthermore, the geometric model is of limited accuracy for the real numbers are specified with a limited number of significant digits. Therefore, a resizable recognition tolerance must be applied, which is an operational parameter to the connection analysis.

#### 8.3.1 Boundary points

The original bubble check method [Bouts 90, Verwer 91] concerns itself with generation of a minimum number of bubbles with varying radius, that just surround the part. The bubble hierarchy method, expecting that in most cases there is no contact, optimizes for comparison speed and an early conclusion of 'no contact'. When however contact is to be expected and must be analysed accurately, a different approach can be taken. Still most important is to determine the situation of the elements in contact.

For determining contact situations, **boundary points** are introduced here. Boundary points are points on surfaces, edges and in vertices, that are *a maximum distance apart*. The points thus form a grid over the entire part, covering every boundary of the part (see figure 8.6 and figure 8.7). Boundary points of two parts are compared in pairs: one of part A and one of part B. If a pair is close to each other, that is, the distance is smaller than a criterion, the geometric elements that the points are on are assumed to be in contact.

The boundary point method inherits from the bubble method, that the exact geometry is modelled by a set of simple forms that is slightly larger than the original shape. This

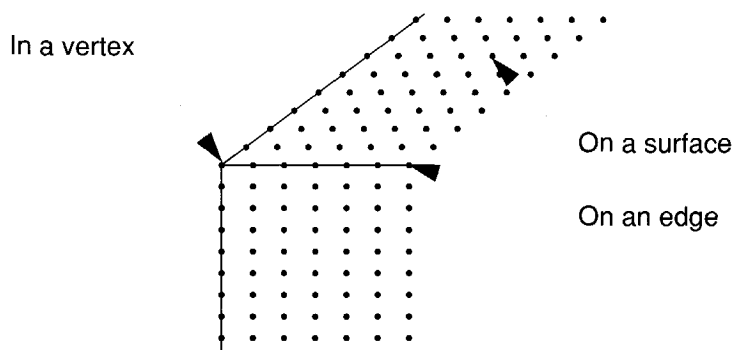


figure 8.6 Boundary points

ensures sensitivity for very small contact areas, such as a vertex on a plane surface (almost undefined contact areas, see figure 5.2 on page 52). In other words, appropriate geometric elements are found to be in contact, indifferent of their mutual position/configuration. Also, there may be a small, controllable clearance between the two parts.

Disadvantages are:

- More contacts than actually exist will be found. A second stage must determine from the found contacts how the obstruction actually fits (discussed in subsection 8.3.4 on page 100).
- The procedure is time-consuming. Matching all pairs of points of large parts can take many minutes of CPU time.
- The storage of the points takes up a considerable amount of memory.

However, in the prototype implementation, it appears that neither memory nor computing time are bottlenecks while comparing complex parts.

The boundary point method is usable for the problem of BRep approximation tolerance and designer's tolerances. By varying the parameters of the algorithm, it can be determined very accurately which surfaces are (meant to be) in contact. Also it is very easy to determine which part of every surface is at least *possibly* in contact. This yields a list of the interesting surfaces, leaving out disturbing not-interesting other surfaces and even parts of surfaces. If one of the parts travels a distance, the comparison can be made again directly by changing the reference frame of the appropriate pointlist (which is the relative position of the part in the product). For

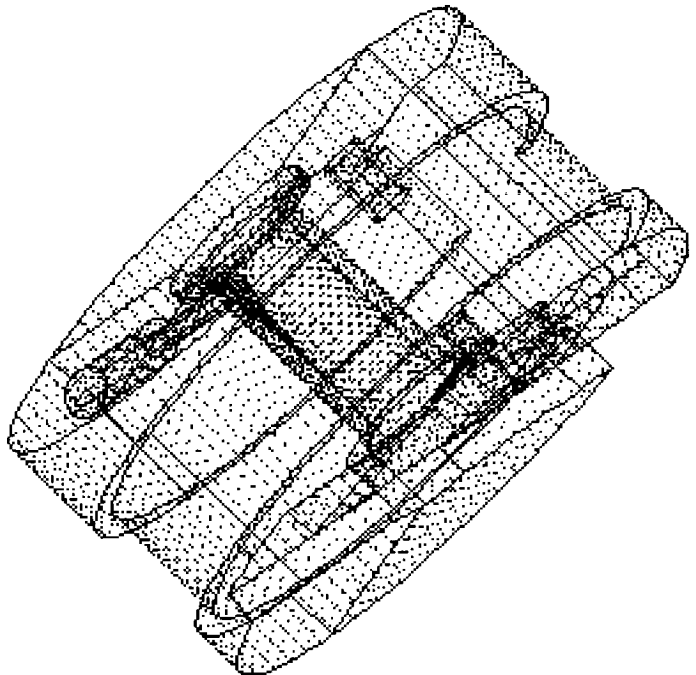


figure 8.7 Boundary points generated on a part  
Partial screen-snapshot of program guif analysing Diac2

larger distances, the analysis might yield new maybe-contact surfaces, which will have to be re-examined by zooming in with a smaller point-distance.

### 8.3.2 The compare criterion

The following variables are used for the boundary point method:

- The **point distance**  $\zeta$ . This is the pitch of the grid.
- The **clearance**  $\xi$ . This is the maximum distance of two parts under study.
- The **criterion**  $\chi$ . This is the distance at which two points of a compare-pair are considered to be in contact.

The maximum distance from any point in a grid is  $\frac{1}{2}\sqrt{2} \cdot \zeta$  (within the plane of the surface, see figure 8.8; for curved surfaces the local plane perpendicular to the normal will do). Perpendicular to that surface, there may be a distance between the two parts: the clearance. The worst case combines these two perpendicular distances. The criterion is the maximum distance that must be accepted, so (see figure 8.9):

$$\chi = \sqrt{\left(\frac{1}{2}\sqrt{2}\zeta\right)^2 + (\xi)^2} = \sqrt{\frac{1}{2}\zeta^2 + \xi^2}$$
. This needs to be calculated once for each pair of parts.

The distance between two points, which must be calculated very frequently, can be calculated with:

$$\delta = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$$
. If  $\delta \leq \chi$ , then the points connect ( $\delta$ -relation). Most points will not be connected, so a less accurate and faster calculation of  $\delta$  would be useful. If  $\Delta x = \Delta y = \Delta z$ , then  $\delta = \sqrt{3} \cdot \Delta x$ . Suppose  $D$  equals to the largest of the three  $\Delta x$ ,  $\Delta y$ ,  $\Delta z$ , then must hold that  $D \leq \delta \leq \sqrt{3} \cdot D$  and the criterion

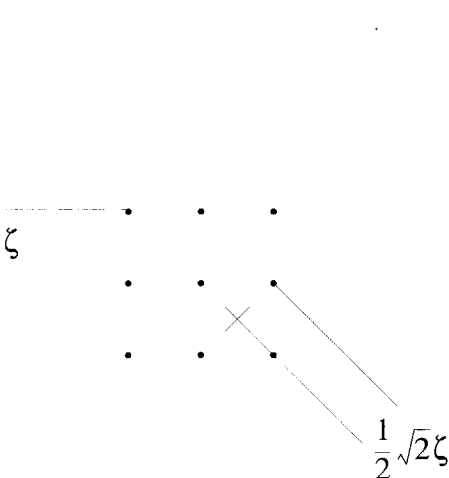


figure 8.8 The maximum distance from a point

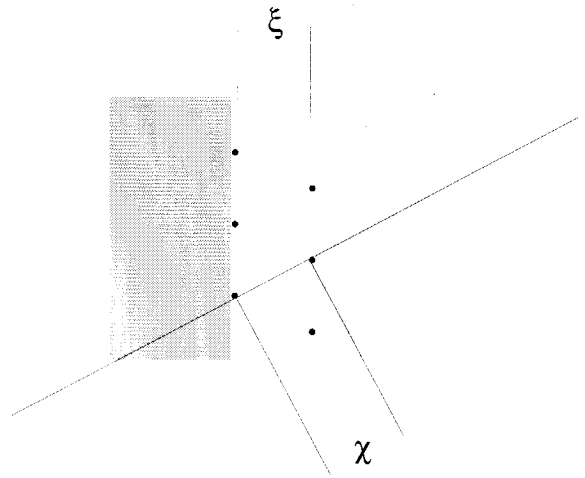


figure 8.9 Construction of the criterion for a pair of parts

relation answers to  $\sqrt{3} \cdot D \leq \chi \Leftrightarrow D \leq \frac{\chi}{\sqrt{3}}$  ( $D$ -relation). There will be points that

in fact do not answer to the criterion when  $\delta$  is calculated exactly, so if the  $D$ -relation holds then the  $\delta$ -relation must be checked also. Naturally determination of a largest number of three is so much faster than calculating  $\delta$  every time, that this comparison method is worthwhile.

### 8.3.3 Boundary point generation

Boundary points must be generated for vertices, (curved) edges and surfaces of every part in a product. The points must be kept with the original geometric elements that they were derived from, so a contact point will mark the element to be in contact. Also, some basic properties have to be stored for the obstruction construction, such as whether a surface point is close to an edge. The generated points are represented in the local part frame, so any displacements of the part do not influence the boundary points list. Comparison requires however that points are compared in one frame, so the position of points relative to the common frame of the product have to be calculated, preceding the comparison.

For vertices, a point is simply put on it. On edges, which are currently approximated with small straight segments, points are generated along the edge, a point distance  $\zeta$  apart.

Surfaces can contain inner profiles, which must be excluded (see example in figure 5.1 on page 51). Surfaces can be curved in one direction (such as the cylinder outer surface), so the generated points must follow the curve. Freely curved surfaces are either approximated in the Brep by tiles, or the exact equation must be known. Every point on the surface that is next to an edge, either of the outer or an inner profile, will be labelled as such. The distance between such a point and the edge is guaranteed to be larger than zero and to have a maximum of the point distance. The grid will be as square as possible, in order to control the distances to be less or equal to the point distance. However, there may be occasions where points are closer to each other than that.

### 8.3.4 Contact situations

After comparing all points of both parts with each other, the results of contact points and their geometric elements have to be interpreted. As said before, there will be more contacts found than actually exist. This is due to the fact that surface points are closer to an edge than the point distance  $\zeta$ , so if an edge is in contact, most adjacent surfaces are found to be in contact as well (see figure 8.10). This can be solved by discarding

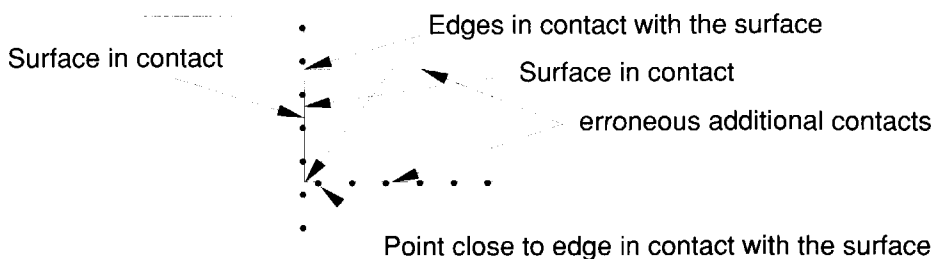


figure 8.10 Erroneous additional contacts found

contact surfaces that are caused only by points *close to an edge* or vertices. The obstruction will be built, using one of the surfaces in contact, preferably the one with the most points in contact. The obstruction type (PLANE, in this case) is determined by evaluating the shape of the surface. The significant points for the obstruction (required for testing valid motion directions) are taken from the edges, which represent the limits of the area in contact.

The recognition of cylinder or cone-type obstructions is similar, only the shape of the surface reveals the nature of the obstruction. It is important to know whether the angle over which the contact extends is larger than  $\pi$  (different obstruction type, see 8.1.2 on page 94). This can be verified by examining the points in contact, with respect to the center of the cylinder.

But what if the close-to-edge-points do represent a valid contact, such as on a rim (see figure 8.11)? This case will be solved by recognizing the edge to surface contacts. The obstruction will be taken from a surface in contact, so the contact plane will be correct.

The contacts of an edge or a vertex in contact with a surface are recognized easily with the boundary points method. The obstruction will be derived from the surface. The freedom that the objects have, rotating around the contact edge (see figure 8.12), is represented by the fact that the obstruction has significant points that lie in one line (the contact edge is projected on the surface, forming a twist axis). This is in fact the same as with the surface to surface, which can rotate around its edges as well, but with different edges as twist axes.

When just two edges are in contact, or even two vertices, they will be detected by the algorithm. The problem is that no surface is present to derive the obstruction from. As an approximation, a plane is generated which runs through the contact point and perpendicular to the Centres Of Gravity of the objects (see figure 8.13). Since the significant points are reduced to one point, the exact orientation of the obstruction plane is less important.

#### 8.4 Resume of the obstruction recognition

The obstruction recognition is performed in 7 steps:

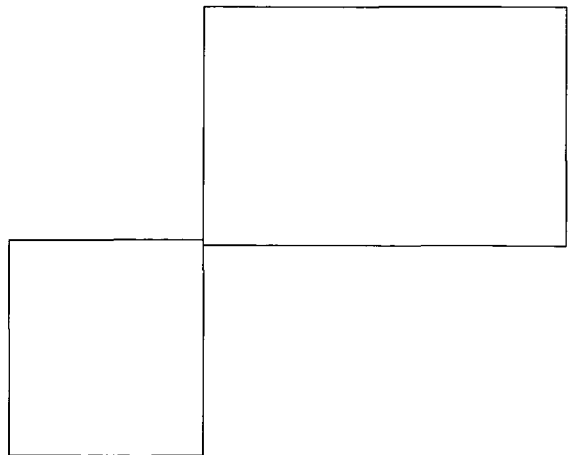


figure 8.11 Flat surface to surface close to edge

- The product description with all geometric parameters is retrieved from the database.
- A quick bubble check is done to see if the parts are close at all.
- The parts are covered with boundary points, generated on geometric elements, with a maximum distance apart.
- Points from two parts are compared in pairs with a criterion, derived from the allowed clearance between the parts.
- If there appears to be any contact, a connection entity is created in the database and the generation/compare sequence is repeated with higher accuracy.
- The elements in contact are evaluated in order to filter out non-existing contacts.
- Obstructions are built from the contacts, deriving the type and the parameters from the contact situation.

The list of obstructions for a connection between two parts is passed onto the procedures that combines and evaluates them, in order to generate an assembly path.

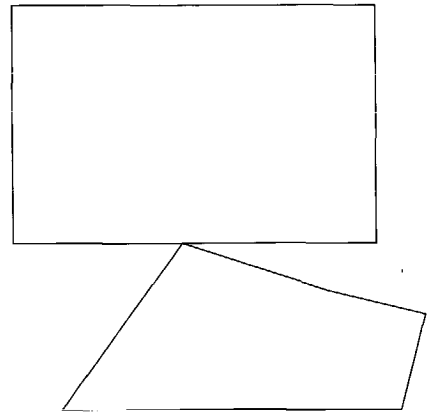


figure 8.12 Surface to edge

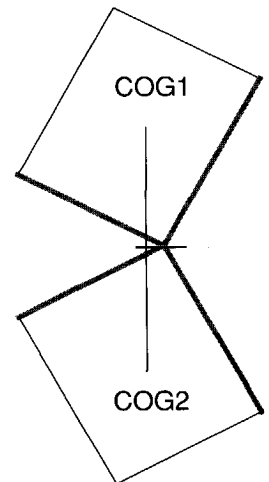


figure 8.13 The obstruction of an edge-edge contact is approximated by a small surface perpendicular to the line COG1-COG2

## 9 Connection Synthesis

This is the final chapter on the connection analysis. In the previous chapters, the tools have been developed to determine a direction set for a given contact situation. The direction set has to be applied in order to investigate possible insertion paths, the insertion point and the approach direction set. This completes the connection model and therefore the product model for assembly.

### 9.1 The connection model attributes

#### 9.1.1 Applying the relocation direction set

Once a relocation direction set for a given situation is found, it must be used to find a next position. The procedure is repeated for the new situation. Alternatives may arise as more directions appear to be valid.

The direction to move in is dictated by the relocation direction set. This may contain more than a few discrete values, in which case a selection must be made to investigate the nature of the alternatives. A larger relocation direction set may indicate that the current position is in fact the insertion point. In case of several discrete values, they all must be applied to determine a path out of the assembly. In case of a set of directions (an area within the  $(\tau, \theta)$  diagram), the extremes and significant directions are to be checked.

Moving a part along a direction will cause changes in the obstruction configuration. Some contacts are directly lost: the part moves away from it. Others remain during the motion: the path is defined along this obstruction, so it slides. These sliding obstructions are ignored during the motion, while any new contacts will be recognized as a significant change of state, which needs to be analysed again.

Initially, the procedure determines a relocation direction set from the obstructions in the assembled, final position (see figure 9.1). The direction found, which is the only one possible in the shown situation, is applied and by using a sweep, the travelling distance is determined (figure 9.2). When a blocking element of the counter part is found, a new obstruction analysis is done, in order to determine the new direction set (figure 9.3).

#### 9.1.2 The insertion point

As soon as the procedure results in a collision free direction, it stops. Apparently, the part has become contact-free from its counterpart and can be disassembled with the found (sequence of) direction(s). This last direction is the main assembly axis, which is stored as an extra attribute. The Insertion Point can be found by examining the last valid obstructions, it can be determined where they are no longer in contact.

Somewhere along this last direction, all obstructions are lost: the contact ceases to exist (see figure 9.4). This candidate IP must be checked, to see if the last motion does provide a collision free path, away from the counter part (figure 9.5)

If there appear to be no collision free directions out of the IP (see for example figure

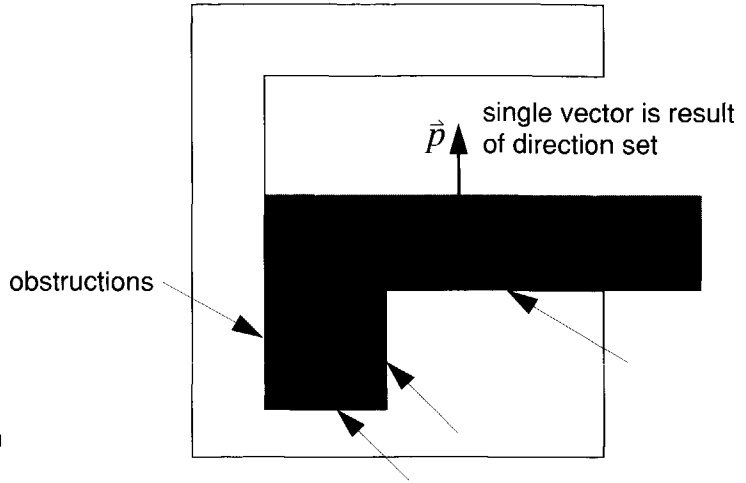


figure 9.1 Initial situation

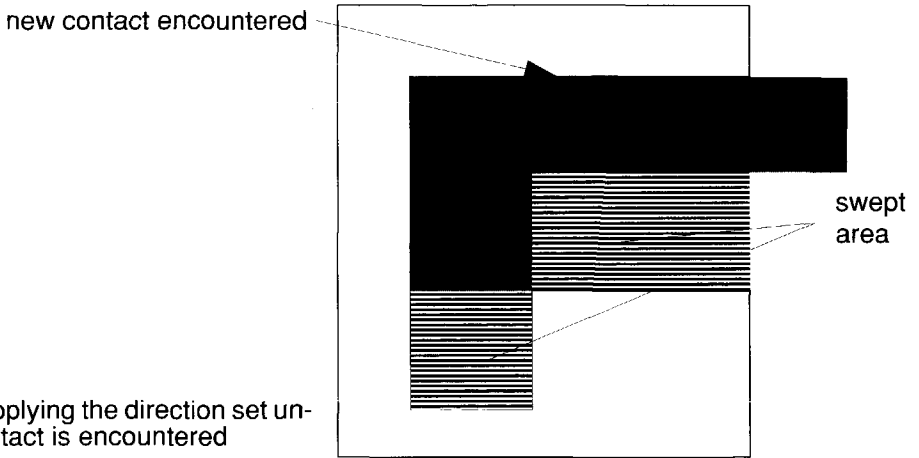


figure 9.2 Applying the direction set until a new contact is encountered

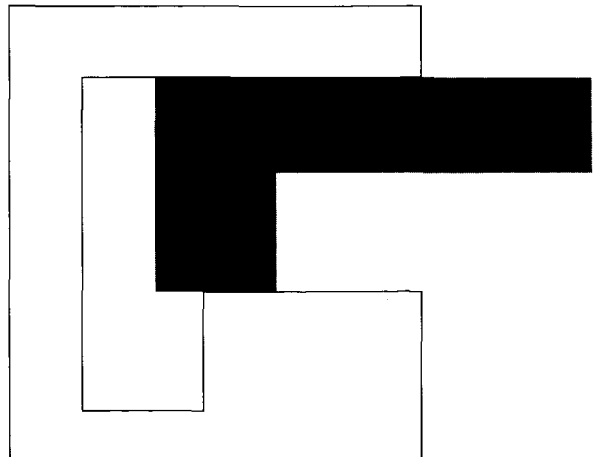
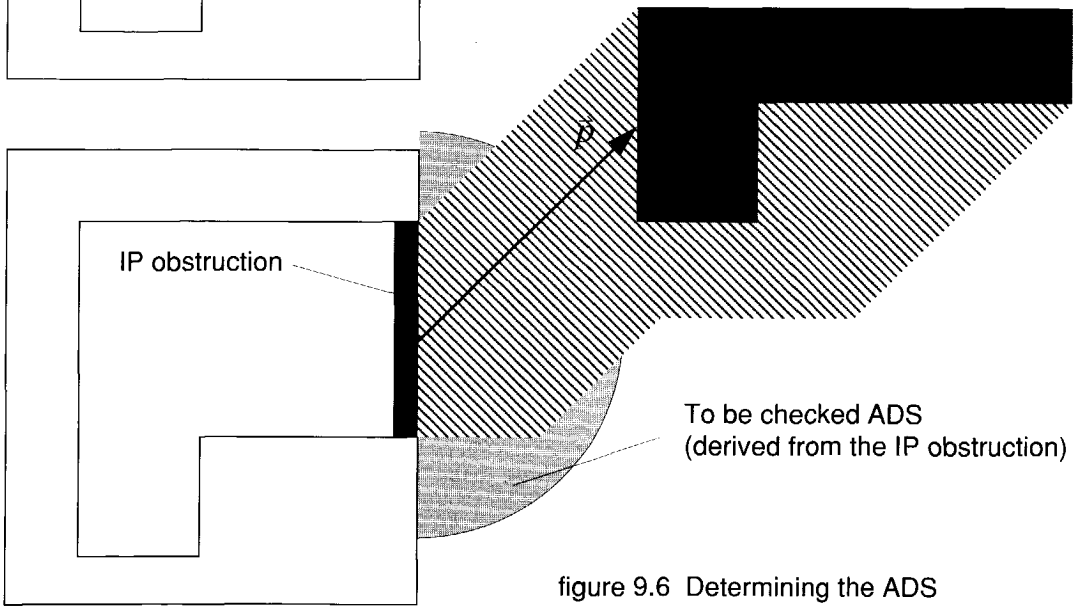
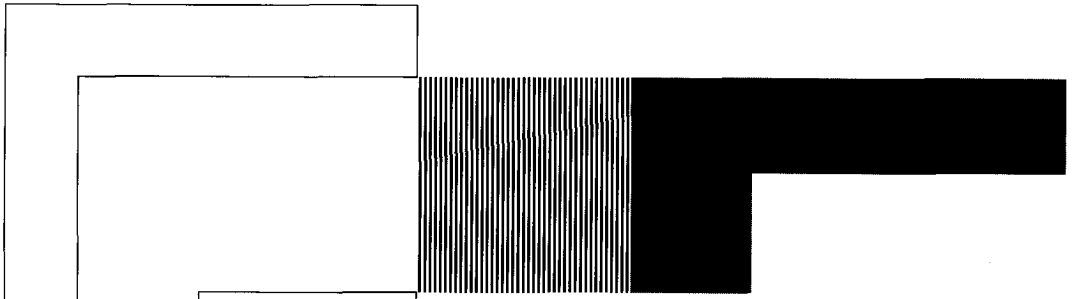
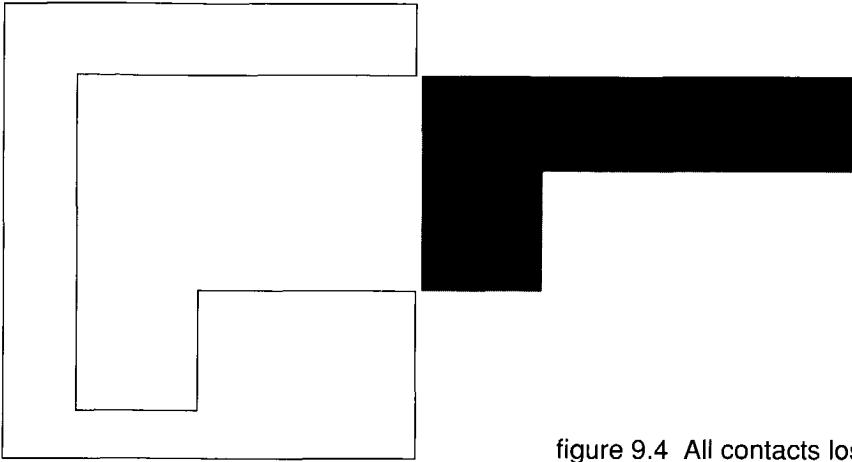


figure 9.3 Applying the new direction set.





4.5 on page 46), the IP will be discarded and the found direction set will be applied in order to generate a new situation. A collision will provide a new obstruction and therefore new insight how to leave the counter part.

### 9.1.3 The approach direction set

The IP can be reached from outside by a range of paths: the approach direction set. In theory, this is a full 5D relocation direction set, including rotations. This description is however not practical. Since the IP is already assumed to be (almost) 'outside', the approach motion is not likely to contain any rotations. This reduces the problem to a translation direction set, described with one  $\alpha, \beta$  diagram. Note that since rotations are not considered, no support points have to be established, and the IP can be the reference of the  $\alpha, \beta$  diagram of the ADS.

The ADS can be found by determining a new direction set in the IP (see figure 9.6). However, since there may be no more contacts in the IP, an obstruction must be generated, a plane perpendicular to the last direction, so moving back cannot occur. This will be called **IP obstruction**, and is always located in the IP, perpendicular to the last direction. Normally, moving back is prevented with discarding the direction that the part came from, but without obstructions no relocation direction set can be calculated. Also, all directions must be totally collision free, so there are other restrictions than with the normal direction set finding.

The direction set found in the IP will by definition contain an area of directions to move in. This area is thoroughly checked for collision free directions to move away, by performing the sweep analysis for several directions (see figure 9.6). The example in figure 9.7 shows that not all directions that are allowed by the imaginary plane obstruction are collision free: the peg may not collide with the rim of the hole.

## 9.2 Performance and limitations of the connection modelling procedure

The connection modeller works well for the common connections, almost all of which are to be reduced to a kind of peg/hole connection. The assembly path almost always consists of only one component, a straight line along the Z-axis, sometimes a different angle and sometimes with a thread. The connection modeller handles these routine matters well. The assembly path, the insertion point and the approach direction set are all generated correctly. One can argue on the ideal placement of the insertion point, for it influences the approach direction a great deal: if the IP is further out of the construction, the approach possibilities grows rapidly.

As stated in the justification of the project, routine tasks such as programming a robot for a common type of connection would be tedious. It would be even an annoyance if a routine task becomes complicated when sensor interaction has to be integrated. So, when able to handle the routine jobs, the CAD/CAM system is moving in the right direction.

Several limitations can be identified:

### **Thread**

When the product model is being built, the system scans the object drawing for annotation that seems interesting (everything but a plain dimension without any tolerance). If the dimension appears to belong to a cylinder and it starts with the letter **M**, it assumes a metric thread. The pitch of the thread is looked up in tables, as a

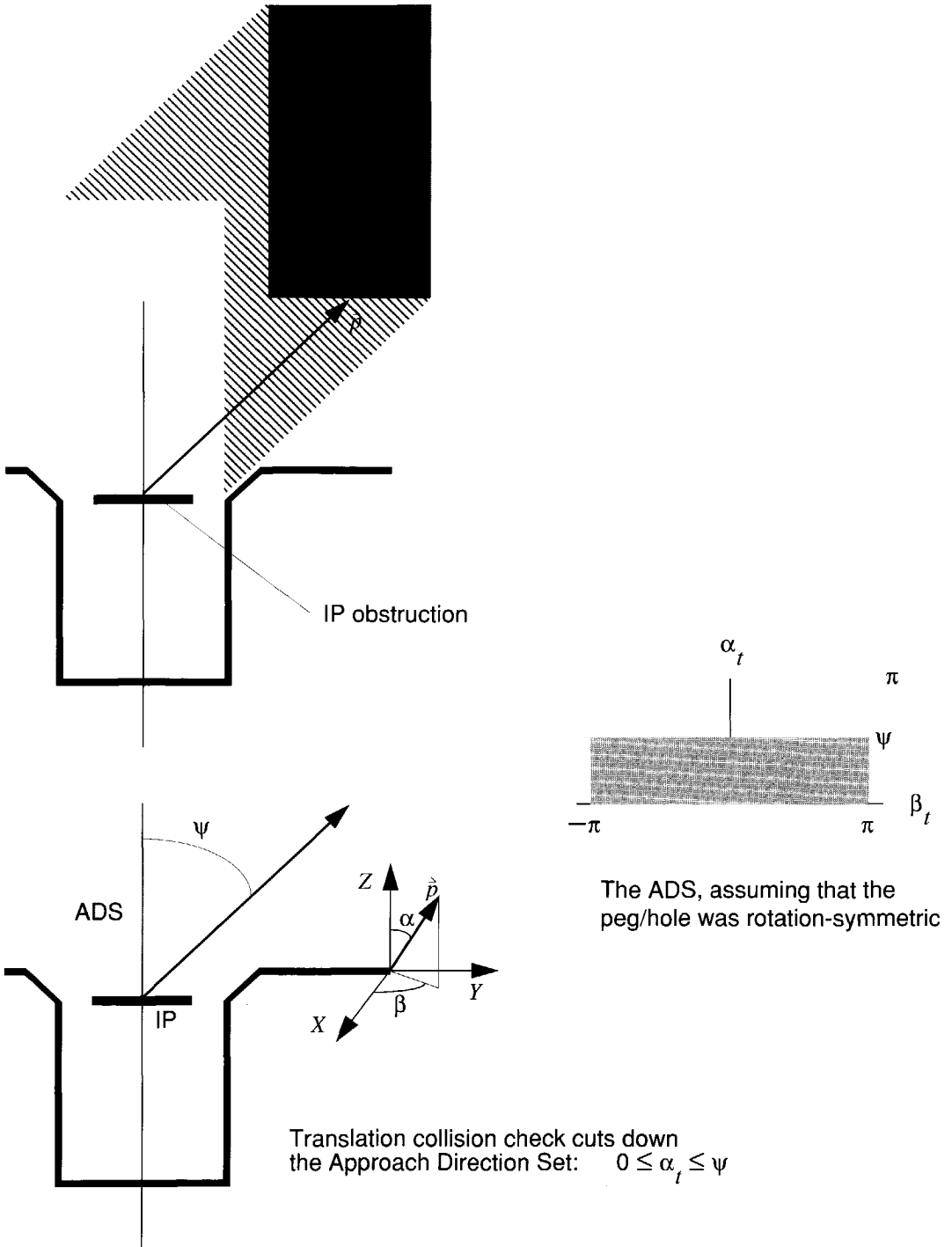


figure 9.7 Limited ADS

function of the diameter. The connection modeller takes this information for the  $\left(\frac{\delta r}{\delta t}\right)$  proportion, and starts calculating with the prescribed screw motion.

Currently, this is the only information on thread that is processed. Other types of thread with a different diameter/pitch proportion and another form of dimension text is not recognized. Of course it would not be very hard to extend the system at this point. The problem would be easily solved by a standard way of representing thread in the product model.

### **Deformation**

A serious limitation of the connection modeller is the inability to cope with deformation. Deformation of materials is the basis of the modern snap-fit connection, which will be applied more and more. Floppy parts such as seals are almost always assembled with special tooling, so user interaction is always required.

Calculation of the mechanism of a snap-fit connection requires knowledge of the material and the behaviour of features under various attempts to assemble them. Disassembly may very well require a complete different path than assembly. To solve the problem of intentional deformation - elastic or even plastic - in assembly, more research is required.

### **Large freedom in the assembly path**

If somewhere in the assembly path an 'open space' is encountered and further on the path narrows again (figure 9.8), the path finding algorithm may easily fail. The search is not guided through the construction: contact is lost and there is no way that the algorithm can find out where to start again. If some obstruction would narrow the gap and lead directly to the next step in the path, there would be no problem. The user could aid the connection modelling by adding a fictional obstruction.

### **Usability of the approach direction set**

Having a starting point for fine motion planning in the form of an insertion point and approach direction set is likely to be useful. However, this is still not proven. Only after experimenting in the context of the follow-up research, it will become clear what



figure 9.8 Widening assembly path

the benefits of the taken approach are. The fine motion planning will still have to go into the details of geometric reasoning, in spite of having a good starting point. Nevertheless, the automatic assembly sequence planning has already showed to need the information on a approach set.

#### **Access constraints for assembly sequence planning**

The product model for assembly only contains information on binary connections: it does not concern itself with parts with multiple contacts. A given situation in which a number of parts are already assembled, yields a new problem of approach of a next part to its assembly position. The approach directions and insertion points of its counterparts are known however, so a new combination analysis can be performed. The intersection of the approach directions of all present counterparts give a new approach direction, which is as a new, complex counterpart. If the approach direction is empty, this sequence is not possible.

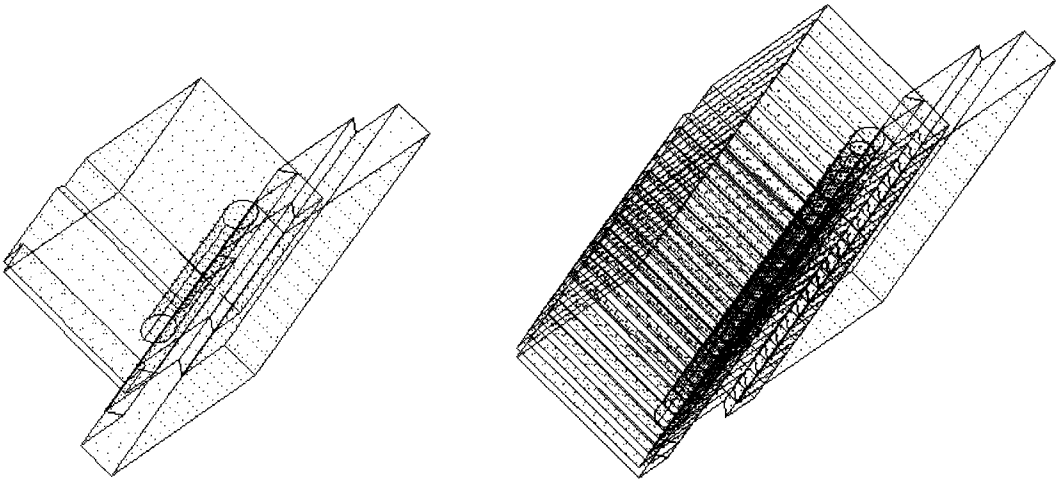


figure 9.9 Visualisation of the sweep from IP to assembled position.  
Partial screen-snapshots of program guif analysing Diac2



PART 3  
EPILOGUE

# 10 Evaluation

## 10.1 Overview of the implementation

The current implementation is a working CAD/CAM system, producing a product model and a connection model. It runs on SUN workstations (SUN3 and/or SPARC) and uses the following software subsystems:

- The Medusa CAD system from CIS/Prime. The interactive 2D/3D system is used to create product drawings and 3D models. Several utilities (MEDMIF, MedProp, Bacis2, HIP) are used to extract drawing and model information in a readable form.
- The Ingres relational database management system from Ingres/ASK. The product model is implemented as a set of tables, except for the geometric models which are kept in structured files created by Medusa (MIF). Several additional supporting tables

action/statistic	Product <sup>1</sup>				[]
	DIAC-1	DIAC-2	DIAC-3	Cranfield	
Read product from sheets <sup>2</sup> (on SUN3)	7.8	19.8	11.0	6.3	min.
Generate geometry <sup>3</sup> (on SUN3)	12.1	34.0	10.8	7.2	min.
Analyse contacts (on SPARC)	144	1134	204	861	min.
Build connection model <sup>4</sup> (on SPARC)	282	311	29	529	min.
Build product description (on SPARC)	0.9	2.7	1.4	1.2	min.
Number of parts <sup>5</sup>	21	25	16	19	#
connections <sup>6</sup>	39	70	5	35	#
avg. surfaces/part	24	31	26	37	#
avg. generated boundary points /part	306	616	460	583	#
simple <sup>7</sup> assembly path	39	70	3	35	#
compound assembly path	0	0	2	0	#
automatic connection models created	37	70	3	35	#
user-specified connections	2	6	2	0	#

1. see for a description of the products, appendix A. on page 131.

2. initially, without having to update database information; includes the Medusa interface utilities

3. includes the Medusa MIF generator and a currently very inefficient circle assignment algorithm

4. this includes the sweeping and repeatedly analysing the new contact situation

5. parts inside clumps counted

6. clumps regarded as one part

7. simple in sense of only one transformation, so includes for example screwing

figure 10.1 Table of results of the CAD/CAM system

are used to store intermediate results and additional data on the geometry.

- The product modeller (refer to figure 6.1 on page 66: processes *1 Read CAD sheets* and *2 Generate geometry*). This is a set of programs, written in C and Basis2 (the Medusa programming language), that interface to the CAD system and the database. The Medusa utilities are called to write specified drawing information to files, which are then interpreted. If assembly drawings are found, the system recursively descends the product structure and interrogates all drawings belonging to the product.
- The PDM interface (*5 Build product description*). On request, a full product description, including geometry, can be built in memory, available for processing by other software. Whenever a drawing appears to be updated since the last interpretation, the parts of the product that are newer will be re-interpreted by calling the product modeller.
- The connection modeller (refer to figure 6.1 on page 66: processes *3 Analyse contacts* and *4 Build connection model*). These programs implement most of the theory described in the previous chapters. It takes the product description, analyses it and stores a connection model in the database. The implementation shows the feasibility of the theory but has not been made specifically complete, robust or fast. Therefore some parts are rather slow and/or simple. The direction sets are only fit for either translation, rotation or a fixed proportion between the two. Only two out of 151 connections in the four tested products required a compound assembly path. Although the algorithm is capable of handling those, it was not incorporated in the prototype. However the general case of arbitrarily placed obstructions is dealt with. Only the obstructions were implemented that are present in the four tested products (plane, cylindrical face, cylinder and thread, see appendix B. on page 139). There is no optimization of the best insertion path.
- The interactive graphic tools (process *6 Graphic User Interface*). Several tools are provided to show graphically what the results are of the product and connection modelling (based on information stored in the database). The actual graphic user interface allows a user to change data concerning the product and connection model.

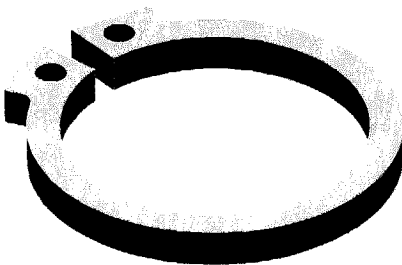


figure 10.2 Circlip

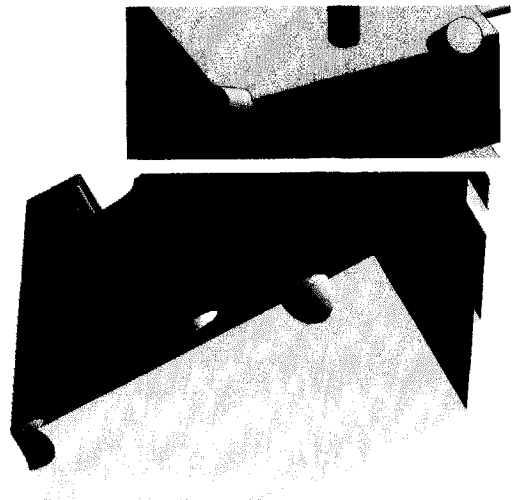


figure 10.3 The oblique pegs and holes in DIAC-3



## 10.2 Results

The product modeller appears to be fairly reliable when it comes to processing products. There were problems with the Medusa utilities and the accuracy that Medusa works with, so the system could not determine the correct position on the drawing of some features. But after a period of testing with several complex products, the product modeller works well.

The product modeller needs a product to be fully specified by its top level assembly sheet and will interpret it batchwise. This means that errors will be found only when processing the drawings and models. However, when no errors are found, the software will run unattended. The connection modeller runs unattended as well.

The connection modeller can only handle contacts with the predefined obstructions. When obstructions are encountered with curved surfaces or when deformation is required to (dis)assemble, the connection modeller fails. It does however show that automatic interpretation of common connections between parts in a product is feasible.

The performance of the programs and some statistics are collected in the table in figure 10.1 on page 111. The first section shows the execution times for the processes, except for the interactive GUIF. Since the Medusa utilities are currently only available on SUN3, the two CAD interfacing processes are logged on a SUN3, while the database runs remotely on a faster SPARC station. Note that the connection analysis processes take a long time, which depends on the complexity of the product (indicated by the average number of surfaces per part) and of course on the number of parts. The average number of generated boundary points per part indicates the size of the compared areas.

Remarkable is the small number of connections that require more than one transformation (compound assembly path). Only the DIAC-3 product, that has been

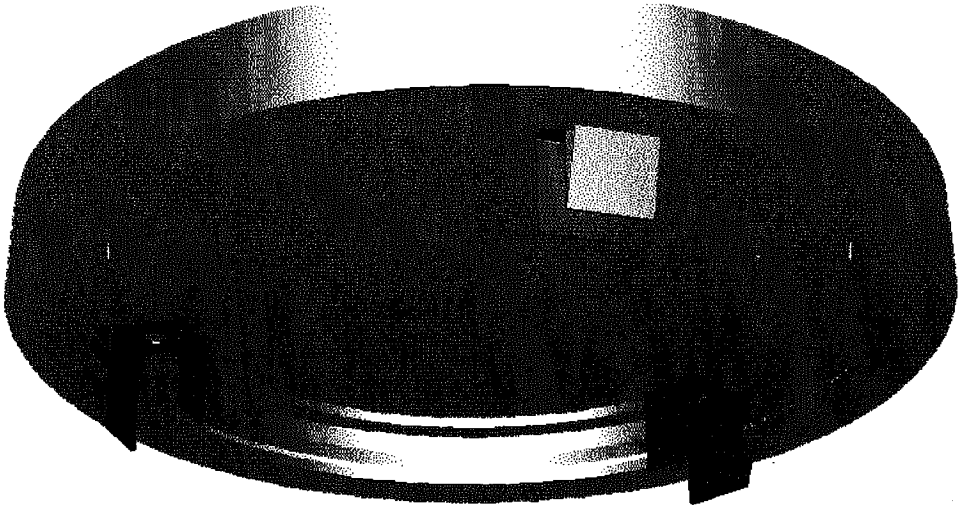


figure 10.4 The lid of Diac-2 with the three springs

designed especially to test some complicated connections [Donselaar 90], contains one three-component assembly path. The one two-component path connection of DIAC-3 is a snap-fit, that could not be handled automatically in the first place. Therefore the algorithm to find collisions by sweeping (see subsection 9.1.1 on page 103) was not implemented.

The connection modeller stumbles, as expected, on parts that are elastically deformable. In DIAC-1, there are two circlips (see figure 10.2) that holds the axes in place. The user will have to specify the connection model once and copy it to the two failed connections. This part would need special tooling anyway. In DIAC-2, the lid is mounted on the casing, held by three springs. The connection lid-spring (three times) is correctly modelled according to the notion of the connection modeller: it does not find any problem in removing the spring (see figure 10.4 on page 113) because the casing is not present. The sequence planner will find that the springs cannot be assembled with its screws when the lid is already mounted. Here we find a serious flaw: the sequence planner will have to go back to the connection modeller to correct the error.

The concept of clumps influences the figures of the table. Parts are counted, regardless of whether they are in a clump, but connections between parts in a clump are not analysed. The parts in a clump come to the cell as one, so the connection modeller treats them as one as well. Examples are found in the two pegs that are pressed into the casing of DIAC-2.

Note that the pegs in DIAC-3 that are to be mounted with an oblique angle (see figure 10.3 on page 112) cause no problem. Neither the angle of the assembly direction nor the odd-shaped edge of the hole) disrupt the connection modeller. The snap fit peg (figure 10.5) again requires deformation, and is therefore not modelled.

### 10.3 Comparison with related research

The related research survey showed that previous connection analysis algorithms suffered from the following drawbacks:

**Either:**

- unable to combine connection features
- Limited recognition and combination due to main-axis approach (regarding only properties along axis of orthogonal coordinate frame)
- Limited by making a classification of the connection.

**or/and:**

- unable to accept parameterized geometry
- unable to accept drawing attributes

The proposed system has limited ability to accept drawing attributes. It is unique however in the ability to look for and interpret designer's meanings with connections. It 'follows' the contact topology by assigning seeking and imperative properties. The designed features of a product that are meant for assembly, get translated to attributes that direct the search for an assembly path.

The obstruction math is a quite general and flexible method to determine the degrees

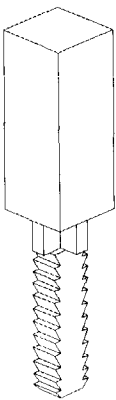


figure 10.5 The snap fit peg of Diac-3

of freedom of an object and generate a number of possible travelling paths.

## 10.4 Conclusions

### **CAD interfacing will remain a problem for applications seeking a product modeller**

Commercial CAD systems are not very fit to make capable interfaces to, but it appears to be possible. All drawing information can be extracted and stored in a product data model. The product modeller is very specific to Medusa. However, all commercial CAD systems will suffer from poor interfacing facilities, one way or the other, so a general product modeller will only be possible when a standard such as STEP has been accepted (see next chapter for more comment on that). The exercise of creating a product modeller for a specific application and with a specific CAD system shows the problems that may be expected. It is not likely that these problems will be solved completely when a STEP based CAD system is introduced. Every system might use a specific subset of STEP and give its own interpretation of STEP's *applications* (see subsection 3.3.2 on page 26).

### **The connection model can be a valuable extension to a general product model.**

The connection model as an enhancement to a product model provides a de-coupling of fine motion planning and sequence- and coarse motion planning. It can be derived from the CAD-extracted data, without any knowledge of equipment that is to be used for the actual assembly. Since the model is rather simple for a human to understand assembly properties of the product. It can also serve as an intermediate to various complex assembly planning software. It could be incorporated in STEP as the application 'Assembly'.

The automatic synthesis of an enhanced product model is a good tool for automatic assembly planning for sensor-based robotic systems. It makes CAD/CAM for assembly more feasible by defining a capable stationary model between CAD and process planning.


### **Connection analysis with obstruction combination handles most connection situations.**

For the relatively simple connection situations that occur frequently in industrial products (rigid, well-designed parts), the connection modeller performs well. However, very few products will *not* contain some connection that can not be recognized, so fully automatic assembly planning will remain not feasible for the time being.

## 10.5 Recommendations for further research

A future view on product modelling can be found in the next chapter. The connection modelling as presented in this thesis is a technique that can be applied if further developed. Some aspects have not been implemented fully and more testing should be done before the technique is mature. Furthermore, since the performance in time is rather poor, optimizations should be explored, depending on the application.

The current user interface of the product analysis system is stand-alone, while other applications in the assembly planning and control require user interfaces as well. Since the product model and the product model based graphics are the same, integration of the user interfaces is possible. However, consistency towards the user should be paid



attention to and the functionality should be carefully redesigned.

The application area of connection analysis needs to be explored more thoroughly, in order to direct the research. Two area's can be identified:

- As a tool for automatic assembly analysis
- As an assembly evaluation tool for product design.

Automatic assembly analysis is yet to prove to be valuable. Only when the current advanced robot programming tools are really commonly used and the number of application grows, then automatic assembly planning will become interesting. Tools such as the connection analysis algorithm may be used then to program the sensor-rich assembly machine. The basic idea of obstruction combination may solve specific problems of the programming system.

# 11 Future CAD/CAM techniques

## 11.1 Introduction

The research described in this thesis aimed to solve some of the problems that arise when using CAD data for assembly planning. For a number of reasons, the CAD system at which the design process takes place, was regarded as a black box, that produces data on a fully specified product. The interpretation of this data produces a new product model with extensions on assembly characteristics. This action was referred to as pre-planning, an extra step to facilitate the assembly planning: CAD/CAM for assembly planning.

Trends towards integration will also affect the boundary between CAD and CAD/CAM. In the case of CAD/CAM for assembly, the system can benefit from making the assembly characteristics more explicit in the extended product model. The assembly pre-planning can be integrated, using the techniques on geometric reasoning to show the user assembly effects and implications of the current product model.

This chapter gives the author's view on the future integration of assembly modelling and interactive product modelling. Especially the issue of data management requires special attention.

## 11.2 The product model

### **Manufacturing Data**

In the architecture presented in the second chapter, a product specification is interpreted in order to produce the manufacturing data necessary for producing the product. Functionally, the manufacturing data belongs to the product model. The main entity is a product and the manufacturing data is only valid for producing and assembling/manufacturing the parts of **this** product. In this project, the manufacturing data consequently has been joined with the product data in the product model.

Integration of functions requires a common usage of data. In order to be able to store, exchange and interface to the product model, it should be a standard definition such as STEP. Integration will cost too much if every application is to be adapted to a certain non-standard model.

However, it is very unlikely to expect a product model standard definition to incorporate all kinds of manufacturing data. This data is very specific for the available equipment and manufacturing technology. A more realistic view would be not to define the manufacturing data with the product model, but to provide handles in the product model to attach to. This poses problems on the data-management: whenever a product has been changed or even removed, what consequences does it have for the manufacturing data? Therefore, from the viewpoint of data management, both types of data should be regarded as an integral part of the product model. See figure 11.1.

The product model STEP has provisions to store application data such as calculations and process data. The results refer to (parts of) geometries that are also in the model and is therefore an extra set of attributes to the geometry, see also section 3.3. For

example, the assembly planning could use this information for locating elements that are meant to deform during the assembly (snap fit), if so indicated by the designer.

### Product data

The product model is coherent: most data is attached to main entities such as products and parts. Therefore a hierarchical, object-oriented approach can be taken to implement the product database. This type of data-organization has great advantages in searching and maintaining the data. Moreover, this principle should be consequently followed in describing the product. At the top-level, a product and its parts form an hierarchy of objects, to which the Bill Of Materials (BOM) is attached. The geometry is as an attribute to a part. Data-elements such as tolerances refer to elements in the geometry and should be treated as attributes to those elements.

Annotation in a drawing specifies cryptically something about a *quality* of a geometrical element. Therefore these specifications should be attached directly to the three-dimensional geometrical elements, see figure 11.2.

Attributes attached to an element can refer to higher levels in the object-hierarchy as well: an attribute such as material is attached to the complete geometry, so every part of the geometry is supposed to be of that same material. When the scope of an attribute is defined as, say, the complete geometry, it will be *inherited* by all existing and new elements *belonging* to that geometry, if not defined otherwise (overrides).

Take, for example, the dimension and tolerance of a cylinder. In 2D, this will be drawn as a dimensioning construction on a circle or rectangle representing the projection of the cylinder. In reality, the dimension and tolerance are attributes of the cylindrical feature. This poses constraints and demands on the cylindrical surface such as shape and dimension, but also on the circular face at the end of the cylinder. Characteristics such as roughness, colour and shininess (for a 3D shaded viewer or for a vision system in manufacturing) can be attached to a particular surface directly or to a complete object.

The data management appears to be an important issue. The next section presents an architecture that can deal with the product model data management.

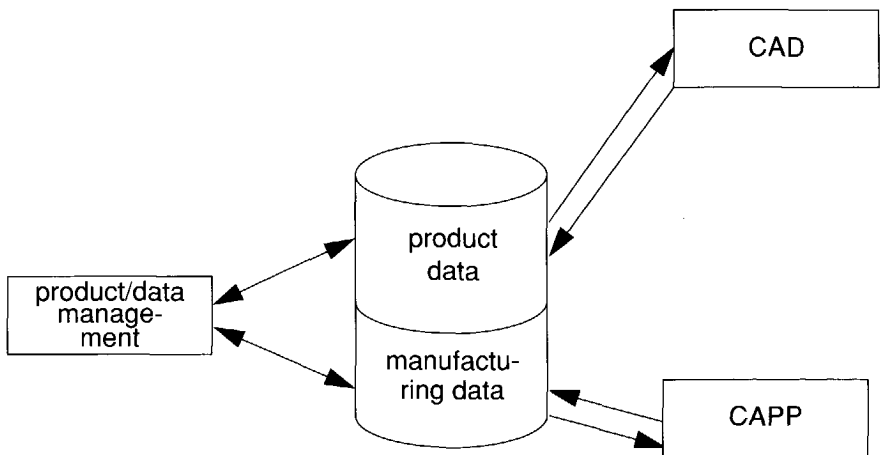


figure 11.1 Integration of manufacturing data

## 11.3 Data based product modelling

### 11.3.1 Database management

Maintenance of databases in general is a task that should not be underestimated. In particular a product database requires special attention, because of its size and strategic value for the company. It is very important for a company to have a manageable number of 'live' partnumbers and a minimum number of obsolete parts. For service on old products it may be necessary to keep some parts in the system that are not manufactured anymore. Also, it is very desirable to have designers re-use parts that have been created previously. This requires an on-line database that is well-maintained: up-to-date, easy to access and to expand. Accessibility is a problem, for it is not obvious what to search for: name or identification is usually not known, just function and global form.

A person or department in charge of the product management has the authority to issue part numbers and authorizes the introduction of new products into the system and has responsibility for the contents of the product database (also see figure 11.1). When removing a product or part, the manufacturing data should be removed as well.

Whenever an object (geometry) is called (instanced) into the assembly area, it should be introduced to the part-tree of the product. The other way around, if an entry in the product structure is edited, the changes should be reflected in the physical assembly. In general, creating and maintaining the BOM and the assembly structure is done primarily by the designer, referred to as product structure management. This requires a very complex and capable database system, which must be integrated with the system, but with very user-friendly interfaces.

### 11.3.2 Data working levels: a proposition

The user must be able to access and edit all other data in a comprehensive and structured way. A geometric modeller/editor/display system is required to manipulate the geometries. The data editor could be a generic database interface, but this is user-unfriendly. The complexity and coherence of the data requires special purpose editors

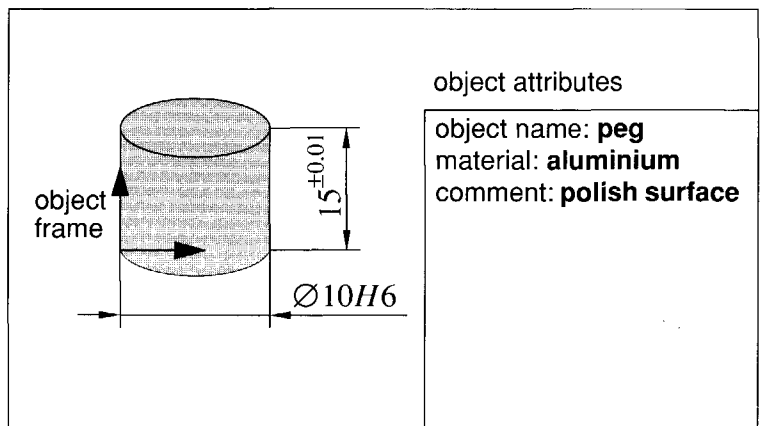


figure 11.2 Object specification with attributes and annotation

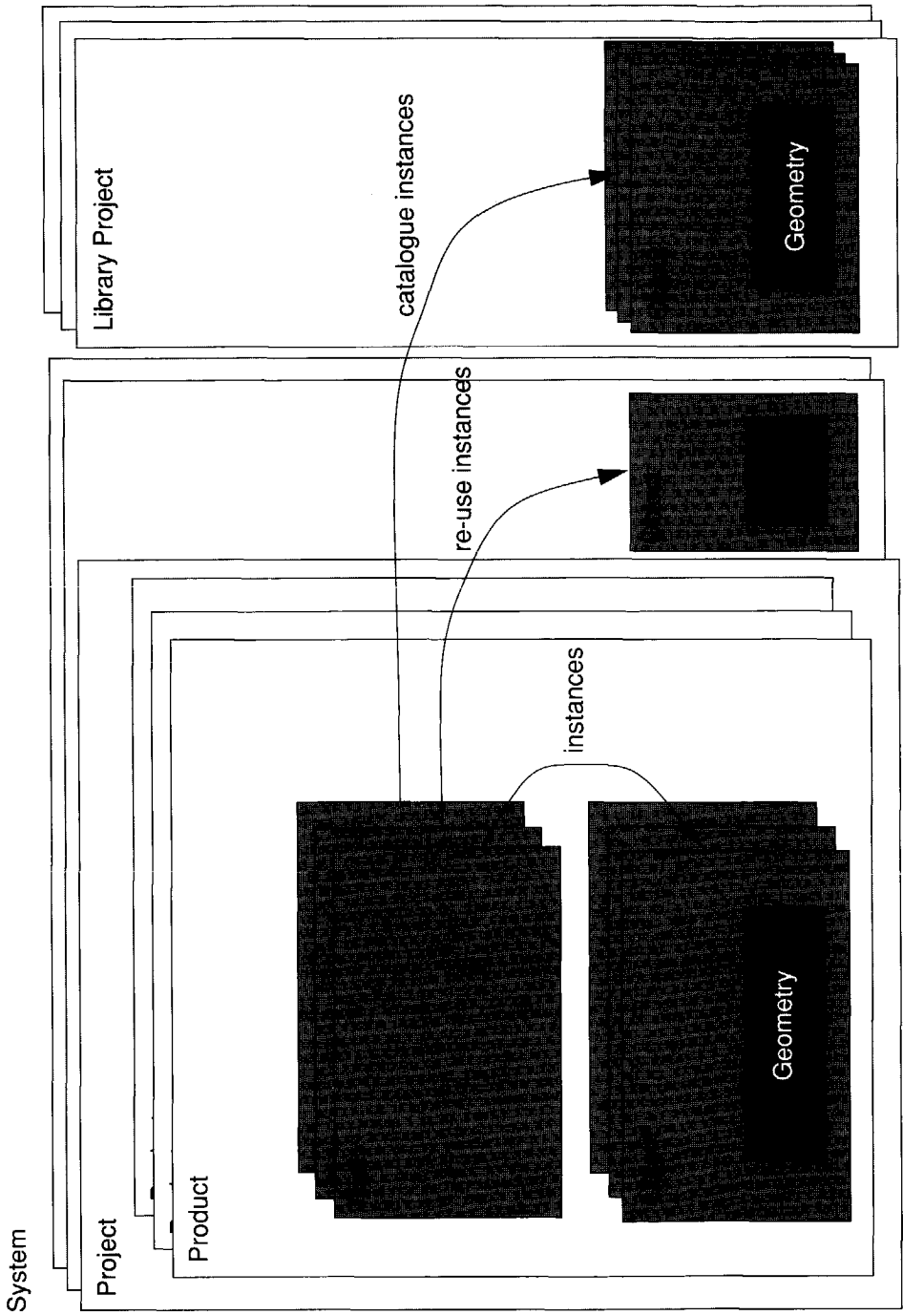


figure 11.3 Schematic representation of CAD system working levels



as well. We need to make distinctions between definitions of original items (such as an object), instances (the part is an instance of an object), references (a product is made up of a collection of parts) and proportions in the relations (see discussion in chapter 3). It is very hard for a user to find his way in this data jungle, so a structuring is called for. The data can be edited in six levels, each of which gives access to specific data, privileges to change it, and an environment (context editor) for the user interaction.

The proposed six working levels are (see figure 11.3):

- System: the level for the System manager/ Application Manager/ Database Administrator. Allows global access to data.
- Project: an environment that holds predefined defaults for all attributes and manages and gives access to all data levels, such as all products within this project.
- Product: the product structure contains the subassemblies and parts in a particular product. A subassembly and a product are not really different: a product may be incorporated by another product. They are tied together by the project. Entities on this level (the products) are not usually shared among projects.
- Part: definition of part attributes, which distinguishes it from physical objects, such as a position. Not shared.
- Object: the physical objects are the entities in the system that define the geometry. Physical objects are defined and maintained within one project, but sharing is encouraged: reusability. However, changes to shared objects should agree with all projects in which they are used.
- Geometry: basic entities that make up the geometry, such as CSG-primitives, surfaces and edges. Sharing is defined exclusively by the system for the callable primitives (basic drawing operations). Editing only via the geometric modeller.

With this structure of working levels, the system can be managed in a precise and logical manner. The problem of reusability of objects can be dealt with, by providing sophisticated search facilities on the object level. The search conditions are not exact: there is no one-to-one match for an object. The user may specify vague information on function, application, global dimension, indication for a name. A set of keywords and a short description of functionality to every created object will make it more reusable, analogous to a literature search. The search facility probably must be equipped with techniques such as language interpretation, dialogue ((multiple choice) questions asked by the system) and geometric reasoning.

The working levels should not be confused with the product data model. The product data model defines the data-entities and the types and relations of attributes. The working level defines model how *access to that data is managed*.

A special case of object search is the use of catalogues. Standard parts such as screws and bearings will be purchased and are to be selected from a catalogue provided by standardization institutes or parts suppliers. The company might have policies on preferences and prices, so these catalogues must be adapted. Also, the company might have a private catalogue of preferred in house manufactured parts, that should be available to designers. These catalogue parts are in fact objects, which can be instanced as parts in a product. The structure above would allow each library of catalogue parts to become a project, in which no products are defined, but only readable objects.

An important feature is working with empty items. Compare product design with writing a book: often the table of contents is created while the chapters and sections

do not exist yet. Empty parts, subassemblies and products must be manipulated while there is no definition present. Due to the description above, this was already inherently true for projects, for there do not have to be products present. This allows, for instance, a project manager to set up a project with empty subassemblies and/or parts in which several people work on different parts of the design.

## 11.4 Assembly design

Working with assemblies and complete products has created the need for a clear data management, as is described above. This section will discuss the specification of assemblies by a designer.

### 11.4.1 Design for assembly

First, the distinction must be made between **Design For Assembly (DFA)** and **assembly design support**. DFA is a method based on rules to make the product easier to assemble, for instance:

- reduce the number of parts in the product.
- either well recognizable and distinguishable part-features or indifferent (multi-symmetric) parts.
- feedability of parts (non-tangling, distinguishable).
- modern assembly methods: snap fit, less fastening features (self-locking).

Boothroyd [Boothroyd 82] is the most well-known advocate of DFA. Most alternative methods are derived from the original ideas by Boothroyd. Assembly design support is the active support the designer receives from the system while specifying assemblies. DFA could be implemented as one of the mechanisms to automatically help to evaluate assemblability, besides mechanisms to determine assembly sequence or generate/adjust assembly related geometry. In that case, assembly design support uses much of the techniques that are developed for assembly process planning.

### 11.4.2 Assembly design support

Active support by the system can help the user with specifying the product. An implementation of a straightforward standalone assembly support system (created by the Dutch TNO/MI) holds a database of assembly methods and -parts. It can be queried and an assembly method can be suggested by the system by specifying the characteristics of the needed assembly. It can come up with a method such as welding or with a kind of fastener such as a screw.

If the product model is extended with assembly characteristics, the CAD/CAM system can access and edit these as well as the other product attributes. Even more, calculations concerning the assembly can be used during the design as a tool for evaluating assembly properties of connected parts.

Concurrent design of the product and the complete assembly process is not feasible. Too much of the assembly process depends on the assembly system (the robot and its peripherals). However, the pre-planning described in this thesis merely involved properties that can be derived from the configuration of parts and their contacts.

The obstruction concept assumes that every contact area has a meaning for the assembly of the parts. During the development of parts within a product, this meaning can be made explicit and the consequences can be shown directly. The obstruction analysis can be an evaluation tool for the designer.

The obstruction analysis algorithms (which are fast enough for interactive use) can show how the specified assembly can be taken apart and what alternatives remain for approaching the assembly.

Fitting is a facility that can be used while designing an assembly: When a feature of one part is to be connected to a feature of another part, the system can adjust dimensions in such a way that they fit. With inference checking, an assembly can be checked afterwards to see if there are no parts of two objects overlapping. The intersection volumes can be shown in order to make the corrections.

Referring to the concept of empty items in previous section, the idea of fitting can get another meaning as well. When specifying the assembly connections for a major functional part, often one is not much interested in its counter part. The counter part just offers a support function for the functional part. In this fashion, obstructions could be a kind of functional design feature: just specify the support contacts without directly specifying a complete counter part. The obstruction is an attribute to a connection of the functional part and an empty part.



## References

### A

- Asea 87 on page 48, 83  
Asea Robotics: Quaternions, or how to twist a robot<sup>1</sup>. Asea, oct 1987.

### B

- Baartman 90 on page 24, 42  
J.P. Baartman: PARTAS, Process planning for part assembly. TU Delft report FPA90.053
- Biemans 89 on page 16, 17  
F.P.M. Biemans: A reference model for manufacturing planning and control. TU Twente 1989, PhD thesis.
- Boneschanscher 88 on page 42  
N. Boneschanscher, J.H.M. van der Drift: Automatic assembly sequence planning. TU Delft report WPS88.024.
- Boothroyd 82 on page 39, 124, 133  
G. Boothroyd et.al.: Automatic Assembly. M. Dekker Inc., New York, 1982.
- Bottema 79 on page 48, 83  
O. Bottema, B. Roth: Theoretical kinematics. North-Holland, 1979.
- Bouts 90 on page 57, 97  
E. Bouts: Collision detection with bubble hierarchies. TU Delft/Technical Physics report 1990.
- Brady 82 on page 43  
M. Brady et. al.: Robot motion, planning and control. The MIT press, 1982.
- Brussel 84 on page 39  
H. van Brussel et. al.: LOLA, an advanced end-point level off-line robot programming system. Journal of Robotic Systems, 1(2), 1984.
- Buurman 90 on page 19  
J. Buurman, D.J.W. Bierhuizen: A two stage object identification system in the Delft Intelligent Assembly Cell. SPIE conference, Boston, 1990.

1. Note!: this document contains errors in the calculation of twisting angle and twisting axis from the twisting matrix (section 1.3).

  
C

Cutkosky 89 on page 22, 57

M.R. Cutkosky, J.M. Tenenbaum: A methodology and computational framework for concurrent product and process design. Stanford University/Mechanical Engineering report 1989.

## D

Donselaar 90 on page 52, 116, 131

E.J. van Donselaar: Stageverslag TU Delft. HTS Rotterdam 1990, report.  
*Report of training*

## E

Emmerik 90 on page 53

M.J.G.M. van Emmerik: Interactive design of parameterized 3D models by direct manipulation. PhD thesis, Delft University Press, 1990.

## F

Friederichs 89 on page 59

H.W.M.P. Friederichs: Feature recognition voor assemblage-planning. TU Delft report WPS89.034.  
*Feature recognition for assemblage-planning*

Friederichs 91 on page 60, 62, 81

H.W.M.P. Friederichs: Ontwikkeling van een Montage-verbindingen herkenningssysteem. TU Delft report FPA91.013  
*Development of an assembly recognition system*

## G

Grabowski 89 on page 26

H. Grabowski, R. Anderl, M. Schmitt: Das Produktmodellkonzept von STEP. VDI-Z 131 (1989) Nr. 12.

## H

Heemskerk 90 on page 22, 23, 24, 39, 41, 44

C.J.M. Heemskerk: A concept for computer aided process planning for flexible assembly. TU Delft PhD thesis; The Netherlands 1990.

Houten 91 on page 59

F.J.A.M. van Houten: PART, a computer aided process planning system. TU Twente 1991, thesis.

## J

Jasperse 88 on page 44

H.B. Jasperse: Verbindingsmethoden en hun montagebeweging. WPS88.048.  
*Joining methods and their assembly path*

Jonker 88 on page 17, 131

P.P. Jonker et. al.: The architecture of the Delft intelligent assembly cell. TU Delft

report 1988.

- Joshi 87 on page 59  
S.B. Joshi, T.C. Chang: CAD-interface for automated process planning. CIRP seminar on manufacturing systems, Pennsylvania 1987.

## K

- Knook 91 on page 72  
F.A. Knook: Analyse en toepassing van vlakvergelijkingen in Medusa. TU Delft, report FPA91.035.  
*Analysis and application of face equations in Medusa*

- Ko 87 on page 43, 55  
H. Ko and K. Lee: Automatic assembling procedure generation from mating conditions. In: Comp.Aided Design, jan 87.

- Krishnan 91 on page 55  
S.S. Krishnan, A.C. Sanderson: Reasoning about geometry constraints for assembly sequence. IEEE International Conference on Robotics and Automation 1991, California.

## L

- Lieberman 77 on page 42  
L.I. Lieberman, M.A. Wesley: AUTOPASS. IBM Journal of research and Development, vol 21, jul 77.

- Lozano-Perez 83 on page 39, 42, 54  
Th. Lozano-Pérez, M.T. Mason, R.H. Taylor: Automatic synthesis of Fine-Motion Strategies for Robots. Massachusetts Institute of Technology 1983, A.I. Memo 759.

## M


- Mazon 89 on page 54, 55  
I. Mazon and R. Alami: Representation and propagation of positioning uncertainties through manipulation robot programs. LAAS Report 89042; France 1989.

- McConalogue 91 on page 30, 37  
D.J. McConalogue: Object-oriented databases: how do they differ from the relational and what are the implications for CAD?. Techno-data '90, pp 141-163, Berlin 1990

## R

- Reijers 90 on page 17  
L.N. Reijers, H.J.L.M. de Haas: Flexibele produktie automatisering 2, Productiesystemen. De Vey Mestdagh, 1990.  
*Flexible production automation*

- v. Rijn 91 on page 25  
A.M.C. van Rijn: Natural language communication between man and machine. TU Delft 1991, PhD thesis.



Rong-Kwei 88 on page 59

Rong-Kwei Li: A part-feature recognition system for rotational parts. Int. Journal of production research, vol 26 no. 9, 1988.

## S

Schlechtendahl 87 on page 26

E.G. Schlechtendahl: Specification of a CAD\*I Neutral File for CAD geometry. Springer-Verlag 1987.

Sekiguchi 83 on page 43, 55

H. Sekiguchi et. al.: Study on Automatic determination of assembly Sequence. annals of the CIRP, vol 32/1/1983.

Shpitalni 89 on page 55

M. Shpitalni, G. Elber, E. Lenz: Automatic Assembly of three-dimensional Structures via Connectivity Graphs. Annals of the CIRP, vol. 38/1/1989.

Storm 88 on page 52, 131

T.Storm: Product specification for DIAC. TU Delft report WPS88.059.

Stroustrup 87 on page 30

B. Stroustrup: The C++ programming language. Addison-Wesley 1987.

## T

Ter Bekke 91 on page 30, 31

J.H.ter Bekke: Semantic data modelling in a relational environment. TU Delft 1991, PhD thesis.

## V

Veltheer 89 on page 22, 59, 60

A. Veltheer: Identification of connections between parts in assembly products. TU Delft, report WPS89.043.

Vergeest 89 on page 26

J.S.M. Vergeest: STEP-standaard biedt mogelijkheden voor betere uitwisseling van CAD/CAM-data. De Constructeur, augustus 1989/nr. 8.  
*STEP offers better exchange of CAD/CAM data*

Verwer 91 on page 54, 57, 97

B.J.H. Verwer: Distance Transforms. TU Delft 1991, PhD thesis.

## W

Weerd 90 on page 37

P. van der Weerd: Data abstractions in object-oriented simulation. TU Delft report (TWI)90-07.

Weule 89 on page 43, 55

H. Weule, Th. Friedmann: Computer-aided Product Analysis in Assembly-planning. Annals of the CIRP, vol. 38/1/1989.



**Y**  
Yourdon 88 on page 65  
Yourdon: Structured Analysis Workshop. Yourdon Inc., 1988.

**Z**  
Zussman 90 on page 43, 44, 55  
E. Zussman, E. Lenz, M. Shpitalni: An Approach to the Automatic Assembly Planning Problem. Annals of the CIRP, vol. 39/1/1990.



# APPENDICES

## A. The DIAC specification products

This appendix describes products that are used to test the developed systems. It is based on original work done in the context of the Delft Intelligent Assembly Cell [Storm 88] and [Donselaar 90]. An additional product has been tested as well, the Cranfield Benchmark.

### A.1 Design criteria

DIAC is meant to be a research production cell, designed to be highly flexible and capable of assembling a range of products. Among the characteristics of the production facility are [Jonker 88], small series of products and the production of families of products. In order to specify the physical properties of the cell and to focus all research groups in the project to a joint goal, two products were designed. These products are not benchmarks in the sense of a test for an existing assembly cell. Normally, the design of an assembly cell should be based on a clearly described (group of) products. In the case of DIAC, the characteristics of the system prevail, and the products are designed to meet that demand. Therefore, they are called *specification products*.

The products that can be assembled with DIAC meet the following criteria [Storm 88]:

- Designed for Assembly: the product does not conflict with the rules of Design for Assembly [Boothroyd 82].
- Size limits: The completed product is within a cube of (200 x 200 x 200 mm). Parts should be smaller than ca. 150 mm and there are no miniatures (very small parts: < 5 mm).
- Weight limits: Parts should be not heavier than 1 kg each. The weight of one single assembly does not exceed 5 kg.
- Maximum number of parts is 25.

For testing and specification purposes, two products have been designed that meet the criteria above. These products contain several aspects of assembly that can be tested:

- repeating parts, both in similar functions (several locking screws) and other (same screw used for attaching).
- identifiable shapes. Some shapes have intentional non-symmetric aspects and others are symmetrical, in various ways.
- insertion with and without force
- insertion from several directions

### A.2 Prismatic product: DIAC-1

The first product consists primarily of prismatic parts (see figure A.1 on page 132). Several repetitions are present in this product. The most significant connection of this product is the dove-tail. The listing of the BOM as generated from the PDM database<sup>1</sup>:

stacked assembly

circlip

dove tail connection

repeating assemblies

parts used for different functions,  
but instanced from same object (screw)

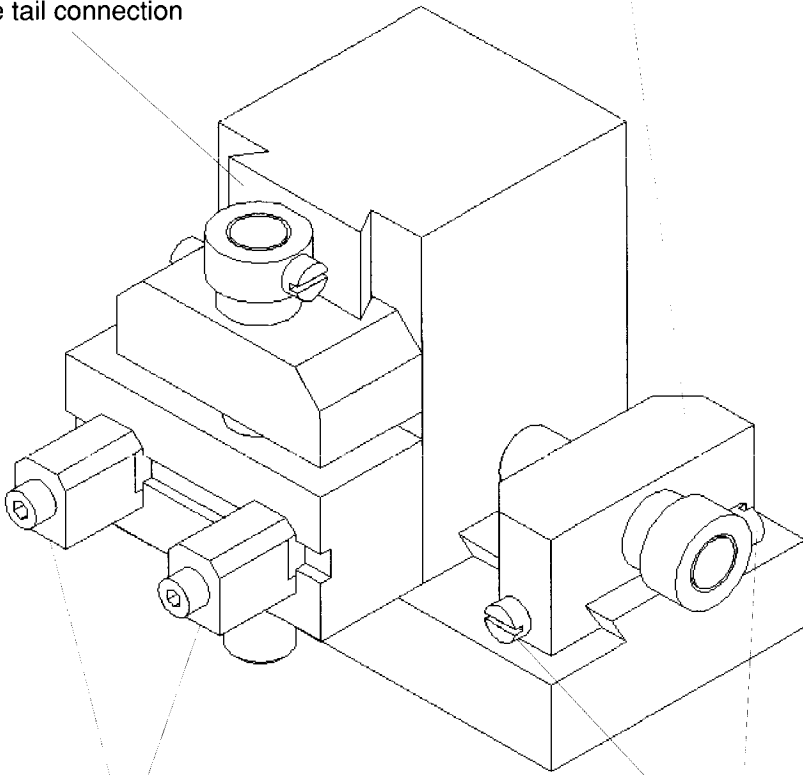
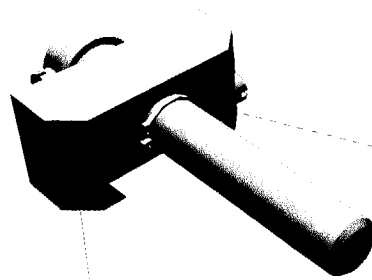


figure A.1 DIAC-1

Bill Of Materials for product 1, Diac\_1. Total of 21 parts

Nr	Name	Cnt.	catalogue	Material
1	borgschroef	4		nylon
2	inbusbout	2		St50
3	seegerring	2		St50
4	kartelknop	2		Cu-Zn
5	schroefspil	2		St50
6	spie	2		Cu-Zn
7	steunblok	2		Cu-Zn
8	lagerblok schroefsp	2		Cu-Zn
9	vert. schuifblok	1		Cu-Zn
10	hor. schuifblok	1		Cu-Zn
11	bovenkantelplaat	1		Cu-Zn

1. partial output listing of program **pdm\_bom**

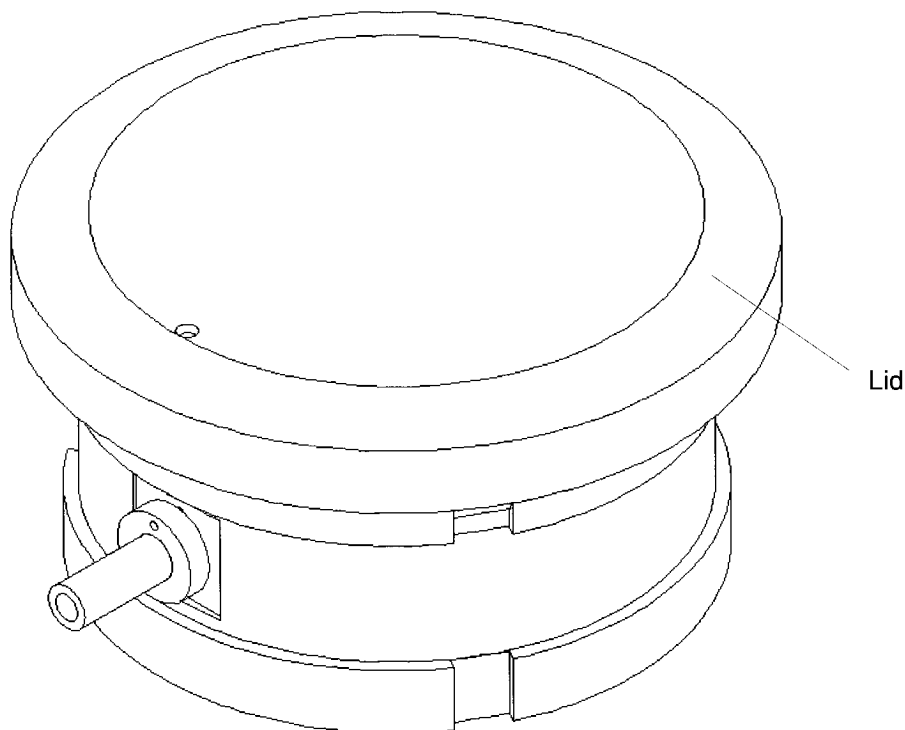


figure A.2 DIAC-2

Note that some of the objects (#1,2,3) actually should have the catalogue field filled in, for they are bought parts, but the drawing did not contain a catalogue specification.

### A.3 Cilindric product: DIAC-2

The second product consists primarily of cylindrical parts. The most significant connection of this product is the fit of the plastic hood with springs. There are two pegs that are pressed in place, so they are presented to the connection modeller as part of a clump. The listing of the BOM:

press-fit peg

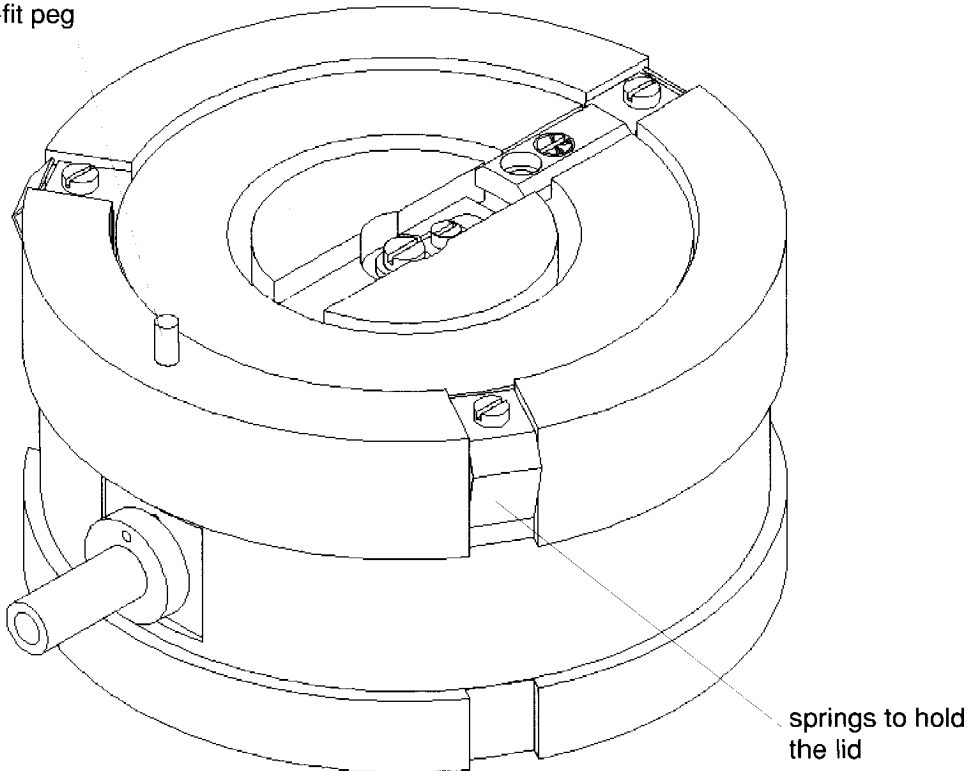


figure A.3 DIAC-2 without the lid

Bill Of Materials for product 2, Diac\_2. Total of 25 parts

Nr	Name	Cnt.	catalogue	Material
12	veer	3		St50
13	M4schr	4		St50
14	deksel	1		perspex
15	toevoerpijp	1		Cu-Zn
16	steunring	1		Cu-Zn
17	huis	1		Al
18	afsluitring	1		Cu-Zn
19	aansluitpijp	1		St50
20	paspen4x15	1		St50
21	regelstift	1		Cu-Zn
22	borgring	1		Cu-Zn
23	borgstrip	1		Cu-Zn
24	regelhuis	1		Cu-Zn
25	schroefM5X12	1		St50
26	M3schr	1		St50
27	blokkeerstrip	1		Cu-Zn
28	bodemring	1		Cu-Zn
29	binnenhuis	1		Cu-Zn
30	paspen4x18	1		St50
31	paspen2x10	1		St50

#### A.4 The special connection product: DIAC-3

Diac-3 was designed to test some special connection configurations. It contains several connections that can not be recognized by the connection modeller in the current implementation.

Bill Of Materials for product 3, Diac 3. Total of 16 parts

Nr	Name	Cnt.	catalogue	Material
32	inbusbout	2		St50
33	pen45	1		
34	pen4545	1		
35	volvopen	1		
36	Title	1		
37	tapeind	1		
38	volvobus	1		
39	Title	2		
40	M8moer	1		
41	ring	1		
42	lepelstaaf	1		
43	lepelblok	1		
44	bovenblok	1		
45	onderblok	1		Al

## A.5 The Cranfield Benchmark

The Cranfield benchmark is a simple assembly benchmark, designed by the Cranfield Institute. It only contains peg/hole connections and one screw. There are however several pegs that have to be placed under an angle, so a SCARA robot would not be able to assemble it without special tooling.

Bill Of Materials for product 4, CRANFIELD. Total of 19 parts

Nr	Name	Cnt.	catalogue	Material
46	screwM4	1		St50
47	lock_pin	8		St50
48	spacer	4		Al
49	lever_top	1		Al
50	spacer_piece	1		Al
51	shaft	1		Al
52	lever	1		Al
53	side_plate	2		Al



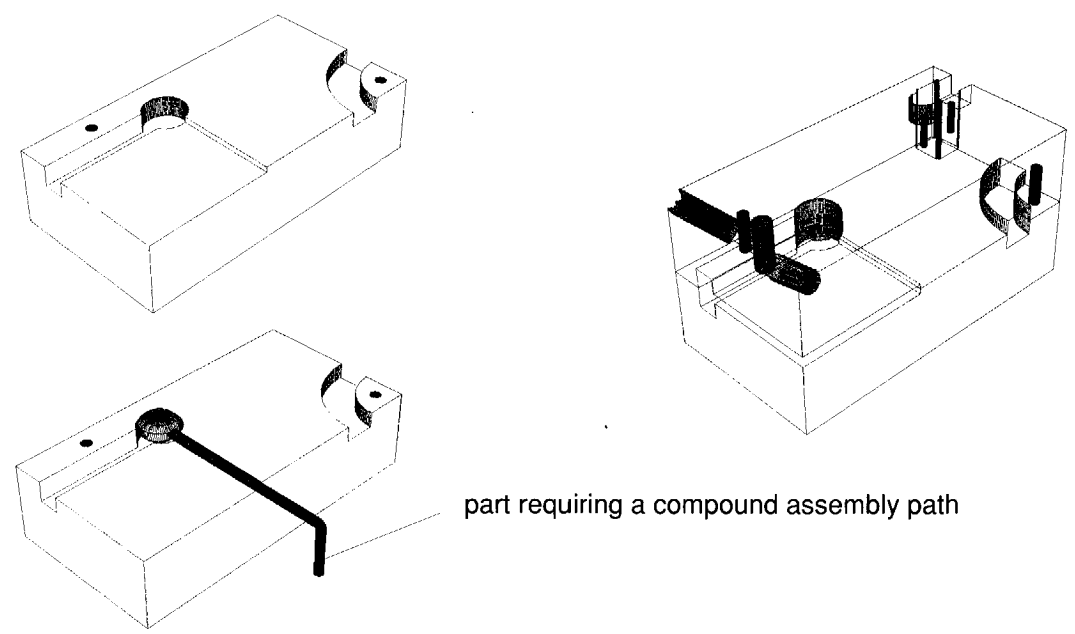
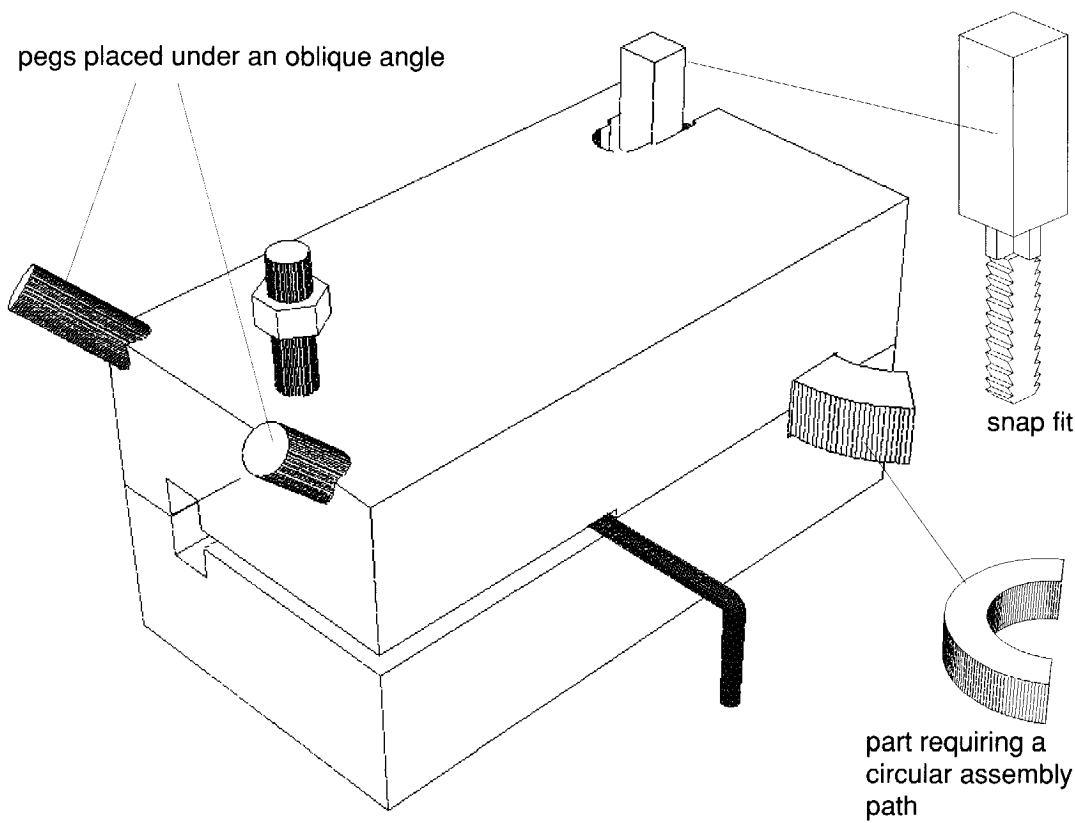


figure A.4 Diac-3

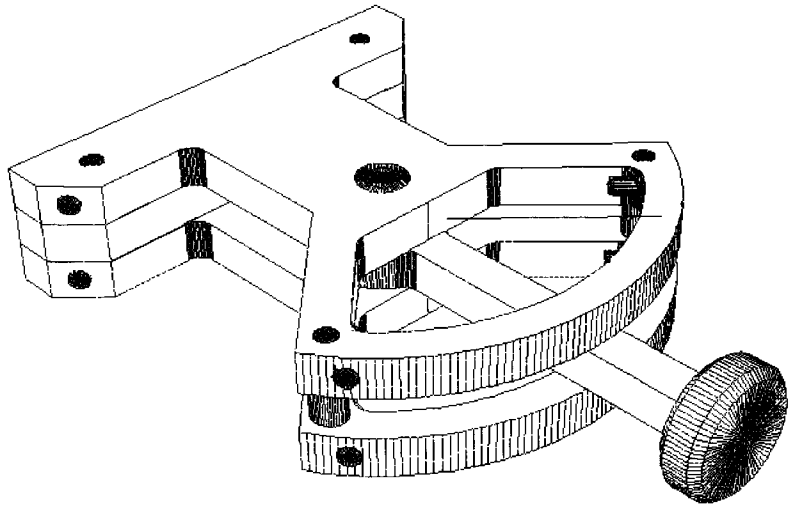
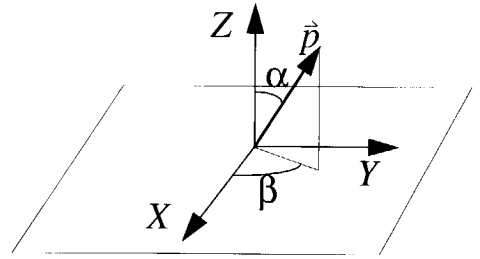


figure A.5 The Cranfield benchmark

# B. Obstruction prototypes

## B.1 Plane



**Properties of the obstruction frame:**

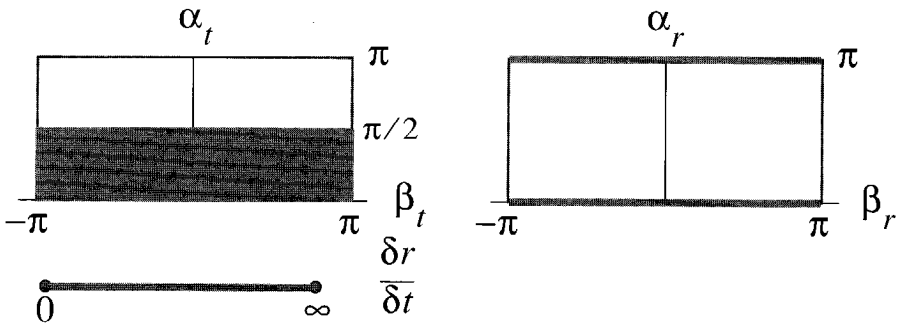
orientation: Z-axis perpendicular to the frame, points in the direction of the moving part.

position: somewhere on the plane

**Significant points:**

Points along the edges of the contact area, as far apart as possible.

**Relocation direction set**

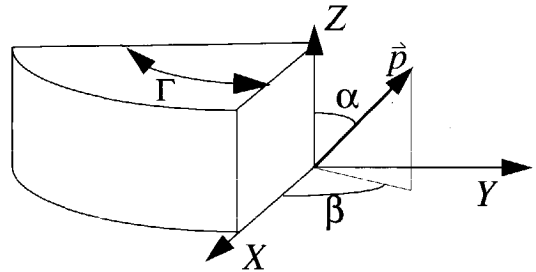


The normal is a constant vector:  $\vec{n} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Support points may be moved over the plane, Translation matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## B.2 Cylindrical face



### **Properties of the obstruction frame:**

orientation: Z-axis perpendicular to the frame, points in the direction of the moving part.

position: somewhere on the plane

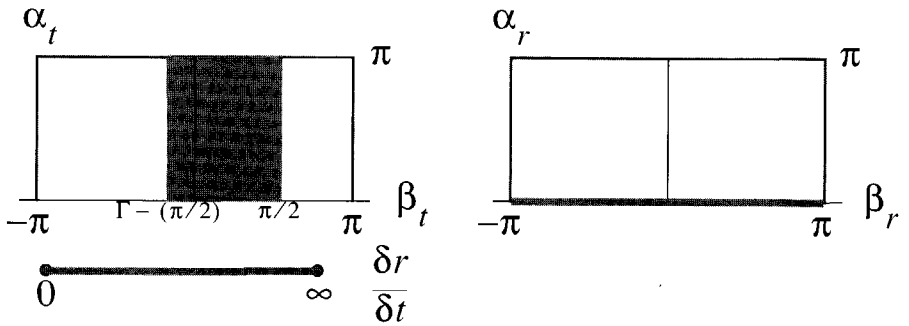
### **Significant points:**

Points along the edges of the contact area.

### **Parameters:**

The angle  $\gamma$  over which the contact extends, while  $\Gamma \leq \pi$ .

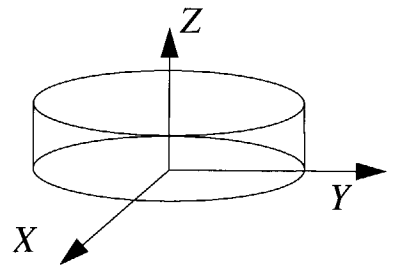
### Relocation direction set



The normal is a function vector:  $\vec{n} = \begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \\ 0 \end{bmatrix}, 0 \leq \gamma \leq \Gamma$

Support points may be moved along the Z axis, Translation matrix:  $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

### B.3 Cylinder



#### Properties of the obstruction frame:

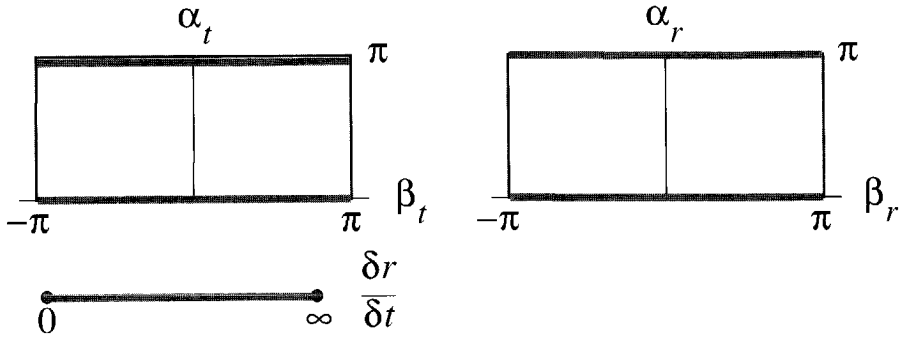
orientation: Z-axis in centre of cylinder

position: somewhere on centre of cylinder

#### Significant points:

Not necessary (imperative twist axis)

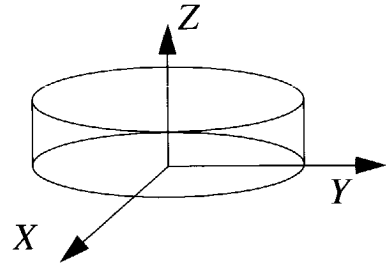
### Relocation direction set



The normal is not relevant (imperative twist axis).

Support points may be moved along the Z axis, Translation matrix: 
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

### B.4 Thread<sup>1</sup>



#### Properties of the obstruction frame:

orientation: Z-axis in centre of cylinder

position: somewhere on centre of cylinder

1. assumed is right-handed thread

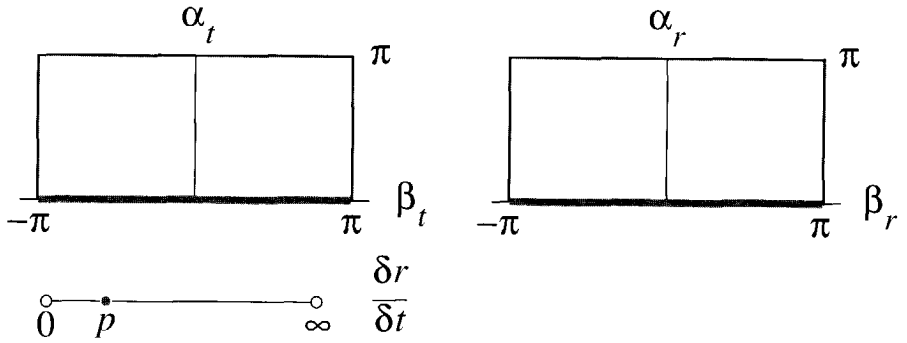
**Significant points:**

Not necessary (imperative twist axis)

**Parameters:**

$p$ , where  $p = 2\pi/\text{pitch}$  [rad/m]

**Relocation direction set**



The normal is not relevant (imperative twist axis).





## List of figures

figure 2.1 Major facility functions and interfacing databases	15
figure 2.2 reference model of MPCs [Biemans 89]	16
figure 2.3 System borders and elements of the FAC planning and control.	18
figure 2.4 Product development and process planning	23
figure 3.1 (incomplete) Overview of the STEP-IPIM entity hierarchy	28
figure 3.2 Mapping of the main PDM entities on STEP	29
figure 3.3	31
figure 3.4	31
figure 3.5	32
figure 3.6	33
figure 3.7	33
figure 3.8	33
figure 3.9	33
figure 3.10	34
figure 3.11	35
figure 3.12	35
figure 3.13	35
figure 3.14	36
figure 3.15	36
figure 4.1 Phases of an Assembly Action	40
figure 4.2 Example relation network and clusters	41
figure 4.3 Eccentric multi-stage peg	44
figure 4.4 Connection model definition	45
figure 4.7 A transformation of a frame	46
figure 4.5 A potential problem	46
figure 4.6 By changing the Insertion point and direction, the approach route is collision free.	46
figure 4.8 Translation	47
figure 4.9 Rotation Matrix: unity vector notation	47
figure 4.10 Definition of a direction	49
figure 5.1 An object with curved surfaces in tiles and inner profiles	51
figure 5.2 Contact forms	52
figure 5.4 The contact situation vertex-surface	56
figure 5.3 Contact topology	56
figure 5.6 Syntactic pattern recognition: grammar of a flat bottom hole	57
figure 5.5 Bubble collision check.	57

figure 5.7 Attributed Adjacency Graph of an object	58
figure 5.8 Examples of AAG features	59
figure 5.9 Normal and direction vectors for contact point $i$	60
figure 5.10 Cylindrical obstruction: two rotation possibilities, A is more likely	61
figure 5.11 Combination of two pure translation direction sets L and R	62
figure 6.1 Process oriented overview of the Product analysis system	66
figure 6.2 Example of dimensioning in a mono sheet	67
figure 6.3 Mono sheets and instancing	68
figure 6.4 Definition of the main entities in the relational version of the PDM	70
figure 6.5 Mono sheet in Medusa	72
figure 6.7 Mapping of ambiguous 2D dimensioning to 3D reality	74
figure 6.6 The chord tolerance	74
figure 6.8 The GUIF	76
figure 6.9 The parts in the final position (as read from the CAD assembly sheet)	78
figure 6.10 Specifying translations by using geometrical elements of the individual parts	79
figure 6.11 The insertion point, interactively specified	79
figure 7.1 Definition of a direction	81
figure 7.2 Relocation vector $m_i$ in a point $i$ in an object	81
figure 7.3 The twist axis and the direction in a point	82
figure 7.4 A contact with a cylindrical face	82
figure 7.6 Mounting a screw	84
figure 7.5 Relocation direction set representation	84
figure 7.7 Unmounting a screw	85
figure 7.9 Alternative twistaxes for a contact with a cylindrical face	86
figure 7.8 Normal and direction vectors in contact point $i$ , for rotation around twist axis.	86
figure 7.11 Irreconcilable imperative twist axes	88
figure 7.10 An imperative and an optional twist axis	88
figure 7.12 Using two obstructions to describe a contact configuration	89
figure 7.13 A separate rotation and translation analysis	90
figure 7.14 Translation transformation and combination	92
figure 8.1 A cylinder in sections has a single continued obstruction.	93
figure 8.2 A discontinuous obstruction must be evaluated into separate ones.	93
figure 8.3 Distant twistaxis checked with significant points of the obstruction	94
figure 8.4 Initial check of contact: intersection check of bubbles around COG	95
figure 8.5 A part of a relation network.	96
figure 8.6 Boundary points	97
figure 8.7 Boundary points generated on a part	98
figure 8.8 The maximum distance from a point	99
figure 8.9 Construction of the criterion for a pair of parts	99
figure 8.10 Erroneous additional contacts found	100

figure 8.11 Flat surface to surface close to edge	101
figure 8.12 Surface to edge	102
figure 8.13 The obstruction of an edge-edge contact is approximated by a small surface perpendicular to the line COG1-COG2	102
figure 9.3 Applying the new direction set.	104
figure 9.1 Initial situation	104
figure 9.2 Applying the direction set until a new contact is encountered	104
figure 9.4 All contacts lost	105
figure 9.5 Check for valid insertion point	105
figure 9.6 Determining the ADS	105
figure 9.7 Limited ADS	107
figure 9.8 Widening assembly path	108
figure 9.9 Visualisation of the sweep from IP to assembled position.	109
figure 10.1 Table of results of the CAD/CAM system	111
figure 10.2 Circlip	112
figure 10.3 The oblique pegs and holes in DIAC-3	112
figure 10.4 The lid of Diac-2 with the three springs	113
figure 10.5 The snap fit peg of Diac-3	114
figure 11.1 Integration of manufacturing data	118
figure 11.2 Object specification with attributes and annotation	119
figure 11.3 Schematic representation of CAD system working levels	120
figure A.1 DIAC-1	132
figure A.2 DIAC-2	133
figure A.3 DIAC-2 without the lid	134
figure A.4 Diac-3	137
figure A.5 The Cranfield benchmark	138



## Index

## Numerics

	3-D detailing	21
A		
	accessibility	42
	ADS	48
	Aggregation	30
	applications	27
	approach direction set	45, 48, 106
	approach phase	39
	architecture	15
	assembly feature	59
	assembly features	44
	assembly path	40
	assembly process model	39
	assembly sequence	41
	Attributed Adjacency Graph	59
B		
	Bill Of Materials	25
	BOM	25
	boundary points	97
	Boundary Representation	26, 27
	Brep	26
	bubble	95
	bubble comparison	56
C		
	CAD*I	26
	catalogue parts	69
	Center Of Gravity	25
	class	30
	Classification	30
	classification	43, 44
	clearance	99
	CL-file	27
	clump	53, 71
	clumps	46
	cluster	42
	COG	25, 95



collision avoidance	42
comid	71
compare-pair	99
compliant motion	40
concurrent engineering	21
connection	31, 42
connection (table)	71
connection feature	61
connection model	43
Constructive Solid Geometry	26
Contact analysis	97
contact area	94
contact phase	39
contact topology	55
Control Model	17
counter part	40
criterion	99
CSG	26
C-space	43

## D

Data Flow Diagram	65
data object	30
Delft Intelligent Assembly Cell	13
design feature	21
DIAC	13
direction set	62
drawing	65
D-relation	100
DXF	26

## E

Express	27
extend angle	90

## F

FAC	14
Facility Model	17
feature	57, 87
final position	40
fine motion planning	42
freedom matrix	44

## G

generalization	31
Grammar	59
guarded motion	39



I	IGES	26
	imperative twist axis	88
	insertion path	44
	insertion phase	40
	insertion point	40, 44, 103
	Instance	31
	instancing	71
	Integrated Product Information Model	27
	IP obstruction	106
	IPIM	27
L		
	layer cluster	42
M		
	Macro substitution	53
	Manufacturing	17
	manufacturing features	21
	Manufacturing Planning and Control System	17
	Manufacturing Resource Planning	26
	mating conditions	60
	mechanical engineering drawing	69
	Medusa	73
	methods	30
	mono sheet	69
	Mount	39
	MPCS	17
	MRP	26
N		
	normal vector	86
O		
	object	30
	object (table)	71
	Object Oriented	30
	obstruction	60, 61, 81, 87
	obstruction prototypes	61, 87, 93
	orthogonal frame	49
P		
	part (table)	71
	path	81
	PDES	26
	PDM	22
	PDM System	22
	physical object	30



	point distance	99
	pre-planning	24
	product (table)	71
	Product analysis system	65
	Product Data Exchange using STEP	26
	Product Data Model	22
	Product Model	17
	production planning	17
	profile	72
	proportion	31, 82
	Proportional relation	31
Q		
	quaternions	48
R		
	RDBMS	37
	relation network	42, 95
	relocation direction set	81, 103
	Relocation space	81
	Rotation Matrix	49
S		
	selective relocation direction set	87
	sequence planning	41
	SET	26
	shape	71
	sheets	65
	significant points	94
	sort cluster	42
	stability	42
	stack cluster	42
	STEP	26
	STEP-IPIM	27
	strategy	39
	structured file	37
	support point	85
	sweep	103
	Syntactic pattern recognition	59
T		
	task planning	42
	tiles	53
	Topology	27
	transformation	48
	twist axis	83



## V

VDAFS	26
vertex	27
vertices	27
voxels	56